Robert Meersman
Zahir Tari
Angelo Corsaro et al. (Eds.)

LNCS 3292

On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops

**OTM Confederated International Workshops and Posters
GADA, JTRES, MIOS, WORM, WOSE, PhDS, and INTEROP 2004
Agia Napa, Cyprus, October 2004, Proceedings**



DOA



Springer

Lecture Notes in Computer Science 3292

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison
Lancaster University, UK

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler
University of Surrey, Guildford, UK

Jon M. Kleinberg
Cornell University, Ithaca, NY, USA

Friedemann Mattern
ETH Zurich, Switzerland

John C. Mitchell
Stanford University, CA, USA

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz
University of Bern, Switzerland

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Bernhard Steffen
University of Dortmund, Germany

Madhu Sudan
Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos
New York University, NY, USA

Doug Tygar
University of California, Berkeley, CA, USA

Moshe Y. Vardi
Rice University, Houston, TX, USA

Gerhard Weikum
Max-Planck Institute of Computer Science, Saarbruecken, Germany

This page intentionally left blank

Robert Meersman Zahir Tari Angelo Corsaro
Pilar Herrero María S. Pérez Milena Radenkovic
Victor Robles Corrado Santoro Antonia Albani
Klaus Turowski Mustafa Jarrar Aldo Gangemi
Erik Duval Peter Spyns Alexios Palinginis (Eds.)

On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops

OTM Confederated International Workshops and Posters
GADA, JTRES, MIOS, WORM, WOSE, PhDS, and INTEROP 2004
Agia Napa, Cyprus, October 25-29, 2004
Proceedings

Springer

eBook ISBN: 3-540-30470-3
Print ISBN: 3-540-23664-3

©2005 Springer Science + Business Media, Inc.

Print ©2004 Springer-Verlag
Berlin Heidelberg

All rights reserved

No part of this eBook may be reproduced or transmitted in any form or by any means, electronic, mechanical, recording, or otherwise, without written consent from the Publisher

Created in the United States of America

Visit Springer's eBookstore at: http://ebooks.springerlink.com
and the Springer Global Website Online at: http://www.springeronline.com

GADA

Pilar Herrero
María S. Pérez
Milena Radenkovic
Victor Robles

JTRES

Angelo Corsaro
Corrado Santoro

MIOS

Antonia Albani
Klaus Turowski

WORM

Mustafa Jarrar
Aldo Gangemi

WOSE

Erik Duval
Peter Spyns

PhDS

Peter Spyns
Antonia Albani
Alexios Palinginis

INTEROP

Hervé Panetto
Monica Scannapieco
Martin Zelm

This page intentionally left blank

OTM 2004 General Co-chairs' Message

The General Chairs of OnTheMove 2004, Larnaca, Cyprus, are once more proud to observe that the conference series we started in Irvine, California in 2002, and continued in Catania, Sicily last year, has turned out to be a concept that attracts a representative selection of today's research in distributed, heterogeneous yet collaborative systems, of which the Internet and the WWW are its prime examples.

Indeed, as such large, complex and networked intelligent information systems become the focus and norm for computing, it is clear that one needs to address and discuss in a single forum the implied software and system issues as well as methodological, theoretical and application issues. This is why the OnTheMove (OTM) Federated Conferences series covers an increasingly wide yet closely knit range of topics such as data and Web semantics, distributed objects, Web services, databases, workflows, cooperation, ubiquity, interoperability, and mobility. OnTheMove wants to be a primary scientific forum where these aspects for the development of internet- and intranet-based systems in organizations and for e-business are addressed in a quality-controlled fundamental way. This third, 2004 edition of the OTM Federated Conferences event therefore again provided an opportunity for researchers and practitioners to understand and publish these developments within their respective as well as within their broader contexts.

OTM first of all co-locates three related, complementary and successful main conference series: DOA (Distributed Objects and Applications), covering the relevant infrastructure-enabling technologies, ODBASE (Ontologies, DataBases and Applications of SEmantics) covering Web semantics, XML databases and ontologies, and CoopIS (Cooperative Information Systems) covering the application of these technologies in an enterprise context through, for example, workflow systems and knowledge management. Each of these three conferences treats its specific topics within a framework of (a) theory, (b) conceptual design and development, and (c) applications, in particular case studies and industrial solutions.

Following and expanding the example set in 2003, we solicited and selected quality workshop proposals to complement the more "archival" nature of the main conferences, with research results in a number of selected and more "avant garde" areas related to the general topic of distributed computing. For instance, the so-called Semantic Web has given rise to several novel research areas combining linguistics, information systems technology, and artificial intelligence, such as the modeling of (legal) regulatory systems and the ubiquitous nature of their usage. We were glad to see that in 2004 several of the Catania workshops reemerged with a second edition (notably WoRM and JTRES), and that four other workshops could be hosted and successfully organized by their respective proposers: GADA, MOIS, WOSE, and INTEROP. We trust that their audiences mutually productively and happily mingled with those of the main conferences.

A special mention for 2004 is in order for the new Doctoral Symposium Workshop where three young postdoc researchers organized an original setup and formula to bring PhD students together and allow them to submit their research proposals for selection. A limited number of the submissions and their approaches were independently evaluated by a panel of senior experts at the conference, and presented by the students in front of a wider audience. These students also got free access to all other parts of the OTM program, and only paid a heavily discounted fee for the Doctoral Symposium itself. (In fact their attendance was largely sponsored by the other participants!) If evaluated as successful, it is the intention of the General Chairs to expand this model in future editions of the OTM conferences and so draw in an audience of young researchers to the OnTheMove forum.

All three main conferences and the associated workshops share the distributed aspects of modern computing systems, and the resulting application-pull created by the Internet and the so-called Semantic Web. For DOA 2004, the primary emphasis stayed on the distributed object infrastructure; for ODBASE 2004, it was the knowledge bases and methods required for enabling the use of formal semantics; and for CoopIS 2004 the main topic was the interaction of such technologies and methods with management issues, such as occurs in networked organizations. These subject areas naturally overlap and many submissions in fact also treat envisaged mutual impacts among them. As for the earlier editions, the organizers wanted to stimulate this cross-pollination with a shared program of famous keynote speakers: this year we got no less than Roberto Cencioni of the EC, Umesh Dayal of HP Labs, Hans Gellersen of Lancaster University, and Nicola Guarino of the Italian CNR! As before we encouraged multiple-event attendance by providing authors with free access to other conferences or workshops of their choice.

We received a total of 350 submissions for the three conferences and approximately 170 in total for the workshops. Not only can we therefore again claim success in attracting a representative volume of scientific papers, but such a harvest allowed the program committees of course to compose a high-quality cross-section of worldwide research in the areas covered. In spite of the large number of submissions, the Program Chairs of each of the three main conferences decided to accept only approximately the same number of papers for presentation and publication as in 2002 and 2003 (i.e., an average of 1 paper out of 4 submitted, not counting posters). For the workshops, the acceptance rate varied but was stricter than before, about 1 in 2, to 1 in 3 for GADA and WoRM. Also, for this reason, we decided to separate the proceedings into two books with their own titles, with the main proceedings in two volumes and the workshop proceedings in a separate, third volume, and we are grateful to Springer for their suggestions and collaboration in producing these books. The reviewing process by the respective program committees as usual was performed very professionally and each paper in the main conferences was reviewed by at least three referees. It may be worthwhile to emphasize that it is an explicit OnTheMove policy that all conference program committees and chairs make their selections completely

autonomously from the OTM organization. Continuing an equally nice (but admittedly costly) tradition, the OnTheMove Federated Event organizers decided again to make ALL (sizeable!) proceedings available to ALL participants of conferences and workshops, independent of their registrations.

The General Chairs really are especially grateful to all the many people who were directly or indirectly involved in the setup of these federated conferences and in doing so made them a success. Few people realize what a large number of people have to be involved, and what a huge amount of work, and, yes, risk organizing an event like OTM entails. In particular we therefore thank our eight main conference PC co-chairs (DOA 2004: Vinny Cahill, Steve Vinoski, and Werner Vogels; ODBASE 2004: Tiziana Catarci and Katia Sycara; CoopIS 2004: Wil van der Aalst, Christoph Bussler, and Avigdor Gal) and our 15 workshop PC co-chairs (Angelo Corsaro, Corrado Santoro, Mustafa Jarrar, Aldo Gangemi, Klaus Turowski, Antonia Albani [2x], Alexios Palinginis, Peter Spyns [2x], Erik Duval, Pilar Herrero, Maria S. Perez, Monica Scannapieco, Paola Velardi, Herve Panetto, Martin Zelm) who, together with their many PC members, did a superb and professional job in selecting the best papers from the large harvest of submissions. We also thank our Publicity Chair (Laura Bright) and Publication Chair (Kwong Yuen Lai), and of course our overall Workshops Chair (Angelo Corsaro).

We do hope that the results of this federated scientific event contribute to your research and your place in the scientific network. We look forward to seeing you at next year's edition!

August 2004 Robert Meersman, Vrije Universiteit Brussel, Belgium
Zahir Tari, RMIT University, Australia

This page intentionally left blank

Organization Committee

The OTM (On The Move) 2004 Federated Workshops, involving six workshops (i.e., GADA – Workshop on Grid Computing and Its Application to Data Analysis, JTRES – Workshop on Java Technologies for Real-Time and Embedded Systems, MIOS – Workshop on Modeling Inter-organizational Systems, WORM – Workshop on Regulatory Ontologies, WOSE – Workshop on Ontologies, Semantics, and E-Learning, PhDS – PhD Symposium, and INTEROP – INTEROP NoE Dissemination Workshop), were proudly supported by RMIT University and Vrije Universiteit Brussel.

Executive Committee

OTM 2004 General Co-chairs	Robert Meersman (Vrije Universiteit Brussel) and Zahir Tari (RMIT University)
GADA 2004 PC Co-chairs	Pilar Herrero (Universidad Politecnica de Madrid), María S. Pérez (Universidad Politécnica de Madrid), Milena Radenkovic (University of Nottingham), and Victor Robles (Universidad Politecnica de Madrid)
JTRES 2004 PC Co-chairs	Angelo Corsaro (Washington University), Corrado Santoro (University of Catania)
MIOS 2004 PC Co-chairs	Antonia Albani (University of Augsburg) and Klaus Turowski (University of Augsburg)
WORM 2004 PC Co-chairs	Mustafa Jarrar (Vrije Universiteit Brussel) and Aldo Gangemi (ISTC-CNR)
WOSE 2004 PC Co-chairs	Erik Duval (Katholieke Universiteit Leuven) and Peter Spyns (Vrije Universiteit Brussel)
PhDS 2004 PC Co-chairs	Antonia Albani (University of Augsburg), Alexios Palinginis (ETHZ), and Peter Spyns (Vrije Universiteit Brussel)
INTEROP 2004 PC Co-chairs	Monica Scannapieco (University of Rome "La Sapienza"), Paola Velardi (University of Rome "La Sapienza"), Hervé Panetto (University Henri Poincaré, Nancy 1), and Martin Zelm (CIMOSA Association)
Publication Chair	Kwong Yuen Lai (RMIT University)
Organizing Chair	Skevos Evripidou (University of Cyprus)
Publicity Chair	Laura Bright (Oregon Graduate Institute)

GADA 2004 Program Committee

José Mare Benford
Chris Greenhalgh
José Luis Bosque
David Walker
Alberto Sanchez
Jon Garibaldi
Toni Cortes
Jemal Abawajy
Javier Segovia
Heinz Stockinger
Andrea Omicini
Omer Rana
Rajkumar Buyya
Miguel Marquina
DomenicoLaforenza
Asunción Gómez
Savas Parastatidis
Angélica de Antonio
Phil Lord
Jesús Carretero
Félix García
Steve Chiu
Vicente Martín
Francisco Rosales
Jan Humble

JTRES 2004 Program Committee

Greg Bollella
Joe Cross
Ron Cytron
Peter Dibble
Chris D. Gill
Giuseppe Di Giore
Miguel De Miguel
Marco Fargetta
Marisol Garcia Valls
Doug Jensen
Doug Lea
Doug Locke
Joe Loyall
M. Teresa Higuera-Toledano
David Holmes
Kelvin Nielsen
Agostino Poggi
John Regehr
Martin Rinard
Douglas Schmidt
Jan Vitek Purdue
Andy Wellings

MIOS 2004 Program Committee

Antonia Albani
Bernhard Bauer
Jorg Becker
Radu D. Bercaru
Peter Bernus
Christoph Bussler
Emmanuel delaHostria
Jan L.G. Dietz
Klaus R. Dittrich
Rony G. Flatscher
Michael Huhns
Zahir Irani
Stefan Klein
Peter Loos
Masao J. Matsumoto
Arturo Molina
Moira Norrie
Oliver Roos
Michael Rosemann
Klaus Turowski
Vijay K. Vaishnavi

WORM 2004 Program Committee

Aldo Gangemi
André Valente
Andrew Stranieri
Bart Verheij
Cecilia M. Sjoberg
Carole Hafner
Giancarlo Guizzardi
Guiraude Lame
Jaime Delgado
Jerome Euzenat
Joost Breuker
Leonardo Lesmo
Marc Wilikens
Mariano F. Lopez
Mustafa Jarrar
Nicklas Lundblad
Richard Benjamins
Rita Temmerman
Robert Meersman
Said Tabet
Valentina Tamma
Youssef Amghar

WOSE 2004 Program Committee

Lora Aroyo
Erik Duval
Wayne Hodgins
Robert Farrell
Paul LeFrere
Ambjorn Naeve
Wolfgang Nejdl
Daniel Rehak
Tyde Richards
Peter Spyns
Frans Van Assche
Martin Wolpers

PhDS 2004 Program Committee

Sonia Bergamaschi
Jaime Delgado
Jan Dietz
Werner Nutt

INTEROP 2004 Program Committee

Giuseppe Berio
Arne Berre
Nacer Boudjlida
Tiziana Catarci
David Chen
Thomas Deproix
Frank W. Jaekel
Maria-Jose Nunez
Michele Missikoff
Hervé Panetto
Colin Piddington
Richard Stevens
Martin Zelm

This page intentionally left blank

Table of Contents

Posters of the 2004 CoopIS (Cooperative Information Systems) International Conference

Dynamic Publish/Subscribe in Mobile Peer-to-Peer Systems 1
Eiko Yoneki, Jean Bacon

The Virtual Cooperative Project Taking into Account Work Semantic Meaning in Architectural Cooperative Design 3
Mohamed Bouattour, Gilles Halin, J. Claude Bignon

Collaborative Fuzzy Agents Using Constraint Logic Programming over Finite Domains 5
Susana Muñoz-Hernández, Jose Manuél Gómez-Pérez

Design a High-Level Language for Large Network Security Management 7
Jangha Kim, Byungwook Song, Kanghee Lee, Sangwook Kim, Daesik Choi, Jungtaek Seo

A Context-Based Awareness Mechanism for Mobile Cooperative Users ... 9
Manuele Kirsch-Pinheiro, Marlène Villanova-Oliver, Jérôme Gensel, Hervé Martin

Access Control-Based Distribution of Shared Documents 12
Sonia Mendoza, Alberto L. Morán, Dominique Decouchant, Ana María Martínez Enríquez, Jesus Favela

Service-Oriented Architecture of TeleCARE 14
Victor Guevara-Masis, Hamideh Afsarmanesh, Louis O. Hertzberger

Feedback Partitioning and Decomposition-Oriented Analysis of Cyclic Workflow Graphs 17
Yongsun Choi, J. Leon Zhao

Towards Implementing Consistency in OMG MOF-Based Repository Systems 19
Ilia Petrov, Stefan Jablonski, Marc Holze

Privacy-Preserving ID-Based Service Provision in Ubiquitous Computing Environments 21
Keon Myung Lee, Sang Ho Lee

Agents for Decentralised Process Design 23
Iain Stalker, Nikolay Mehandjiev, Georg Weichhart, Kurt Fessl

Posters of the 2004 ODBASE (Ontologies, Databases, and Applications of Semantics) International Conference

Knowledge Signatures for Information Integration 26
Judi Thomson, Andrew Cowell, Patrick Paulson, Scott Butner, Mark Whiting

Efficient Processing of Huge Ontologies in Logic and Relational Databases 28
Timo Weithöner, Thorsten Liebig, Günther Specht

Representing and Reasoning for Spatiotemporal Ontology Integration 30
Nacera Bennacer, Marie-Aude Aufaure, Nadine Cullot, Anastasiya Sotnykova, Christelle Vangenot

Effective and Efficient Melody-Matching Method in a Large-Scale Music Database 32
Sung-Phil Heo

Posters of the 2004 DOA (Distributed Objects and Applications) International Conference

A CORBA-Based System for the Monitoring of Petroleum Fields 34
Diogo Salim, Thais Batista, Adelardo Medeiros

Context-Aware Distributed Deployment of Component-Based Applications 36
Dhouha Ayed, Chantal Taconet, Nawel Sabri, Guy Bernard

Towards an Incremental Process Model Based on AOP for Distributed Component-Based Software Development 38
Alexandre Alvaro, Daniel Lucrédio, Antonio Francisco do Prado, Eduardo Santana de Almeida

Supporting Nomadic Desktop Computing Using an Internet Application Workbench 40
Stefan Paal, Reiner Kammüller, Bernd Freisleben

A Distributed Object Platform to Achieve Nomadic Interactions 44
Domenico Cotroneo, Armando Migliaccio, Stefano Russo

A Virtual-Machine-Based Middleware 46
Alcides Calsavara, Agnaldo K. Noda, Juarez da Costa, Carlos Kolb, Leonardo Nunes

Workshop on Grid Computing and Its Application to Data Analysis (GADA)

PC Co-chairs' Message 48
Editors: Pilar Herrero, María S. Pérez, Milena Radenkovic, Victor Robles

Simulation of Mars Impact Cratering on a Grid Enviroment 50
Eduardo Huedo, Alain Lepinette, Rubén S. Montero, Ignacio M. Llorente, Luis Vázquez

A Grid Environment for Simulation and Optimization and a First Implementation of a Biomedical Application 59
Silke Halstenberg, Karl-Uwe Stucky, Wolfgang Süß

Application of Grid Security to Personal Distributed Environments 68
Allan Tomlinson, Scarlet Schwiderski-Grosche

An Adaptive Cost Model for Distributed Query Optimization on the Grid 79
Yahya Slimani, Faiza Najjar, Najla Mami

GDIS: A Service-Based Architecture for Data Integration on Grids 88
Carmela Comito, Domenico Talia

Managing Long Running Queries in Grid Environment 99
Ruslan Fomkin, Tore Risch

Improving Distributed Data Mining Techniques by Means of a Grid Infrastructure 111
Alberto Sánchez, José M. Peña, María S. Pérez, Víctor Robles, Pilar Herrero

Improving Grid Services Security with Fine Grain Policies 123
Fabrizio Baiardi, Fabio Martinelli, Paolo Mori, Anna Vaccarelli

Resource Dissemination Policies on Grids 135
Marta Beltrán, Jose L. Bosque, Antonio Guzmán

A Grid Framework for Approximate Aggregate Query Answering on Summarized Sensor Network Readings 144
Alfredo Cuzzocrea, Filippo Furfaro, Giuseppe M. Mazzeo, Domenico Saccà

Large-Scale Resource Selection in Grids 154
Ali M. Roumani, David B. Skillicorn

A Proposal of a CIM-Based Policy Management Model for the OGSA Security Architecture 165
Félix J. García Clemente, Gregorio Martínez Pérez, Óscar Cánovas Reverte, Antonio F. Gómez Skarmeta

A Grid-Based Approach for Processing Group Activity Log Files 175
Fatos Xhafa, Santi Caballé, Thanasis Daradoumis, Nan Zhou

Design, Implementation, and Evaluation of a Resource Management Multiagent System for a Multimedia Processing Grid 187
Isaac Chao, Ramon Sangüesa, Oscar Ardaiz

Towards a Peer-to-Peer Event Broker Grid in a Hybrid Network Environment 198
Eiko Yoneki, Jean Bacon

Bioinformatics Data Access Service in the ProGenGrid System 211
Giovanni Aloisio, Massimo Cafaro, Sandro Fiore, Maria Mirto

MSF: A Workflow Service Infrastructure for Computational Grid Environments 222
Seogchan Hwang, Jaeyoung Choi

Using PROTEUS for Modeling Data Mining Analysis of Proteomics Experiments on the Grid 232
Mario Cannataro, Pietro Hiram Guzzi, Tommaso Mazza, Pierangelo Veltri

Workshop on Java Technologies for Real-Time and Embedded Systems (JTRES)

PC Co-chairs' Message 244
Editors: Angelo Corsaro, Corrado Santoro

Real–Time Teaching with Java: JPR^3 246
Diego Alonso, Juan A. Pastor, Bárbara Álvarez

Toward a Component-Based Embedded Java-Oriented Operating System 256
Frédéric Dang Tran, Jean-Philippe Fassino, Olivier Lobry, Jacques Pulou, Nicolas Rivierre

Studying the Behaviour of the Single Parent Rule in Real-Time Java 268
M. Teresa Higuera-Toledano

Supporting Mixed Criticality Applications in a Ravenscar-Java Environment 278
Hao Cai, Andy Wellings

Optimizing Real-Time Embedded Systems Development Using a RTSJ-Based API 292
Marco A. Wehrmeister, Leandro B. Becker, Carlos Eduardo Pereira

Lightweight Firm Real-Time Extensions for Low Memory Profile Java Systems 303
Helge Böhme, Ulrich Golze

Towards an Understanding of the Expressive Power of the RTSJ Scoped Memory Model 315
Andrew Borg, Andy Wellings

Memory Management Based on Method Invocation in RTSJ 333
Jagun Kwon, Andy Wellings

Encapsulation of IEC 61499 Function Blocks Using Real-Time Java According to the RTSJ 346
Marcus Tangermann, Christian Schwab, Arndt Lüder, Luca Ferrarini, Carlo Veber

No Heap Remote Objects: Leaving Out Garbage Collection at the Server Side 359
Pablo Basanta-Val, Marisol Garcia-Valls, Iria Estévez-Ayres

A Time Predictable Instruction Cache for a Java Processor 371
Martin Schoeberl

Indoor Positioning for Location-Aware Applications on Java-Based Mobile Devices 383
Cristiano di Flora, Massimo Ficco, Stefano Russo

Design Patterns for RTSJ Application Development 394
Angelo Corsaro, Corrado Santoro

Separating Soft Real-Time from Functionality Concerns in Complex Object-Oriented Applications 406
Antonella Di Stefano, Marco Fargetta, Giuseppe Pappalardo, Emiliano Tramontana

Workshop on Modeling Inter-Organizational Systems (MIOS)

PC Co-chairs' Message 418
Editors: Antonia, Albani, Klaus Turowski

The e-Lollapalooza GlobalWorkflow Modeler: A Registry-Based Workflow Modeling Methodology and System 419
Kwanghoon Kim, Hyukjae Ryoo, Jaekang Won

Concepts of Flexibility for Efficient Integration of B2B Processes 431
Jelena Zdravkovic

Using Patterns in the Design of Inter-organizational Systems – An Experience Report 444
John Novatnack, Jana Koehler

Managing Ad Hoc Virtual Enterprises in Open SME Networks: The SCOOP Solution 456
Michael Ryba, Gerhard Wickler

Engineering of Dependable Complex Business Processes Using UML and Coordinated Atomic Actions 468
Nicolas Guelfi, Guillaume Le Cousin, Benoît Ries

Cross-Enterprise Business Process Management Architecture – Methods and Tools for Flexible Collaboration 483
Sven Zang, Anja Hofer, Otmar Adam

From the Analysis of Cooperation Within Organizational Environments to the Design of Cooperative Information Systems: An Agent-Based Approach 495
Djamel Benmerzoug, Zizette Boufaida, Mahmoud Boufaida

Transforming UMM Business Collaboration Models to BPEL 507
Birgit Hofreiter, Christian Huemer

Using BPEL Process Descriptions for Building Up Strategic Models of Inter-organizational Networks 520
Dominik Schmitz, Gerhard Lakemeyer, Günter Gans, Matthias Jarke

A Meta Ontology for Organizations 533
Jan L.G. Dietz, Nathalie Habing

A Model-Driven Approach to Designing Cross-Enterprise Business Processes 544
Bernhard Bauer, Jörg P. Müller, Stephan Roser

A Component-Based Architecture for Protocol Vector Conversion in Inter-organizational Systems 556
Bettina Bazijanec, Christian Winnewisser, Antonia Albani, Klaus Turowski

Workshop on Regulatory Ontologies (WORM)

PC Co-chairs' Message 568
Editors: Mustafa Jarrar, Aldo Gangemi

Ontology-Based E-government Thematic Services Based on Topic Maps 569
Ioannis Tsampoulatidis, Dimitrios Tzovaras, Michael G. Strintzis

Workflow Description of Digital Rights Management Systems 581
Silvia Llorente, Eva Rodríguez, Jaime Delgado

Automatic Classification and Analysis of Provisions in Italian Legal Texts: A Case Study 593
Roberto Bartolini, Alessandro Lenci, Simonetta Montemagni, Vito Pirrelli, Claudia Soria

Engineering an Ontology of Financial Securities Fraud 605
Gang Zhao, John Kingston, Koen Kerremans, Frederick Coppens, Ruben Verlinden, Rita Temmerman, Robert Meersman

Extracting Legal Propositions from Appellate Decisions with Text Discourse Analysis Methods 621
Woojin Paik, Jee Yeon Lee

Cases and Dialectical Arguments – An Approach to Case-Based Reasoning 634
Bram Roth, Bart Verheij

Modeling the Tax Code 652
Eric Melz, Andre Valente

Modelling Changes in Ontologies 662
Johann Eder, Christian Koncilia

An Ontology-Based Model for Representing "Bundle-of-Rights" 674
Maria-Teresa Sagri, Daniela Tiscornia, Aldo Gangemi

Workshop on Ontologies, Semantics, and E-learning (WOSE)

PC Co-chairs' Message 689
Editors: Erik Duval, Peter Spyns

Improving LOM-Based Interoperability of Learning Repositories 690
Germán M. Rivera, Bernd Simon, Juan Quemada, Joaquín Salvachúa

Domain Ontology as a Resource Providing Adaptivity in eLearning 700
Galia Angelova, Ognian Kalaydjiev, Albena Strupchanska

Towards a Global Component Architecture for Learning Objects: An Ontology Based Approach 713
Katrien Verbert, Joris Klerkx, Michael Meire, Jehad Najjar, Erik Duval

Human-Generated Learning Object Metadata 723
Andrew Brasher, Patrick McAndrew

Extracting Sub-ontology from Multiple Ontologies 731
Dazhou Kang, Baowen Xu, Jianjiang Lu, Peng Wang, Yanhui Li

MAZETTE: Multi Agent MUSETTE for Sharing and Reusing Ontologies 741
Jesus Arana, Salima Hassas, Yannick Prié

Literate Ontology Development 753
Achim Mahnke, Bernd Krieg-Brückner

An E-learning Concept Mindmap and the Management of the Underlying Learning Objects 758
Jeanne Schreurs, Gerd Balette, Rachel Moreau

Theory and Semi-automatic Generation of Bridge Ontology in Multi-ontologies Environment 763
Peng Wang, Baowen Xu, Jianjiang Lu, Yanhui Li, Dazhou Kang

Integrating Ontologies into Learning Management Systems – A Case of Czech 768
Pavel Smrž

Towards an Ontological Support for eLearning Courses 773
Antonio De Nicola, Michele Missikoff, Frederica Schiappelli

PhD Symposium (PhDS)

PC Co-chairs' Message 778
Editors: Peter Spyns, Antonia Albani, Alexios Palinginis

Management of Dynamic and Imperfect User Context Information 779
Andreas Schmidt

Bridging Information Gaps in an OPM-Based System Development Lifecycle 787
Avi Soffer

Revising and Managing Multiple Ontology Versions in a Possible Worlds Setting 798
Pieter De Leenheer

On the Verification of Web Services Compatibility: A Petri Net Approach 810
Manu De Backer

Usability of Visual Information Retrieval Metaphors for Object-Oriented Databases 822
Mariusz Trzaska, Kazimierz Subieta

Automated Compatibility Tests for Business Related Aspects of Software Components 834
Johannes Maria Zaha

Formal Infrastructure for Modelling Intelligent Agents with Agent UML and Petri Nets 842
Marina Bagic

Model-Driven Design of Distributed Applications 854
João Paulo A. Almeida

INTEROP NoE Dissemination Workshop (INTEROP)

INTEROP NoE: Interoperability Research for Networked Enterprises Applications and Software 866
Hervé Panetto, Monica Scannapieco, Martin Zelm

Author Index 883

This page intentionally left blank

Dynamic Publish/Subscribe in Mobile Peer-to-Peer Systems

Eiko Yoneki and Jean Bacon

University of Cambridge Computer Laboratory
Cambridge CB3 0FD, United Kingdom
{Eiko.Yoneki, Jean.Bacon}@cl.cam.ac.uk

There is a diversity of large scale network environments from Internet-scale peer-to-peer systems to sensor networks. Mobile ad hoc networks (MANET) lie between these extremes. The combination of mobile devices and ad-hoc networks is best managed by the creation of highly dynamic, self-organizing, mobile peer-to-peer systems. Asynchronous communication is essential to support anonymous coordination of communication in such ubiquitous environments. There have been efforts to create efficient multicast communication for MANET. When the scale of group communication is small, multicast can be efficient. However, if the group gets larger, and a multicast group must define a specific topic for each receiver, such as a consumer alert with a fine-grained subscription, this requires a more efficient communication mechanism such as a publish/subscribe paradigm. Publish/subscribe becomes powerful in mobile ad hoc environments, when all the nodes are not in an exclusively ad hoc topology. In this case some nodes may be connected to the Internet backbone or may be relay nodes for different network environments. For example (see Fig.1), event broker grids are formed over mixed networks. An event publisher broker node can act as a gateway from a sensor network, performing data aggregation, and distributing aggregated data to other mobile networks based on the contents. Broker nodes that offer data aggregation services can coordinate data flow efficiently. There is a strong need to introduce event-based middleware support.

Event-based middleware is based on the publish/subscribe communication paradigm. It became popular, because asynchronous and multipoint communication is well suited for distributed computing applications. The first event-based middleware systems were based on the concepts of channel or topic communication. These systems categorize events into pre-defined groups. In an attempt to overcome the limitation on subscription declarations, the content-based model has been introduced, which allows subscribers to use flexible querying languages to declare their interests with respect to event contents.

We propose a dynamic publish/subscribe system in mobile P2P environments, which integrates an extended ODMRP (On-Demand Multicast Routing Protocol) and content-based messaging. ODMRP is simple and scalable, which avoids the drawbacks of multicast trees in MANET environments such as intermittent connectivity and frequent tree configuration. Thus, ODMRP is chosen for the underlying data dissemination mechanism. Because of dynamic MANET environments, multi-hop routing has to consider many contexts from physical constraints, location, and mobility. ODMRP improves its performance

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 1–2, 2004.
© Springer-Verlag Berlin Heidelberg 2004

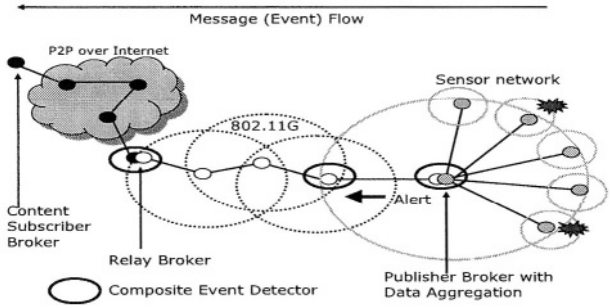


Fig. 1. Message Flow by Event Brokers

using mobility and location information. Many contexts belong to the network, and are outside the scope of middleware. On the other hand, the semantic contexts from upper layers should be used for building an efficient communication by the network layer component. Here, there is a need to exchange contexts among applications, the middleware tier, and the network layer component to build an optimized data dissemination structure and group maintenance mechanism. The proposed system uses content-based subscriptions as inputs, when ODMRP builds the routing table to disseminate the events. However the interface is generic, and any contexts from the middleware tier can be applied. The approach is conceptually similar to Active Networks, which allow users to inject customized programs into the nodes of the network. Content-based subscriptions at a subscriber broker node are aggregated and summarized into a compact data format in Bloom filters, and the publisher broker node defines the multicast group by examining the propagated subscriptions. This dynamic group setting is one of the difficult challenges, and the current approach includes representing in a graph and clustering by K-means. Content-based routing is computationally more expensive than explicit-address routing or topic-based routing. In wired peer-to-peer network environments, content-based subscriptions are used to construct routing itself. However, over dynamic mobile ad hoc network environments, the multicast channel life is ephemeral and subscriptions are more specific. In such network environments, epidemic style data dissemination should work better. Thus, the proposed approach to establish a dynamic channel would have good use for realizing content-based subscriptions.

Note that it is necessary to develop data structures and algorithms that can introduce efficient subscription propagation and event matching. Naturally XML is a good candidate along with XPath as a subscription language. Event brokers provide a rich subscription language including an event correlation service by the embedded composite event detector. Preliminary simulation results show that performance is improved by introducing a publish/subscribe scheme; the reduction of network traffic and the reliability improvement given by the mesh topology are especially significant. Evaluation requires many different aspects such as complexity of subscriptions and messages, efficiency of global subscription aggregation, mobility patterns, network traffic load, and others over the dynamic environments, and the further experiments are in progress.

The Virtual Cooperative Project Taking into Account Work Semantic Meaning in Architectural Cooperative Design

Mohamed Bouattour, Gilles Halin, and J. Claude Bignon

CRAI, Centre de recherche en Architecture et Ingénierie, 2 rue Bastien Lepage, 54000 Nancy, France
{bouattou, halin, bignon}@crai.archi.fr
http://www.crai.archi.fr

An act of design is a designer's interaction within a group of designers. This group is characterized by instruments, competence and proficiency. Every act constitutes a round-trip between the designer group and a common base. Design is often conducted in a collective way. Points of view are often shared between contributors and decisions about different project aspects are submitted for common approbation.

In this context, current Computer Supported Cooperative Work tools don't sufficiently take in account the cooperative dimension and the implicit nature of building designer work. These systems must help us to obtain maximum data which allows good objectivity of evaluation. Nevertheless, their design complexity results from the same cooperative work complexity (obligation to share time, tracing action difficulties, conformist attitudes, non-effective contribution, influences and dominance, overflow of data, coordination problems, etc.).

The Virtual Cooperative Project (VCP) is a project that we had initiated in the codesigning domain, having as a target the definition, the design and the realization of a model able to assist the cooperative design in architecture. This model will be the support for the development of a cooperative system. This project rests on the use of Digital Project notion and the semantic meaning of objects. This research represents a new approach because it not based on management of documents but on all data relative to works.

In fact, when concentrating on the data exchanged during the project design, the works are the main focal point. A work represents a physical or spatial object making up the basic brick of an architectural project... In the same way, every project work holds some relation with its 'environment': with the actors who design it, the documents that represent it and the activities that create it.

Our research proposes a statement of building collective work, its context and tools. Then we present one vision of the digital project deduced from the analysis of the IFC model "Industry Foundation Classes". This first world model answer is adapted from the industry domain but their building sector adjustment raises some data exchange problems especially in the compatibility of the different profession versions of the D.M. We will evoke aspects of these problems that are fundamental for our cooperation research works.

We justify the interest of a new model of cooperative design where the relational organization of the project is taken into account. This model proposes

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 3–4, 2004.
© Springer-Verlag Berlin Heidelberg 2004

an adaptive and navigable vision of the project and takes account of the current practices flexibility. The meta- modelling approach allowed us to model our cooperative needs and the architecture of a system. The development of this system that we name Bat'Map, constitutes both an aid of cooperative design and a project management help.

In the objective to validate the Bat'Map cooperative system as well as his meta-model, we conduct an experimentation to assist cooperation design in a building project. The script adopted for the experience describes the design of a building and makes interacting actors in a distributed asynchronous mode of coordination.

This research aims to give an account of the D.M evolution of a project and of its works during its life cycle. The identification of the different states of works enumerated previously will allow actors to have a clearer idea of every work and D.M statute. The semantics thus obtained will allow actors to adapt their vision of the design evolution and to avoid wrong decisions. Finally, visualization of the different D.M updates allows us to have a trace of actors' actions (author, date and modification objects), saves time in the identification of changes and permits us to specify the respective legal responsibilities linked to modification, creation or forgotten works.

References

Katzy, B., Ma, X.,. A research note on virtual project management systems, 8th International Conference on Concurrent Enterprising, (June2002) 518

Rueppel, U., Meissner, U., Theiss, M.,. An agent-based Plateform for Collaborative Building Engineering, 9th International Conference on Computing in Civil and Building Engineering, Taiwan, (April 2002) 1053

Salber, D., Coutaz, J., Decouchant, D., Riveill, M.,. De l'observabilité et de l'honntêté : le cas du contrôle d'accès dans la Communication Homme-Homme Médiatisée, CLIPS/BULL - IMAG, IHM'95, Toulouse, (Octobre 1995) 28

Ellis, C., Wainer, J.,. A conceptual model of groupware, In Proc. CSCW'94, ACM, (1994) 79

Dourish, P., Edwards, K., Larmaca, A. and Salisbury, M.,. Presto: An Experimental Architecture for Fluid Interactive Document Spaces, ACM Transactions on Computer-Human Interaction, Vol. 6, No. 2, (1999) 133-161

I.A.I, International Alliance for Interoperability,. IFC Technical Guide, enabling interoperability to the AEC/FM Industry, Version 2.x, (2000)

Jeng, T.,. Coordination of distributed design activities: A rule-driven approach, CAAD 9th Conference, Computer Aided Architectural Design, Futures, (2001) 416

Hanser, D.,. Proposition of an auto coordination model in design situation, application in building domain, PHD Thesis, CRAI, Nancy, (2003) pp85.

Halin, G., Hanser, D., Bignon, J.C.,. User Adaptive Visualization of Cooperative Architectural Design, International journal of architectural computing, Issue 01, Volume 02, (2004) 89-107

Collaborative Fuzzy Agents Using Constraint Logic Programming over Finite Domains*

Susana Muñoz-Hernández and Jose Manuél Gómez-Pérez

School of Computer Science, Technical University of Madrid (UPM)

The introduction of Fuzzy Logic into Logic Programming (LP) has resulted into the development of several Fuzzy Prolog systems. These systems replace the inference mechanism of Prolog with a fuzzy variant which is able to handle partial truth. Most of these systems implement the fuzzy resolution introduced by Lee, examples being Prolog-Elf, Fril Prolog, and F-Prolog. Truth values associated to fuzzy variables can be represented in an ordeal of different flavors, such as real numbers, percentiles, intervals, unions of intervals, and continuous or discrete functions on different domains. [GMV04] presented a Fuzzy Prolog Language that models interval-valued Fuzzy Logic, implemented using CLP($\mathcal{R}$). This Fuzzy Prolog system uses both the original inference mechanism of Prolog, and constraint handling facilities provided by CLP($\mathcal{R}$) to represent and manipulate the concept of partial truth. In that approach, a truth value is a finite union of sub-intervals on [0, 1] [1]. A truth value will be propagated through the rules by means of an *aggregation operator*. The system is user-extensible simply by adding new *aggregation operators* to the library.

Given many of the most interesting fuzzy problems deal with a discrete range of truth values in this work we have implemented a discrete Fuzzy Prolog system. We represent ranges of discrete truth values using Constraint Logic Programming over Finite Domains (CLP($\mathcal{FD}$) [Van89])). This allows to produce finite enumerations of constructive answers instead of constraint expressions. We have also added a new aggregation operator called *inter_m*, intended to obtain a consensus amongst different fuzzy agents, which provides an interval of truth values common to all variables [2].Our implementation is not based on CLP($\mathcal{R}$) (like the original Fuzzy Prolog) but CLP($\mathcal{FD}$) is used instead, allowing to represent truth values discretely. Fuzzy Prolog clauses are expanded at compilation time to CLP($\mathcal{FD}$) clauses and CLP($\mathcal{FD}$) solvers are applied to solve discrete fuzzy problems.

CLP augments LP semantics with constraint handling capabilities, including the ability to solve them by means of user-transparent constraint solvers. Constraints can come in very different flavors. Examples of constraint systems are [dis]equations over $\mathcal{R}$, $\mathcal{Q}$, or $\mathcal{FD}$ whose variables range over finite sets with a

* This work has been partially funded by the European Union IST program under contract IST-2001-34717, Amos, and by Spanish MCYT projects TIC 2002-0055, CUBICO and TIC 2003 - 01036, SOFFIE.

[1] An interval is a particular case of union of one element, and a unique truth value is a particular case of having an interval with only one element.

[2] If such consensus did not exist (intersection void) the minimum is returned as a conservative measure.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 5–6, 2004.
© Springer-Verlag Berlin Heidelberg 2004

complete order among their elements. $\mathcal{FD}$ is one of the more widely used constraint domains, since the finiteness of the domain of the variables allows, in the worst case, a complete traversal of each variable range when searching for a solution. Operationally, a CLP($\mathcal{FD}$) program first declares initial ranges for variables, then a set of relationships are set up among them, and finally a search procedure is called to bind the variables to *definite* values. The first phase (setting up equations) fires a process called *propagation,* in which inconsistent values are removed from the domain of the variables. Usually, this state does not end with a definite value for each variable, which is sought for in a second process, called *labeling,* where variables are assigned values within their domains until all of them have a unique value satisfying all the equations in the problem. If some assignment is inconsistent with the equations, the system backtracks, looking for alternative assignments, and firing more propagation.

Besides the expressiveness of the discrete approach, another advantage of representing fuzzy models through finite domains is that some of the existing techniques and algorithms of distributed constraint programming can be borrowed. In this context, the knowledge of the problem, i.e. the constraints that represent it, is typically spread among a number of agents. We present an extension of the Ciao Prolog language that implements further work on the Asynchronous Backtracking algorithm (ABT) to execute CLP($\mathcal{FD}$) programs in a distributed fashion, and apply it to the case of collaborative fuzzy agents systems. ABT lacks several features which are desirable for a complete collaborative agents system. It only considers the case of just one variable per agent, does not include distributed propagation (only labeling), and provides no means to detect termination. In our approach we have extended ABT with these features. To support distributed propagation a minimal, dynamic spanning tree is built using the Dijkstra-Scholten algorithm that covers all the agents and reduces the number of messages exchanged. Termination of propagation or labeling is detected by means of a Chandy-Lamport algorithm. This allows to combine both phases until a problem can be declared as solved or failed.

In this work, we provide the formal framework for our discrete Fuzzy Prolog, we describe the details of its implementation using CLP($\mathcal{FD}$) and we take advantage of this implementation to solve complex distributed fuzzy problems in a collaborative agents system.

References

[GMV04] S. Guadarrama, S. Muñoz, and C. Vaucheret. Fuzzy prolog: A new approach using soft constraints propagation. *Fuzzy Sets and Systems, FSS,* 144(1):127–150, 2004. ISSN 0165-0114.

[Van89] P. Van Hentenryck. *Constraint Satisfaction in Logic Programming.* MIT Press, Cambridge, MA, 1989.

Design a High-Level Language for Large Network Security Management

Jangha Kim[1], Byungwook Song[1], Kanghee Lee[1], Sangwook Kim[1], Daesik Choi[2], and Jungtaek Seo[2]

[1] Department of Computer Science, Kyungpook National University
1370 Sankyuk-dong Buk-gu, Daegu, 702-701, Korea
{jhkim, bwsong, khlee, swkim}@cs.knu.ac.kr
[2] National Security Research Institute
161 Gajeong-dong Yuseong-gu, Daejeon, 305-350, Korea
{dschoi, seojt}@etri.re.kr

Abstract. A common breach of network security is the class of attacks called Worm-virus. This paper proposes a language called Triton whose goal is to efficiently and effectively safeguard large network security for management system. We represent abstract language that supports various functions. This high-level language abstracts the control behavior of the network nodes, which has various setting-up methodology. We shall describe the feature of Triton language and give some example of the preliminary Triton implementation on the test-bed.

1 Introduction

Malicious codes and worms comprise the largest portion of the loss caused the security problem in the Internet. Small worms such as the "Blaster" spread quickly through the enormous network. It causes the network to lock down within an hour or so[1]. The situation worsens before it can be monitored and notified by the supervisor. Since the network is not available, it becomes hard to serve a node with an order. It is for this reason that the abstract ACL policy language and hierarchy security management system are necessary in delivering security information. The Triton is an abstract high-level language that manages the lower node and the network node through the administrator of the Domain.

2 Triton Language

It is necessary to describe the heterogeneous and configurative information of various network nodes. This approach provides the highest Domain with a methodology to manage the large-scale network. A high-level Triton offers various political functions that manage lower network nodes. This mechanism provides five perspectives of a description:

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 7–8, 2004.
© Springer-Verlag Berlin Heidelberg 2004

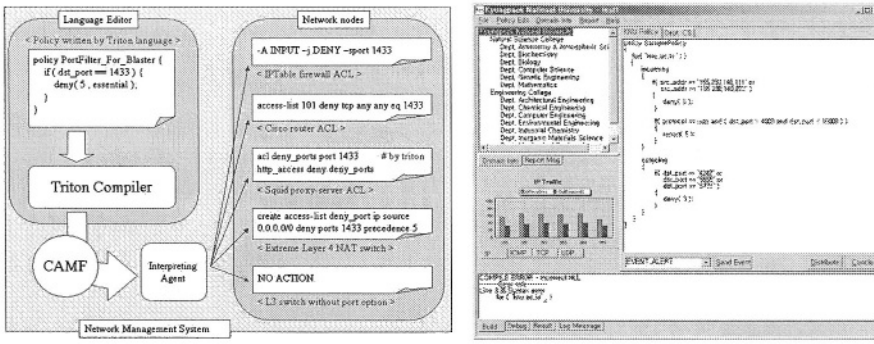


Fig. 1. An abstract policy for network ACL and an administrator interface

- Abstract description for the configuration of a lower node
- Mechanism of policy compliance
- Grouping
- Event for a polymorphous policy
- Semi-structured communication form

Figure 1 shows the process of converting the abstract policy into a configuration of the lower node. The editor delivers a semi-structured ACL delivery form, in an intermediate form, to the Interpreter that has information on the lower nodes. The Interpreter agent generates the ACL commands according to the each feature of the lower nodes. We implement the compiler that converts the high-level language to the intermediate form. For code generation, use the Win32 versions of the MKS LEX & YACC and the algorithm of tree traversals.

3 Conclusion

The Triton language is independent of specific network devices and is familiar with the existing programming language of the large-scale network manager. This paper proposes a language that is analogous to the C language and describes network security ACL policies. The Triton is able to control the lower nodes using the CAMF in intermediate form. It is different from the other policy languages. The policy manager has summarized and abstracted the information on large-scale networks to set up a coherent ACL policy. Additionally, if the Domain server accommodates a policy delivered from the other trustable Domains, the lower Domain then, is improving in security. This gives a tried manager the important function of managing the security of the large-scale network.

References

1. Mohit Lad, Xiaoliang Zhao, Beichuan Zhang, Dan Massey and Lixia Zhang : "An Analysis of BGP Update Burst during Slammer Attack," IWDC, Calcutta, 2003

A Context-Based Awareness Mechanism for Mobile Cooperative Users

Manuele Kirsch-Pinheiro, Marlène Villanova-Oliver, Jérôme Gensel, and Hervé Martin

Laboratoire LSR – IMAG, BP 72 – 38402 Saint Martin d'Hères Cedex, France
{kirsch, villanov, gensel, martin}@imag.fr

Abstract. This paper presents a context-based awareness mechanism designed for users who access web-based collaborative systems using mobile devices. The limited capabilities of such devices (reduced display size, limited battery and memory capacity...) constitute intrinsic constraints which make even more tenuous the issue of delivering relevant information to collaborative mobile users. We propose here an object-based representation of the notion of context and an awareness mechanism which filters the available information according to both the physical (user's location and device) and organizational (knowledge relative to the collaborative process) contexts of the user.

Keywords: Awareness, context-aware computing, mobile computing, collaborative systems.

1 Motivations

This work aims at providing an awareness support for Web-based collaborative systems, particularly those supporting asynchronous work. These systems can be accessed by web-enabled mobile devices (PDAs, cellular phones...), and consequently exposed to the problems their use may cause [2]. The information overload problem addressed by awareness mechanisms [1] [4] embedded in collaborative systems, is made more accurate when using mobile devices due to their physical constraints (see [2]). For delivering some relevant information to mobile users involved in a collaborative task, one has to take into account not only their physical context (information about their location and the device they use), but also their organizational context (knowledge relative to the collaborative process). Existing works concerning awareness support do *not* integrate these two aspects of the user's context. We propose here a representation of the notion of context which includes both the physical and the organizational views of the user's context and we propose an awareness mechanism which exploits this representation for filtering adequately the information delivered to users.

2 A Context-Based Awareness Mechanism

We have designed an object-based representation of the notion of context [3] as a UML schema in which classes represent both the user's physical context (classes

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 9–11, 2004.
© Springer-Verlag Berlin Heidelberg 2004

location, device, application) and the user's organizational context (classes *group, role, member, calendar, activity, shared object* and *process*). The context itself is represented by a class *context description,* which is a composition of these concepts. This UML schema has been implemented as a knowledge base using the java API of the object-based knowledge representation system AROM. This allows us to describe the context of a mobile user accessing the collaborative system. This awareness mechanism is built using a framework for awareness support called BW [4], which adopts an event-based model. In this model, all the awareness information delivered to the user is supported by events. Events are defined by the system developer and may contain any useful information about a specific topic related to the collaborative process. This process intends to determine, using the context representation, which events are relevant in the user's current context. The filtering process performed by the awareness mechanism is based on the concept of general profiles. General profiles represent the preferences and the constraints the system should satisfy for each context element (a member, a role, a device...). For instance, profiles of group members and roles describe the preferences of users performing some roles the awareness mechanism has to take into account. Thus, these profiles define the event types that should be delivered to the user and a priority order for them. A profile can be associated with a time interval which indicates when events must be sent. It can also be associated with some context conditions which describe which events must be sent. Profiles are classes of the knowledge base and are associated with context description. This association reflects that (i) events are produced in a certain context, (ii) each profile has a context in which it can be applied, and (iii) once a mobile user accesses the system, she/he is doing so through a specific context. The filtering process is performed in two steps. First, when a collaborative user logs, all the profiles which correspond to this user in the context of the application currently executed are selected. Then, the selected profiles apply. The first step is performed by comparing the content of the context description instances of both, the active user and the profile: if the context description instance of the profile has the same content or is a subset of the current user's context description instance, then this profile is selected. In the second step, the awareness mechanism compares the conditions associated with the selected profiles to the information carried by the available events. Among all events, the awareness mechanism selects only those which match these conditions. This matching is achieved as follows: for each selected profile, among the available events, those whose type corresponds to a type indicated in the profile are selected. If the profile is also labeled by a time interval, the algorithm restrains the selected events to those which have occurred (or should occur) in this interval. Then, it applies the context conditions, by checking, for each selected event, if its context description satisfies these conditions. If the event satisfies all conditions, it will be delivered to the user. Otherwise, the mechanism will discard it. At the end of this process, the user will receive a limited set of events, considered as relevant for her/his current context.

3 Conclusions

This paper has presented an awareness mechanism which uses an object-based context representation in order to perform a context-based filtering process. The adopted context representation was implemented using the AROM system (see [3]),

and the filtering process was implemented using the BW framework [4], We intend to improve the filtering process by refining the definition of the *general profiles,* as well as the definition of the *subset* relation.

References

1. Espinosa, A., Cadiz, J., Rico-Gutierrez, L., Kraut, R., Scherlis, W., Lautenbacher, G.: Coming to the wrong decision quickly: why awareness tools must be matched with appropriate tasks. CHI Letters, CHI 2000, vol. 2, n° 1. ACM Press (2000) 392-399.
2. Jing, J., Helal, A., Elmagarmid, A.: Client-server computing in mobile environment. ACM Computing Surveys, vol. 31, n° 2. ACM Press (1999) 117-157
3. Kirsch-Pinheiro, M., Gensel, J., Martin, H.: Representing Context for an Adaptative Awareness Mechanism. To appear in X International Workshop on Groupware (CRIWG'04), San Carlos, Costa Rica, 05-09 Sept. 2004. Springer-Verlag (2004)
4. Kirsch-Pinheiro, M., Lima, J.V., Borges, M.R.S.: Framework for Awareness Support in Groupware Systems. Computers in Industry, vol. 52, n° 3. Elsevier (2003) 47-57

Access Control-Based Distribution of Shared Documents

Sonia Mendoza[1], Alberto L. Morán[1,2], Dominique Decouchant[1],
Ana María Martínez Enríquez[3], and Jesus Favela[4]

[1] Laboratoire "Logiciels Systèmes, Réseaux", Grenoble, France
{Sonia.Mendoza, Alberto.Moran, Dominique.Decouchant}@imag.fr
[2] Facultad de Ciencias, UABC, Ensenada, B.C., México
[3] Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, D.F., México,
ammartin@mail.cinvestav.mx
[4] Departamento de Ciencias de la Computación, CICESE, Ensenada, B.C., México,
favela@cicese.mx

Abstract. The PIÑAS platform provides an authoring group with support to collaboratively and consistently produce shared Web documents. Such documents may include costly multimedia resources, whose management raises important issues due to the constraints imposed by Web technology. This poster presents an approach for distributing shared Web documents to the authoring group's sites, taking into consideration current organization of the concerned sites, access rights granted to the co-authors and storage device capabilities.

1 Introduction

Data distribution in the CSCW field concerns the way in which the shared data of a groupware application is distributed to the involved sites. Distribution strategies mainly differ in three aspects: 1) the number of data replicas available in an application at a given moment; 2) the data representation at the local site; and 3) the mobility of data around the network. As these aspects can influence the working style of an authoring group, a groupware platform should provide flexible enough distribution support to allow the developer to implement different strategies. Previous work, e.g. [1], identifies data distribution as an important issue in groupware design and development. However, only few platforms, e.g. [2], take flexibility into account for their distribution support. Moreover, they are mainly intended to support synchronous work and do not address important issues arising from the need to support collaborative work on the Web, such as the current organization of the concerned sites, co-author's access rights and storage device capabilities.

2 Distribution of Shared Web Documents

The proposed approach is based on the multi-site author principle [3], which relates an author with several working and storage sites possibly located in different places around the world. Based on this principle, the PIÑAS platform [4] supports nomadic/disconnected work and provide high availability of Web documents. Moreover,

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 12–13, 2004.
© Springer-Verlag Berlin Heidelberg 2004

associations among working and storage sites allow authors to automatically access documents from close sites, using more reliable environment (e.g. LAN).

Distribution is carried out regarding the current organization of working and storage sites, author access rights and storage device capabilities. At creation time, a document is private and belongs to its creator. Therefore, the document is distributed to the creator sites for low-latency access and persistent storage. When the document owner decides to share it with others, he creates an authoring group, fragments the document and finally grants roles to co-authors on each fragment and resource. Afterwards, the document redistribution to the co-author's sites can take place. When a co-author has access rights on a fragment or resource, a permanent replica can be saved in his storage sites. Also, a temporal replica can be created in his current working site to allow rapid feedback at the user interface. Otherwise, permanent and temporal proxies are respectively kept in his storage and working sites in order to save storage space. On the other hand, if a co-author has access rights on a costly resource, but he can not access it due to device constraints, a temporal proxy is created in his current working site, even if a permanent replica is maintained in his storage sites.

3 Conclusion and Future Work

By providing developers with an autonomous service for distributing documents, they can build groupware applications that support different distribution strategies. Also, an authoring group can rely on an infrastructure that automatically distributes their documents to make them available during a working session and to persistently store them in their Web sites. In addition to these features, our infrastructure ensures that documents are updated and kept consistent even in unreliable environments such as the Internet. However, there are still some open issues. We intend to provide flexible support for managing consistency of coauthored documents. We also aim at supporting automatic detection of neighboring sites. Thus, a working site can get documents from the closest storage site available. Other important issue concerns the adaptive support for propagating document state changes.

References

1. P. Dourish, Using Metalevel Techniques in a Flexible Toolkit for CSCW applications, ACM Transactions on Computer-Human Interaction, 5(2):109-155, June 1998.
2. S. Lukosch, Adaptive and Transparent Data Distribution Support for Synchronous Groupware, 8th International Workshop on Groupware, Springer-Verlag, LNCS 2440, pp. 255-274, La Serena (Chile), September 2002.
3. S. Mendoza, D. Decouchant, A.M. Martínez and A.L. Morán, Adaptive Resource Management in the PIÑAS Web Cooperative Authoring, 2nd International Atlantic Web Intelligence Conference, Springer-Verlag, LNAI 3034, pp. 33-43, Cancún (Mexico), May 2004.
4. A.L. Morán, D. Decouchant, J. Favela, A.M. Martínez, B. Gonzalez and S. Mendoza, PIÑAS: Supporting a Community of Co-Authors on the Web, 4th International Conference on Distributed Communities on the Web, Springer Verlag, LNCS 2468, pp. 114-125, Sydney (Australia), April 2002.

Service-Oriented Architecture of TeleCARE

Victor Guevara-Masis, Hamideh Afsarmanesh, and Louis O. Hertzberger

Faculty of Science, University of Amsterdam
Kruislaan 403, 1098 SJ, Amsterdam, The Netherlands
{vguevara, hamideh, bob}@science.uva.nl
http://www.science.uva.nl/CO-IM/

Abstract Developers of advanced services for tele-assistance need a way to reuse the functionality of resources, being hardware devices or software applications, at their disposal. The reusability of resources becomes even more important, when considering the continuous development of facilities (e.g. the emerging domotics) and services (e.g. the panic button) in this domain. The overall goal of the TeleCARE project is the provision of a configurable infrastructure supporting collaborative networked environments for tele-assistance applied to elderly care. It designs an open flexible horizontal infrastructure based on a Service-Oriented Architecture (SOA) and a multi-agent system.

1 Introduction

The overall goal of the IST 5FP TeleCARE project is the design and development of a configurable framework solution for tele-assistance to assist the elderly [1]. Seen as a Collaborative Network (CN) environment, TeleCARE focuses on the creation and maintenance of an integrated care system, involving partners from different areas and organizations in order to provide services and corresponding products (value-added applications), that rely on the resources of other autonomous partners. Most CNs infrastructures do not provide complete support to build services based on incompatible frameworks. To solve this problem, the tendency is to build loosely coupled and dynamically bound components based on services. In [5] some insights about how to deploy services in CNs are explored and the importance of describing the interfaces to the applications in order to eradicate duplicate processes is addressed.

2 Service-Orientation of TeleCARE for Collaborative Tele-assistance

The service-oriented architecture (SOA) is derived from self-contained and modular components. These components are accessible within the networked environment and provide a set of functionalities (or operations) to different users [2]. Using a SOA, the TeleCARE platform attempts to achieve proper interoperability by allowing "pluggable" developments in agent-based applications.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 14–16, 2004.
© Springer-Verlag Berlin Heidelberg 2004

The TeleCARE resource model definition is largely based on the service-oriented paradigm, which allows the publishing of functionality descriptions of services in a registry that can be used to search, select and combine individual services. The concept has been extended in TeleCARE to provide agent-based access to the descriptions, as well as defining access rights to secure resources. TeleCARE resources are either *Devices* or *Vertical Services,* and their functionality are described using the Universal Plug and Play Device Architecture [3] standard and Web Service Definition Language (WSDL) [4] respectively.

The TeleCARE resource model (see Figure 1) supports all involved actors, namely the resource provider, resource broker and resource requester, with performing their service related tasks: description, advertisement, discovery and invocation.

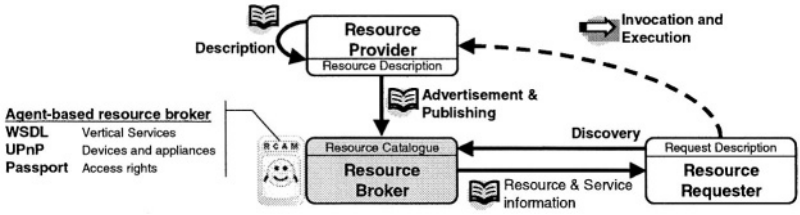


Fig. 1. TeleCARE resource model

The Resource CAtalogue Management (RCAM) is an automatic resource broker and a key component of TeleCARE infrastructure. It is an agent-based repository of all resource descriptions for devices and vertical services. Through RCAM, resource providers advertise the functionality of their resources, while resource requesters (being software agents or humans) search for appropriate resources and their internal services, and obtain the necessary information to invoke them. The resource discovery is performed through two different interfaces, a Web-access for human access and an agent communication protocols (native and FIPA ACL) for computer-based queries. Furthermore, RCAM also manages access rights for resources, supporting both security and privacy of end users that are crucial aspects in the tele-assistance domain.

From a developer's point of view, RCAM allows resource composition and supports significant on-the-fly integration of new tele-assistance applications. RCAM facilitates the gradual addition of new software and appliances to TeleCARE environment, supported by a set of standards and by an agent-based infrastructure.

References

1. Camarinha-Matos L. and Afsarmanesh H., "TeleCARE: Collaborative virtual elderly care support communities," The Journal on Information Technology in Healthcare, vol. 2:2, pp. 73-86, 2004.
2. Tsalgatidou A. and Pilioura T., "An Overview of Standards and Related Technology in Web Services," International Journal of Distributed and Parallel Databases, vol. 12, pp. 135-162, 2002.

3. UPnP Forum, "The Universal Plug and Play Architecture," http://www.upnp.org/download/UPnPDA10_20000613.htm, 2000.
4. W3C Consortium, "Web Services Description Language (WSDL)," http://www.w3.org/TR/wsdl, 2001.
5. Yang J., Heuvel W.J.V.d., and Papazoglou M.P., "Service Deployment for Virtual Enterprises," Proceedings of the Workshop on Information Technology for Virtual Enterprises, Queensland, Australia, IEEE Computer Society, 2001.

Feedback Partitioning and Decomposition-Oriented Analysis of Cyclic Workflow Graphs

Yongsun Choi[1] and J. Leon Zhao[2]

[1] Dept. of Systems Management & Engineering, Inje University, Kimhae, Korea
yschoi@inje.ac.kr
[2] Dept. of MIS, University of Arizona, Tucson AZ 85721, USA
lzhao@bpa.arizona.edu

Abstract. This paper introduces a novel method of verifying a cyclic workflow model by partitioning it into acyclic subgraphs. The iterative classification of nodes according to the feedback structure and derivation of acyclic subgraphs are illustrated. We also introduce the concepts of two-phased verification of control flows and decomposition-oriented analysis, which is generally easier to comprehend and verify individually than the whole workflow model.

1 Introduction

Workflow analysis and design is a fundamental task in business process management and becoming more important with the recent surge of e-business process automation efforts [1]. Although hierarchical decomposition of a complex workflow process is a useful step in workflow analysis and design, until now there has been no reported work of its automated support in the literature. This paper presents a decomposition-oriented method of verifying a cyclic workflow model by partitioning it into a set of acyclic subgraphs based on its feedback structure.

2 Partitioning into Acyclic Flows and Decomposed Analysis

Cycles in workflow models, needed for purposes of rework and information feedback, make workflow analysis more complex. Our method identifies and partitions each acyclic flow from the given cyclic workflow model. At the first iteration of partition, our method decomposes the *main flow*. The main flow is defined as a maximally connected subgraph from the *Start node* to the *End node,* spanned by all nodes reachable to the *End node* without multiple visits to any nodes contained. If the original workflow model has nested feedback structures, our method proceeds on partitioning further to the remaining subgraph(s) that contain(s) cyclic structures. Otherwise, our method identifies and partitions each additional acyclic *feedback flow* from the remaining subgraph(s). The complexity of partitioning step can be estimated as $O(qr|\boldsymbol{T}|) < O(|\boldsymbol{N}|\cdot|\boldsymbol{T}|)$, with q as the maximum degree of feedback, r as the average number of

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 17–18, 2004.
© Springer-Verlag Berlin Heidelberg 2004

subgraphs that need to be further partitioned in each iteration, and $|N|$ and $|T|$ as the number of nodes and arcs in the workflow graph, respectively. Figure 1 shows a workflow graph with nested cycles and the normalized graph with five acyclic subflows after partitioning.

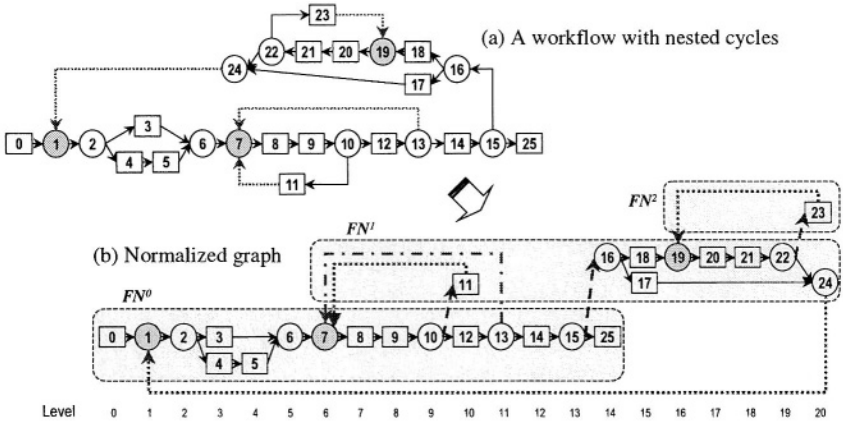


Fig. 1. Normalization of a workflow graph with nested cycles

Our method can detect certain types of structural conflicts, during partition, caused by inappropriate composition of feedback flows. And then, it allows a cyclic workflow model to be analyzed further with several smaller subflows, which are all acyclic. For instance, our method allows the extended graph reduction technique [2] to handle cyclic workflow models and reduces the complexity from $O((|N|+|T|)^2 \cdot |N|^2)$ [1, 2] to $O((|N|+|T|)^2 \cdot |N|^2 / s^3)$ when the original model is partitioned into s acyclic graphs.

3 Concluding Remarks

Hierarchical decomposition of a workflow model by means of feedback partitioning is unique and has several distinctive features useful for workflow analysis and design. Each acyclic flow is a candidate for a sub-workflow, the knowledge of which is helpful for managing large-scale workflows. The two-phased verification and the decomposition-oriented analysis with each acyclic flow can make workflow analysis easier and more efficient.

References

1. Aalst, W.M.P. van der: The Application of Petri Nets to Workflow Management. J. Circuits, Systems and Computers **8** (1998) 21-66
2. Lin, H., Zhao, Z., Li, H., Chen, Z.: A novel graph reduction algorithm to identify structural conflicts. Proc. 35th Hawaii Int. Conf. On Sys. Sci. (HICSS'02) **9** (2002) 289

Towards Implementing Consistency in OMG MOF-Based Repository Systems

Ilia Petrov, Stefan Jablonski, and Marc Holze

Chair for Database Systems, Department of Computer Science, University of Erlangen-Nürnberg, Martensstrasse 3, Erlangen, D-91058, Germany
{Petrov, Jablonski, Holze}@informatik.uni-erlangen.de
http://www6.informatik.uni-erlangen.de/

Abstract Repository systems handle the management of metadata and meta-models. They act as data store with a distinguishing characteristic - it provides a custom-defined and adaptable system catalogue. Preserving the consistency of the repository data is a major challenge. In this paper we propose a novel algorithm for enforcing structural consistency in MOF-based repositories, which is implemented in iRM/RMS - a prototypical OMG MOF-based repository system.

1 Introduction

Repository systems are "shared databases about engineered artifacts" [2]. They facilitate integration among various tools and applications, and are therefore central to an enterprise. Loosely speaking repository systems resemble data stores with a new and distinguishing feature – a customizable system catalogue. Repository systems exhibit an architecture comprising several layers of metadata [3], e.g. repository application's instance data (M_0), repository application's model (M_1), meta-model (M_2) and meta-meta-model (M_3). In comparison to database systems repository systems contain an additional metadata layer, M_3, allowing for a custom definable and extensible system catalogue (M_2). Preserving consistency between the different layers is a major challenge specific to repository systems.

2 Repository Consistency

This paper discusses various aspects of repository consistency. Its major contribution is the proposed approach to enforcing consistency in OMG MOF-based repositories. Here we also discuss its implementation – the iRM/RMS module of the iRM project [1]. In addition, this paper dwells on some run-time aspects of consistency, which are not touched upon in the MOF specification version 1.4 [3] such as repository transactions. Consistency in repository systems has several aspects: (a) Operational consistency – relates to how repository applications interact with the RMS. Operational consistency is based on the notion of repository transactions. Two sub-aspects need to be considered: concurrent multi-client access; and cooperative atomicity [4], i.e. atomic, structurally consistent repository updates spanning multiple

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 19–20, 2004.
© Springer-Verlag Berlin Heidelberg 2004

operations, (b) Metadata integrity – comprises the notions of well-formedness and structural integrity. Well-formedness ensures the syntactical correctness of the model definitions within a meta-layer. It is guaranteed by the RMS API or an import utility. Structural integrity relates the conformance of objects on one level to type definitions on the adjacent higher meta-level. Structural integrity results from the strict enforcement of the type-instance relationship across meta-layers. Various metadata standards [24, 3] use the term consistency, which in our view is too broad.

The lack of structural integrity would mean that modifications through repository applications may render the repository data inconsistent. This will violate the basic type-instance relationship among adjacent meta-layers. Models M_{n-1} may then be created or modified inconsistently with their meta-models M_n.

The concept of repository transactions is central to enforcing structural consistency. iRM/RMS repository management system is designed to handle concurrent multi-client operations. To ensure isolation the iRM repository API provides a locking mechanism based on long exclusive X-locks and short shared S-locks. The employed locking mechanism is 2PL compatible. It extends the traditional locking [14, 29, 33], to accommodate the multiple meta-levels (M_0 .. M_2).

In iRM/RMS well-formedness, of this type, is enforced by the Metadata Manager alone. Some examples are: (i) enforce the properties of generalization hierarchies and containment hierarchies. For example, no class or package can be defined as super types of themselves; no name conflicts with super-type elements are allowed; (ii) check whether containment rules are satisfied; (iii) check whether miscellaneous properties are properly set.

The algorithm for structural integrity we propose handles the modifications to all types of MOF model elements. Some of the most important cases are: binary associations and association ends; multiplicities of association ends and attributes; attributes and attribute types; classes; Containment hierarchy and Generalization hierarchy; packages; object- meta-object relationship.

3 Conclusions and Future Work

In this paper we propose a novel approach to enforcing structural consistency in OMG MOF based repository systems. It ensures the structural consistency of the repository data after any modification operation. The proposed approach is tightly coupled with the concept of repository transactions. Extending the employed locking mechanism to increase the concurrency is a major challenge.

References

[1] Petrov, I., Jablonski, S.: An OMG MOF based Repository System with Querying Capability - the iRM Project. To appear In Proceedings of iiWAS 2004.
[2] Bernstein, P., Dayal, U.: An overview of repository technology. In Proc. of VLDB 1994.
[3] Object Management Group: Meta Object Facility Specification Version 1.4.
[4] Ozsu, T.: Transaction Models and transaction management in Object-oriented database management systems. In Advances in Object-oriented Data-base Systems, 1994.

Privacy-Preserving ID-Based Service Provision in Ubiquitous Computing Environments

Keon Myung Lee and Sang Ho Lee

School of Electric and Computer Engineering, Chungbuk National University,
and Advanced Information Technology Research Center(AITrc)*, Korea
kmlee@cbnu.ac.kr

Privacy preservation is vital because a lot of privacy-sensitive information can be collected and distributed in the ubiquitous computing environment. [1] The anonymity-based approach is one of well-known ones for privacy preservation, which allows users to use pseudonyms when they join in a service area of the ubiquitous computing environment in order to hide the users' identity (ID). [2,3] This paper proposes an architecture to enable ID-based services in the anonymity-based platform. The architecture employs the so-called *white page agent* which maintains the current pseudonyms of users and thus allows users to get the pseudonyms of other users. Although the white page agent maintains the pseudonyms of users, it is enforced not to disclose the association of users' real ID with their pseudonyms by adopting security protocols.

The proposed architecture comprises of several agents: a white page agent, a set of zone agents taking charge of physical zones, a set of user agents providing user-interfaces, a set of user directory agents updating the pseudonym records on behalf of users, and a set of application agents corresponding to applications. Users register their current pseudonyms to the white page agent when they change their pseudonym. Both users and applications can get the pseudonyms of correspondents from the agent once they have proper key information. The database of the white page agent contains the records made of the pair (real ID of user i, a list of user i's current pseudonyms encrypted with different encryption keys). In order to allow a user to update the encrypted pseudonym field of her record in the white page agent without revealing her identity, the architecture asks the i's user directory agent to register new pseudonym on behalf of user i in the following way: The i's user agent sends to the white page agent the pseudonym update request encrypted in a secret key shared with its directory agent. Then the white page agent broadcasts the message to all directory agent without figuring out its contents. Upon the broadcasted message, multiple directory agents including the i's directory agent come to update their records residing in the white page agent. To hide who actually updates her pseudonym, the directory agents other than the i's directory agent also update their records by inserting meaningless noise in the records.

When a user (or an application) i wants to communicate with a user j, i needs to know the current pseudonym of j. At the moment, user i is assumed to know the real ID of user j. The i's user agent asks the white page agent about the pseudonym of user j. After obtaining the pseudonym, user i communicates

* This work was supported by Korea Science and Engineering Foundation via AITrc.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 21–22, 2004.
© Springer-Verlag Berlin Heidelberg 2004

with user j using the pseudonym. To avoid the eavesdropping, the pseudonym request message is encrypted using the public key of the white page agent. The request message contains a secret key as well as the counterpart's ID. The secret key is later used to encrypt the response message which consists of a collection of pairs (real ID, a list of encrypted pseudonyms). In the list, each encrypted pseudonym element is the pair (friend key f_k, pseudonym) encrypted using the friend key f_k. If user i is-a friend of user j, i must have a friend key for j's pseudonym. Therefore, the i's user agent can recover the pseudonym for user j from the response message.

In the anonymity-based system, users adopt pseudonyms to hide their real IDs and use them as their temporary ID. Therefore, pseudonyms should be unique across the networks. The architecture employs a naming scheme to associate pseudonyms with the locations at which the users with the pseudonyms are placed. All agents but user agents running on users' mobile devices are connected to the Internet and thus they have corresponding IP(Internet Protocol) addresses. In order to provide the IP address resolution for pseudonyms, the white page agent maintains a database which keeps the mapping information between the pseudonyms (exactly to say, zone IDs) and IP addresses, the proximity information among zone IDs, and addressing information for entities such as user directory agents, application agents, etc. When a user or an application wants to communicate with a user on the Internet, she first gets the addressing information for the user from the white page agent. The zone agents to which they belongs have IP addresses and the user agents are distinguished by their temporary agent IDs within their zone.

Only friends of a user are allowed to get her pseudonym. When a user or an application i establishes a friendship with user j, the following protocol is exercised: The i's user agent sends an encrypted friendship request message to the white page agent which broadcasts it to all user directory agent. Then j's user directory agents gets the information about i from the i's user directory agent, and asks the user j whether she accepts. If she accepts it, the j's user directory agent creates a friend key for i, informs i of the friend key through the j's user directory agent, and then updates j's record of the white page agent so as to take care of the friend key.

The proposed architecture enables ID-based services such as peer-to-peer communication, buddy service, safety alert service, in a privacy-preserving manner in the ubiquitous computing environment. As the further studies, there remain how to incorporate the delicate access control into the architecture and how to reduce communication overhead among agents.

References

[1] G. Myles, A. Friday and N. Davies. Preserving Privacy in Environments with Location-Based Applications. *IEEE Pervasive Computing* 2(1). (2003). 56-64.

[2] A. R. Beresford and F. Stajano. Location Privacy in Pervasive Computing. *IEEE Pervasive Computing* 2(1). (2002). 46-55.

[3] X. Jiang and J. Landay. Modeling Privacy Control in Context-aware Systems. *IEEE Pervasive* 1(3). (2002).

Agents for Decentralised Process Design – Extended Abstract

Iain Stalker[1], Nikolay Mehandjiev[1], Georg Weichhart[2] and Kurt Fessl[2]

[1] Department of Computation, UMIST, PO Box 88, Manchester M60 1QD, UK
[2] Profactor Produktionsforschungs GmbH, Im Stadtgut A2, A-4407 Steyr/Gleink

Keywords: Agents, Workflow Management, Decentralisation, Process Languages.

To improve competitiveness in complex and dynamic markets, many businesses seek agile partnerships. As a result, the paradigm of the Virtual Enterprise (VE) — a temporary network of cooperating enterprises in which there is a distribution of control, or *decentralisation* — has attracted much attention in recent years [1]. Successful collaboration demands a tight, but flexible integration of individual business processes across the VE, and a key element to achieving this is the use of workflow management systems to control these processes. Traditionally, workflow systems are based on two alternative approaches: *model-driven workflow* (e.g. [2]); and *situated dynamic planning and negotiation between intelligent software agents* (e.g. [3]). Each has strengths and drawbacks and some researchers, e.g. [4], have tried to get the "best of both worlds" by combining the two paradigms. However, these efforts largely take *ad hoc* decisions regarding the places and roles of agents in workflow management and construction. The approach reported here differs from these in the use of a formal model of the cognitive aspects of our domain constructed *a priori. This allows us to determine the place of agents in a systematic fashion.*

To fix in mind the requirements of decentralised workflow management systems, we introduce a very simple scenario. The desired outcome of the global workflow of our VE is a working bicycle: this is our top-level goal. Consider, an organisation assembles the components but does not manufacture the necessary parts: it needs to work with manufacturers of appropriate components. Moreover, the assembly of the components must follow particular sequence and presupposes that the components are available as required. Thus, negotiation among (potential) partners is needed to integrate the component workflows within the global VE workflow. This view is somewhat simplistic, for example, we have not considered payment terms, logistics and so forth. Yet, even this simple usage scenario intimates the need for complex interactions between design and control processes, within different viewpoints and at different levels of abstraction. Our main interest lies in the *strategic level,* which addresses interorganisational aspects and includes the design of the global workflow; in particular, how agents may be brought fruitfully to bear upon the activities of this level.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 23–25, 2004.
© Springer-Verlag Berlin Heidelberg 2004

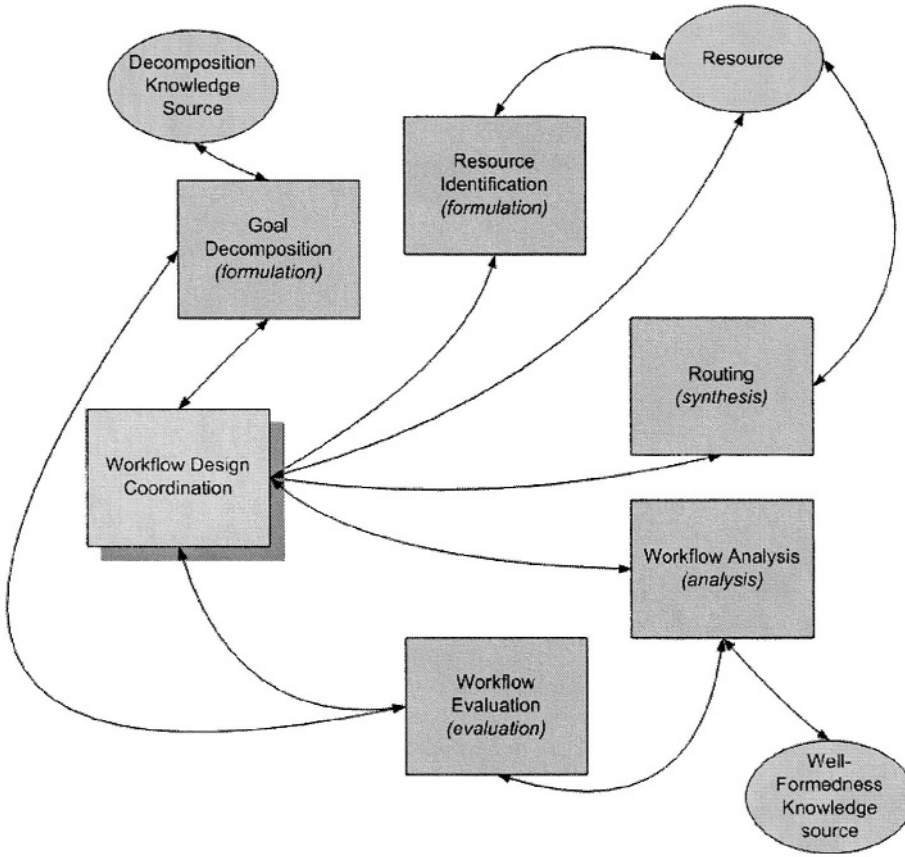


Fig. 1. Functional Elements of Workflow Design. The rectangles denote functional blocks; ellipses denote resources or knowledge sources, as indicated; and the arrows indicate exchange of information.

We suggest the *function-behaviour-structure* framework [5] as an organising schema for the strategic level of workflow design. We use it to conceptualise the design process through which workflow structures are devised; and to help determine the role of software agents. We apply the FBS framework once we have established the desired outcome of the global workflow, using it systematically to identify where agents might be applied. In particular, we clarify the decision space for workflow design and construction, distinguishing a number of categories of knowledge and the transformations between these categories. We use this separation of domain knowledge from the computational processes which operate upon it to highlight where agents may be applied and identify their contributions to the design and construction of decentralised workflows. We suggest some areas for integration with model-driven aspects.

The design of a workflow is intimately related to the formation of a core team of partners who will work together on a project. We distinguish two modes of interaction between the team formation and workflow design: team formation precedes the

workflow design; and team formation is interwoven with the workflow design. In the first case we treat team formation as an independent, preliminary application of the FBS framework.

Applying the FBS framework to the strategic level allows us to identify a number of functional blocks and their interactions; illustrated in Fig. 1. We use the functional blocks distinguished to identify potential opportunities for the application of agents. We are guided by the definition of a (software) agent as a computer system capable of flexible, autonomous action which exhibits characteristics of *proactivity, reactivity* and *social ability* [6]. For example, *Goal Decomposition* is a prime candidate for the use of a personal assistant agent to facilitate a human designer; whereas the use of a model-driven non-agent approach may be more appropriate for *Workflow Analysis.*

We have used this approach to propose necessary elements of a generic framework which can be specialised to a particular application according to end-user requirements. Indeed, the next step is to apply the proposed elements to a set of concrete usage scenarios. In addition to the immediate further work of identifying these scenarios and eliciting the appropriate user requirements, we identify a number of interesting, longer-term research issues attendant upon the development of an application specific implementation: for example, modelling and integrating the worldviews of potential end users.

References

1. Camarinha-Matos, L.M, Afsarmanesh, H.: Virtual Enterprise Modeling and Support Infrastructures: Applying Multi-agent System Approaches. In M. Luck et al. (Eds.): ACAI 2001, LNAI 2086, pp. 335-364.
2. van der Aalst, W., van Hee, K.: *Workflow Management: Models, Methods, and Systems,* Cooperative Information Systems, The MIT Press, 2002.
3. Berry, P M and Drabble, B, SWIM: An AI-based System for Workflow Enabled Reactive Control. In proceedings of the IJCAI Workshop on Workflow and Process Management held as part of IJCAI-99, Stockholm, Sweden, August 1999
4. Jennings, N.R., Faratin, P., Norman, T.J., O'Brien, P., Wiegand, M.E., Voudouris, C., Alty, J.L., Miah, T., Mamdani, E.H.: ADEPT: Managing Business Processes using Intelligent Agents. In Proceedings of BCS Expert Systems 96 Conference (Intelligent Systems Integration Programme Track), pages pp. 5-23, Cambridge, UK.
5. Gero, J.S.: Design prototypes: A knowledge representation schema for design, *AI Magazine,* Winter:26-36, 1990.
6. Wooldridge, M. and Jennings, N R. Intelligent agents: theory and practice. *The Knowledge Engineering Review.* **10**(2), 115-152.

Knowledge Signatures for Information Integration

Judi Thomson[1], Andrew Cowell[2], Patrick Paulson[2], Scott Butner[2], and Mark Whiting[2]

[1] Department of Computing & Information Science,
University of Guelph, Guelph, Ontario N1G 2W1, Canada
judi@cis.uoguelph.ca
http://www.cis.uoguelph.ca
[2] Pacific Northwest National Laboratory,
Richland, Washington 99352, USA
{andrew, patrick.paulson, scott.butner, mark.whiting}@pnl.gov
http://www.pnl.gov

Abstract. Information analysis, scientific discovery, web navigation and many other information related activities rely on the collection, analysis, and integration of unstructured, disparate data. In spite of significant research efforts, automatic methods for information integration are still vastly inferior to human capabilities. Tasks which are simple for humans, such as recognizing the underlying similarities between superficially different objects or sorting out the semantics of ambiguous statements, continue to present significant technical challenges for automated processes.

Introduction

This paper presents the concept of *knowledge signatures;* simple constructs that capture the semantics of data within a particular context or domain. Knowledge signatures can be used to:

- Characterize the semantics of data while maintaining a concise, manipulable representation.
- Facilitate quantitative comparison of semantic information, i.e., sorting/ordering operations and the calculation of semantic distance between two objects.
- Provide for a level of data fusion and differencing through a set of basic mathematical operations such as addition and subtraction.

A knowledge signature is a real-valued vector derived from a data item and a domain-specific ontology. Each element of the vector characterizes the strength of the association between the data item and a single concept described by the ontology. The vector representation can be manipulated mathematically with relative ease and vector comparisons can be made between knowledge signatures that are created using the same ontology or set of concepts. We call the vector a knowledge signature because it describes the semantic content of the data item with a numerical representation that is similar to the data signatures obtained from traditional statistical approaches to data characterization [1].

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 26–27, 2004.
© Springer-Verlag Berlin Heidelberg 2004

Knowledge signatures can be used by any number of integration processes to guide the automated integration of information. Generation of knowledge signatures exploit tacit knowledge contained in the ontologies to improve the understanding of data semantics. The signatures also provides a type-independent mechanism for comparing pieces of data and predicting the semantic effects of aggregating two or more data elements.

Procedurally, the creation of a knowledge signature comprises two distinct operations: *observation* and *refinement.* Observation determines the degree of association [2] between data and ontological concepts relying primarily on tacit knowledge and data analysis techniques. Refinement uses the axioms and structure of the domain ontology to discover associations between data and additional concepts, and to adjust the associations discovered by observation. Including only concepts relevant to the current task can reduce the size of the resulting signature significantly.

Our concept of a knowledge signature does not stipulate how observation and refinement are performed. The methods used for observation will often depend on the type of data being analyzed and those for refinement on the nature of the ontology. We are investigating several approaches to observation, including statistics, heuristics, and through human annotation. Approaches to refinement — including logical representations, ontological relationships, probabilistic networks, and graph similarity algorithms — are also being considered.

Individual knowledge signatures for discrete pieces of data are useful as summaries of what the data denotes (e.g., a classification tasks), but the approach is even more powerful when the signatures for several pieces of data are used in an information integration task. For example, because knowledge signatures represent mathematical representations of a data object's semantics, they provide a means to formally represent questions such as: "What happens when these two pieces of data are merged?" and "How similar are these two pieces of data?" Our efforts are focused on providing accurate, repeatable answers to these two questions, even when the data under consideration is of differing types.

Work to date has focused on the development of an overall architectural framework for calculating and manipulating knowledge signatures, and the implementation of a prototype set of observers and refiners based on this architecture. Testing of these prototype implementations, using a subset of data extracted from the Reuters 21578 [3] data set, is currently underway.

References

[1] Wong, P.C., Foote, H., Leung, R., Adams, D., Thomas, J. (2000) "Data Signatures and Visualization of Very Large Datasets". IEEE Computer Graphics and Applications, **20**(2).

[2] Salton, G., Wong, A., Yang, C.S. (1975) "A Vector Space Model for Automatic Indexing". Communications of the ACM, **18**(11).

[3] Lewis, D.D. (1997) "Reuters-21578 Test Collection, Distribution 1.0." http://www.daviddlewis.com/resources/testcollections/reuters21578, January 1997.

Efficient Processing of Huge Ontologies in Logic and Relational Databases

Timo Weithöner, Thorsten Liebig, and Günther Specht

University of Ulm, D-89069 Ulm
{weithoener | liebig | specht}@informatik.uni-ulm.de

Abstract. Today ontologies are heavily used in the sematic web. As they grow in size reasoning systems can't work without secondary storage anymore. Thus database technology is required for storing and processing huge ontologies. In this paper we present an efficient technique for representing and reasoning with ontologies in databases. We also present some benchmarking results in comparison with previous approaches.

1 Mapping Ontologies into Logic Programs

Converting ontologies into description logic programs (DLP; DLP covers most of OWL[1] Lite plus a portion of OWL DL, namely general concept inclusions with disjunction and qualified existential qualification on the left hand side) would allow to use deductive database systems as reasoners. A straightforward "Direct Mapping" approach (DiMA) to convert ontologies into a DLP was suggested in [1]. This approach maps every class or property definition into a rule and every class-instance or instance-property-instance relationship into a fact. The following gives a brief example of some frequently used statements:

DL	DLP
$A \sqsubseteq B$	B(X) :- A(X).

DL	DLP
$i : A$	A(i).

DL	DLP
$B \sqcap C \sqsubseteq A$	A(X) :- B(X), C(X).

DL	DLP
$A \sqsubseteq B \sqcap C$	B(X) :- A(X). C(X) :- A(X).

Obviously, this approach has some conceptual drawbacks:

- First, the concept names cannot be used as query results. For that reason it is impossible to get an answer to the queries like "give me all classes the individual *I* is instance of".
- The mapping typically results in only a few facts per literal. But the number of different rules grows linear with the complexity of the ontology.
- The names of the relations used and the structure of the rules involved vary from ontology to ontology. Consequently precompilation is impossible and query optimization becomes a real issue in this approach.

We overcome these drawbacks with our approach, which we call the Meta Mapping approach.

[1] OWL is the Web Ontology Language, see http://www.w3.org/TR/owl-features/

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 28–29, 2004.
© Springer-Verlag Berlin Heidelberg 2004

2 The Meta Mapping Approach (MeMA)

The "Meta Mapping" from OWL DL into DLPs has an emphasis on low computational complexity and high representational flexibility. Two things have to be done to overcome the limitations mentioned above: First the rules and facts are pushed to a meta level, where names of concepts and properties become arguments of "meta predicates". Second we construct a constant set of rules valid for all ontologies accompanied by a set of fact predicates with constant names. As a result concept and property names can be used as query results as well as inputs and query formulation and optimization is eased.

Asserting instance `i` to class `C` results in instantiating the binary relation `type("i","C")`. In MeMA subclass relationships or any other kind of constructors are converted into facts which state, that the ontology defines such relationships. The following table shows some more examples:

DL	DLP
$A \sqsubseteq B$	isSub("A", "B").
$B \sqcap C \sqsubseteq A$	isSub("I1", "A"). intersectionOf("I1", {"B", "C"}).

DL	DLP
$i : A$	type("i", "A").
$A \sqsubseteq B \sqcap C$	isSub("A", "I2"). intersectionOf("I2", {"B", "C"}).

In order to reflect the underlying semantic of the introduced meta relations, we have to add adequate rules[2]. E.g. `type(I,X) :- type(I,Y), isSub(Y,X).` defines that if an individual `I` is instance of concept `Y` and `Y` is subclass of concept `X`, `I` is also an individual of class `X`. All rules are completely independent of any concrete ontology vocabulary and can thus be used for every ontology. With the combination of the ontology specific facts and the general rule we can easily perform common A- and TBox queries within the Meta Mapping approach.

3 Comparison

When benchmarking both mappings it turned out that in DiMA even a linear growth in relations results in fatal performance during preprocessing while loading the program into the CORAL deductive database. The same operation takes only seconds in the MeMA. Our experiments also observed a linear growth of processing time for class instance and subclass querying with an increasing number of individuals for both approaches. However, only in case of the MeMA better results are reached by logic databases with secondary storage indexing mechanisms. We thus get logarithmic behavior for the MeMA.

In consideration of the above we propose the Meta Mapping because of its significant conceptual advantages, higher expressivity and better performance for storing and evaluation of large scale real world ontologies in logical databases.

References

1. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description Logic Programms: Combining Logic Programms with Description Logic. In: Proceedings of the 12^{th} International World Wide Web Conference, Budapest, Hungary (2003)

[2] The complete rule set with 23 rules is provided at http://www.informatik.uni-ulm.de/ki/Liebig/MM-ruleset.txt

Representing and Reasoning for Spatiotemporal Ontology Integration

Nacera Bennacer[1], Marie-Aude Aufaure[1], Nadine Cullot[2], Anastasiya Sotnykova[3], Christelle Vangenot[3]

[1] École Supérieure d'Électricité, 3 rue Joliot-Curie, F- 91192 Gif-sur-Yvette, France
{Nacera.Bennacer, Marie-Aude.Aufaure}@Supelec.fr

[2] Université de Bourgogne, UFR Science et Techniques, 21078 Dijon Cedex, France
Nadine.Cullot@u-bourgogne.fr

[3] Swiss Federal Institute of Technology in Lausanne, Database Laboratory, Switzerland*
{Anastasiya.Sotnykova, Christelle.Vangenot}@epfl.ch

1 Motivation

The World-Wide Web hosts many autonomous and heterogeneous information sources. In the near future each source may be described by its own ontology. The distributed nature of ontology development will lead to a large number of local ontologies covering overlapping domains. Ontology integration will then become an essential capability for effective interoperability and information sharing. Integration is known to be a hard problem, whose complexity increases particularly in the presence of spatiotemporal information. Space and time entail additional problems such as the heterogeneity of granularity used in representing spatial and temporal features. Spatio-temporal objects possess intrinsic characteristics that make then more complex to handle, and are usually related by specific relationships such as topological, metric and directional relations. The integration process must be enhanced to tackle mappings involving these complex spatiotemporal features. Recently, several tools have been developed to provide support for building mappings. The tools are usually based on heuristic approaches that identify structural and naming similarities [1]. They can be categorized by the type of inputs required for the analysis: descriptions of concepts in OBSERVER [2], concept hierarchies in iPrompt and AnchorPrompt [3] and instances of classes in GLUE [4] and FCA-Merge [5], However, complex mappings, involving spatiotemporal features, require feedback from a user to further refine proposed mappings and to manually specify mappings not found by the tools.

2 Our Contribution

We aim at helping users capturing meaningful mappings between spatiotemporal ontologies, and to reason about them. Our approach to interrelate spatiotemporal data

* Work by authors from EPFL is done in the framework of the EPFL Center for Global Computing and is supported by the Swiss National Funding Agency OFES as part of the European projects KnowledgeWeb (FP6-507482) and DIP (FP6-507483).

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 30–31, 2004.
© Springer-Verlag Berlin Heidelberg 2004

sources relies on two well-known formalisms, conceptual models and description logics (DL). As the conceptual model, we employ MADS, a spatiotemporal conceptual model with multi-representation capabilities [6]. MADS has rich spatiotemporal semantics that is easily understood by a wide circle of designers. For the reasoning services, we exploit the expressive power of the $\mathcal{ALCRP}$ $(\mathbf{S}_2 \oplus \mathbf{T})$ logic [7] that augments basic $\mathcal{ALC}$ description logic with a spatiotemporal concrete domain. The first phase of our integration approach consists in defining the source schemas in MADS. Inter-schema mappings are then defined between the source schemas. As defining inter-schema mappings is an error-prone activity, we check the compatibility of the set of mappings. For that purpose, the set of mappings is translated into description logic [8] with spatial and temporal roles and axioms defined in $\mathcal{ALCRP}$ $(\mathbf{S}_2 \oplus \mathbf{T})$. Exploiting the inference engines developed for DL models, we then can focus on detecting inconsistencies and implicit information in the set of mappings. From the set of validated mappings, integration patterns for a unified ontology are then proposed. As further work, we plan to design a framework that would combine tools like the MADS editor to describe source schemas with an automatic translator from MADS to a description logic formalism, and finally, with a DL reasoner like Racer enhanced with spatiotemporal semantics. A detailed description of our approach is available
in [9].

References

1. S. Staab, R. Studer (Eds.) Handbook on Ontologies. International Handbooks on Information Systems, Springer-Verlag 2004.
2. E. Mena, A. Illarramendi, V. Kashyap and A. Sheth: OBSERVER: An Approach for Query Processing in Global Information Systems Based on Interoperation across Preexisting Ontologies. Distributed and Parallel Databases - An International Journal 8(2), 2000.
3. N. F. Noy, and M. A. Musen: Anchor-PROMPT: Using non-local context for semantic matching. In Workshop on Ontologies and Information Sharing, IJCAI 2001.
4. A. Doan, J. Madhavan, P. Domingos and A. Y. Halvey: Learning to Map between Ontologies on the Semantic Web. In Proceedings of the Eleventh International World Wide Web Conference, WWW 2002, Hawaii, US.
5. G. Stumme, A. Maedche: FCA-MERGE: Bottom-Up Merging of Ontologies, In Proc. of International Joint Conference on Automated Reasoning, IJCAR 2001.
6. A. Sotnykova, S. Monties, S. Spaccapietra: Semantic Integration in MADS Conceptual Model. In H. Bestougeff, B. Thuraisihgham (Eds.) Heterogeneous Information Exchange and Organizational Hubs, Kluwer, 2002.
7. V. Haarslev, C. Lutz, R. Möller: A Description Logic with Concrete Domains and a Role-forming Predicate Operator. Journal of Logic and Computation, 9(3), 1999.
8. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati: Description Logic Framework for Information Integration. In Proc. of the 6th International Conference on the Principles of Knowledge Representation and Reasoning, Trento (I), June 2-5, 1998.
9. A. Sotnykova, M-A Aufaure, N. Bennacer, N. Cullot and C. Vangenot: Semantic Mappings in Description Logics for Database Schema Integration. Journal of Database Semantics, Springer LNCS, submitted for publication.

Effective and Efficient Melody-Matching Method in a Large-Scale Music Database

Sung-Phil Heo

Service Development Laboratory, Korea Telecom
17, Woomyeon-dong, Seocho-gu, Seoul, 137-792 Korea
hsphil@kt.co.kr

Abstract. Rapid progress in hardware and software technologies has made it possible to manage and access large volumes of music data. The effectiveness (high-accuracy) and efficiency (high-speed) of retrieval techniques for music information data are key issues in the area of large-scale MIR systems. Retrieval performances by conventional methods, however, have not been good enough in large-scale music information retrieval systems. DP matching generally has high retrieval accuracy. However, its retrieval time depends on the size of the database. Thus, to improve the effectiveness and efficiency of music information retrieval in large-scale music databases, two techniques propose: fuzzy-quantization based on melody representation and error-tolerant linear matching. The fuzzy-quantization method uses membership functions based on a probability distribution. The error-tolerant linear matching method is a linear-matching method that considers the possibility of the insertion or omission of notes. Using these techniques, higher recall rates were achieved compared to conventional methods. Retrieval time in pre-selection was about 0.5 seconds per query in large-scale DB, including database loading.

1 Introduction

In order to make effective use of the huge amounts of multimedia content, information retrieval systems are needed. Conventional information retrieval (IR) research has been mainly based on text information. However, this research cannot be directly applied to a multimedia information retrieval system because the text-based IR method retrieves desired text documents using a search query consisting of a number of text-based keywords [1]. For example, in conventional text-based music information retrieval systems, retrieval keys mainly consist of the text information such as the singer's name, composer, title of the piece of music, or lyrics of the song. On the other hand, content-based retrieval enables users to directly represent what they want and it makes the retrieval system easier to use [2]. The system described here is a content-based retrieval system for music that accepts hummed tunes as queries even if the name of the song or the singer cannot be recalled or remembered.

In this paper, an MIR system is integrated in a two-stage matching method for effective and efficient music retrieval in a large-scale database. The first stage is preselection based on linear matching and the second stage is fine-matching using continuous DP. The first stage pre-selection is carried out through linear matching. At this time, the

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 32–33, 2004.
© Springer-Verlag Berlin Heidelberg 2004

pre-selection algorithm gives each candidate a similarity score. When the degree of similarity becomes higher, the music of the DB resembles the humming. The *top-n* songs with the highest scores are chosen as the candidates, which are passed on to the first stage. In the second stage, DP-based fine-matching is carried out on the candidates. The fine-matching gives each candidate a DP distance score [3]. When the DP distance becomes smaller, the music of the DB resembles the humming. That is, the DP distance is zero when the humming and the music in the DB are identical. Next, the unified score is calculated using an integrated function. Finally, the MIR system shows the user the top-*m* ranked music according to the unified score as a retrieval result.

2 Effective and Efficient Melody-Matching Method

This paper proposes a new pre-selection method for use in large-scale music retrieval systems. Since the matching algorithm that uses DP is too slow for large-scale MIR systems, the reduction of the number of candidate songs using a pre-selection method is indispensable for a usable large-scale MIR system. It is a two-stage process that tries to achieve high selectivity in the pre-selection process followed by a *fine-matching* process. The quantization-based melody representation and distance-based melody representation were investigated through experiments. The results showed that the quantization-based method performs better. Then, we looked at the performance of a fuzzy quantization [4] technique for further improvements. Finally, experiments were undertaken to determine the optimal parameter settings and show the overall performance.

In order to investigate the performance of the proposed method, music retrieval experiments were carried out. The large-scale music database has 10,155 songs that consist of 155 children's songs and 10,000 pieces automatically generated by n-gram. The probabilities of n-gram [5] were estimated from the 155 children's songs. Humming data sung by five subjects were used as the test data. The pre-selection result was evaluated using two factors: pre-selection rate and recall rate. It is ideal for a pre-selection method to have a lower pre-selection rate and a higher recall rate.

The result showed that a recall rate of approximately 90% was obtained when the pre-selection rate was 10%. Retrieval time in the pre-selection was about 0.5 seconds per query in the large-scale DB, including database loading.

References

1. D. Feng, W.C. Siu and H. Zhang, "Multimedia Information Retrieval and Management: Technological Fundamentals and Applications," Springer-Verlag, 2003.
2. Anni R. Coden, Eric W. Brown and Savitha Srinivasan, "Information Retrieval Techniques for Speech Applications (LNCS 2273)," Springer-Verlag, 2002.
3. Sung-Phil Heo *et al.,* "Multiple Pitch Candidates based Music Information Retrieval Method for Query-by-Humming," In International Workshop on AMR2003, pp. 189-200, 2003.
4. Zadeh, L.A., "Fuzzy sets," Information and Control, Vol. 8, pp. 338-353, 1965.
5. L.R. Rabiner and B.H. Juang, "Fundamentals of Speech Recognition," Prentice-Hall, 1993.

A CORBA-Based System for the Monitoring of Petroleum Fields*

Diogo Salim[1,2], Thais Batista[1], and Adelardo Medeiros[2]

[1] Informatics Department and [2] Department of Computer and Automation Engineering,
Federal University of Rio Grande do Norte, Campus
Universitário Lagoa Nova, 59072-970, Natal, RN, Brazil
{diogo@interjato.com.br, thais@ufrnet.br, adelardo@dca.ufrn.br}

1 The Architecture of the Monitoring System

This paper presents the definition and implementation of a CORBA-based system for the distributed monitoring of automated oil wells located in onshore petroleum fields. Conducted by the automation of industrial processes in the petroleum and gas area, the development of computational distributed systems, based on emerging middleware technologies and designed to monitor and actuate in such particular domain, has been strongly worldwide stimulated. The monitoring of the oil wells' behavior can serve as a powerful tool for the oil production analysis. Since the monitoring system takes part of a larger oil production system with heterogeneous and distributed modules, CORBA and its services provide support for transparent interaction among such modules. The implemented system has the following goals: to register characteristics associated with the operational process of oil extraction in automated wells of oil fields, and based on this collected information, to provide descriptive and behavioral results in order to make possible an efficient monitoring process and to also supply control procedures. To support multiple clients' invocations, the system provides concurrent monitoring, control tasks and data distribution.

Building the monitoring application is a very demanding effort since it must satisfy a set of requirements deriving from several research areas, arising from the concepts in automation, development and implementation of wireless communication, and also the implementation of a distributed system composed of heterogeneous modules. The parts of the system are divided into four sub-areas, as illustrates Fig. 1.

The **hardware and instrumentation of the rod pump wells** are the position and load sensors, a Programmable Logic Controller (PLC) and a radio modem.

The **field communication network infrastructure** provides a master-slave communication between the PLC installed in every rod pump well and the Central Monitor. Since most of the rod pumps are usually located in remote and uninhabited areas, information between hosts is transported via a wireless communication by means of radio modems installed and connected to the PLC.

* This work has the financial support of ANP – Brazilian Petroleum Agency

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 34–35, 2004.
© Springer-Verlag Berlin Heidelberg 2004

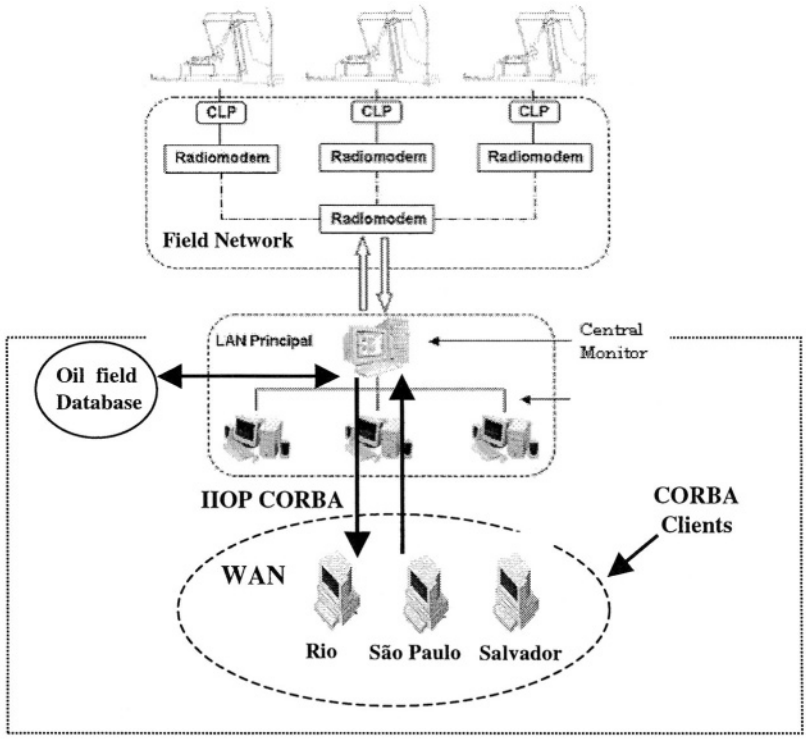


Fig. 1. Architecture of the Monitoring System

The **Central Monitor** (CM) links the universe of the monitored petroleum fields and the remote clients. Its major task is to communicate with oil wells, to store the collected data and to conduct the remote monitoring of oil extraction. CM supports distributed and heterogeneous aspects in order to make possible an HOP communication between remote clients and servers based on CORBA. The following tasks consist in the major events performed either by CM or accomplished by a remote client: (1) **Automatic Data Sampling**: a periodic task that accesses the monitoring data collected and stored in every well and transmits it to CM; (2) **System Monitoring**: After updating the received data, the system processes and tests the integrity and status conditions of the corresponding information. Several procedures are implemented in order to analyze the well's operational pumping process. Based on the collected information, automatic or human-assisted decisions can be taken.

The **Client** is the module of the system by which the users can access the oil monitoring application and remotely operate it. Communication and interaction with the CM is carried out in a distributed and transparent mode, performing thus remote data requisitions and control procedures requested by the users.

The results show that by implementing the monitoring system based on CORBA, a number of remarkable features in this specific petroleum-monitoring domain have been introduced, providing thus portability, reconfiguration and support for transparent communication between heterogeneous and distributed components.

Context-Aware Distributed Deployment of Component-Based Applications

Dhouha Ayed, Chantal Taconet, Nawel Sabri, and Guy Bernard

GET / INT, CNRS Samovar 5157, 9 rue Charles Fourier, 91011 Évry, France

1 Introduction

Software deployment refers to all the activities performed after the development of a software in order to make it available to users. These activities mainly consist of installation, configuration, activation of the software but also of reconfiguration, update and de-installation of the software [1].

With recent advances in wireless networking technologies and mobile computing devices, making applications context-aware becomes essential. The reason is that, with these new technologies, device resources (e.g. memory, battery, CPU) may be scarce, and application execution context (e.g. user location, device screen size) is variable [2].

Context information may be taken into account at different times of the application life cycle. We argue that the deployment is an effective period to take context state into account. Context information allows the deployer to install an application suited to each execution context. We introduce in this presentation the concept of context-aware deployment [3].

Context-aware deployment has to be processed just-in-time (i.e. when a user accesses to a service). Just-in-time deployment enables users to automatically install and configure applications suited to their needs, the resources of their computers and their surrounding environments. Because software is installed at access time only and may be de-installed afterwards, just-in-time deployment enables scarce resources saving. Furthermore, just-in-time deployment implies the automation of the deployment tasks and thus relieves the user from the deployment repetitive tasks.

Our work focuses on multi-component applications which are described by an assembly. An application assembly defines component instances and their connections [4]. With multi-component applications, deployment may be distributed [5]. In this way, some application components may be hosted by the user's computer, while other components are hosted by computer servers.

The solution we present in the poster is dedicated to context-aware distributed deployment of multi-component applications.

2 Context-Aware Deployment

We summarize briefly in this section: (1) the type of decisions allowed by a context-aware deployment, (2) the data models proposed for context-aware deployment and (3) the deployment platform we have designed.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 36–37, 2004.
© Springer-Verlag Berlin Heidelberg 2004

During context-aware deployment, the context has the following impacts: (i) the choice of the application structure (components' assembly) which may vary according to the context - the application structure defines the type of each component instance; (ii) since several implementations may be available for each given component type, context-aware deployment has to choose the suited implementation for each component instance; (iii) the placement of each component within a set of computers is another choice of just-in-time deployment; and finally, (iv) the configuration of each component has to be set according to the context.

We define two data models for the description of information necessary for context-aware deployment. Firstly, a data model to define the context which is relevant for the deployment. Secondly, a data model to define the rules necessary to make the deployment choices. For those two data models, the descriptions may be defined at two levels: (i) generic (i.e. applicable to any application) and (ii) specific (i.e. applicable to a given set of applications).

In order to make a deployment service context-aware, we have added a set of components to an existing middleware deployment service layer. Some of these components are hosted by the user terminal and others by the deployment provider domain. These components include context collectors, repositories and a deployer adapter. The role of the deployer adapter is to generate a deployment plan in which all the deployment decisions have been taken according to the context.

3 Conclusion

Our context-aware deployment scheme may be seen as a deployment preprocessor which generates a fixed deployment plan. The deployment is then realized by the underlying middleware deployment service. A first implementation of the context-aware deployer has been developed. It has been coupled with the OpenCCM middleware platform to make context-aware deployment of CCM applications. The implementation is under evaluation on some demonstrator applications.

References

1. Hall, R.S., Heimbeigner, D., Wolf, A.L.: A cooperative approach to support software deployment using the software dock. In: International Conference on Software Engineering. (1999) 174–183
2. Dey, A.K: Understanding and Using Context. Personal and Ubiquitous Computing Journal **5** (2001) 4–7
3. Ayed, D., Taconet, C., Bernard, G.: A Data Model for Context-Aware Deployment of Component-based Applications onto Distributed Systems. Component-oriented approaches to context-aware systems Workshop ECOOP'04 (June 2004)
4. OMG: UML Superstructure 2.0. Adopted Specification ptc/03-08-02 (August 2003)
5. OMG: CORBA Component Model Version 3.0. Adopted Specification formal/2002-06-65 (June 2002)

Towards an Incremental Process Model Based on AOP for Distributed Component-Based Software Development

Alexandre Alvaro[1], Daniel Lucrédio[2], Antonio Francisco do Prado[2], and Eduardo Santana de Almeida[1]

[1] Federal University of Pernambuco
C.E.S.A.R – Recife Center for Advanced Studies and Systems
{aa2, esa2}@cin.ufpe.br
[2] Computing Department – Federal University of São Carlos, Brazil
{lucredio, prado}@dc.ufscar.br

Abstract. In spite of recent and constant researches in the Component-Based Development area, there is still a lack for patterns, processes and methodologies that effectively support either the development "for reuse" and "with reuse". This short paper presents a process model that integrates the concepts of component-based software engineering, frameworks, patterns, non-functional requirements and aspect-oriented programming. This process model is divided in two activities: Domain Engineering and Component-Based Development. An aspect-oriented non-functional requirements framework was built to aid the software engineer in this two activities.

1 Introduction

One of the most compelling reasons for adopting component-based approaches is the premise of reuse. The idea is to build software from existing components primarily by assembling and replacing interoperable parts. One of the most important approaches for achieving reuse is the Component-Based Development (CBD).

However, the Non-Functional Requirements (NFRs) many times are not considered in components development activities. There are some reasons that can help us to understand why these requirements are not explicitly dealt with, such as the difficult to deal with different NFRs because they often compete with each other and with the functional requirements

To help to overcome the difficulties of treating different NFRs, Aspect-Oriented Programming (AOP) [1] may be a good choice. Thus, in order to aid the Software Engineer in the integration of the NFRs into design decisions during the software development process, a framework may be used to represent, organize and analyze NFRs [2], increasing quality and reducing time and costs.

In this context, motivated by ideas of reuse, component-based development, non-functional requirements and Aspect-Oriented Programming, this work proposes an Incremental Process Model based on AOP for Distributed Component-Based Software Development.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 38–39, 2004.
© Springer-Verlag Berlin Heidelberg 2004

2 Towards an Incremental Process Model Based on AOP for the Distributed Component-Based Software Development (DCBSD)

In order to enable CBD with aspect-oriented support for NFRs, such as distribution, for example, an Incremental Process Model (IPM) was defined.

The IPM is divided in two activities, as shown in Figure 1, using SADT [3] notation. In the first activity, **Domain Engineering (DE)**, the problem domain requirements are identified and organized in a series of reusable information. Software components are specified, designed – including the NFRs design, which were performed through aspect-oriented NFRs framework support– and tested, being then stored into a repository. In the next activity, **Component Based Development (CBD),** the software engineer may build applications that reuse these components, consulting the repository to find and reuse them. Still, the aspect-oriented NFRs framework aids the software engineer in the application development.

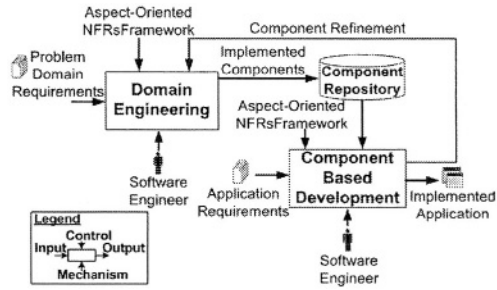


Fig. 1. Aspect IPM.

3 Conclusion

The main contribution of this work is to propose an Incremental Process Model based on AOP for DCBSD, providing a high reutilization degree, guiding the software engineer during the development and reuse of software components.

A preliminary evaluation was performed, indicating that this Incremental Process Model provides several benefits, standing out the reduction of the interlacement that occurs when combining different functional and non-functional requirements. This results, at the end of the process, in a code with increased modularity, flexibility and overall quality. However, a more complete evaluation is being planned, in order to obtain greater degree of certainty in these affirmations.

References

1. G. Kiczales; J. Lamping; A. Mendhekar; C. Maeda; C.V. Lopes; J.M. Loingtier; J. Irwin, Aspect-Oriented Programming (AOP). In 11th European Conference on Object-Oriented Programming (ECOOP) - LNCS, Springer-Verlag, Finland, 1997.
2. J. Mylopoulos; L. Chung; B. Nixon, Representing and Using Non-Functional Requirements: A Process Oriented Approach. IEEE Transaction on Software Engineering, v. 18, no. 6, June, 1992, pp. 483-497.
3. D.T. Ross, Structured Analysis (SA): A Language for Communicating Ideas. IEEE Transaction on Software Engineering. v. 3, no. 1, January, 1977, pp. 16-34.

Supporting Nomadic Desktop Computing Using an Internet Application Workbench

Stefan Paal[1], Reiner Kammüller[2], and Bernd Freisleben[3]

[1] Fraunhofer Institute for Media Communication
Schloss Birlinghoven, D-53754 St. Augustin, Germany
stefan.paal@imk.fraunhofer.de
[2] Department of Electrical Engineering and Computer Science, University of Siegen
Hölderlinstr. 3, D-57068 Siegen, Germany
kammueller@pd.et-inf.uni-siegen.de
[3] Department of Mathematics and Computer Science, University of Marburg
Hans-Meerwein-Strasse, D-35032 Marburg, Germany
freisleb@informatik.uni-marburg.de

Abstract. Nomadic Internet users typically employ different computers to access applications from different locations. A particular problem in this scenario is the deployment and hosting of applications which do not run on a remote server but have to be dynamically deployed on the currently involved client node. In this paper, we present an Internet Application Workbench which provides a pervasive application environment across heterogeneous Internet nodes.

1 Introduction

Several proposals have been made to establish the web browser as the universal user interface to a *pervasive application environment* supporting nomadic computing [1]. The basic idea is to provide the illusion of an application environment which is not bound to a specific computing device but is traveling with the user as she or he switch to another device and which can be nevertheless uniformly and transparently accessed. In this context, a particular application scenario is *nomadic desktop computing* and the alternate employment of various desktop computing devices by a nomadic user who is moving from one station to another one [2]. A basic problem in this application scenario is the dynamic deployment of applications and the seamless customization of the specific desktop computing system to keep the illusion of a pervasive application environment for each user [3]. In this paper, we briefly present an *Internet Application Workbench* which provides a uniform graphical user interface across different desktop computing systems. It is built upon a cross-platform application environment that enables the deployment and execution of an application on-demand across heterogeneous computing systems. It automatically adapts itself without user intervention and provides a suitable runtime environment for each deployed application. As a result, the nomadic user is moving from one desktop computing system to the other and always gets a personalized Internet application workbench along with her or his configured application running in a pervasive application environment.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 40–43, 2004.
© Springer-Verlag Berlin Heidelberg 2004

2 Nomadic Desktop Computing

The general objective of *nomadic desktop computing* is the illusion of a pervasive and personalized application environment across heterogeneous desktop computing systems. Nomadic users can travel around and employ various desktop computers to work with their applications. Consequently, instead of installing and configuring desktop applications of a user on a single personal computer only, the applications and the related configurations are dynamically and seamlessly deployed on different nodes whenever the user moves, as shown in fig. 1.

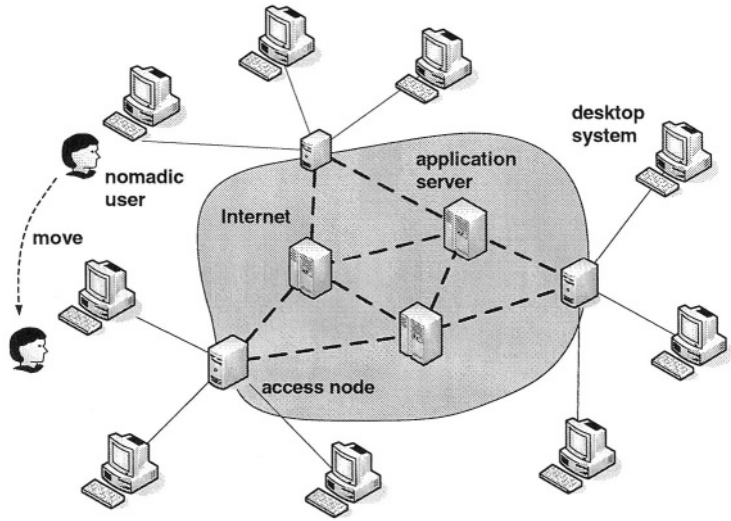


Fig. 1. Nomadic Desktop Computing

Furthermore, similar to the success of the web and its graphical browser approach, a pervasive application environment supporting nomadic desktop computing should also provide a uniform graphical interface independent of the currently involved desktop computing system. It is not only used to control a single application but also used to run multiple applications within the same window, like a web browser concurrently opening various web pages. Finally, the pervasive application environment should be globally deployable in an existing infrastructure like the Internet and manage common deployment and composition tasks.

We have identified several requirements of a pervasive application environment supporting nomadic desktop computing, e.g. graphical desktop interface, distributed application management, autonomic platform configuration and cross-platform personalization. Certainly, there are several approaches addressing these requirements such as browser computing, rich thin client computing, fat client computing and personal computing. They mainly differ in the amount of specific application code which is deployed on the desktop computer and the degree of dynamically configuring the application environment. However, none of them currently provides the illusion of a pervasive application environment automatically configured and personalized across heterogeneous and unknown desktop computing systems.

3 Internet Application Workbench

In this section, we briefly present the proposed *Internet Application Workbench* which is supposed to provide the illusion of a pervasive application environment across heterogeneous computing systems. It offers a graphical desktop interface written in Java and is based on a cross-platform application environment [4] that also supports the launching of applications written in different programming languages.

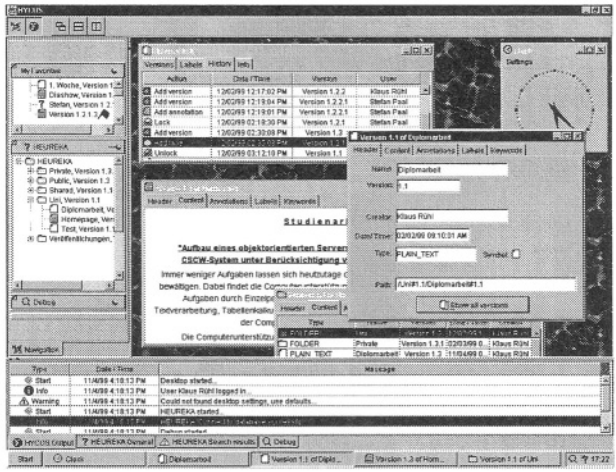


Fig. 2. Uniform Graphical User Interface

An essential issue of nomadic desktop computing is the dynamic installation and deployment of the required applications on the currently employed client node. To this end, we split up the deployment process in two parts, as shown in fig. 3. The first part uses Sun Java Web Start and is only run for the initial deployment of the cross-platform application environment. A corresponding JNLP file is stored on a web server and contains the basic libraries and initial settings of our cross-platform application environment. After Java Web Start has launched the Internet Application Workbench, a graphical desktop interface is provided to the user. It can be used to further customize the workbench and browse for further applications, components and profiles using the passed addresses of related repositories. For instance, the user selects an application configuration from the *application repository* which indicates the runtime configuration and the modules needed to launch the selected application. After that, a suitable application environment is prepared using the information from *the platform configuration* and the required application modules are retrieved from the *module repository.* Finally, the application environment is customized according to the application and the user profile loaded from the *profile repository.*

As a result, the Internet Application Workbench is not relying on a static, server-side deployment and predefined composition strategy such as Sun JNLP but it is dynamically adapted depending on the currently employed platform and requested application. Furthermore, dynamic personalization of the Internet Application Workbench and related applications is supported using profile repositories without bothering users to synchronize the current configurations on their own.

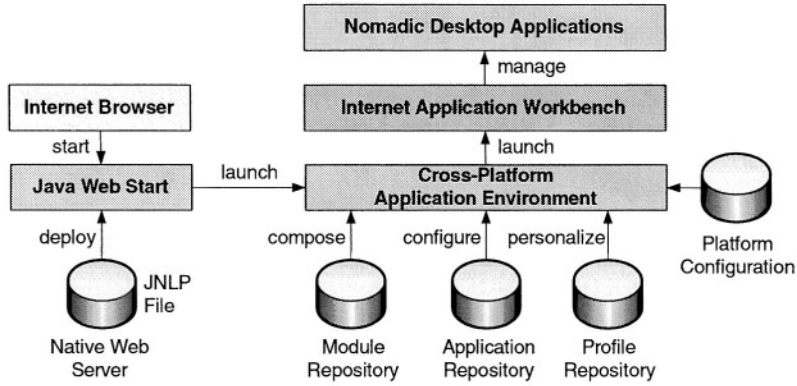


Fig. 3. Dynamic Deployment and Customization of the Internet Application Workbench

4 Conclusions

In this paper, we have briefly presented the so called *Internet Application Workbench* which is able to launch and control dynamically requested Java applications. It is not installed and bound to a single desktop computing system but it is transparently deployed and customized on the currently employed node on request; hence it travels with nomadic users as they move from one station to another. It has been built as part of our cross-platform application environment [5] and is basically meant as an interface to a pervasive application environment supporting nomadic desktop computing in heterogeneous environments like the Internet. In this context, we use the approach as an extension of our Internet portal *netzspannung.org* [6] to provide an advanced user interface to the community workspace. Further development is focusing on the migration of an active working session to another desktop computing system and the automatic composition of the recent running applications there.

References

1. Kleinrock, L. Nomadic Computing and Smart Spaces. IEEE Internet Computing. Vol. 4, Nr. 1. IEEE 2000. pp. 52-53.
2. Lyytinen, K., Yoo, Y. The Next Wave of Nomadic Computing. Information System Research. Vol. 13, Nr. 4. Informs 2002. pp. 377-388.
3. Wood, K. R., Richardson, T., Bennett, F., Harter, A., Hopper, A. Global Teleporting with Java: Toward Ubiquitous Personalized Computing. IEEE Computer. Vol. 30, Nr. 2. IEEE 1997. pp. 53-59.
4. Paal, S., Kammüller, R., Freisleben, B. A Cross-Platform Application Environment for Nomadic Desktop Computing. Objects, Components, Architectures, Services, and Applications for a NetworkedWorld. Erfurt, Germany. Springer 2004. (to appear)
5. Crossware - Middleware for Cross-Platform Application Environment. http://crossware.org
6. netzspannung.org, Communication Platform for Digital Art and Media Culture. http://netzspannung.org

A Distributed Object Platform to Achieve Nomadic Interactions*

Domenico Cotroneo, Armando Migliaccio, and Stefano Russo

Dipartimento di Informatica e Sistemistica– Universitá di Napoli Federico II
Via Claudio 21,80125 Napoli, Italy
{cotroneo,armiglia,sterusso}@unina.it

1 Rationale

Network infrastructures composed of wireless access points (e.g. IEEE 802.11 APs) connected via a LAN have been enabling a form of mobile computing known as Nomadic Computing (NC). In order to manage migrations of a mobile terminal among different wireless networks, such infrastructures are often decomposed in several domains: a domain can be a building floor, a building room, or a certain zone of a campus. Since the traditional middleware are unsuitable for mobile computing [1], during the past years a great deal of research has been conducted. Research efforts have been progressed along the following directions: i) providing mechanisms, such as context awareness, reconfiguration, spontaneous discovery; ii) dealing with QoS aspects such as security. While we recognize these studies as fundamental milestones for the pursuit of new middleware for mobile computing, most of them do not effectively explore how the distributed interaction (such as RPC or RMI) has to take place.
There is a need for a communication broker acting as a building block for the realization of higher level middleware services. This work proposes the Esperanto Broker (EB), a communication platform aiming to address the following issues: i) providing developers with an interaction paradigm, powerful and simple to program; ii) supporting terminals mobility.

2 Overall Architecture

In order to take into account the above mentioned requirements, the EB provides: i) an object oriented abstraction of paradigms proposed in WSDL specification (preserving a structured and typed programming abstraction); and ii) mechanisms to support terminal mobility (i.e., implementing handover procedures and decoupling interacting entities at middleware layer). In this way, an

* This work has been partially supported by the Italian Ministry for Education, University and Research (MIUR) in the framework of the FIRB Project "Middleware for advanced services over large-scale, wired-wireless distributed systems (WEB-MINDS), and by Regione Campania in the framework of "Centro di Competenza Regionale ICT".

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 44–45, 2004.
© Springer-Verlag Berlin Heidelberg 2004

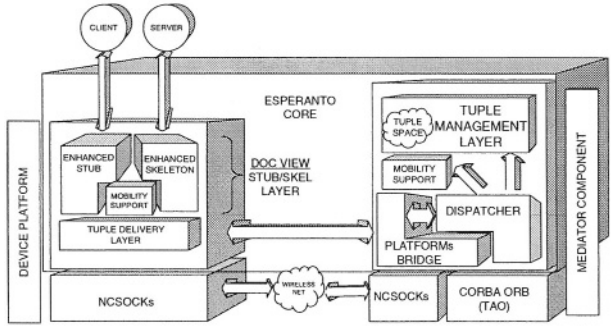


Fig. 1. The Esperanto Broker layered architecture

application/middleware designer can model software components as a set of objects that are distributed over mobile terminals and interact each other via remote method invocations, despite of terminal movements among different NC domains.The layered architecture implemented by EB is depicted in figure 1. EB is composed of two platforms, *Device-side* and *Mediator-side,* for the following reasons: i) as stated in [1], in order to take account for mobility, there is a need of decoupling the distributed counterparts by means of an intermediary. The mediator platform adopts the tuple space model for decoupling object interactions both in time and space. Moreover, as stated by [2], there is a need to implement mobility management at middleware level, as well. The mediator implements handover procedures when a mobile terminal changes its current NC domain. In order to simplify such procedures, a mediator is assigned to each NC domain in which the network infrastructure is decomposed. Mediators are connected via the fixed network and cooperate each others for managing terminal mobility; ii) as stated in [3] the Distributed Object Computing improves software quality as compared with other approaches (e.g. publish/subscribe or message passing). The device platform provides developers with an enhanced IDL language which encompasses all the WSDL interaction types. It also encapsulates service method invocations into a list of tuples which enables the Mediator to keep the interaction decoupled. Finally, the EB appears to application/middleware developer like an ORB for distributed computation, taking into account terminal mobility via the mobile-enabled tuple space.

References

1. C. Mascolo, L. Capra, and W. Emmerich. Mobile Computing Middleware. *Lecture Notes In Computer Science, advanced lectures on networking,* pages 20 – 58, 2002.
2. K. Raatikainen. Wireless Access and Terminal Mobility in CORBA. Technical Report, December 1997. OMG Technical Meeting, University of Helsinky.
3. S. Yau and F. Karim. A Lightweight Middleware Protocol for Ad Hoc Distributed Object Computing in Ubiquitous Computing Environments. *In Proc. of 6th IEEE Int. Symp. on Object-Oriented Real-Time Distributed Computing,* pages 14–16, 2003.

A Virtual-Machine-Based Middleware

Alcides Calsavara, Agnaldo K. Noda, Juarez da Costa, Carlos Kolb, and Leonardo Nunes

Pontifícia Universidade Católica do Paraná
Programa de Pós-Graduação em Informática Aplicada
Rua Imaculada Conceição, 1155, Prado Velho, Curitiba, PR, Brazil
{alcides,anoda,juarez,kolb,leonardo}@ppgia.pucpr.br
http://www.ppgia.pucpr.br/~alcides

1 Introduction

Distributed computing introduces great complexity in software systems development, deployment and maintenance. As a consequence, building quality distributed software systems is hard and relies fundamentally on programmers expertise and good tool assistance. A key factor for the success of a development system is the capability to easily create prototypes, that is, create a preliminary version for the target system where its requirements can be quickly implemented, debugged, tested and simulated. Currently, a number of distributed software systems development tools exist, but they hardly favor learning about distributed computing and hardly favor prototyping because they are typically designed either to satisfy industrial standards – industrial perspective – or to experiment new concepts – research perspective. Industrial tools are concerned with productivity and software efficiency and robustness. Research tools normally have complex user interfaces and require the knowledge of particular concepts. Therefore, there is a need for software development tools where programmers can both learn about distributed computing – pedagogical perspective – and build quality distributed software systems through prototyping – experimental perspective.

2 Objectives

VIRTUOSI Project aims at building a toolkit to assist in developing and executing distributed software systems from both pedagogical and experimental perspectives. From pedagogical perspective, VIRTUOSI will permit programmers to be taught about distributed computing in a structured manner. From experimental perspective, VIRTUOSI will permit programmers to create prototypes which are mature and robust; such prototypes will be the basis for easily developing the corresponding operational releases by using specific technologies. Thus, VIRTUOSI will assist in developing distributed software systems of great quality in a short period of time, since programmers will be better trained and will be able to implement and test critical system requirements in a controlled manner.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 46–47, 2004.
© Springer-Verlag Berlin Heidelberg 2004

3 Architecture

The VIRTUOSI toolkit encompasses many aspects of distributed computing and of software engineering. It should comprise artifacts to build software systems and a full-fledged distributed runtime system. The pedagogical perspective requires an environment where a programmer can write a program by using a simple yet powerful set of abstractions, and then test that program in a way that all abstractions employed can be easily traced, i.e., translations from programming abstractions to runtime structures should be minimized. Another requirement from the pedagogical perspective is that the environment should be as neutral as possible with respect to the actual runtime platform in order to avoid unnecessary distractions. Finally, the pedagogical perspective requires an environment where the programmer can easily select which system aspects should be either transparent or translucent in a given moment. The experimental perspective, on the other hand, requires an environment where real-world applications can be quickly developed and carefully tested.

The VIRTUOSI runtime environment is composed of a collection of communicating virtual machines – a runtime environment for distributed software systems similar to *middleware* platforms. VIRTUOSI Project adopts object orientation as the paradigm for both applications development and runtime system. The runtime system should preserve all object-oriented abstractions in order to minimize translations that could make it difficult debugging applications; that helps in fast building prototypes – the experimental perspective – and helps programmers to understand better programming concepts – the pedagogical perspective. The reflective approach allows programmers to change systems behaviour in two levels: application level and runtime system level. From pedagogical perspective, programmers can selectively choose what system features should be or not transparent, so that it is possible to study each feature individually or combine just some of them. And, from experimental perspective, programmers have a great dynamism for developing software systems as components can be easily replaced and tested.

4 Implementation

The code interpreted by a VIRTUOSI machine is a *graph of objects* which represent the original source code and organized in such a way that all the semantics is preserved and made available at runtime. Such objects are instances of classes defined in the VIRTUOSI metamodel. Such complex representation – when compared to Java bytecode, for example – is justified in our project because we intend to provide full support to computational reflection.

Currently, a full-fledged version of the toolkit is under development. To date, we have developed a programming language and corresponding compiler, a prototype version of a code interpreter (the virtual machine kernel), a mechanism for transparent remote method invocation and a mechanism for object mobility. In the near future, we expect to develop a mechanism for concurrency control and also provide for event-based distributed programming.

Workshop on Grid Computing and Its Application to Data Analysis (GADA) PC Co-chairs' Message

We wish to extend a warm welcome to GADA 2004, the First International Workshop on Grid Computing and its Application to Data Analysis. This workshop will be held in conjunction with the On The Move Federated Conferences 2004 (OTM 2004).

The complexity of current computational systems requires the deployment of new technologies, which constitute a challenge in the field of Communications and Distributed Systems. The OTM 2004 Federated Conferences provide a global scenario in which researchers can extend their background to different related research areas, such as Data and Web Semantics, Distributed Objects, Web Services, Databases, Cooperation, Interoperability and Mobility.

In the last decade, Grid computing has become one of the most important topics to appear and one of the most widely developed fields. Research into Grid computing is making rapid progress, owing to the increasing necessity of computation resources in the resolution of complex applications. The great challenge is the complete integration of heterogeneous computing systems and data resources with the aim of providing a global computing space. The achievement of this goal will involve revolutionary changes in the field of computation, enabling seamless resource and data sharing across networks.

GADA 2004 aims to provide a forum for novel topics related to Grid computing, providing an opportunity for researchers to discuss and identify key aspects of this important area.

The set of technical papers presented here is the result of a difficult and thorough review process. This year the workshop received 58 submissions of high quality from which the 19 papers making up the technical programme were selected. The number of submissions and the quality and diversity of the resulting programme are testimony of the interest in this up-and-coming area.

This workshop could not have taken place without considerable enthusiasm, support and encouragement as well as sheer hard work. Many people have earned the thanks of those who attended and organized GADA 2004. In particular, we would like to gratefully thank:

- The many supporters of OTM 2004 for their contributions to the conference. Many of these people have been involved with the OTM conferences since several years.
- Those members of the Workshop Program Committee who gave their time and energy to ensure that the conference maintains high technical quality and runs smoothly. The many individuals we owe our thanks to are listed in this volume.
- All those who submitted to the workshop. The set standard was higher than our expectations and reflects well on the research work in the community.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 48–49, 2004.
© Springer-Verlag Berlin Heidelberg 2004

We would also like to acknowledge the organizers of the OTM 2004 conferences for the support and encouragement they extend to this workshop. The close cooperation between GADA 2004 and the OTM 2004 organization allows us to contribute to the growth of this research community.

August 2004

Pilar Herrero
Maria S. Perez
Victor Robles
Milena Radenkovic
(GADA 2004 Workshop Co-chairs)

Simulation of Mars Impact Cratering on a Grid Enviroment*

Eduardo Huedo[1], Alain Lepinette[1], Rubén S. Montero[2], Ignacio M. Llorente[2,1], and Luis Vázquez[3,1]

[1] Laboratorio de Computación Avanzada, Simulación y Aplicaciones Telemáticas, Centro de Astrobiología (CSIC-INTA), Associated to NASA Astrobiology Institute, 28850 Torrejón de Ardoz, Spain.
[2] Departamento de Arquitectura de Computadores y Automática, Facultad de Informática, Universidad Complutense de Madrid, 28040 Madrid, Spain.
[3] Departamento de Matemática Aplicada, Facultad de Informática, Universidad Complutense de Madrid, 28040 Madrid, Spain.

Abstract. Marine-target impact cratering simulation plays an important role in the study of past martian seas. In order to develop profiles for future exploration missions and to understand the morphologies for future investigations, a large number of simulations have to be done. This paper presents some experimental results obtained with the execution of impact cratering simulations in a Globus-based Grid environment through the Grid *W*ay framework. Some performance metrics are presented to demonstrate the suitability of this platform for running High Throughput Computing applications.

1 Introduction

Impact cratering is an important geological process of special interest in Astrobiology that affects the surface of nearly all celestial bodies such as planets and satellites. The detailed morphologies of impact craters, which will not be described in this work, show many variations from small craters to craters with central peaks. Furthermore, a water layer at the target influences lithology and morphology of the resultant crater. That is the reason why marine-target impact cratering simulation plays an important role in studies which involve hypothetical martian seas [1].

In this study, we analyze the threshold diameter for cratering the seafloor of an hypothetical martian sea during the first steps of an impact. The numerical simulation of this process involves the execution of a high number of tasks, since the search space of input parameter values includes the projectile diameter, the water depth and the impactor velocity. Furthermore, the execution time of each task is not uniform because of the different numerical properties of each

* This research was supported by Ministerio de Ciencia y Tecnología, through the research grant TIC 2003-01321 and 2002-12422-E, and by Instituto Nacional de Técnica Aeroespacial "Esteban Terradas" (INTA) – Centro de Astrobiología.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 50–58, 2004.
© Springer-Verlag Berlin Heidelberg 2004

experimental configuration. Grid technology is a promising platform to execute this kind of applications, since it provides the end user with a performance much higher than that achievable on any single organization. However, the scheduling of each task on a Grid involves challenging issues due to the unpredictable and heterogeneous behavior of both the Grid and the numerical code. For this reason, the application will be executed on the Grid through the Grid *W*ay framework, that provides the adaptive and fault tolerance functionality required to harness Grid resources.

Results of these analysis can be used to develop a search criteria for future investigations, including techniques that will be used in future Mars exploration missions to detect buried geological structures using ground penetrating radar surveys, as the ones included in the ESA Mars Express and planned for NASA 2005 missions. The discovery of marine-target impact craters on Mars would also help to address the ongoing debate of whether large water bodies occupied the northern plains of Mars and help to constrain future paleoclimatic reconstructions [1].

We describe the target application and the Grid *W*ay framework in Sections 2 and 3, respectively. In Section 5, we present some experimental results, obtained in our research testbed, summed up in Section 4. Finally, we end with some conclusions.

2 Simulation of Impact Cratering

The impact process can be described as a transfer of energy process. The initial kinetic energy of the projectile does work on the target to create a hole –the crater– as well as heating the material of both projectile and target. We focus our attention in high-velocity impacts which can be separated into several stages dominated by a specific set of major physical and mechanical processes.

The main stages are contact and shock compression, transient cavity growth by crater material ejection, and finally, transient cavity modification. Impact cratering begins with a sufficient compression of target and projectile materials. The energy released by deceleration of the projectile results in the formation of shock waves and its propagation away from the impact point. The projectile's initial kinetic energy redistributes into kinetic and internal energy of all colliding material. The internal energy heats both the projectile and target and, for strong enough shock waves, this may result in melting and vaporization of material near the impact zone.

To describe the impact process we solve equations of motion for compressible media using a hydrocode. The standard set of equations of motion expresses 3 basic law: mass, momentum, and energy conservation. It must be combined with the equations of state (EOS), a system of relationships which allow us to describe the thermodynamic state for materials of interest. In its basic form, an EOS should define what is the pressure in the material at a given density and temperature. In an extended form, an EOS should define also the phase state of the material (melting, vapor, dissociation, ionization process) as well as all

useful derivatives of basic parameters and transport properties (sound speed, heat capacity, heat conductivity, etc.).

Numerical simulations use the Eulerian mode of SALE-B, a 2D hydrocode modified by Boris Ivanov based on SALES-2 [2]. The original hydrocode, Simplified Arbitrary Lagrangian-Eulerian (SALE), permits to study the fluid-dynamics of 2D viscous fluid flows at all speeds, from the incompressible limit to highly supersonic, with an implicit treatment of the pressure equation, and a mesh rezoning philosophy. The PDE solved are the Navier-Stokes equations. The fluid pressure is determined from an EOS and supplemented with an artificial viscous pressure for the computation of shock waves. SALES-2 can also model elastic and plastic deformation and tensile failure.

The code is based on finite difference approximations to the differential equations of motion. The algorithm used in the code is separated into four sections: problem set-up, cycle initialization, Lagrangian computation, and cycle completion.

3 The Grid*W*ay Framework

The Globus toolkit [3] supports the submission of applications to remote hosts by providing resource discovery, resource monitoring, resource allocation, and job control services. However, the user is responsible for manually performing all the submission stages in order to achieve any functionality: selection, preparation, submission, monitoring, migration and termination [4,5]. Hence, the development of applications for the Grid continues requiring a high level of expertise due to its complex nature. Moreover, Grid resources are also difficult to efficiently harness due to their heterogeneous and dynamic nature. In a previous work [6], we have presented a new Globus experimental framework that allows an easier and more efficient execution of jobs on a dynamic Grid environment in a "submit and forget" fashion. The Grid *W*ay framework provides resource selection, job scheduling, reliable job execution, and automatic job migration to allow a robust and efficient execution of jobs in dynamic and heterogeneous Grid environments based on the Globus toolkit.

Grid *W*ay achieves an efficient execution of Parameter Sweep Applications (PSA) in Globus-based Grids by combining: *adaptive scheduling, adaptive execution,* and *reuse of common files* [7]. In fact, one of the main characteristics of the Grid *W*ay framework is the combination of adaptive techniques for both the scheduling and execution [8] of Grid jobs:

- *Adaptive scheduling:* Reliable schedules can only be issued considering the dynamic characteristics of the available Grid resources [9,10,11]. In general, adaptive scheduling can consider factors such as availability, performance, load or proximity, which must be properly scaled according to the application needs and preferences. Grid *W*ay periodically gathers information from the Grid to adaptively schedule pending tasks according to the application demands and Grid resource status [7]. Grid *W*ay periodically gathers information from the Grid and from the running or completed jobs to adaptively

schedule pending tasks according to the application demands and Grid resource status [7].

- *Adaptive execution:* In order to obtain a reasonable degree of both application performance and fault tolerance, a job must be able to migrate among the Grid resources adapting itself to events dynamically generated by both the Grid and the running application [12,13,14]. Grid *W*ay evaluates each *rescheduling event* to decide if a migration is feasible and worthwhile [6]. Some reasons, like job cancellation or resource failure, make Grid *W*ay immediately start a *migration* process. Other reasons, like "better" resource discovery, make Grid *W*ay start a *migration* process only if the new selected resource presents a higher enough rank. In this case, the time to finalize and the migration cost are also considered [6,15].
- *Reuse of common files:* Efficient execution of PSAs can only be achieved by re-using shared files between tasks [11,16]. This is specially important not only to reduce the file transfer overhead, but also to prevent the saturation of the file server where these files are stored, which can occur in large-scale PSAs. Reuse of common files between tasks simultaneously submitted to the same resource is achieved by storing some files declared as *shared* in the Globus GASS cache [7].

In the case of adaptive execution, the following rescheduling events, which can lead to a job migration, are considered in Grid *W*ay [6,8]:

- Grid-initiated rescheduling events:
  - "Better" resource discovery [15].
  - Job cancellation or suspension.
  - Remote resource or network failure.
- Application-initiated rescheduling events:
  - Performance degradation.
  - Change in the application demands.

In this work, we do not take advantage of all the Grid *W*ay features for adaptive execution, since they are not supported by the application. In order to fully support adaptive execution, the application must provide a set of restart files to resume execution from a previously saved checkpoint. Moreover, the application could optionally provide a performance profile to detect performance degradations in terms of application intrinsic metrics, and it could also dynamically change its requirement and ranking expressions to guide its own scheduling process [8]. Therefore, we only consider adaptive execution to provide fault tolerance by restarting the execution from the beginning.

4 The Research Testbed

Table 1 shows the characteristics of the machines in the research testbed, based on Globus Toolkit 2.4 [3]. The testbed joins resources from 5 sites, all of them connected by RedIRIS, the Spanish research and education network. The geographical distribution of sites is shown on Figure 1. This organization results in

Table 1. Characteristics of the machines in the research testbed.

Name	Site	Nodes	Model	Speed	Memory	OS	Job mgr.
hydrus	DACYA-UCM	1	Intel P4	2.5GHz	512MB	Linux 2.4	fork
cygnus	DACYA-UCM	1	Intel P4	2.5GHz	512MB	Linux 2.4	fork
cepheus	DACYA-UCM	1	Intel PIII	600MHz	256MB	Linux 2.4	fork
aquila	DACYA-UCM	1	Intel PIII	700MHz	128MB	Linux 2.4	fork
babieca	LCASAT-CAB	5	Alpha DS10	450MHz	256MB	Linux 2.2	PBS
platon	REDIRIS	2	Intel PIII	1.4GHz	512MB	Linux 2.4	fork
heraclito	REDIRIS	1	Intel Celeron	700MHz	256MB	Linux 2.4	fork
ramses	DSIC-UPV	5	Intel PIII	900MHz	512MB	Linux 2.4	PBS
khafre	CEPBA-UPC	4	Intel PIII	700MHz	512MB	Linux 2.4	fork

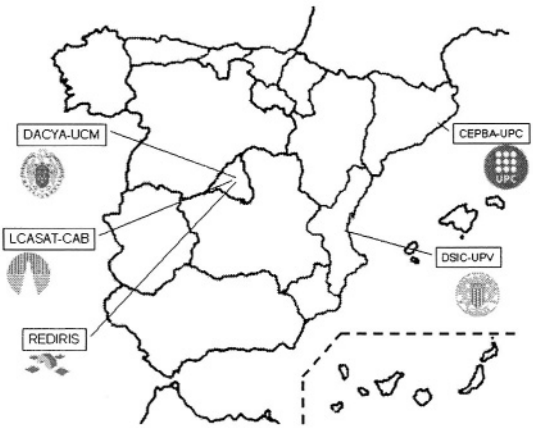


Fig. 1. Geographical distribution of sites in Spain.

a highly heterogeneous testbed, since it presents several architectures, processor speeds, job managers, network links... In the following experiments, cepheus is used as client, and it holds all the input files and receives the simulation results.

5 Experimental Results

We deal in this study with vertical impacts, as they reduce to 2D problems using the radial symmetry. All simulations were conduced with spherical projectiles. We run a set of simulations with a low-resolution computational mesh in a Grid environment. The non-uniform computational mesh of the coarse simulations consists of 151 nodes in horizontal direction and 231 nodes in vertical direction and the total nodes describes half of the crater domain because of axial symmetry. The mesh size progressively increases outwards from the center with a 1.05 coefficient to have a larger spatial domain. The central cell region around the impact point where damage is greater, more extended than the crater area, is a

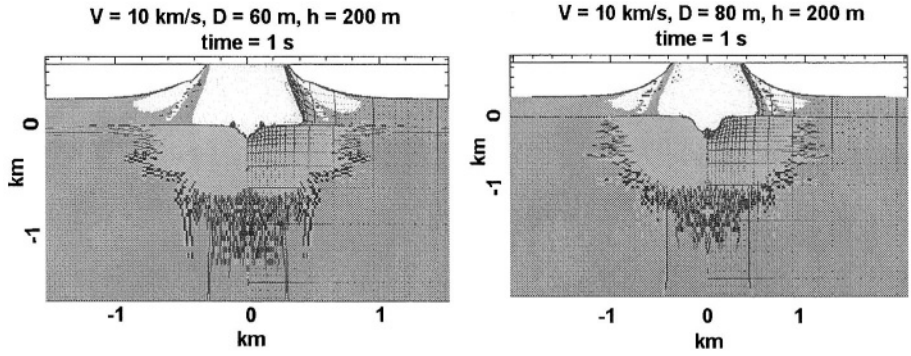


Fig. 2. Timeframes of the opening cavities at 1 second time using the 60 m impactor (left-hand chart), and the 80 m impactor (right-hand chart) with 200 m water depth and a velocity of 10 Km/s for the impactor.

regular mesh 80 nodes resolution in both x and y direction, and also describes half of the damaged zone. We use a resolution of 10 nodes to describe the radial projectile.

For a fixed water depth, we used 8 cases of projectile diameter in the range of 60 m to 1 Km, and 3 cases of impactor velocity: 10, 20 and 30 Km/s. Calculations were performed for 3 cases of water depth: 100, 200 and 400 m. Once fixed the projectile velocity and the water depth of the hypothetical ocean, we search to determine the range for the critical diameter of the projectile which can crater the seafloor. Therefore, in this study we have to compute 72 cases. The execution on a Grid environment permits to compute the 72 cases in a fast way and to obtain the diameter range of interest.

Figure 2 shows the timeframes of the opening cavities at 1 second time using the 60 and the 80 m impactor with 200 m water depth and a velocity of 10 Km/s for the impactor. The shape difference between the 60 m case and the 80 m case illustrates the water effect. Due to the water layer, in that case, the impactor diameter has to be larger than 80 m to crater the seafloor.

The execution time for each task is not uniform, since the convergence of the iterative algorithm strongly depends on input parameters, besides the differences produced by the heterogeneous testbed resources. Moreover, there is an additional variance generated by the changing resource load and availability. Therefore, adaptive scheduling is crucial for this application. Figure 3 shows the dynamic throughput, in terms of mean time per job, as the experiment evolves. The achieved throughput is 3.87 minutes per job, or likewise, 15.5 jobs per hour. Total experiment time was 4.64 hours. Table 2 shows the schedule performed by Grid *W*ay, in terms of number of jobs allocated to each resource and site. Most of the allocated jobs were successfully executed, but others failed and had to be dynamically rescheduled.

Given the results on Table 2, we can calculate the fault rate for each resource or site. It is clear that all resources (sites), but babieca and ramses (LCASAT-

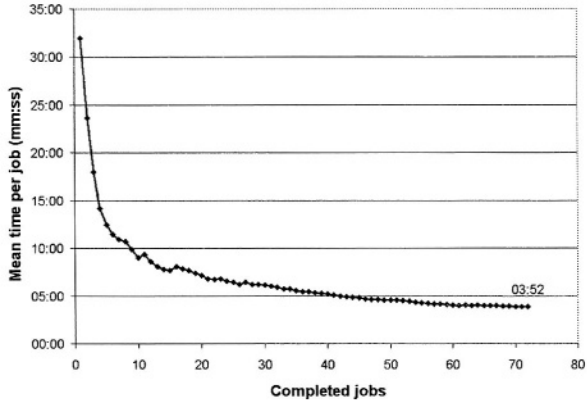


Fig. 3. Dynamic throughput, in terms of mean time per job.

Table 2. Schedule performed by Grid*W*ay, in terms of jobs allocated to each resource (left-hand table) and site (right-hand table).

Resource	Allocated	Done	Failed	Ratio
aquila	4	4	0	0%
babieca	24	18	6	25%
cygnus	7	7	0	0%
heraclito	2	2	0	0%
hydrus	8	8	0	0%
khafre	12	12	0	0%
platon	5	5	0	0%
ramses	29	16	13	45%
Total	91	72	19	21%

Site	Allocated	Done	Failed	Ratio
CEPBA-UPC	12	12	0	0%
DACYA-UCM	19	19	0	0%
DSIC-UPV	29	16	13	45%
LCASAT-CAB	24	18	6	25%
REDIRIS	7	7	0	0%
Total	91	72	19	21%

CAB and DSIC-UPV), have a fault rate of 0%, and the two failing resources (sites) have a fault rate of 25% and 45%, respectively, which supposes an overall fault rate of 21%. These failures are mainly due to a known Globus problem (bug id 950) related to NFS file systems and the PBS job manager. This problem is mitigated, but not avoided, on babieca, where a patch related to this problem was applied.

Figure 4 shows the achieved throughput, also in terms of mean time per job, by each site and by the whole testbed for the above schedule. In the right axis, the distributed or Grid speed-up (i. e. the performance gain obtained by each site) is also shown. We think that is a valuable metric for resource users and owners on each site to realize the benefits of Grid Computing, since results like this can help to curb their selfishness [5].

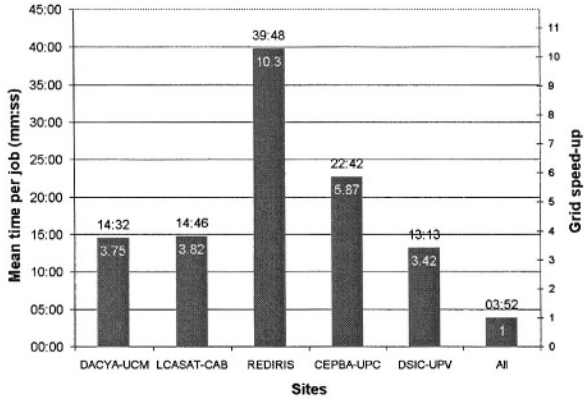


Fig. 4. Throughput, in terms of mean time per job (left-hand axis and values on top of columns) and Grid speed-up (right-hand axis and values inside columns), for each site and for the whole testbed (rightmost column, labelled as "All").

6 Conclusions

Studies about marine-target impact cratering will help to develop a search criteria for future investigations and exploration missions to Mars and, if they are successful in their hunt, they would also help to address the ongoing debate of whether large water bodies existed on Mars and, therefore, they would help to constrain future paleoclimatic reconstructions.

This kind of studies requires an huge amount of computing power that is not usually available in a single organization. Grid technology, and Globus in particular, allows the federation of resources from different organizations, with respect for each site autonomy, to help in the construction of virtual organizations. However, efficient execution on Grids involves challenging issues.

The Grid *W*ay framework provides an easier and more efficient execution of jobs on dynamic and heterogeneous Grids. That has been demonstrated by performance metrics like the Grid speed-up, which is a valuable metric for resource users and owners to realize the benefits of sharing resources over the Grid.

Acknowledgement. We would like to thank all the research centers that generously contribute resources to the experimental testbed. They are the European Center for Parallelism of Barcelona (CEPBA) in the Technical University of Catalonia (UPC), the Department of Computer Architecture and Automatics (DACyA) in the Complutense University of Madrid (UCM), the Department of Information Systems and Computation (DSIC) in the Polytechnic University of Valencia (UPV), the Laboratory of Advanced Computing, Simulation and Telematic Applications (LCASAT) in the Center for Astrobiology (CAB), and the Spanish Research and Education Network (RedIRIS). All of them are part of the Spanish Thematic Network on Grid Middleware.

We would like to also thank Jens Ormö and Jesús Martínez-Frías, in the Planetary Geology Laboratory at CAB, for their help in understanding the impact cratering simulations.

References

1. Ormö, J., Dohm, J.M., Ferris, J.C., Lepinette, A., Fairén, A.: Marine-Target Craters on Mars? An Assessment Study. Meteoritics & Planetary Science **39** (2004) 333–346
2. Gareth, S.C., Melosh, H.J.: SALES 2: A Multi-Material Extension to SALE Hydrocode with Improved Equation of State and Constitutive Model. Available at http://www.lpl.arizona.edu/~gareth/publications/sales_2 (2002)
3. Foster, I., Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit. Intl. J. Supercomputer Applications **11** (1997) 115–128
4. Schopf, J.M.: Ten Actions when Superscheduling. Technical Report GFD-I.4, Scheduling Working Group – The Global Grid Forum (2001)
5. Schopf, J.M., Nitzberg, B.: Grids: The Top Ten Questions. Scientific Programming, special issue on Grid Computing **10** (2002) 103–111
6. Huedo, E., Montero, R.S., Llorente, I.M.: A Framework for Adaptive Execution on Grids. Intl. J. Software – Practice and Experience (SPE) **34** (2004) 631–651
7. Huedo, E., Montero, R.S., Llorente, I.M.: Experiences on Adaptive Grid Scheduling of Parameter Sweep Applications. In: Proc. 12th Euromicro Conf. Parallel, Distributed and Network-based Processing (PDP2004), IEEE CS (2004) 28–33
8. Huedo, E., Montero, R.S., Llorente, I.M.: Adaptive Scheduling and Execution on Computational Grids. J. Supercomputing (2004) (in press).
9. Berman, F., et al.: Adaptive Computing on the Grid Using AppLeS. IEEE Trans. Parallel and Distributed Systems **14** (2003) 369–382
10. Buyya, R., et al.: Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computation Grid. In: Proc. 4th Intl. Conf. High Performance Computing in Asia-Pacific Region (HPC Asia). (2000)
11. Casanova, H., et al.: Heuristics for Scheduling Parameter Sweep Applications in Grid Environments. In: Proc. 9th Heterogeneous Computing Workshop. (2000)
12. Allen, G., et al.: The Cactus Worm: Experiments with Dynamic Resource Discovery and Allocation in a Grid Environment. Intl. J. High-Performance Computing Applications **15** (2001)
13. Lanfermann, G., et al.: Nomadic Migration: A New Tool for Dynamic Grid Computing. In: Proc. 10th Symp. High Performance Distributed Computing (HPDC10). (2001)
14. Vadhiyar, S., Dongarra, J.: A Performance Oriented Migration Framework for the Grid. In: Proc. 3rd Intl. Symp. Cluster Computing and the Grid (CCGrid). (2003)
15. Montero, R.S., Huedo, E., Llorente, I.M.: Grid Resource Selection for Opportunistic Job Migration. In: Proc. 9th Intl. Conf. Parallel and Distributed Computing (Euro-Par 2003). Volume 2790 of LNCS. (2003) 366–373
16. Giersch, A., Robert, Y., Vivien, F.: Scheduling Tasks Sharing Files on Heterogeneous Master-Slave Platforms. In: Proc. 12th Euromicro Conf. Parallel, Distributed and Network-based Processing (PDP 2004), IEEE CS (2004) 364–371

A Grid Environment for Simulation and Optimization and a First Implementation of a Biomedical Application

Silke Halstenberg, Karl-Uwe Stucky, and Wolfgang Süß

Forschungszentrum Karlsruhe GmbH
Institute for Applied Computer Science
P.O. Box 3640, 76021 Karlsruhe, Germany
{halsten, stucky, suess}@iai.fzk.de

Abstract. Simulation and optimization tools are used to design and develop complex systems e.g. in technical or medical fields and generally require a huge hardware and software resources. Therefore grid computing offers a chance to use distributed resources. This paper describes an overall concept of a heterogeneous grid environment for simulation and optimization problems. The basic idea of this concept is that the application is described as an instantiation of a workflow. The grid middleware, especially resource management, has to be adapted to this workflow-based concept. Another aspect of our concept is a modular and recursive structure of the grid environment to achieve flexibility and scalability. The first reference application of this concept is a biomedical system for planning refractive surgery of a human eye and for simulating in ophthalmologic research. It is called "Virtual Eye". Here, a prototype of this grid system shall be discussed.

1 Introduction

In the past few years, simulation and optimization have been very important parts of our work and subject of a number of research projects. Our applications were found in different domains like mechanics, fluidics, and optics and various fields like microsystems technology and medical systems. To simulate and optimize systems from these areas, vast hardware and software resources are necessary. To increase the performance, a distributed environment has to be used. Further improvements can be reached by using grid computing services, as shown by several other grid developments, for example Geodise [1] and Nimrod [2].

There are various types of grid applications with different resource requirements. For a grid application which requires high performance and high throughput, computing power is the most important resource. In data grids storage capacity is also essential. In [3] other resources, such as software, software agents or users are defined. Additional resources are grid services and information. In this respect, a grid is considered as a mechanism to integrate these different resources.

We developed a concept for a grid environment to simulate and optimize complex systems in technical or medical fields. As a reference application, we choose the biomedical system "Virtual Eye". It is suitable for grid application for the following different reasons:

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 59–67, 2004.
© Springer-Verlag Berlin Heidelberg 2004

- All relevant types of resources are required: computing power for simulations according to accuracy, storage capacity for storing models and results and software like simulation tools.
- Several parts of the system need different operating systems (Unix, Windows). Different databases will be used and single personal computers as well as clusters will be included. Thus it is a system with high heterogeneity.
- The system will be used carrying out numerous parameter studies which require high performance and high throughput.

Here, parameter studies shall be discussed primarily. They are relevant to support the further development of the system "Virtual Eye", for example, to verify simulation models. Ophthalmologic research of surgical techniques is another field that benefits from parameter studies and, thus, from a grid implementation. For these purposes, the resources of computing power and software are the most relevant resources. Consequently, a computational grid providing these relevant resources is considered only. In the future, storage capacity will be the next important grid resource to be integrated in our grid environment.

2 Concept

Our concept of a heterogeneous grid environment for simulation and optimization is based on describing the grid task as an instantiated workflow, called *application job*. An application job consists of a workflow definition and the corresponding data. Thereby any user-defined structure of the workflow is allowed (for example parallelism, sequences, splitting or joining) [4].

Handling of application jobs requires a dedicated grid middleware. This grid middleware receives the application job from the application and analyzes and distributes parts of the application job in the grid. For this purpose, the grid middleware divides the application job into single *grid jobs* as described by the workflow. After processing these single grid jobs, the grid middleware composes the overall result from the intermediate results and sends it back to the application. Fig. 1 shows the principle of our grid environment for simulation and optimization.

Defining application jobs with workflows allows the description of parallelism and the usage of heterogeneous resources. Moreover, it is independent of a specific application. Generally, essential features of a grid middleware are resource management, data management, security and information services [5]. Concerning our grid environment concept, the first important feature discussed in this paper is resource management.

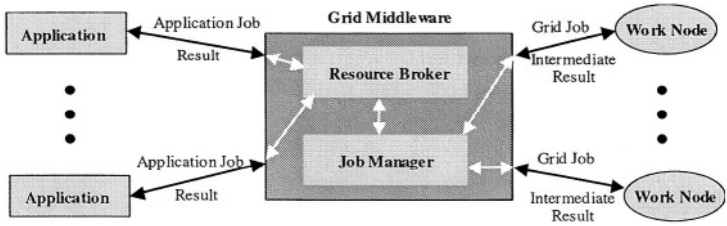


Fig. 1. Basic principle of our grid environment for simulation and optimization

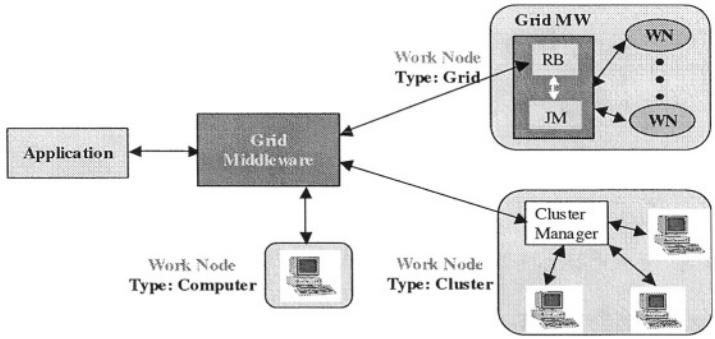


Fig. 2. Different types of work nodes

We divide resource management into two services, the resource broker and the job manager. The resource broker receives the application job that consists of a workflow definition and the corresponding data. It analyzes the workflow and divides the application job into single grid jobs. The resource broker acquires the capacities of the work nodes and schedules the distribution of the single grid jobs.
The job manager gets each single grid job and the assigned work nodes. It is responsible for the execution of the grid job on the assigned work node. After a grid job has been completed, the job manager transmits the intermediate result or maybe an occurring error to the resource broker.

Three different types of work nodes are distinguished, as shown in Fig. 2.

The simplest work node consists of only one computer which is supervised by the grid middleware. The other two types of work nodes possess an intelligent unit which manages and controls the corresponding computers or work nodes. One is a work node of the type cluster, where the intelligent unit is a cluster manager that controls the computers in a local area network. The other is a work node of the type grid. Such a grid work node has the same structure as the overall grid environment described in this paper. This allows for an arbitrary, scalable increase in the computing power in a recursive manner. For communication between the grid middleware and a work node, standardized interfaces are required.

3 Application

As a reference application of our grid environment concept, the biomedical system "Virtual Eye" is chosen. The "Virtual Eye" system has been developed by our institute and combines biomechanical and optical simulations. Following a brief description of the system, a first prototype of the "Virtual Eye" in a grid environment shall be discussed.

3.1 Virtual Eye

The "Virtual Eye" system supports patient-specific diagnosis and therapy planning in corneal surgery and ophthalmologic research [6,7,8]. On the basis of biomechanical

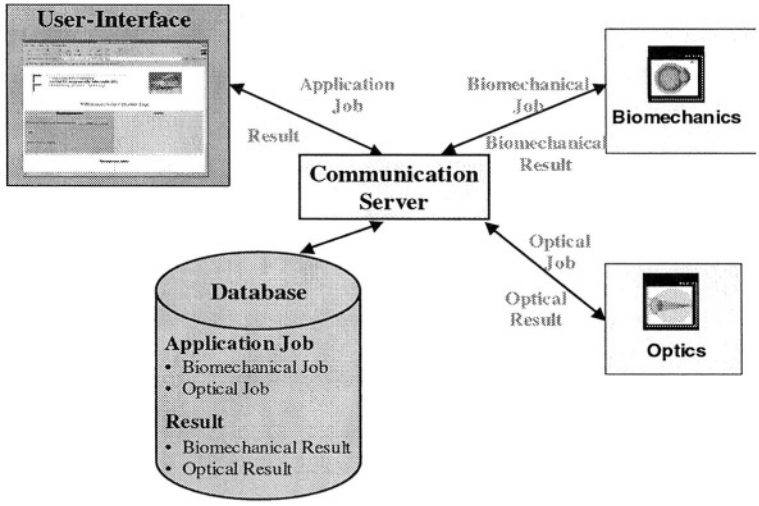


Fig. 3. Structure of the web based "Virtual Eye" system

All simulations of the human eye necessary for this purpose are very time-consuming and the required simulation tools are very expensive, complex tools. Consequently, the "Virtual Eye" system in its pre-grid implementation is developed as a distributed and web-based simulation service using the computer capacities existing at our research institution. Fig. 3 shows the system with its main components.

The end users of the virtual eye (e.g. hospitals) do not need any complex and expensive simulators because of the web interface of the system. Only a personal computer or a workstation with a connection to the internet is required. Another advantage of this approach is that the necessary resources can be shared among several users.

For simulating the biomechanical behavior the finite element method (FEM) is used. The FEM model describes the cornea of the human eye. Variations in geometry induced by mechanical modifications during corneal surgery (cuts, ablation) will be simulated using the FEM. These variations strongly depend on the intra-ocular pressure. For the simulations the simulator ANSYS[1] is used, which runs on Sun workstations with a Solaris operating system. Further enhancement of the "Virtual Eye" extends the eye model and surgery techniques to other parts of the eye (e.g. lens, bulbus).

Using the FEM results, the changed refraction of the eye can be calculated by simulating the optical behavior of the eye. Optical simulation is based on photometric calculations. For this purpose, rays are generated on the basis of an object pattern. These rays are propagated through the optical system "eye" and collected to form an image on the retina. Based on these results, visual acuity is determined. Optical simulations are accomplished with SOLSTIS[2] that runs under Windows on a PC. In the future, wave front optics will be integrated in the system.

---

[1] ANSYS is a trademark of ANSYS, Inc.
[2] SOLSTIS is a trademark of OPTIS

3.2 Grid Application "Virtual Eye"

The main task in the development of the grid system "Virtual Eye" is to substitute the communication server of the pre-grid version (see Fig. 3) by a grid middleware. The communication server of the web based system contains the functions also provided by services of the middleware in a grid. Whereas grid middleware contains standardized, application independent software for multiple purposes, the communication server consists of just the functions relevant for the "Virtual Eye" and is designed specifically for this application. The structure of the grid application is shown in Fig. 4.

Users of the grid application like ophthalmologists of a hospital can use the same web interface as in the current web based "Virtual Eye" system, but the underlying computing power, simulation and storage resources are distributed over different providers. Storage resources are important for hospitals storing patient specific data and simulation results and also for simulation providers storing simulation models. Computing resources and software resources (e.g. optical and biomechanical simulators) can also be distributed over the grid and can change dynamically depending on the availability of the resources. The grid middleware is responsible for distribution, execution, error handling, data storage and data mining.

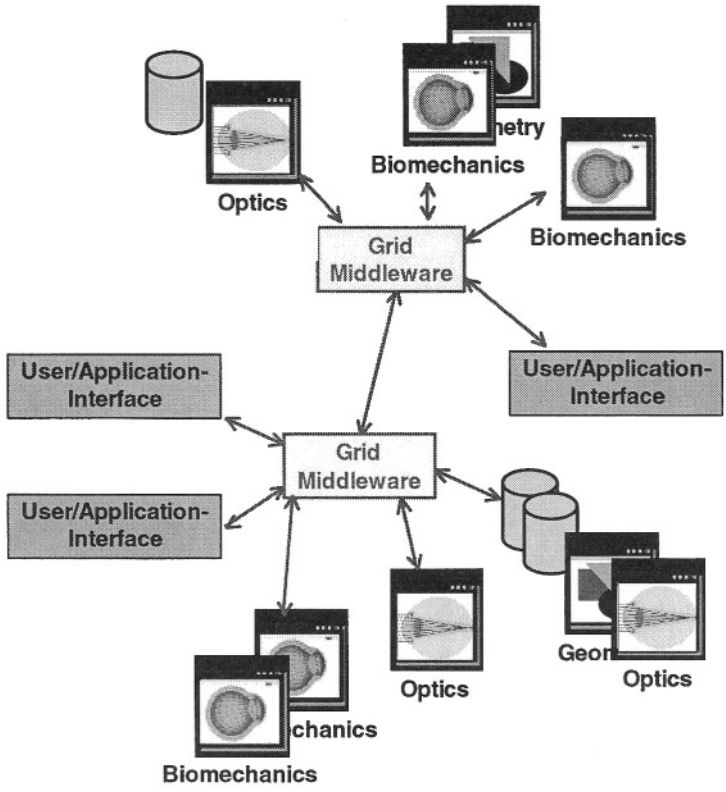


Fig. 4. Concept of the grid application "Virtual Eye"

The grid middleware gets an application job, generates the single grid jobs and distributes them to the different work nodes, e.g. single computers or another grid.

3.3 Prototype

Our grid environment concept is based on workflows. Each application job (see chapter 2) consists of a workflow specification and associated data. The global concept will allow for more complex workflows than those used in the "Virtual Eye" system. The workflows of the "Virtual Eye" system describe mainly sequences of simulations and mathematical calculations and are shown in Fig. 5.

By parsing the application job the resource broker determines the number of single grid jobs and the software required for each grid job. Based on this information, the resource broker assigns a work node and the associated data to each single grid job. The single grid job is then passed to the work node by the job manager.

As a concrete example of the approach, "Workflow 1" shall be described (see Fig. 5). This workflow consists of three parts, biomechanical simulation, the polynomial fitting step, and optical simulation. These three parts depend on each other. First, a biomechanical simulation using the FEM simulator ANSYS is required. Using Grid information services the resource broker identifies the properties of the work nodes, e.g. the installed software and free capacity. It selects an adequate work node for the

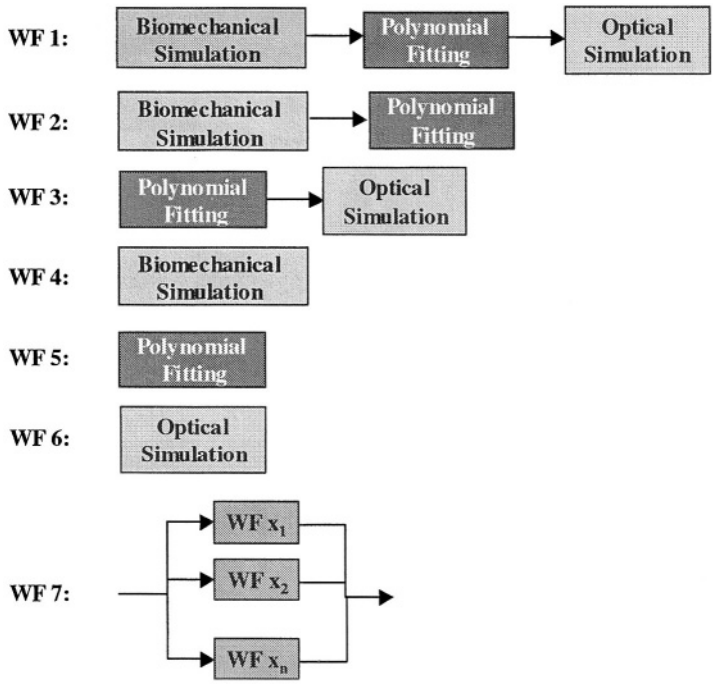


Fig. 5. Workflows of the system "Virtual Eye"

first grid job of the workflow. The job manager executes this grid job and returns the result to the resource broker. Based on this result, the second grid job of the workflow is generated. Again, a work node is selected for this grid job by the resource broker and the job is executed by the job manager. Respectively, the procedure is repeated for the third part of the workflow, the optical simulation. Upon completion of the total workflow, the resource broker sends the results back to the application.

Each workflow can be processed parallel to each other as shown in Fig. 5 ("Workflow 7"). However, the parallel processing depends on the availability of the resources. Parameter studies are described by various parallel workflows. Generally they cannot be scheduled on homogeneous cluster nodes because single grid jobs require software packages running on different operating systems.

3.4 Implementation

A first basic prototype was implemented using the Globus toolkit [9] as grid middleware and JavaCoG [10] as interface between our application and Globus. A snapshot of the user interface of our prototype is shown in Fig. 6.

Our first implementation contains two different types of work nodes: On the one hand work nodes of the type "Computer" and on the other hand work nodes of the type "Grid". We developed a first prototype of a grid middleware consisting of a basic resource broker and a job manager. The grid middleware and the user interface are implemented in Java. The main task of the resource broker in the current prototype is the creation of single grid jobs. The implemented job manager is responsible for the

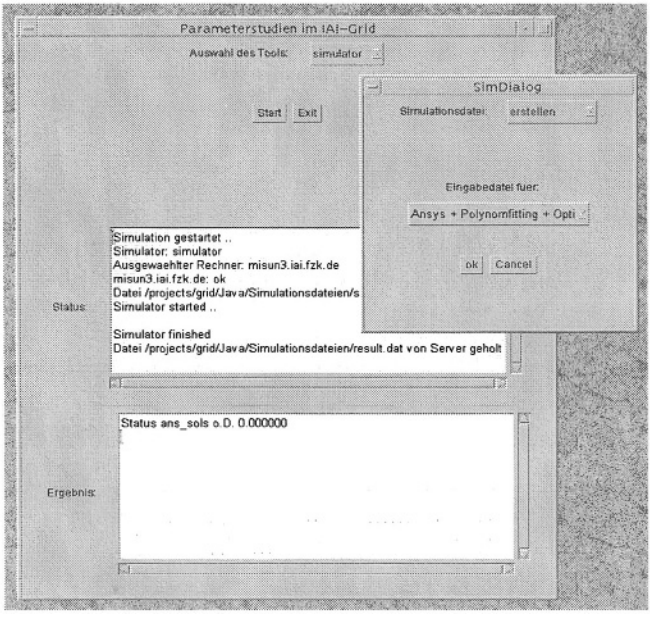


Fig. 6. Prototype of the "Virtual Eye" grid system

execution of the single grid jobs and the transfer of input data and the result between the grid middleware and the work node. In the future (see Chapter 4), the prototype will be extended by enhancing the resource broker and in particular by optimizing the planning and distribution of the grid jobs.

4 Conclusion and Future Work

This paper presents our concept of a grid environment for simulation and optimization. A basic idea of this concept is that a grid application is described as an instantiation of a workflow. A grid middleware tailored to this workflow-based concept is specified. The resource management of our first grid middleware consists of a resource broker and a job manager. Additionally, a modular and recursive structure of the grid environment is envisaged by our concept.

As a reference application of this concept, the biomedical system "Virtual Eye" is described. A first prototype of the "Virtual Eye" in a grid environment is discussed. Implementation of this reference application demonstrates the feasibility of our concept. In the future, this first prototype will be extended as follows:

First, the hardware and software resources will be increased by integrating further work nodes in our environment. For this, an improved resource broker will be required. To optimize resource management, it is intended to use the optimization tool HyGLEAM [11] that was developed by our institute. HyGLEAM is a hybrid of evolutionary algorithms and deterministic local search methods.

Secondly, the analyzing capability of the resource broker will have to be enhanced with increasing complexity of workflows.

Thirdly, an improved data management will be required with the further development of the reference application, the "Virtual Eye", and its intensive use by ophthalmologic institutions. Hence, the resource of storage capacity will gain importance.

Fourthly, our grid environment will have to be improved by a web service interface.

References

1. Xue, G., Fairman, M. J., Pound, G. E., Cox, S. J.: Implementation of a Grid Computation Toolkit for Design Optimisation with Matlab and Condor. In: Kosch, H., Böszörményi, L., Hellwagner, H. (eds.): Proceedings of Euro-Par 2003, Lecture Notes in Computer Science, Vol. 2790. Springer-Verlag, Berlin Heidelberg New York (2003) 357–365
2. Buyya, R., Abramson, D. and Giddy, J. Nimrod/G: An Architecture of a Resource Management and Scheduling System in a Global Computational Grid, HPC Asia 2000, Beijing, China, May 14-17, (2000) 283-289
3. Manola, F.: Characterizing Computer-Related Grid Concepts. Object Services and Consulting Inc., March 1999, http://www.objs.com/agility/tech-reports/9903-grid-report-fm/html
4. Fischer, L. (Ed.): Workflow Handbook 2001, ISBN 0-9703509-0-2
5. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organisations. Intern. Journal of Supercomputer Applications, Vol. 15(3), (2001) 200-222

6. Scherer, K. P., Eggert, H., Guth, H., Stiller, P: Application of simulation techniques in human eye corneal. In Proceedings of the 13th International Conference of the Society for Medical Innovation and Technology, SMIT & CARS, Berlin, June 27-30, (2001)
7. Scherer, K. P., Eggert, H., Guth, H., Halstenberg, S., Quinte, A.: Simulations and material modeling of special biological tissue. In Proceedings of the 22nd International Conference of Modeling, Identification and Control (MIC), Innsbruck (Austria), Acta Press, Anaheim, (2003), 421-425
8. Eggert, H., Guth, H., Hellmann, A., Scherer, K. P., Stiller, P., Süß, W.: An Internet Medical Application for the Ophthalmology. In Proceedings of the IASTED International Conference on Internet and Multimedia Systems and Application, Honolulu, Hawaii, August 13-16, 2001, Acta Press, ISBN 088986-299-0, 148-153
9. The Globus Alliance - Globus Toolkit http://www.globus.org
10. The Globus Alliance - Commodity Grid Kits: http://www-unix.globus.org/cog/
11. Jakob, W.: HyGLEAM – An Approach to Generally Applicable Hybridization of Evolutionary Algorithms. In: Merelo, J.J., et. al. (eds.): Proceedings of PPSN VII, Lecture Notes in Computer Science, Vol. 2439. Springer-Verlag, Berlin Heidelberg New York (2002) 527–536

Application of Grid Security to Personal Distributed Environments

Allan Tomlinson and Scarlet Schwiderski-Grosche

Mobile VCE Research Group, Information Security Group, Royal Holloway, University of London, Egham, Surrey, TW20 0EX, England
{Allan.Tomlinson,Scarlet.Schwiderski-Grosche}@rhul.ac.uk

Abstract. As Grid technology evolves it is becoming evident that the inclusion of mobile devices in a Grid environment will benefit both the Grid and mobile network communities. Many mobile devices, however, have limits on their computational resources that make it difficult to achieve this goal. The Personal Distributed Environment (PDE) is a new concept in mobile networks which, by constraining the problem, overcomes the resource limitation. This paper introduces the concept of the PDE and compares it with the Grid environment. The paper focuses on the specific problem of how to secure the mobile device and in particular how Grid security may be applied to this new environment. The PDE allows simple Grid security mechanisms to be adapted and applied within the PDE boundaries, while allowing a gateway device to support open standards as they emerge. In adopting a hybrid approach, a migration path to open standards is provided, as the computational resources of mobile devices increase.

1 Introduction

Since its conception, Grid computing has evolved to encompass an increasing number of applications. Much of this development has been attributed to the growth of the Internet, e-commerce and business-to-business applications [1]. As new technologies and business models emerge, it is likely that this trend will continue, and Grid technology will continue to find use in new areas.

Mobile communications represent one potential new area where Grid technology may be applied. In parallel with the evolution of Grid computing, mobile communications technology has developed to the stage where wireless networks are now becoming commonplace in the home environment. The potential for new applications based around the convergence of mobile devices and Grid technology is now becoming apparent to both the research and business communities [2,3].

An example of this awareness is evident in the Mobile VCE project. This is a collaborative project between industry and academia which, having a strong background in mobile communications, is seeking to integrate heterogeneous wireless technologies to provide a virtual network for personal communications. The concept, known as the Personal Distributed Environment (PDE) [4], is similar to the Virtual Organisation (VO), although there are some differences

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 68–78, 2004.
© Springer-Verlag Berlin Heidelberg 2004

between the two. In general, the objectives of the Grid are to allow resource sharing between members of a community [5], whereas the objective of the PDE is to allow a single user to access a wide range of services from a multitude of personal devices via a multitude of heterogeneous networks, all of which may be changing with time and location [4]. PDE services are often data-intensive, and range from the simple provision of a global address book and universal email service, to the delivery of high bandwidth multimedia content over heterogeneous networks to a continuously changing collection of devices. Although the ultimate objectives of the PDE may differ from the objectives of the Grid, the means of achieving these goals are very similar. This paper compares the PDE with the Grid environment, concentrating on the common aspects of the security architecture, and shows how Grid security may be applied to the PDE.

Section 2 describes the concept of the PDE, and the difference in objectives between the PDE and the VO. Similarities between the implementation of the two environments are presented in section 3, which begins by comparing the scenario described by Foster et al. in [6], with a PDE analogy. Continuing in a similar manner as [6], the characteristics of the two environments are then compared, and summarised in Table 1. This section then proceeds to identify a common set of security requirements and policies, which lead to a proposal, in section 4, of how Grid security may be applied to the PDE.

2 Background

2.1 Personal Distributed Environments

With the increasing deployment of wireless communications systems, a growing demand is envisaged from consumers who wish to have access to advanced services via their mobile devices. The potential therefore exists to deliver such services to personal mobile devices using a broad range of communications technologies including telecommunications, data, and broadcast networks. Consumers may be able to connect some of their devices to form Personal Area Networks (PANs), using technologies such as Bluetooth [7,8]. The same users may also have access to separate home or office networks, or even third party publicly accessible devices such as printers, displays, or sensors. Many of these networks will exist in independent security domains, for example the user's home network and a corporate network. Taking the personal networking concept one step further, to allow interaction between personal devices using disparate communications technologies is the idea behind the PDE [4]. The concept is illustrated in Fig. 1.

The heterogeneous nature is an important aspect of this environment. Another key feature is the dynamic nature of the PDE. The PDE changes not only with the services the user accesses, but with the user's location and local access technology. This access technology could be UMTS; wireless LAN; broadcast networks; or even wired gateways to a core network. Since the user is mobile, network access will make use of a dynamic combination of techniques depending on the technology that is available in the current location. The PDE concept en-

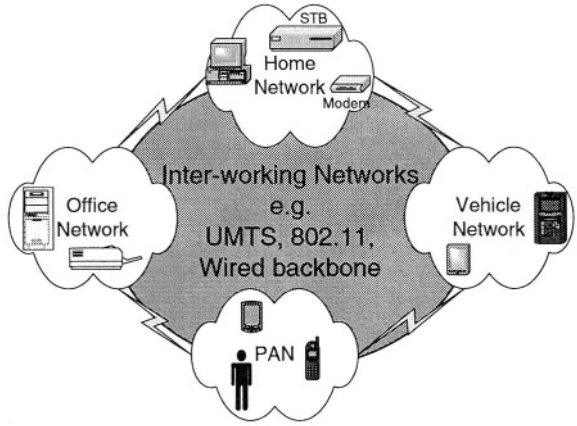


Fig. 1. Personal Distributed Environment

sures that the user's communications environment continuously adapts to meet their needs.

In addition to adapting the environment to network technology, individual mobile devices in the PDE may be designed to conserve energy. Consequently the topology of the PDE will change as these devices disappear and re-join at a later time, either on demand or as batteries are recharged.

Although the PDE involves the construction of virtual networks, the personal nature of the environment demands that it is designed to meet the needs of a single user rather than a Virtual Organisation. This user centric philosophy is fundamental to the concept of the PDE, with the emphasis on delivering services to individual consumers.

There are obvious differences between the objectives of the Grid environment and the objectives of the PDE. There are, however, many similarities in the conditions required to allow those objectives to be met. This suggests that closer investigation of Grid technology will be important in the definition of the PDE architecture. The following focuses on the security architecture, and compares the requirements of both environments. The comparison reveals how elements of Grid security may be adapted for use with the PDE and how the PDE may make use of this to interact with third party resources and services.

3 Comparison

3.1 Comparison of Scenarios

In their paper proposing a Grid security architecture, Foster et al. [6] describe a scenario to illustrate the security issues. The following compares this scenario with something more appropriate to the PDE.

Grid Scenario. The Grid scenario begins with a scientist who receives email regarding a new data set and subsequently launches an application to analyse the data. This application delivers code to the remote site where the data is stored and the scientist's code is executed at that site.

The application running on the remote site determines that it needs access to simulation software available at yet another site. It then contacts a broker to locate idle resources that can run the simulation.

The broker then initiates computations on two idle sites, which access parameters on yet another site.

PDE Analogy. A similar PDE scenario would begin with a consumer who receives email on a mobile device reporting new multimedia content available for delivery to his home receiver. The consumer then sends a codec application to his home receiver that allows received content to be formatted in a manner that the mobile device can decode[1]. The application is executed on the home receiver.

The codec determines that the multimedia content is protected by a Digital Rights Management (DRM) mechanism and that payment is required. It then contacts a broker to locate the most cost effective source of the multimedia content and most efficient delivery mechanism.

The broker then locates a network with idle bandwidth, and a service provider who has the content. These entities access other data such as cost and bandwidth parameters from yet another site, somewhere on the network.

3.2 Comparison of Characteristics

Although the objects in the two scenarios are completely different, the underlying architectures and the security issues have a lot in common.

Where the Grid is concerned with a large user population, the PDE is only concerned with single users. However, both environments share a dynamic population of security objects. In the case of the PDE, a greater diversity of security objects is expected, as various businesses compete to provide services.

Both the Grid and the PDE allow access to a large dynamic pool of resources. In the case of the Grid, this is often computational resources or data, available on machines in diverse organisations who have agreed to collaborate to achieve a common goal. In the PDE, the resources will be service providers who compete to provide the best service to the user. The PDE user will also have a large dynamic collection of devices on which received data may be accessed.

In a PDE, applications may initiate processes on other resources, and release them later. This is not likely to happen to the same extent as in the Grid, but

[1] It is unlikely that a mobile device will have the resources to decode all formats of content. Similarly, the source device may not have the encoder applications required by all possible receivers. The remote device, however, could send the appropriate encoder application to the source device, as described in this scenario.

Table 1. Comparison of Grid and PDE Characteristics

Grid	PDE
Large user population	Single user
Unicast and multicast	Unicast, multicast, and broadcast services
Mainly background processing	Some background processing
IP based	Heterogeneous, but IP at network layer
Computation intensive	Communications intensive
Large dynamic resource pool	
Many trust / security domains	

it will be required, for example to run a codec or a security application on a remote device to allow services to be received on a local device.

The processes involved in delivering a service may communicate by a variety of methods. In addition to unicast and multicast that the Grid environment expects, the PDE will also receive broadcast services. Furthermore, in a PDE, since the user is mobile, communication connections will be dynamically reconfigured at lower layers of the protocol stack.

Where service providers and heterogeneous networks are concerned, the number of legacy authentication and authorisation processes supported by the PDE is expected to be greater than encountered in a Grid environment.

Finally, although Grid users may have to manage a collection of credentials, users of a PDE accessing many services will probably have a much larger collection of credentials to manage, and may have some of these assigned dynamically by service providers.

The main differences between the Grid and PDE environments are the size of the user base and the difference in the types of applications that will run. Where the Grid is chiefly concerned with computations the PDE is more concerned with communications. The similarities are the dynamic and heterogeneous collection of security objects which give rise to the same fundamental security problems. This is summarised in Table 1.

3.3 Comparison of Security Requirements

It is clear from the comparison of characteristics that the PDE and the Grid environment are faced with similar fundamental security problems. This gives rise to a common sub-set of security requirements for both environments

There are general requirements that apply to most security architectures as well as the PDE and the Grid. These include provision of authentication, authorisation, integrity and confidentiality. However, there are some security requirements that are particularly germane to the PDE and the Grid environment. The Grid requirements have been described in [6] and [9] and are compared with the PDE in the following:

Interoperability with local security solutions: As with the Grid, the PDE consists of devices that operate in different security domains but which interact on a global level. Local security solutions will exist in each domain, but it is unlikely that they will be compatible with security solutions in other domains or at the global level. Since these local security solutions cannot be modified, both the Grid the PDE security architectures need to interoperate with existing security solutions. Neither architecture, therefore, can impose any specific security mechanisms on local domains.

Single Sign-On: Both the PDE and the Grid integrate pre-existing infrastructures, each of which may have its own particular authentication mechanisms in place. These may be, for example, Kerberos, public key based, or perhaps even customised legacy systems. The latter is particularly true for the PDE when service providers are considered, such as those already delivering broadcast content. In both the Grid and the PDE, users need to authenticate to many different devices, networks, and services or resources. In the commercial environment, it is also likely that the user may even act in different roles depending on the resource being accessed. For example, as a mobile user accessing a telecommunications network; a student accessing a campus network; or a customer accessing an online bookshop. So, both the Grid and the PDE require a single sign-on solution, allowing the user to authenticate only once in order to access resources in all environments.

Protection, Revocation, and Renewal of Credentials: In both environments, security credentials are required at various layers of the OSI model. At the link layer, these credentials depend on the corresponding wired (e.g. Ethernet) or, especially in the case of the PDE, wireless (e.g. Bluetooth or IEEE 802.11) technology. The user centric nature of the PDE and the diversity of technologies used by personal devices mean that managing these credentials is perhaps more of a challenge than with the Grid. At the network layer, it is reasonable to assume that both environments will use IP and IPsec, and at the transport layer SSL/TLS is assumed. In addition to the management of credentials at those layers, user credentials are required in both environments at a global level, above the transport layer, but below the application layer (where the user services run). All these credentials in both the Grid and the PDE require protection, and mechanisms to manage their revocation and renewal. A further challenge, particularly for the PDE, is that depending on the technology, the end points of the security associations may differ, for example hop to hop at the link layer and host to host at the network layer.

Delegation: The Grid security architecture requires delegation of authority to processes acting on the user's behalf. Similarly, the PDE may delegate authority to agents acting on the user's behalf. These agents search for resources and negotiate cost before presenting options to the PDE user. To do this, the agents require authority to access networks, devices and services on behalf of the user.

Uniform Credentials, Certification Infrastructure, and Cryptographic Algorithms: Although one of the initial requirements proposed for

Grid security, the development of Web Services security may diminish the importance of this requirement to future Grid networks. In the PDE however, the user has control over their own personal devices, and since some of these devices may have limited computational resources, this remains a useful requirement. While different security mechanisms exist in the different sub-networks of a PDE, uniform mechanisms are required at the global PDE level. These mechanisms unify the existing solutions of the heterogeneous and dynamic environment.

3.4 Comparison of Security Policies

The common requirements described above, are fundamental to both the PDE and the Grid security architecture. These requirements led to the definition of a security policy for the Grid which applies to a large extent to the PDE.

Multiple Trust Domains: The first policy statement defined in [6] is that the Grid environment consists of multiple trust domains. This applies to the PDE although perhaps to a lesser degree. While the user may control devices within the PDE to some extent, a number of devices will exist on different networks with different security policies. For example, a user may have files stored on a device on a corporate or campus network which is a trust domain in its own right distinct from the user's mobile or home network. Within a single trust domain, only local security policies apply. Neither the Grid nor the PDE impose any additional policies within a local domain. Furthermore, since both Grid and PDE environments maintain local trust domains alongside the global Grid or PDE domain, both environments require a mapping between global and local subjects.

Mutual Authentication: The policy of mutual authentication between entities in different trust domains applies to both the Grid and the PDE. The commercial nature of the PDE, when compared to the generally more collaborative nature of the Grid, however, presents a potentially more hostile operating environment. In this respect, mutual authentication is particularly important to the PDE. In both environments an authenticated global subject, mapped to a local subject is assumed to be equivalent to being locally authenticated.

One distinction between the PDE and the Grid arises where global authentication is concerned. In [6], the user logs on to the Grid via a user proxy process running on a physically accessible client. In the PDE, for reasons explained in section 4, this user proxy may not always be hosted on the particular device used to access the PDE. This places a higher security requirement on the global authentication mechanism for the PDE. Once authenticated to a local domain, the Grid policy of keeping all access control decisions in the hands of local system administrators also applies to the PDE.

Delegation: The policy of delegation, allowing programs or processes to act on behalf of a user is important in the Grid environment where sequences of computations may be executed on multiple resources and access to data stored in several locations may be required. In the PDE environment, a policy of allowing delegation is required for a different reason. In the PDE, a broker or agent may act on behalf of a user [10]. This agent may search for services required

by the user and subsequent means of delivering these services. Therefore a PDE agent may require access to service providers and network operators while presenting the user's credentials to determine the availability and cost of resources. Furthermore, devices such as displays or printers may be available for public use. Where such devices have to temporarily join the PDE, it may be beneficial to delegate some degree of authority to the new device. So, although the motivation may be different, both the Grid and the PDE have need of this policy.

Sharing Credentials: One policy proposed for the Grid that does not have a counterpart in the PDE is that of permitting multiple processes to share a single set of credentials. It is not expected that the PDE will require as many processes as the Grid to be operating in parallel on the same resource. Using unique credentials for each process is not expected to have the same impact on scalability in the PDE.

4 Application of Grid Security

The emergence of the Open Grid Services Architecture (OGSA) [1] clearly indicates a convergence between Grid and Web Services architectures [11]. The benefits of collaboration between networks of mobile devices and the Grid are also becoming apparent [2,3]. However, many mobile devices, such as those forming the PDE, will be low power communications devices with limited computational resources. This potential limitation presents a challenge in the adoption of OGSA or Web Services standards. It is not clear that all devices will have the computational resources available to support these standards. In considering this challenge, the preceding comparison indicates a possible solution, at least for the PDE.

Although the PDE and the Grid may have quite distinct objectives, the two environments share many similarities as has been demonstrated in section 3. The differences in the two environments, where they exist, may now be exploited to simplify the security architecture for the PDE. For example, unlike the Grid, the PDE will be based on devices owned and managed by a single user. Therefore, when compared with the Grid, the construction of tables to map a global identity and credentials to local counterparts, as suggested in [6], is a relatively simple task. On the other hand, many of the devices forming the PDE will have limited computational resources. Although this places restrictions on the device's ability to support the OGSA security architecture, the ability to implement a simple security architecture based on [6] is entirely feasible.

Furthermore, not all devices in the PDE will be resource limited. There will be at least one gateway device that will have the resources available to support OGSA. Therefore, while maintaining a simple security architecture inside the PDE, the gateway will be able to support the OGSA security architecture to provide an interface to the outside world.

Within the PDE, the security architecture is expected to be similar to the architecture proposed by Foster et al. [6]. The main difference will be the location of the user proxy. Where the Grid user connected to the Grid from a desktop

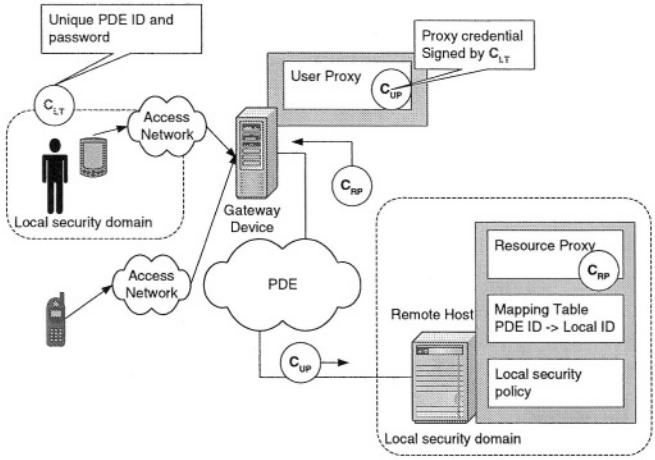


Fig. 2. Grid security architecture adapted to PDE

computing platform, the PDE user will connect to the PDE from one of many mobile devices. To pursue the Grid analogy would require all mobile devices to be capable of supporting the user proxy and all associated security protocols. This requirement may not be feasible for all mobile devices. The alternative proposed here is to implement the user proxy on the PDE gateway device.

Another challenge in defining security architecture of the PDE lies with the heterogeneous nature of communications at the link layer. Due to the diversity of technologies available to mobile devices, this layer, unlike the Grid, may have to be secured using legacy security mechanisms. However, once secured, a common mechanism may be used to secure all devices at the network layer. Fig. 2 illustrates how the security architecture proposed by Foster et al. [6] may be adapted for use within the PDE.

In this architecture the gateway device is a logical entity that supports the user proxy. This logical entity is available to all mobile devices using IP at the network layer although individual devices may have to secure access to the host via an access network using legacy security mechanisms at the link layer. Long term security credentials C_{LT} can then be used to create the user proxy. Short term user proxy credentials C_{UP} may then be created and exchanged with remote resource credentials C_{RP} to authenticate between different security domains as described in [6]. Once authenticated, the user's PDE ID may be mapped to an ID in the local domain by the mapping protocol also described in [6], and authorisation rights granted accordingly. The mapping protocol requires the user to log on separately to the global and local security domains and issue concurrent mapping requests from each domain. In a VO it may be impractical to physically access remote sites to achieve this, the personal nature of the PDE however, makes this approach quite feasible.

5 Conclusion

The resource constraints of mobile devices present a challenge to the application of Grid technology to mobile networks. The PDE represents a controlled subset of the latter, where Grid technology may be adapted to provide security.

Mobile VCE are working to apply the above ideas from Grid security to the PDE where the processing power of distributed resources is limited, but managed by a single user [12,13]. Work is also underway to explore how full OGSA security can be employed via a gateway device to enable the interaction between PDE and third parties. This architecture does not preclude the use of OGSA within the PDE for those devices that can support it. This hybrid architecture therefore allows migration to full OGSA based security if the computational resources of mobile devices increase in the future.

Acknowledgment. The work reported in this paper has formed part of the PDE area of the Core 3 Research programme of the Virtual Centre of Excellence in Mobile & Personal Communications, Mobile VCE, www.mobilevce.com, whose funding support, including that of EPSRC, is gratefully acknowledged. Fully detailed technical reports on this research are available to Industrial Members of Mobile VCE.

References

1. Foster, I., Kesselman, C., Nick, J.M., Tuecke, S.: The physiology of the grid - an open grid services architecture for distributed systems integration. Technical report, Global Grid Forum (2002)
2. Phan, T., Huang, L., Dulan, C.: Challenge: integrating mobile wireless devices into the computational grid. In: Proc. The Eighth Annual International Conference on Mobile Computing and Networking, (MOBICOM '02), Atlanta, ACM, ACM Press (2002) 271–278
3. Lam, K.Y., Zhao, X.B., Chung, S.L., Gu, M., Sun1, J.G.: Enhancing grid security infrastructure to support mobile computing nodes. In Chae, K., Yung, M., eds.: WISA '03. Number 2908 in LNCS, Springer-Verlag (2003) 42–54
4. Dunlop, J., Atkinson, R.C., Irvine, J., Pearce, D.: A personal distributed environment for future mobile systems. In: Proc. 12th IST Mobile and Wireless Communications Summit, Aviero, Portugal, IST (2003) 705–709
5. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the Grid: enabling scalable virtual organization. The International Journal of High Performance Computing Applications **15** (2001) 200–222
6. Foster, I., Kesselman, C., Tsudik, G., Tueke, S.: A security architecture for computational grids. In: Proceedings of the 5th Conference on Computer and Communications Security, pub:ACM (1998) 83–92
7. Bluetooth: Specification of the Bluetooth system. Technical Report v1.2, Bluetooth (2003)
8. Shepherd, R.: Bluetooth wireless technology in the home. Electronics and Communication Engineering Journal **13** (2001) 195–203

9. Nagaratnam, N., Janson, P., Nadalin, A., Siebenlist, F., Welch, V., Tuecke, S., Foster, I.: Security architecture for open grid services. Technical Report GWD-I (draft-ggf-ogsa-sec-arch-01), GGF OGSA Security Workgroup (2003)
10. Irvine, J.: Adam Smith Goes Mobile: Managing Services Beyond 3G with the Digital Marketplace (Invited Paper). In: European Wireless 2002, Florence, Italy, EUREL (2002)
11. Kreger, H.: Web Services Conceptual Architecture. Technical Report V1.0, IBM (2001)
12. Schwiderski-Grosche, S., Tomlinson, A., Irvine, J.M., Goo, S.K.: Security challenges in the Personal Distributed Environment. In: Proc. VTC Fall04, Los Angeles, USA (2004) to appear.
13. Schwiderski-Grosche, S., Tomlinson, A.: Initialising the personal distributed environment. unpublished (2004)

An Adaptive Cost Model for Distributed Query Optimization on the Grid

Yahya Slimani[1], Faiza Najjar[2], and Najla Mami[1]

[1] Dept. of Computer Science, Faculty of Sciences of Tunis, 1060 Tunis, Tunisia
yahya.slimani@fst.rnu.tn, mnajla@yahoo.com
[2] Computer Science and Engineering Department Southern Methodist University PO BOX 750122 Dallas TX 75275-0122 Tel. 214-768-3716 Fax 214-768-3085
faiza.najjar@engr.smu.edu

Abstract. Distributed query processing is fast becoming a reality. With the new emerging applications such as the grid applications, distributed data processing becomes a complex undertaking due to the changes coming from both underlying networks and the requirements of grid-enabled databases. Clearly, without considering the network characteristics and the heterogeneity, the solution quality for distributed data processing may degrade. In this paper, we propose an adaptive cost-based query optimization to meet the requirements while taking network topology into consideration.

Keywords: Grid computing, Beowulf clusters, Distributed databases, Distributed query optimization, Cost model.

1 Introduction

The vision of grid technologies where computers, storage and other computational resources are connected to yield the benefits from collaboration, mutual sharing and sophisticated interaction of autonomous and geographically dispersed resources, is becoming a reality. The term distributed query processing (DQP) can have slightly different implications depending on the specific context in which it is used. In Distributed Database Systems [9] where the data is distributed over logically interrelated databases in a computer network, query processing benefits from operating in a controlled environment. Recently, DQP has become popular due to performance gains obtained from grid architectures that offer a relay infrastructure for both data distribution and parallel query processing (independent, pipelined, and intra-operator). Communication is an important component of DDB processing since data needs to be transferred from one site to another. Since there is usually a set of alternative execution plan to represent the input query, it is essential to predict the cost of a given execution plan. The cost estimation must be efficient and accurate because optimization is only as good as its cost estimation. However, existing approaches proposed in literatures to estimate communication cost are very simple and idealized: it is only a function of the amount of data transmitted. This is obviously not enough

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 79–87, 2004.
© Springer-Verlag Berlin Heidelberg 2004

on unexpected changes in the performance of the communication networks. However, if the grid is to support a wider range of applications, both scientific and otherwise, then database integration into the grid will become important. For example, many applications in the life and many business applications are heavily dependent on databases. Furthermore, database management systems provide many facilities that are recognised as being important to grid environments. Recently, grid research as well as distributed database research tackled the problem of query estimation and optimization in such an environment [11]. This paper will examine the estimation of the communication cost for optimizing query processing in Beowulf cluster's. In this paper, we propose a generic cost model to optimize the query processing in a kind of grid architecture, e.g. beowulf clusters [1,2]. Two main features characterize this model:

- Adapativity: it is an active research area in distributed query processing [4]. It is important that the cost model can respond to changes in the environment by adjusting its behaviour for any size of beowulf cluster (single cluster, multi-clusters, and so on).
- Generality: our cost model is generic in the sense that it is not dependent on any particular kind of network and can support network heterogeneity.

The remainder of this paper is organized as follows. Section 2 presents an overview on query processing and optimization. Section 3 describes traditional cost models in distributed and parallel environments. Section 4 describes our cost model with emphasis in estimation of communication related to network heterogeneity. Section 5 illustrates our generic cost based optimization. Finally, section 6 concludes this paper and draws some future works.

2 Review of Relational Distributed Query Processing

Nowadays, distributed query processing is becoming both feasible and needed. It is feasible because of recent technological advances and it is needed because it is cost-effective. Given a declarative query, the query processor first devises a procedural plan and then executes the best plan to produce the query result. An execution plan specifies how precisely the query is to be executed. The optimizer's decisions are based on data properties such as cardinalities and predicate selectivities and an environmental conditions such as network speed and machine load. In this context, join is the most critical operation because data transmissions are necessary if the joining relations are not located at the same site [10]. The optimizer must also decide at which site each operation is to be executed. Among the many different strategies that have been proposed for distributed query optimization, those based on the reduction of the referenced relations by means of semi-joins have received considerable attention. Since this problem is NP-hard [13], heuristics were generally used.

2.1 Cost Model Background and Notations

The problem considered is the query processing in a distributed relational database system. We assume that a query is in the form of conjunctions of equi-join clauses. We define a join query graph, $G(V, E)$, as follows: V is the set of relations referenced by the query and E is the set of edges. An edge $(R_i, R_j) \in E$, if there exists a join predicated on some attribute of R_i and R_j.

In most previous work in distributed databases [14], it is assumed that the cost of executing a query can mainly be expressed in terms of the total amount of inter-site data transmission required.

The cost for executing a distributed query is modeled as the amount of data transmitted between sites (TC_{ij}). TC_{ij} is estimated by a linear function of the amount of data (say X) transmitted between the sites S_i and S_j.

$$TC_{ij}(X) = C_{ij} + D_{ij}X$$

where C_{ij} corresponds to the startup cost and D_{ij} is the speed of the transmission. Communication costs significantly depending on the locations of the sending and receiving sites. Consequently, the asymmetry feature between upload path and download path is also considered in our work in the network topology since TC_{ij} and TC_{ji} may differ from each other.

The quantitative property that summarize the instances of a database is its *database profile.* A database profile describes several aspects of relation, such as the number of tuples, the number of distinct values, the distribution of values, and the correlation between attributes.

We denote by $|S|$ the cardinality of a relation S. Let w_A be the width of an attribute A and w_{R_i} be the width of a tuple in R_i. The size of the total amount of data in R_i can then be denoted by $||R_i|| = w_{R_i}|R_i|$. For notational simplicity, we use $|A|$ to denote the cardinality of the effective domain of the attribute A, i.e. the number of distinct values of A in the whole database. $R_i(A)$ denotes the set of distinct values for the attribute A appearing in R_i. A selectivity model is used to predict the reduction effect of semi-joins. $\rho_{i,A}$ is called the selectivity of the semi-join $R_j \ltimes_A R_i$. $\rho_{i,A}$ is defined as a real number ranging from 0 to 1. According to [5], the selectivity factor ($\rho_{i,A}$) of a semi-join is estimated as $\frac{|R_i(A)|}{|A|}$.

Semi-joins can either greatly reduce communication cost, or in other common cases, increase this cost. Moreover, it is important to verify that a semi-join $R_j \ltimes_A R_i$ is *effectual,* i.e. if the cost incurred by this semi-join is less than the cost of the benefit which is computed in terms of avoided future transmission cost.

3 Cost Models Overview

Database cost models provide the foundation for query optimizers to derive an efficient execution plan. Such models consist of two parts: a logical and a physical component. The logical cost component depends only on the properties of the data stored in the database, the operators in the query, and the order in

which these operators are to be evaluated (as specified by the query execution plan). Hence, the logical cost component is independent of the (algorithm) and/or implementation used for each operator. Given the data volumes, the physical cost component is needed to discriminate the costs of the various algorithms and implementations of each operator. The query optimizer uses this information to choose the most suitable algorithm and/or implementation for each operator [3,6,12].

Relational query optimizers transform declarative queries into query plans, and dataflow programs that can be executed efficiently. To perform their combinatorial calculations, optimizers rely on simple statistical cost models to predict the performance of candidate (sub)programs. The cost model is an important component in cost-based optimization. Several models have been proposed for parallel and distributed database systems. In such systems, query processing requires four major resources: CPU, I/O, memory and a network for communication. Therefore, the cost function which is linear, computes a cost value for a query plan as follows:

$$Query\ processing\ cost = CPU\ cost + I/O\ cost + communication\ cost$$

A cost model should reflect all these costs. Many cost models focus on a specific parameter which is considered dominant.

In conclusion, we can say that the query optimizer's choice of an execution plan is not only dependent on the query, but strongly influenced by a large number of parameters describing the database and the hardware environment.

4 Generic Cost Model

In this section, we propose a new cost model to compute the communication cost needed to execute a query in a Beowulf architecture. Invariably, the complexity of optimization problem requires some simplifications and assumptions in the cost model. We assume that: (i) query optimization concerns heteregeneous environments such as multi-cluster's and grid arhitectures; (ii) heterogeneity of communications is the dominant factor in our model; (iii) without loss of generality, we focus only on the total communication cost, though it is easy to include other factors.

The cost of shipping X amount of data from site i to site j is defined as follows:

$$C_{ij}(X) = \alpha_{ij} X + \beta_i \tag{1}$$

where α_{ij} is the communication cost to transfer a unit of data from site i to site j and β_i is the startup cost. One common assumption used in several cost models is that the cost communication of one unit of data between every pair of sites, i and j, in a distributed system is constant, i.e. $C_{ij} = C_{ji} = C_i$. Note that this assumption is unrealistic for real life distributed systems. A simple example that

refutes this assumption is that the communication cost depends on the current load of the network connection.

Now, starting with the above formula, we propose a new cost model that takes into account the network heterogeneity existing in Beowulf cluster. We proceed by case analysis and we distinguish two cases that will be discussed in the subsequent sections.

4.1 Intra-cluster Communication

We consider two subcases: intra-site and inter-site communication.

Intra-site communication. Suppose two nodes i and j located in the same site p. In this case, the cost model is the same as the one defined by (1). This is easy to verify because the nodes of site p are connected by the same bus.

Inter-site communication. We consider a cluster u in which a node i, located on the site p, needs to transfer data to node j in site q. We denote by d the diameter of the graph connecting the different switches in the cluster. The communication cost to achieve this transfer, in the worst case, is defined as follows:

$$C^{u}_{p_i q_j}(d, X) = C_{p_i p_c}(X) + C_{p_c q_c}(d, X) + C_{q_c q_j}(X) \tag{2}$$

where:

1. $C_{p_i p_c}(X)$ denotes the transmission cost between node i and the switch p_c which links the site p to the other sites in the Beowulf cluster.
2. $C_{p_c q_c}(d, X)$ is the transmission cost between two switches (in our case p_c and q_c). It is expressed as $d(\alpha_2 \times X + \beta_2)$. In this expression, α_2 refers to the network bandwidth and β_2 represents the initialization time and latency among the switches.
3. $C_{q_c q_j}(X)$ denotes the transmission cost between switch q_c and node j.

4.2 Inter-cluster Communication

In the case of Beowulf cluster's, each cluster has a *gateway* node with two network interfaces: one interface that lies the gateway node to switches within the cluster, and a second interface that links this cluster with another in the whole architecture. Hence, the transmission cost between two clusters u and v is defined by the following formula:

$$C^{uv}_{p_i q_j}(d, X) = C^{u}_{p_i g_u}(d, X) + C_{g_u g_v}(X) + C^{v}_{g_v q_j}(d, X) \tag{3}$$

where:

1. $C_{g_u g_v}(X) = \sum_{i=1..k}(\alpha_i X + \beta_i)$. This parameter represents the communication cost between two clusters, through their respective gateway nodes.

2. $C^{u}_{p_i g_u}(d, X)$ denotes the transmission cost between a node i on site p and the gateway node in cluster u.
3. $C^{v}_{g_v q_j}(d, X)$ denotes the transmission cost between a node j on site q and the gateway node in cluster v.

5 Cost Based Optimization Heuristic

Given two successive steps of a sequence of semi-joins, $R_i \ltimes R_j$, $R_i \ltimes R_k$ (or $R_j \ltimes R_i$, $R_k \ltimes R_i$), these steps may be performed in any order with no effect on the cost of the semi-join sequence. The problem of LOS (Local Optimization of Semi-joins) heuristic is to compute for each relation R_i, the most effectual set of semi-joins to reduce R_i [8].

We say that R_i is locally fully reduced if $\{j \ / \ R_i \ltimes_A R_j$ is feasible[1]$\}$. We denote by $RD_i = \{j / R_i \ltimes_A R_j$ is effectual$\}$, the set of index reducers of the relation R_i. Let $G = (V, E)$ be a join query graph with n nodes $(R_i)_{i=1,n}$. For each R_i to be reduced, and m candidate semi-joins $(R_i \ltimes R_{j1}, R_i \ltimes, \ldots, R_i \ltimes R_{jm})$, determine a subset of these semi-joins such that the total transmission cost, including the cost of applying semi-joins and that of transferring R_i to the processing site, is minimized. This problem can be transformed into an optimization problem termed *Sum Product Optimization* [8,14], in which the objective function is to be minimized. Thus, our objective is to find the set of the most locally effectual semi-joins (called applicable semi-joins), $AP_i \subseteq RD_i$, such that the total transmission cost (TC_i) of each relation R_i is minimized.

LOS heuristic has been explored by generating a 1-PSJ heuristic. 1-PSJ heuristic proceeds as follows: it repeatedly evaluates the benefit and cost of the feasible semi-join $(R_i \ltimes R_j)$. If the semi-join is effectual, it is inserted into a sorted list of index reducers RD_i $(j \in RD_i)$, until there is no effectual semi-join left. Furthermore, removing an effectual semi-join may minimize the extra costs incurred by semi-joins [8]. From RD_i, find the subset of applicable semi-joins $(AP_i \subseteq RD_i)$ such that the total transmission cost is minimized and all applicable semi-joins to R_i are executed in parallel [8]. Applicable semi-joins allow forward size reduction of relations and reduce the amount of data transmission to perform a join. Note that applicable semi-joins include only the most locally effectual semi-joins since all semi-joins to R_i are considered at the same time, local optimality (with respect to R_i) can be attained. Finally, in order to reduce each relation in the query, we apply a divide-and-conquer algorithm. The total cost (TC) is minimized if all transmission cost (TC_i) are minimized simultaneously. The details of this heuristic algorithm are given in [7]. 1-PSJ heuristic is polynomial; its complexity is $O(nm)$, where m is the number of joins between n relations. The heuristic has been tested on various random cyclic join graphs. Our results show that our heuristic leads to a good solution.

[1] $R_i \ltimes_A R_j$ is feasible if and only if $R_i \bowtie_A R_j$ is implied in the given query.

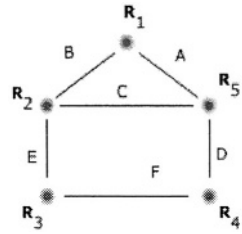


Fig. 1. An example of join query graph

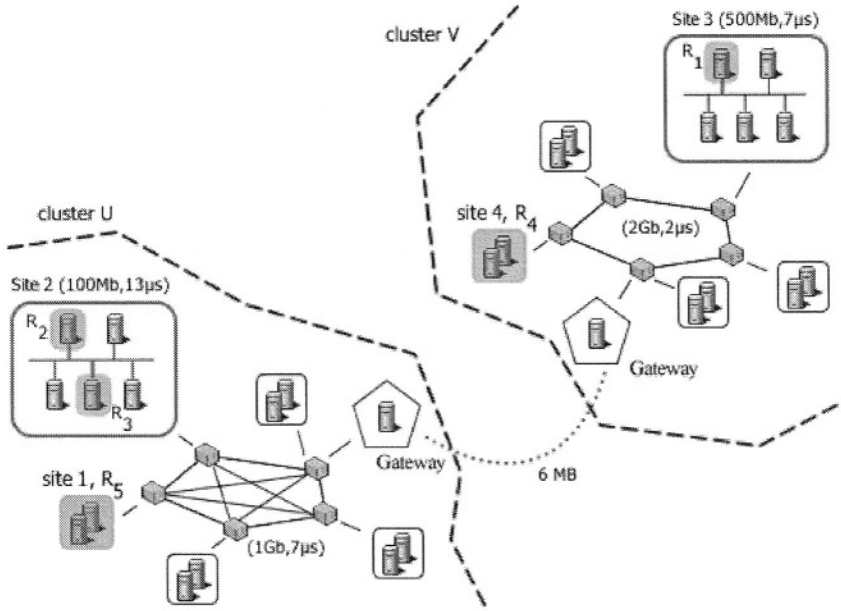


Fig. 2. Beowulf cluster's interconnection

5.1 Illustrative Example

To illustrate the generic cost model, proposed in this paper, we consider the query graph and database profiles, given in figure 1 and in tables 1 and 2 respectively. In this example, we use the grid of figure 2 with network characteristics described in the table 3.

After reducing locally the different relations involved in the join query, all resulting relations are sent to the final node where the joins will be performed. Thus, we compute the total transmission cost for different final nodes by summing the cost incurred by semi-joins and the cost of sending the reduced relations. It can be verified that the node containing R_5 is effectively the good final node.

Table 1. Profile 1

R_i	$\|R_i\|$	$\|\|R_i\|\|$
$R_1(A,B)$	1190	3570
$R_2(B,C,F)$	3440	17200
$R_3(E,F)$	1180	2360
$R4(D,E)$	3100	6200
$R_5(A,C,D)$	2152	12912

Table 2. Profile 2

Attribute X	A	B	C	D	E	F
Width of X (W_X)	2	1	3	1	1	1
Cardinality of X ($\|X\|$)	1200	1200	1200	800	1000	600

Table 3. Network characteristics

Network characteristics (bandwith,latency)	Cluster u	Cluster v
Intra-site	(100Mb,13 μ s)	(500Mb,7μ s)
Switch network	(1Gb,7 μ s)	(2Gb,2μ s)
Inter-cluster	6MB	

Table 4. Optimal set of semijoins reducers

R_i	AP_i	Communication cost (μs) to different nodes where R_i are located				
R_1	{2}	411.484	1448.589	1448.589	513.057	1448.589
R_2	{3,5}	4326.301	566.669	1317.813	4326.301	2306.210
R_3	{2}	807.859	195.373	47.332	807.859	403.975
R_4	$\emptyset$	228.306	2441.240	2441.240	0	2441.240
R_5	{1,4}	4410.706	2626.366	2626.366	4410.706	1088.544
Total Cost		9773.172	6711.567	7834.008	10057.924	**6600.014**

6 Conclusion

Distributed query processing is an appealing solution for expressing and efficiently evaluating distributed queries across Grid resources. In this paper, we presented a generic cost model for Beowulf cluster's architecture. Given a query, this model can assists a query optimizer in the choice of an efficient execution plan for this query. The cost model takes into account three communication parameters that can be heteregeneous: communication within a node of a single Beowulf cluster, communication between two nodes of a single Beowulf cluster, and communication between two clusters. The proposed model is generic in the sense where it is not depending on a specific topology. It can be used, for example, for ring and mesh networks that are the two common topologies used in cluster environments. Actually, our cost model is used by two query optimizers

running on two platforms: the first one is a MIMD architecture (IBM SP2 with 32 processors) and a i-cluster composed by 216 e-PC, with 1GB network linking the switches, an Ethernet 100 MB linking a group of nodes and a 2.5 GB network linking the switch connected to the front-end machine.

References

1. J. Dinquel. Network Architectures for Cluster Computing. Technical Report 572, CECS, California State University, April 2000.
2. I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure.* Morgan-Kaufman, 1998.
3. F. Baião G. Ruberg and M. Mattoso. Estimating costs for path expression evaluation in distributed object databases. In *Database and Expert Systems Applications, 13th International Conference, DEXA 2002, Aix-en-Provence, France, September 2-6, 2002, Proceedings,* volume 2453 of *Lecture Notes in Computer Science,* pages 351–360. Springer Verlag, 2002.
4. A. Gounaris, N. W. Paton, R. Sakellarion, and A.A. Fernades. Adaptive query processing and the grid: Opportunities and challenges. In *The 1st Int. Workshop on Grid and Peer-toPeer Computing Impacts on Large Scale Heterogeneous DDB. GLOBE'04,* 2004. invited paper.
5. C-H. Lee and M-S. Chen. Distributed query processing in the internet: Exploring relation replication and network characteristics. In *21st IEEE Int. Conf. on Distributed Computing Systems,* April 2001.
6. S. Manegold, P. A. Boncz, and M. L. Kersten. Generic Database Cost Models for Hierarchical Memory Systems. Technical Report INS-R0203, CWI, Amsterdam, The Netherlands, March 2002.
7. F. Najjar and Y. Slimani. The enhancement of semijoin strategies in distributed query optimization. In David J. Pritchard and J. Reeve, editors, *Euro-Par,* volume 1470 of *Lecture Notes in Computer Science,* pages 528–533. Springer, 1998.
8. F. Najjar and Y. Slimani. Extension of the one-shot semijoin strategy to minimize data transmission cost in distributed query processing. *Information Science,* 114:1–21, 1999.
9. M. T. Ozsu and P. Valduriez. *Principles of Distributed Database Systems.* Prentice-Hall, 2 edition edition, 1999.
10. C. Shahabi, L. Khan, D. McLeod, and V. Shah. Run-time optimizations of join queries for distributed databases over the internet.
11. J. Smith, A. Gounaris, P.Watson, N.W. Paton, Alvaro A.A. Fernandes, and R. Sakellariou. Distributed query processing on the grid. *Int. Jour, of High Performance Computing applications* (*IJHPCA*), *17(4), 2003.,* 17(4), 2003.
12. Y. Theodoridis, E. Stefanakis, and T.K. Sellis. Efficient cost models for spatial queries using r-trees. *Knowledge and Data Engineering,* 12(1):19–32, 2000.
13. C. Wang. The complexity of processing tree queries in distributed databases. In *Proc. 2nd IEEE Symp. Parallel and Distributed Processing,* pages 604–611, December 1990.
14. C. Wang and M-S. Chen. On the complexity of distributed query optimization. *IEEE Trans. on Knowledge and Data Engineering,* 8(4):650–662, 1996.

GDIS: A Service-Based Architecture for Data Integration on Grids

Carmela Comito and Domenico Talia

DEIS, University of Calabria,
Via P. Bucci 41 c,
87036 Rende, Italy
{ccomito, talia}@deis.unical.it
www.deis.unical.it

Abstract. Distributed data sources can be heterogeneous in their formats, schemas, quality, access mechanisms, ownership, access policies, and capabilities. We need models and techniques for managing different data resources in an integrated way. Data integration is the flexible and managed federation, analysis, and processing of data from different distributed sources. Data integration is becoming as important as data mining for exploiting the value of large and distributed data sets that today are available. Distributed processing infrastructures such as Grids and peer-to-peer networks can be used for data integration on geographically distributed sites. This paper presents a service-based architecture for data integration on Grids. The basic model is discussed and its implementation based on the OGSA Globus architecture is described.

1 Introduction

The Grid offers new opportunities and raises new data management challenges that arise from large scale, dynamic, autonomous, and distributed data sources. A Grid can include related data resources maintained in different syntaxes, managed by different software systems, and accessible through different protocols and interfaces. Due to this diversity in resources concerning data, one of the most demanding issue in managing data on Grids is reconciliation of data heterogeneity [5]. Therefore, in order to provide facilities for addressing requests over multiple heterogeneous data sources, it is necessary to provide data integration models and mechanisms.

Data integration is the flexible and managed federation, analysis, and processing of data from different distributed sources. In particular, the rise in availability of web-based data sources has led new challenges in data integration systems for obtaining decentralized, wide-scale sharing of data preserving semantics. These new needs in data integration systems are also felt in Grid settings. In a Grid it is not suitable to refer to a centralized structure for coordinating all the nodes because it can become a bottleneck and, most of all, it doesn't allow to benefit from the dynamic and distributed nature of Grid resources.

Standards for data management services on Grids are emerging [11]: in particular, the Global Grid Forum (GGF) *Database Access and Integration Services* (DAIS-WG)

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 88–98, 2004.
© Springer-Verlag Berlin Heidelberg 2004

Working Group produced a specification for a standard service-based interface to relational and XML databases in the OGSA setting, by introducing a collection of Grid data access services. The *Open Grid Services Architecture Data Access and Integration* (OGSA-DAI) [14] provides a reference implementation of the DAIS-WG recommendation through the standardization of data access interfaces.

However, till today none of those projects actually meets schema-integration issues necessary for establishing semantic connections among heterogeneous data sources. Based on the above mentioned reasons, we propose a decentralized service-based data integration architecture for Grid databases that we refer to as *Grid Data Integration System* (GDIS). Our proposal is motivated by the need to integrate different data models on distributed databases as well as to integrate data from various sources to obtain the required information. The GDIS infrastructure exploits the middleware provided by OGSA-DQP [13], OGSA-DAI, and Globus Toolkit 3 [12]. On top of these building blocks, we designed schema-integration services introducing new port types that extend both OGSA-DAI and OGSA-DQP ones.

The remainder of the paper is organized as follows. Section 2 reports on some current efforts concerning the management of databases on Grids. Section 3 presents a short analysis of data integration systems drawing attention on the *Peer Database Management System* (PDMS) [7] since we adopt the data integration formalism it introduced. Section 4 illustrates the proposal of a service-based Grid Data Integration System. That section describes the logical model of the proposed system and the architecture implementation by using OGSA Services. Finally, Section 5 outlines future works and draws some conclusion.

2 Current Efforts in Grid Databases

The current efforts toward the integration of databases in the Grid are focused on the development of service-based data management middleware [15]. Service-based middleware allows for developing virtual data services that provide transparent access to all databases through a common interface hiding implementation details. Grid database middleware services should provide generic database management functionality, including querying and updating over databases, and transactional coordination of accesses to one or more databases as well. Furthermore, issues concerning data integration formalisms and distributed query processing should also be supported.

The Grid community is adopting Web Services as the basis for Grid middleware through the definition of the *Open Grid Services Architecture* (OGSA) [3, 14]. OGSA introduces stateful, dynamic instantiable objects termed as Grid Services (GSs) [18]. The *Open Grid Services Infrastructure* (OGSI) is the base infrastructure underlying OGSA. It extends Web Services with interfaces for naming and reference of service instances, state management, notification, dynamic service creation. Each Grid Service has a set of service data elements (SDEs) that describe the service with both static and dynamic, instance-related information. Grid Services provide also operations that allow clients to inspect those SDEs and access data using appropriate operations. Moreover, the Grid community is working, on the wake of Grid Services, on the definition and standardisation of data services exploiting the OGSA mechanisms. An OGSA data service [4] is a Grid Service that implements one or

more of four base data interfaces to enable access to, and management of, data resources in a Grid environment.
The Globus Alliance and IBM recently proposed the Web Service Resource Framework (WSRF) [19], a set of specifications designed to merge Grid and Web technologies by enclosing the OGSI concepts in terms of current Web Service standards.

Service-based middleware systems implementing the emerging standards for Grid data access are already available such as OGSA-DAI and OGSA-DQP. OGSA-DAI builds upon OGSA high-level Grid data management utilities consisting in data access components that provide access to both relational and XML databases implementing standard data models. Thus OGSA-DAI extends Grid Services with new services and port types among which the *Grid Data Service* (GDS) port type that accepts an XML document which tells a GDS instance how to interact with a database. The core operation of the GDS port type is `perform` through which a request can be passed to the GDS. GDS instances are created by invoking the *Grid Data Service Factory* (GDSF), whereas the *Database Access and Integration Service Group Registry* (DAISGR) allows clients to search for GDSs and GDSFs.

OGSA-DQP is a service-based distributed query processor exposed as an OGSA Grid Service, aiming to process queries over distributed data sources obtained from multiple services on the Grid, including GDSs provided by OGSA-DAI. OGSA-DQP extends OGSA-DAI with two new services:

(1) a Grid Distributed Query Service (GDQS) that compiles, partitions and schedules distributed query execution plans over multiple execution nodes in the Grid. GDQS implements two port types from the OGSA, GS and NSnk, and two from OGSA-DAI, GDS and GDT. To these, it adds a Grid Distributed Query (GDQ) port type that allows source schemas to be imported.
(2) a Grid Query Evaluation Service (GQES). Each GQES instance is in charge of a partition of the query execution plan assigned to it by a GDQS. GQES instances implement the GS, GDS and GDT port types from the OGSA and OGSA-DAI.

3 Data Integration Systems

The goal of a data integration system is to combine heterogeneous data residing at different sites by providing a unified view of these data. In so doing, it frees users from locating the sources relevant to their query, interacting with each source independently, and manually combining data from the different sources. The two main approaches to the data integration problem are federated database management systems (FDBMSs) and traditional mediator/wrapper-based integration systems.

A federated database system (FDBMS) [16] is a collection of cooperating but autonomous component database systems (DBSs). The software that provides controlled and coordinated manipulation of the component DBSs is called a federated database management system (FDBMS). The DBMS of a component DBS, or component DBMS, can be a centralized or distributed DBMS or another FDBMS. The component DBMSs can differ in such aspects as data models, query languages, and transaction management capabilities.

Traditional data integration systems [9] are characterized by an architecture based on one or more mediated schemas and a set of sources. The sources contain the real data, while every mediated schema provides a reconciled, integrated, and virtual view of the underlying sources. Moreover, the system includes a set of source descriptions that provide semantic mappings between the relations in the source schemas and the relations in the mediated schemas [10].

Two basic approaches for modelling the relation between the sources and the mediated schemas have been proposed. The first approach, called *global-as-view* (GAV), requires that the mediated schema is expressed in terms of the data sources. The second approach, called *local-as-view* (LAV), requires the mediated schema to be specified independently from the sources, and the relationships between the mediated schema and the sources are established by defining every source as a view over the mediated schema. Finally, the GLAV formalism combines the advantages of both the GAV and LAV models.

The rise in availability of web-based data sources has led new challenges in data integration systems in order to obtain decentralized, wide-scale sharing of semantically-related data. Recently, several works on data management in peer-to-peer systems are moving along this direction [1, 2, 6, 7, 8]. All these systems focus on an integration approach not based on a global schema: each peer represents an autonomous information system, and data integration is achieved by establishing mappings among the various peers.

In particular, in [7] is presented the *Peer Data Management System* (PDMS), a P2P-based decentralized data management architecture supporting integration issues for sharing semantically enriched data among relational databases. Any peer in the PDMS system can contribute data, schema information, or mappings between schemas forming an arbitrary graph of interconnected schemas.

Mappings between data sources are realized by using the PPL language allowing for both GAV and LAV style mappings between peers. The mappings are expressed in the form of inclusion and equality of conjunctive queries and datalog rules over data sources. Two types of mappings have been defined in the PPL formalism: *peer mappings* that establish the correspondences between the schemas at different peers, and *storage descriptions* that relate the data stored at a peer to one or more peer schemas. The queries are considered under certain answer semantics.

The key issue of the PDMS system is the *reformulation algorithm* for PPL: when a query is posed over the schema of a peer, the system will utilize data from any peer that is transitively connected by semantic mappings, by chaining mappings, and reformulate the given query expanding and translating it into appropriate queries over semantically related peers. Every time the reformulation reaches a peer that stores data, the appropriate query is posed on that peer, and additional answers may be found.

4 The Grid Data Integration System (GDIS)

In this section, we describe a decentralized service-based data integration architecture for Grid databases that we refer to as *Grid Data Integration System* (GDIS). The main concern of such system is reconciliation of data source heterogeneity.

The design of the GDIS framework has been guided by the goal of developing a decentralized network of semantically related schemas that enables the formulation of queries over heterogeneous data sources. The system also allows distributed queries when the target data sources are located at different sites. In order to achieve this aim we expose data integration utilities as Grid Services that allow for combining or transforming data from multiple heterogeneous sources to obtain integrated or derived views of data.

As described in section 3, two general approaches to the problem of providing high-level data access and integration services over heterogeneous, autonomous, distributed data resources are the federation of database management systems (FDBMSs) and the use of mediator/wrapper middleware.

In the federation approach at every node there always is an autonomous DBMS. Resource allocation is static (e.g., the federation is a rather rigid configuration) and optimisation cannot take advantage of evolving circumstances in the execution environment (e.g., the freeing-up of extra computational resources).

The design of traditional mediator/wrapper integration systems must be done globally and the coordination of mediators has been done by a central administrator which is an obstacle to the exploitation of evolving characteristics of dynamic environments. As a consequence, data sources cannot change often and significantly, otherwise they may violate the mappings to the mediated schema.

Data integration in the Grid environment has to deal with unpredictable, highly dynamic data volumes provided by unpredictable membership of nodes that happen to be participating at any given time. Therefore, it is not suitable to refer to a centralized structure that coordinates the activity of all of the nodes in the system. Our belief is that a decentralized data management approach can effectively exploit the available Grid resources and their dynamic allocation. These are the main motivations that convinced us to adopt the data integration model of the PDMS system.

Peer-to-peer and Grids are distributed computing systems characterized both by heterogeneity, large scale, lack of central control, multiple autonomous administrative domains, possible unreliable components and dynamic configuration. These common aspects allow both systems to benefit one from the other. In fact, OGSA Grids can provide a suitable and reliable infrastructure for P2P applications freeing them from limitations of hierarchical architectures. At the same time P2P architectures address issues and problems common to several Grid applications [17]. This is the case of the PDMS architecture that can be profitably adopted in a Grid-based environment.

4.1 GDIS Logical Model and Abstract Architecture

Based on the above mentioned features, we designed the GDIS system as a set of Grid nodes each of which can accomplish one or more of the tasks typical of a data integration system. These tasks include: (i) *data providers* supplying data source with or without its schema, (ii) *mediators* providing only a schema to which other nodes' schemas may be mapped, (iii) *clients* formulating queries.

The proposed GDIS system offers a wrapper/mediator-based approach to integrate data sources: it adopts the PPL decentralized mediator approach to handle semantic heterogeneity over data sources, whereas syntactic heterogeneity is hidden behind ad-hoc wrappers. As a consequence of these features, some nodes in the system should supply appropriate services addressing the challenges characterizing the design of a

wrapper/mediator-based data integration system. These challenges concern the design of *query reformulation* strategies, the construction of *wrapper* programs, and the design of effective query processing techniques such as a *distributed query processor.*

Thus, we can further classify the nodes in the system on the basis of the functionality they achieve with regard to the above mentioned features, obtaining in such way the following categorization:

- *processing nodes* that supply distributed query processing capabilities aiming to compile, optimise, partition and schedule distributed query execution plans over multiple *execution nodes*; every execution node is in charge of a sub-query execution plan assigned to it by a processing node;
- *data integration nodes* that offer (i) a set of data integration utilities allowing to establish/update mappings, and (ii) the query reformulation algorithm introduced by the PPL integration formalism;
- *mapper nodes* that hold the global catalog containing all the mappings, both schema mappings and storage descriptions, established in the system;
- *wrapper nodes* allowing for access actual data sources.

It should be noted that any node can carry out all the mentioned services.

With respect to the PPL integration formalism, it is possible to identify the following correspondences: a node corresponds to a peer; nodes' schemas to peers' schemas; peer relations are the relations of a node's schema; stored relations are the relations of the actual data stored at a node; peer mappings are the mappings among nodes' schemas; storage descriptions are the mappings among relations stored at a node and one or more nodes' schemas.

Figure 1 shows the abstract architecture of the GDIS system. All the kinds of nodes in the system and typical interactions that take place among them are shown. As above mentioned, a node in the system can assume several different roles on the basis of the reachable functionality. In the described scenario a client node CN, formulates a query over the schema contributed by the mediator node MN and, then, it poses the query (interaction (*1*)) to the node IN holding a reformulator engine. The reformulator performs the PPL query reformulation algorithm that accepts as input the query and the mappings realized in the system and stored on the mapper node MaN (interaction (*2*)). After the reformulated query has been produced, the node IN invokes (interaction (*3*)) the distributed query processor (DQP) on the node PN. The DQP performs the so far produced reformulated query delegating and scheduling portions of the query plan over multiple execution nodes (interaction (*4*)). Every execution node (EN), handling a partition of the query execution plan assigned to it by the processing node, accesses actual data stored in the database. To this aim each node EN (interaction (*5*)) uses the wrapper made available by the node WN that will access data held in the database on node DN (interactions (*6-7*)) and return it (interaction (*8*)) to the node EN. After that, the sub-query results begin to propagate across the execution nodes (interaction (*9*)) until reaching the node PN that collects them into the query result and passes it to the querying node IN (interaction (*10*)). Finally, the node IN propagates the query answer (interaction (*11*)) to the client (node CN). Note that, the dotted arrow among the node DN and the node MN means that there are semantic mappings among the schemas exported by the two nodes.

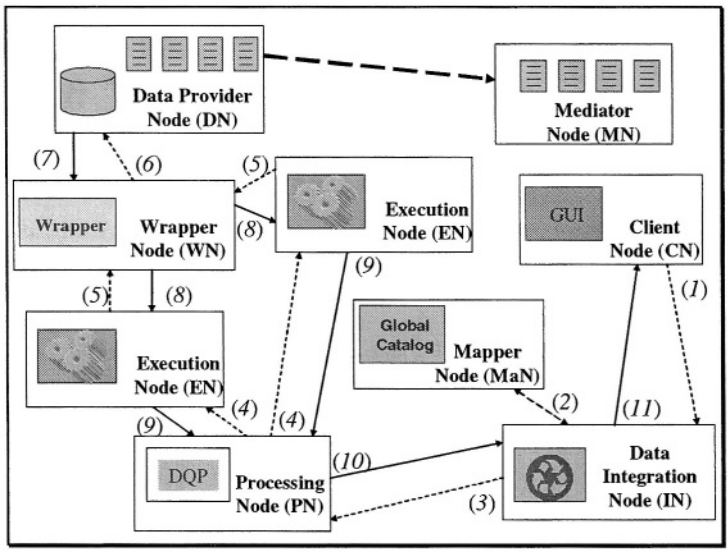


Fig. 1. GDIS Abstract Architecture

4.2 GDIS Architecture Implementation

According to the current trends in Grid computing, we designed our data integration system as a service-based distributed architecture where each node exposes all its resources as Grid Services (GSs) except data resources and data integration facility that are exposed as Grid Data Services (GDSs). In so doing, the GDIS system exploits the middleware provided by OGSA-DAI and Globus Toolkit 3.

The proposed framework is characterized by three core components: a *query reformulation engine,* a *wrapper module,* and a *distributed query processor.*
The implementation choices concerning these components are the following (see Figure 2):

- OGSA-DAI components are employed as wrappers over databases (wrapper nodes);
- OGSA-DQP is applied as distributed query processor. More precisely, execution nodes keep the OGSA-DQP Grid Query Evaluation Service (GQES), whereas processing nodes expose the Grid Distributed Query Service (GDQS) also made available by OGSA-DQP;
- As already explained, the PPL integration formalism is used to handle data heterogeneity. Therefore, the query reformulation engine implements the PPL query reformulation algorithm exposed, as the other integration facilities, through the port types of the proposed OGSA-GDI data integration service, as described in the following sub-section. These integration facilities are provided by data integration nodes.

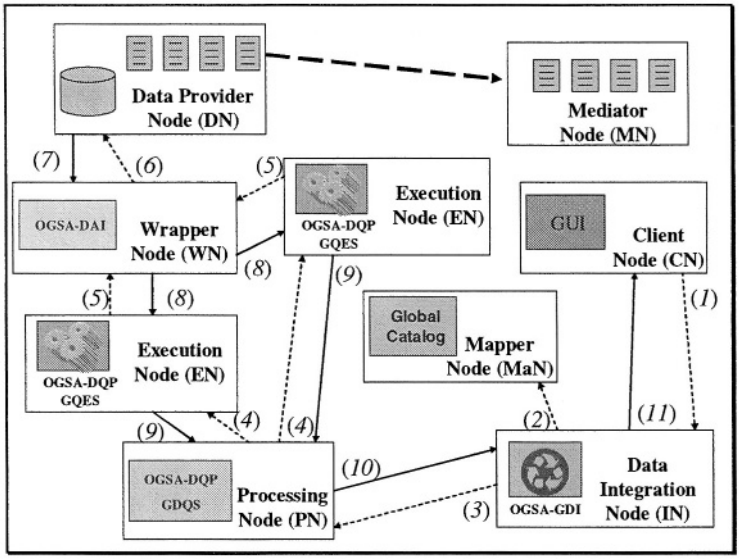


Fig. 2. GDIS Functional Architecture: implementation details

4.2.1 An OGSA-Based Grid Data Integration Service

The GDIS system introduces the OGSA-based *Grid Data Integration* (GDI) service that extends (see Figure 3) OGSA-DAI and OGSA-DQP port types with additional functionality to both enable users to specify semantic mappings among a set of data sources and to execute the above described PPL query rewriting algorithm.

Every GDI instance, other than introducing specific port types, can define some port types from OGSA-DAI and OGSA-DQP. As a Grid Service, GDI must define a Grid Service port type, and can define optional port types, such as Notification Sink (NSnk) and Notification Source (NSrc). Furthermore, GDI implements the port types GDS and GDT from OGSA-DAI, and the port type GDQ from OGSA-DQP. To these port types, it adds the following new ones:

- the *Import Mappings* (IM) port type, through which a client can import the mappings stored in the global catalog. The imported mappings will be exposed by the GDI as SDEs;
- the *Manual Mappings Composition* (MMC) port type, through which a client manually builds either or both schema mappings and storage descriptions;
- the *Automatic Mappings Composition* (AMC) port type, that will allow automatic schema matching utility. The input of this port type are represented by a set of schema mappings obtained by querying the opportune SDEs, and a target schema/stored relation. The output are possible mappings among the target relation and the schemas in the system;
- the output of the two previous port types will be the input of the *Mappings Update* (MU) port type, that allows for update mappings in all the global catalog replicas;

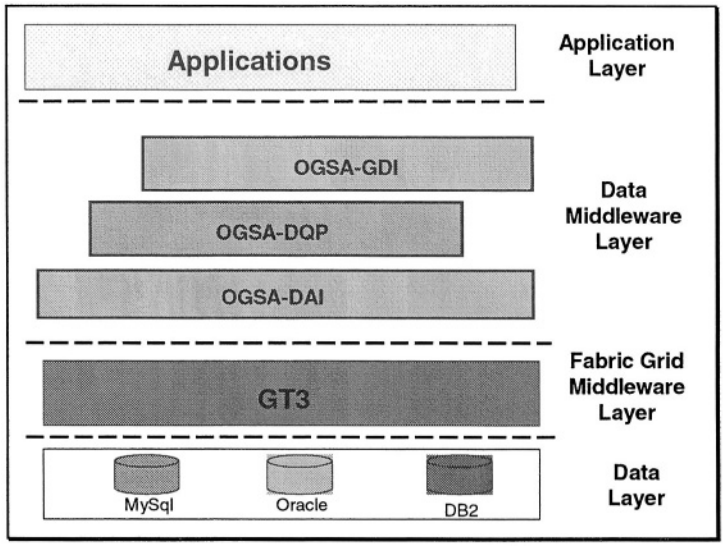


Fig. 3. GDI Internal Architecture

- the *Query Reformulation Algorithm* (QRA) port type, that performs the PPL query reformulation algorithm and receives as input a query, peer mappings and storage descriptions and produces as output a reformulated query that refers to stored relations only. The reformulated query will be the input of the GDS port type offered by the OGSA-DQP GDQS service.

Typical interactions in the system are those that take place when the two following activities are performed:

- joining of new resources in the system,
- submitting a query with consequent execution.

As above stated, GDI instances may implement OGSA-DAI and OGSA-DQP port types. Thus, besides interactions involving the specific GDI port types, in the system take place also OGSA-DAI and OGSA-DQP interactions.

Whether a new node joins the system or a pre-existing one wants to add a new resource (data source, schema, or mappings), it performs the following actions:

1. discover of the factory that can be used to create the desired actual service instance. Through the `FindServiceData` operation the client asks the GS port type of the DAISGR that will return to it the GSH of a GDIF (the factory of the GDI Grid Service);
2. create the GDI instance by using the `CreateService` operation of the Factory port type implemented by the GDIF service;
3. import logical and physical schemas of the participating data sources through the `importSchema` operation of the GDQ port type implemented by the GDI service;

4. import the mappings established in the system via the `importMappings` operation offered by the IM port type of the GDI service;
5. manually build schema mappings and storage descriptions through the `createMappings` operation of the MMC port type given by the GDI service;
6. update all the replicas of the mappings catalog by means of the `updateMappings` operation of the MU port type defined by the GDI service.

The other typical interaction in the GDIS system is query formulation and execution. In order to do that, the client has to perform all the first four above described steps. Then, eventually it has to further invoke the QRA port type in order to obtain the reformulated query. Below is briefly described a typical query request scenario.

Before writing a query a user may wish to access schemas of data sources. To this aim, the user can choose which database schemas to import by means of the GDQ port type of the GDI service. After that, the GDI exposes metadata about the imported resources as SDEs, so the user can examine the schemas of the imported databases by querying the `importedSchemas` SDEs. After importing the schemas of the participating data sources, the client can formulate the query and send it to a reformulator engine through the QRA port type of the GDI service. The QRA port type implements the `reformulation` operation that executes the PPL reformulation algorithm receiving as input the query, in the form of an XML document, and the mappings by asking the `importedMappings` SDEs. The produced reformulated query is then transmitted, in the form of an XML document, to the GDQS service via the `perform` operation of the GDS port type. The next interactions are the same as those of a typical OGSA-DQP execution and hence the partitions of the distributed query execution plan are scheduled for execution at different GQES instances created by the GDQS.

5 Conclusions

The massive amount of data sets that today is available from geographically distributed storage sources, is making data integration as important as data mining and knowledge discovery for exploiting the value of such large and distributed data repositories. Integration and correlation of large data sets demand significant advances in middleware for integrating data from diverse distributed sources. Distributed processing infrastructures such as Grids and peer-to-peer networks can be used for data integration on geographically distributed sites. That's why we designed GDIS, a Grid-based architecture for data integration. This paper presented GDIS as a general service-based architecture for providing data integration on Grids using a decentralized approach. The abstract architecture was discussed and its implementation based on the OGSA-based Globus framework has been described. A prototype of the GDIS system is under development by making use of the Globus Toolkit 3, OGSA-DAI and OGSA-DQP services.

References

1. P. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data Management for Peer-to-Peer Computing: A Vision. In Proc. of the 5th International Workshop on the Web and Databases, WebDB, Madison, Wisconsin, June 2002.
2. D. Calvanese, E. Damaggio, G. De Giacomo, M. Lenzerini, R. Rosati. Semantic Data Integration in P2P Systems. In Proc of Databases, Information Systems, and Peer-to-Peer Computing, 1st International Workshop, DBISP2P, Berlin Germany, September 7-8, 2003
3. I. Foster, C. Kesselman, J. Nick, S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. The Globus Project, www.globus.org. 2002.
4. I. Foster, S. Tuecke, J. Unger. OGSA Data Services, DAIS-WG Informational Draft, 9th Global Grid Forum, 14th August 2003
5. I. Foster, R. L. Grossman. Blueprint for the future of high-performance networking: Data integration in a bandwidth-rich world. Communications of the ACM, Volume 46, Issue 11, November 2003.
6. E. Franconi, G. M. Kuper, A. Lopatenko, L. Serafini. A Robust Logical and Computational Characterisation of Peer-to-Peer Database Systems. In Proc. of Databases, Information Systems, and Peer-to-Peer Computing, First International Workshop, DBISP2P, Berlin Germany, September 7-8, 2003
7. A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema Mediation in Peer Data Management Systems. In Proc. of the 19th IEEE Int. Conf. on Data Engineering (ICDE 2003), 2003.
8. A. Kementsietsidis, M. Arenas, and R. J. Miller. Mapping data in peer-to-peer systems: Semantics and algorithmic issues. In Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD'03), June 2003.
9. M. Lenzerini. Data Integration: A Theoretical Perspective. In Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS), pages 233–246. ACM Press, 2002.
10. A. Y. Levy, A. Rajaraman, and J. J. Ordille, Querying heterogeneous information sources using source descriptions, inVLDB '96, 1996, pp. 251-262.
11. S. Malaika, A. Eisenberg, J. Melton. Standards for Databases on the Grid. ACM SIGMOD Record, Vol. 32, No. 3, September 2003.
12. The Globus Project. The Globus Toolkit 3, Towards the Open Grid Services Architecture. http://www.globus.org/ogsa/
13. M. N. Alpdemir, A. Mukherjee, N. W. Paton, P. Watson, A. Fernandes, A. Gounaris, and J. Smith. OGSA-DQP: A service-based distributed query processor for the Grid. In Proc. of UK e-Science All Hands Meeting Nottingham. EPSRC, 2-4 September 2003.
14. Global Grid Forum, Open Grid Services Architecture - Data Access and Integration, http ://www .ogsa-dai. org.uk
15. N. Paton, M. Atkinson, V. Dialani, D. Pearson, T. Storey and P. Watson. Database Access and Integration Services on the Grid. UK e-Science Programme Technical Report Series Number UKeS-2002-03, National e-Science Centre, UK.
16. A.Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys, 22(3): 183–236, 1990.
17. D. Talia, P. Trunfio. Toward a Synergy Between P2P and Grids. IEEE Internet Computing, vol. 7, no. 4, pp. 94-96, 2003.
18. S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham and C. Kesselman. Grid Service Specification, Draft 5, http://www.gridforum.org/ogsi-wg, November 2002.
19. The WS-Resource Framework, http://www.globus.org/wsrf/

Managing Long Running Queries in Grid Environment

Ruslan Fomkin and Tore Risch

Uppsala Database Laboratory
Department of Information Technology
Uppsala University
P.O. Box 337, SE-751 05 Uppsala, Sweden
{Ruslan.Fomkin, Tore.Risch}@it.uu.se

Abstract. Exceptionally large amounts of both distributed data and computational resources are becoming available through the Grid. This will enable efficient exchange and processing of very large amounts of data combined with CPU intensive computations, as required by many scientific applications. We propose a customizable Grid-based query processor built on top of an established Grid infrastructure, NorduGrid. It allows users to submit queries wrapping user-defined long running operations that filter and transform distributed customized data. Limitations imposed by the used Grid infrastructure influence the architecture. For example, resource requirements have to be specified before Grid jobs are started and delays may occur based on the availability of required resources for a job. We are developing a fully functional prototype to investigate the viability of the approach and its applicability. Our first application area is Particle Physics where scientists analyze huge amount of data produced by a collider or simulators to identify particles.

1 Introduction

The Grid initiative provides an infrastructure for transparent distributed computations among widely distributed computer clusters. Exceptionally large amounts of both distributed data and computational resources are becoming available through Grid infrastructures. Many scientific applications within, e.g. bioinformatics, neuroscience, and space physics require large, scalable, high-performing, CPU demanding, and memory-intensive computations over large amounts of distributed data. They also require representation of not only the traditional tabular data managed by relational database management systems but also representation, modeling, and querying of numerical data, e.g. for vector and matrix algebra.

We are developing a new kind of data manager and query processor, Parallel Object Query System for Expensive Computations (POQSEC) that utilizes the operational Grid infrastructure NorduGrid [41] and leverages the Globus I/O library [17] from the Globus Toolkit [14] for processing declarative queries that access data from storage elements and wrapped external systems on the Grid. POQSEC has support for customizable data representations, allows user-defined long-running distributed computations in queries, and can access conventional relational databases. It processes application specific code on both local data and data distributed through the Grid.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 99–110, 2004.
© Springer-Verlag Berlin Heidelberg 2004

The aim of this system is to achieve high query performance by utilizing large main memories, disk, CPU power and other resources on many distributed nodes accessed through the NorduGrid, which is a distributed peer oriented Grid middleware system that does not rely on a central broker. Computer clusters accessed through NorduGrid have certain restriction with respect to resource allocation, communication, and process management that the POQSEC architecture must cope with and this influences its architecture.

The POQSEC data manager and query processor scales up by utilizing the Grid to transparently and dynamically incorporate new nodes and clusters for the combined processing of data and computations as the database and application demands grow. Conventional databases and file-based Grid storage elements are used as back-ends for data repositories. Extensible and object-oriented query processing and rewrite techniques are used to efficiently combine distributed data and computations in this environment.

Each node in POQSEC has DBMS facilities like storage management, customizable query processing, along with the possibility to call CPU demanding and long running user computations from queries. POQSEC uses an extensible and distributed query engine allowing user-defined data representations and query optimization algorithms that consider calling user-defined CPU intensive query functions. Query transformation rules transparently distribute customized data and their representations to nodes in the clusters. A given user query is rewritten into many distributed execution subplans given knowledge about available resources and observed execution costs. POQSEC nodes are automatically started or closed when needed based on resource demands for the current query workload.

The POQSEC prototype is being developed using as test case data and queries from Particle Physics where large amounts of data describing particles of events are produced by proton-proton collisions. The data is currently produced by simulations at NorduGrid [11] and will be produced by detectors of the Large Hadron Collider (LHC) [26] from 2006. The amount of data produced is huge (10 million events = 20-30 TB) and produced at a very high rate (10 million events over 2-3 weeks) [1]. The data therefore needs to be stored as distributed files in storage elements accessible though the Grid. Many scientists are querying these huge distributed data sets to identify interesting particles. The queries involve regular data comparisons and aggregation operators along with user defined filter operations in terms of C++ based libraries such as the ROOT library [7]. A single such analysis of a single dataset of size 1 million events often takes more than 1 hour to execute on a single machine. Thus, the processing needs to scale up to cover all distributed data produced by LHC [3]. Many scientists will simultaneously submit large amounts of queries and the system, therefore, must be able to manage queries off-line. Better performance can be achieved by identifying when results from running and old queries can be reused.

The remainder of this paper is organized as follows. Related works are discussed in Section 2. Section 3 describes the application from Particle Physics. Section 4 introduces the NorduGrid middleware and its restrictions. The architecture of our system is presented in Section 5. Section 6 gives the query execution strategy used in the system for the application specific queries. Section 7 concludes.

2 Related Work

Relational DBMSs allow the representation of very large and reliable disk-based databases with good performance for business application that store large amounts of structured tabular data. They are furthermore optimized for many short transactions over a few relatively small data items, while the new applications require also the incorporation of long running, expensive, and distributed computations using and producing many large objects and files.

Object-relational DBMSs [40] allow the definition of abstract data types in relational databases having *user defined functions* that are executed in the database server. Object-relational extensions are now part of the SQL-99 standard. However, computationally intensive applications still do not use DBMS technology since they require customized main-memory data structures for realistic performance. They also rely on program libraries written in conventional programming languages and operating on conventional files. Thus a query system utilizing scientific data needs also to be able to incorporate existing scientific libraries and data files into queries. Regular query processors assume cheap primitive operations while the scenarios considered here require optimization of queries with very expensive user-define functions.

In *distributed databases* [33] data distribution is transparent only for queries while data itself is manually distributed using placement conditions. *Parallel databases* are similar, but aim at letting the DBMS transparently and automatically distribute data. Parallel databases normally run on clusters of co-located computers with very fast intercommunication links. Query processing techniques for distributed and parallel databases are used by POQSEC for transparent access to distributed clusters.

Another modern development in the database area is to use large main memories to represent the database [20]. DBMS vendors are developing lightweight main memory relational databases [23] [32] [38] often interoperating with their DBMSs. For high performance, we are utilizing an embeddable main-memory DBMS [35] which includes an extensible and object-oriented query optimizer, i.e. it is an object-relational main-memory DBMS.

Mediators (multi-databases, federated databases) (e.g. [4] [19] [21] [27] [34] [43] [44]) are a modern database approach aiming at transparently integrating (combining) data from many different external data sources. In POQSEC we extend our previous work on integration of heterogeneous data sources [15] [22] [28] for access to Grid-managed external data repositories, relational databases, and to wrap CPU demanding computations.

POQSEC runs on top of existing Grid infrastructures, in particular the Swegrid computational infrastructure [37] and the NorduGrid resource management system [31] [41]. This puts certain restrictions on availability of computational resources and possibility to obtain them compared to systems such as distributed query processors Polar* [42] and LeSelect [5] that assume full control over all resources including computational resources, network management, etc.

One example of utilizing operational Grid infrastructures for transparent execution of jobs is the computational management agent Condor-G [8] [16]. It has its own resource broker to find computational Grid resources. It submits the jobs to a local *gate keeper* of the resource, e.g. the Globus Gatekeeper [14], using the Grid Resource Allocation and Management protocol (GRAM) [9]. In contrast, POQSEC reuses the

resource broker from the NorduGrid middleware [41]. It receives declarative queries unlike Condor-G, which receives traditional batch jobs. Based on the queries POQSEC then generates and submits jobs to NorduGrid for execution.

Rather than manually organizing long-running activities as in workflow systems (e.g. [10]), the long running computations are specified in POQSEC as declarative queries from which the query optimizer automatically generates execution plans consisting of several dependent tasks executed on the Grid and managed by POQSEC.

3 Application

In order to evaluate POQSEC we have chosen an application from Particle Physics. The aim of the application is to analyze data produced by the ATLAS Data Challenges [1] for containing charged Higgs bosons [6]. The analysis is long running because of the huge amount of input data.

The input data are called *events,* which describe collision events between elementary particles. Each event comprises sets of particles of various types such as *electrons, muons,* and sets of other particles called *jets* and sets of event parameters such as missing momentum in x and y directions *(PxMiss* and *PyMiss).* Each particle is described by its own set of parameters, e.g., the ID-number of the type of a particle *(Kf),* momentum in x, y, and z directions *(Px, Py, Pz),* and amount of energy *(Ee).* For example, an event might have one electron, two muons, four jets, and values of PxMiss and PyMiss for the event are -198.237 and 181.605, correspondingly. Examples of values for some particles of the event are presented in Table 1.

Table 1. Values of the particles from the example event

Particle name	Kf	Px	Py	Pz	Ee
Electron	-11	-96.3295	55.0114	-336.974	354.764
Muon	-13	-24.6514	-18.9128	-91.4735	96.6065
Muon	13	-6.23232	-10.2039	-38.1292	39.9601
Jet	2	197.085	8.94369	-165.14	257.281
Jet	2	141.008	-86.3656	-205.205	263.536
Jet	5	-35.3334	-45.2087	29.3406	64.4449
Jet	5	31.1251	-84.9691	-342.232	353.993

These event data are generated by the ATLAS software and stored in files managed by the ROOT library [36]. The size of generated files is from one to several GBytes [2].

The analysis of each event can be divided into several steps. Each step is a condition containing logical, arithmetic, and vector operations over event data. Events satisfying all conditions constitute the final analysis result. Partial results are also produced based on events that satisfy partial conditions.

An example of one of the steps is a *Z-veto cut.* An event satisfies this step if it does not have a pair of opposite charged *leptons,* where each lepton is either a muon or electron, or if it has a pair of opposite charged leptons the invariant mass of the pair is not close to the mass of a Z particle.

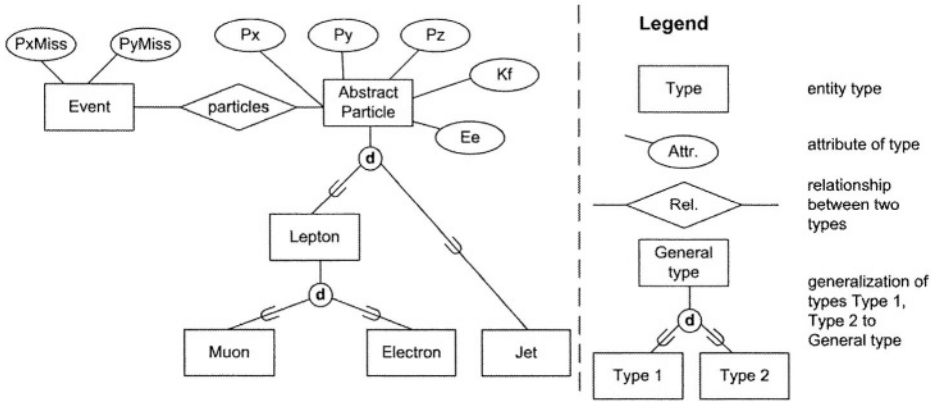


Fig. 1. Basic schema of the application data

This step can easily be expressed as a declarative query, for example, in the declarative language AmosQL [13] that is used in POQSEC. AmosQL has a functional object-relational data model and SQL-like syntax. We developed an object-oriented schema to present the application data in the system. Fig. 1 illustrates the main part of the schema in Enhanced Entity-Relationship notation [12]. Events are represented by type *Event,* which has three attributes, values *PxMiss,* and *PyMiss,* along with a relationship to particles that the event comprises. The particles are represented by three basic types *Electron, Muon,* and *Jet,* and two general types *Lepton* and *AbstractParticle.* The types *Electron, Muon,* and *Jet* correspond to electrons, muons, and jets, respectively. Electrons and muons often participate in analysis queries as one kind of particles, leptons, thus the types *Electron* and *Muon* are generalized by type Lepton. The types *Electron, Muon,* and *Jet* have the same attributes, *Kf, Px, Py, Pz,* and *Ee,* therefore, a general type *AbstractParticle* is introduced.

The Z-veto cut step is expressed as the following AmosQL query:

```
SELECT ev
FROM Event ev
WHERE NOTANY(oppositeLeptons(ev)) OR
      abs(invMass(oppositeLeptons(ev)) - zMass) >= minZMass);
```

The function `invMass` calculates the invariant mass of a pair of two given leptons, `zMass` is the mass of a Z particle, `minZMass` is range of closeness, and `oppositeLeptons` is a *derived* function defined as a query:

```
CREATE FUNCTION oppositeLeptons (Event ev) -> <Lepton, Lepton> AS
SELECT l1, l2
FROM Lepton l1, Lepton l2
WHERE l1 = leptons(ev) AND
      l2 = leptons(ev) AND
      Ee(l1) = -Ee(l2);
```

The function `leptons` returns a set of lepton pairs belonging to a given event, and `Ee` is a function which returns the value of the energy of a given lepton.

All steps of the analysis were expressed declaratively in AmosQL.

4 The NorduGrid Infrastructure

The first prototype of POQSEC is running on the Swedish computational Grid SweGrid [37] on top of the NorduGrid resource management system [31] [41]. The NorduGrid (NG) middleware (also called the Advanced Resource Connector, ARC) provides a uniform job submission mechanism so that a user can submit jobs to any or particular cluster(s) of the Grid. The Nordugrid middleware can be divided in three parts: the *NG client* which contains the *user interface* and the *resource broker,* the *Grid manager,* and the *information system.* The task of the user interface is to process job submissions, job status requests, and data transfer requests by a user. The user specifies jobs and their requirements, such as maximum CPU time, requirements for a cluster where the job is executed, requests to move output data from and to specific storage elements, number of sub-jobs for parallel tasks, requests for cluster nodes with IP connectivity, etc. The jobs are specified using the Extended Resource Specification Language (xRSL) [39]. The user interface includes resource broker functionality. It finds a suitable cluster for executing a submitted job using the information system that provides monitoring and discovery service [25]. On each cluster accessible through the NorduGrid a Grid manager [24] is installed. The task of the Grid manager is to submit a job received from a user interface to the *local batch system* of the cluster and to download input data and upload output data. The NorduGrid guarantees that the cluster fulfills the job requirements and that all input are available on the cluster when the job is started by the local batch system.

There are certain limitations on the usage of computational resources available throughout the NorduGrid, for example:

- The NorduGrid can notify about a job status only by email. However the job status can be obtained by querying the NG client [30].
- A job submitted to NorduGrid has a limit on maximum execution time, which must be set by the user for submission.
- Data produced by a job will be kept on the cluster only for limited amount of time, usually 24 hours.
- A job submitted for parallel execution to NorduGrid has to use the Message Passing Interface (MPI) [29] to be started in parallel.
- When a job is submitted for parallel execution to NorduGrid the number of needed cluster nodes must be specified in the job script.
- The input and output data are handled only at the beginning and at the end of a job.

Other limitations arise from the fact that the NorduGrid middleware does not control clusters directly. Therefore, it cannot guarantee the time when the execution of a submitted job starts.

5 System Architecture

Fig. 2 illustrates the architecture of the POQSEC system running on top of the NorduGrid middleware (NG). The architecture contains four main POQSEC components: *query coordinators, start-up drivers, executors,* and *supervisors.*

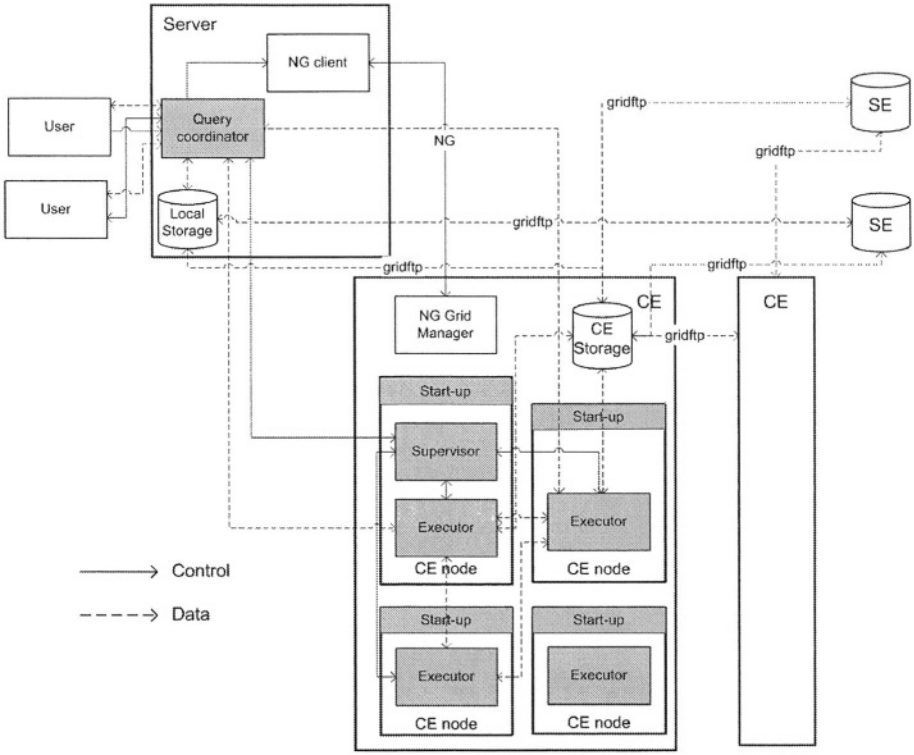


Fig. 2. POQSEC Architecture

NorduGrid components that play important roles in the architecture are the *NG client* and the *NG Grid manager,* which manage jobs submitted to NorduGrid. The *query coordinator* is a POQSEC node that runs on the same server as the NG client. It processes user queries and generates jobs submitted to the NG client. The queries by nature contain database operations such as data selection, projection, and join in addition to application specific operations. The application specific operators are executed as user defined functions calling application libraries such as ROOT [7].

The *computing elements (CEs)* and *storage elements (SEs)* are external resources available through the NorduGrid infrastructure. SEs are nodes managing permanent data repositories. CEs are computational clusters and each of them has working nodes *(CE nodes),* providing local computation, and storage *(CE storage).* The CE storage provides temporary data storage shared between working nodes.

A *user* is an application, running on a client or server machine that interacts with a POQSEC query coordinator. It submits queries to the query coordinator for data processing. The queries often contain computations over large data sets and may therefore be running for long time periods.

Each query coordinator has its own database manager and access to a *local storage,* which is used as temporal file storage. A query coordinator processes queries submitted by users. Some of the queries that do not require computational resources for execution can be executed directly by the query coordinator. Queries expected to be long running are decomposed into subqueries and sent to the NorduGrid for

parallel execution. Long running queries require generating xRSL job scripts describing the requirements for computational resources, the procedures for starting POQSEC modules on CE nodes, and the input and output data for each subquery to be submitted to the NorduGrid client. The query coordinator keeps track of all submitted subqueries by communicating with *supervisors* that monitor and control execution of (sub)queries and run on CE nodes. The query coordinator notifies users about the query status on request. It also sends the query result on a user request when the query execution is finished.

The query coordinator must also deal with job failures. It tracks status of jobs submitted to NorduGrid by polling NG client and when the status is "failed" it resubmits the job.

The NG client software manages the submitted jobs by first extracting from the xRSL scripts requirements for computational resources, such as estimated CPU time, number of required cluster nodes, possibility to connect cluster nodes from an outside cluster, required memory and disk space, version of installed middleware, etc. [39]. The NG client then submits the scripts for execution on clusters fulfilling the requirements. NorduGrid takes care to transfer input data before execution starts, thus guaranteeing that all input data is available when the execution starts, and to transfer result data to a destination specified in the script when execution is finished.

POQSEQ modules are started on CE nodes of the selected cluster when the required number of nodes is available. Each POQSEC module starts by executing a *start-up driver,* which is an MPI program, to initialize the execution. MPI functionality of the start-up driver provides an easy way to choose one node to run a supervisor and to notify start-up drivers of the other nodes about the hostname of the node where the supervisor is running. After the supervisor is started the start-up drivers start *executors* on all the nodes and pass them the hostname of the supervisor. Executors start by registering themselves in the supervisor and then retrieving from the supervisor the information about those other executors they need to communicate with. The executors optimize and execute the queries over the data that is available on the CE storage. Data is sent between executors if it is required for the execution. Produced result is saved in local CE storage for transfer to a SE, another CE, or local storage of the node where the query coordinator is located.

The main task of a supervisor is to monitor and control executions. It communicates with executors to obtain the status and with query coordinators to notify them about the current status.

The architecture assumes the possibility to connect CE nodes with nodes outside CE using IP communication. All communication among POQSEC modules is secure and Globus I/O library [17] is used for this.

6 Query Execution Strategy

The query coordinator decomposes each user query in two levels:

1. *Cluster subqueries* are generated to be executed on clusters as separate NorduGrid jobs.
2. Each cluster subquery is further decomposed into *local subqueries* for parallel execution on the cluster to which it is submitted.

The query optimization has to be done by the query coordinator since required computational resources must be specified in the NorduGrid job scripts when they are submitted.

The first query execution strategy of the system is being developed and evaluated for the application in Section 3.1.

We assume that the data is stored in files of size about 1-2 GB on SEs [2] and accessible through NorduGrid. Execution of a query over one file of such size by one executor on one node takes more than 1 hour. Thus it is reasonable to parallelize execution of the query over one or several files between several executors that run on CE nodes in parallel. Query parallelization is done by partitioning data between executors and executing the same query by different executors over different data partitions. Each executor first accesses a chosen input data file to extract its own portion of data to be processed. Different strategies for file partitioning are being investigated.

Often a user is interested in analyzing many files. Then the query coordinator creates several parallel jobs, one per group of files of the total size 5-10 GB, and for each job it specifies the number of required CE nodes. The query coordinator takes into account that one CE node will be used to merge results produced by other nodes. Each created job is submitted to NorduGrid and the execution is started as described earlier in the architecture section. The executor collocated with the supervisor is used to merge the result; other executors process analysis queries. Each executor optimizes the query and executes a created optimal plan over the data. The executors stream their result data to the executor chosen to do the result merge. The merged result is first saved in the local CE storage and then uploaded by NorduGrid to the local storage of the node where the query coordinator is running. The query coordinator finally merges all the results produced by the jobs and retrieves them to the user.

7 Summary and Continued Work

An architecture has been developed for utilizing the operational NorduGrid infrastructure for managing long running and expensive scientific queries. The architecture is being implemented and evaluated for a Particle Physics analysis application [6]. It is running on top of the NorduGrid infrastructure [41] so special attention is needed to adhere to limitations of this architecture, e.g. that the exact starting time or place of a job cannot be guaranteed and that the number of nodes to use on a cluster must be explicitly given in the job specification. Globus I/O [17] is used for secure communication between distributed POQSEC nodes.

POQSEC utilizes the AMOS II database management system [35] that provides object-relational DBMS functionality, peer to peer communication, declarative query language AmosQL [13], and interfaces to C++ and Java. The kernel is being extended in order to implement the architecture.

The analysis of event data in our implementation is specified declaratively using AmosQL query language. The queries are expressed in terms of logic and arithmetic functions over numbers and vectors. This requires optimization of scientific queries and aggregation functions over numerical data and operations.

In order to be able to read data produced by ATLAS software an object-oriented data analysis framework, the ROOT library [36], was linked to the system. The ROOT library is also going to be used for computations in the application analysis.

The POQSEC system runs under Linux on Swegrid clusters managed by NorduGrid. Parallel query plans are currently created manually by splitting data and queries to investigate trade-offs with the goal to develop automatic query partitioning methods next. Based on this analysis, automatic query decomposition strategies for distributed execution are being investigated.

Another problem that will require special attention and investigation is tracking CPU time so that when the job execution time is approaching deadline the job should be suspended and intermediate results and states should be saved. POQSEC would then resume execution later by restoring the saved state.

Since availability of resources cannot be precisely predicted in advance, adaptive query optimization technique [18] can be used to improve the performance.

Acknowledgments. This work is funded by the Swedish Research Council (www.vr.se/english). We would like also to thank Christian Hansson and professor Tord Ekelöf from the department of Radiation Science, Uppsala University, for providing the particle physics analysis application and data as system test case. The secure communication is implemented by Mehran Ahsant from Center for Parallel Computers, Royal Institute of Technology, Stockholm.

References

[1] *ATLAS Data Challenges,* 2001, [Online], http://atlas.web.cern.cn/Atlas/GROUPS/SOFTWARE/DC/doc/AtlasDCs.pdf

[2] *ATLAS Data Challenge 1, Draft 3.0,* 2003, [Online], http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/DC/DC1/DC1-Report-V3[1].0-22092003.pdf

[3] S. Bethke, M. Calvetti, H.F. Hoffmann, D. Jacobs, M. Kasemann, D. Linglin: *Report oj the Steering Group of the LHC Computing Review,* CERN/LHCC/2001--004, 2001. Available at http://lhcb-comp.web.cern.cn/lhcb-comp/Reviews/LHCComputing2000/Report_final.pdf

[4] O. Bukhres, A. Elmagarmid (eds.): *Object-Oriented Multidatabase Systems: a Solution for Advanced Applications,* Prentice Hall, 1996.

[5] L. Bouganim, F. Fabret, F. Porto, P. Valduriez: Processing Queries with Expensive Functions and Large Objects in Distributed Mediator Systems, *Proc. of Int. Conf. on Data Engineering (ICDE),* 2001

[6] M. Bisset, F. Moortgat, S. Moretti: Trilepton+top signal from chargino-neutralino decays of MSSM charged Higgs bosons at the LHC, *Eur.Phys.J.C30,* 419-434,2003

[7] R. Brun, F. Rademakers: ROOT – An Object Oriented Data Analysis Framework, *Proceedings AIHENP'96 Workshop,* Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86

[8] Condor-G System, [Online], http://www.cs.wisc.edu/condor/condorg/

[9] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Tuecke: A Resource Management Architecture for Metacomputing Systems, *Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategiesfor Parallel Processing,* pp. 62-82, 1998

[10] Directed Acyclic Graph Manager, [Online], http://www.cs.wisc.edu/condor/dagman/

[11] P. Eerola, T. Ekelof, M. Ellert, J.R. Hansen, S. Hellman, A. Konstantinov, B. Kónya, T. Myklebust, J.L. Nielsen, F. Ould-Saada, O. Smirnova, A. Wäänänen: Atlas Data-Challenge 1 on NorduGrid, *ECONF C0303241:MOCT011,* 2003

[12] R. Elmasri, S.B. Navathe: Enhanced Entity-Relationship and UML Modeling, *Fundamentals of Database Systems,* 4th ed., Addison-Wesley, 85-121, 2004

[13] S. Flodin, M. Hansson, V. Josifovski, T. Katchaounov, T. Risch, M. Sköld: *Amos II Release 6 User's Manual,* 2004, [Online], http://user.it.uu. se/~udbl/amos/doc/amos_users_guide.html

[14] I. Foster, C. Kesselman: Globus: A Toolkit-Based Grid Architecture, *The Grid: Blueprint for a New Computing Infrastructure,* Morgan Kaufmann, 259-278, 1999

[15] G. Fahl, T. Risch: Query Processing over Object Views of Relational Data, *The VLDB Journal,* Vol. 6, No. 4, 261-281, 1997. Available at http://www.dis.uu.se/~udbl/publ/vldbj97.pdf

[16] J. Frey, T. Tannenbaum, I. Foster, M. Livny, S. Tuecke: Condor-G: A Computation Management Agent for Multi-Institutional Grids, *Proceedings of the Tenth {IEEE} Symposium on High Performance Distributed Computing (HPDC),* San Francisco, California, pp. 7-9, 2001

[17] *Globus I/O API,* [Online], http://www-unix.globus.org/api/c-globus-3.2/globus_io/html/

[18] A. Gounaris, N.W. Paton, A.A.A. Fernances, R. Sakellariou: Adaptive Query Processing: A Survey, *Proceedings of the 19th British National Conference on Databases: Advances in Databases,* 2002

[19] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y.Sagiv, J. Ullman, V. Vassalos, J. Widom: The TSIMMIS Approach to Mediation: Data Models and Languages, *Journal of Intelligent Information Systems (JIIS),* Kluwer, 8(2), 117-132, 1997

[20] H. Garcia-Molina, K. Salem: Main Memory Database Systems: An Overview, *IEEE Transactions on Knowledge and Data Engineering,* 4(6), 509-516, 1992.

[21] L. Haas, D. Kossmann, E.L. Wimmers, J. Yang: Optimizing Queries across Diverse Data Sources, *23rd Intl. Conf. on Very Large Databases (VLDB'97),* 276-285, 1997

[22] V.Josifovski, T.Risch: Query Decomposition for a Distributed Object-Oriented Mediator System, *Distributed and Parallel Databases,* J. Kluwer, 307-336, 2002. Available at http://user.it.uu.se/~torer/publ/dpdb.pdf

[23] J.S.Karlsson, A.Lai, C.Leung, T.Pham: IBM DB2 Everyplace: A Small Footprint Relational Database System, *17th International Conference on Data Engineering,* 2001.

[24] A. Konstantinov: *The NorduGrid Grid Manager and GridFTP Server, Description and Administrator's Manual,* 2004, [Online], http://www.nordugrid.org/papers.html

[25] B. Kónya: *The NorduGrid Information System,* 2002, [Online], http://www.nordugrid.org/documents/ng-infosys.pdf

[26] The Large Hadron Collider, [Online], http://lhc-new-homepage.web.cern.ch/lhc-new-homepage/

[27] L.Liu, C.Pu: An Adaptive Object-Oriented Approach to Integration and Access of Heterogeneous Information Sources, *Distributed and Parallel Databases,* Kluwer, 5(2), 167-205, 1997.

[28] H.Lin, T.Risch, T.Katchanounov: Adaptive data mediation over XML data. In special issue on "Web Information Systems Applications" *Journal of Applied System Studies (JASS),* Cambridge International Science Publishing, 3(2), 2002. Available at http://user.it.uu.se/~torer/publ/jass01.pdf

[29] The Message Passing Interface (MPI) Standard, [Online], http://www-unix.mcs.anl.gov/mpi/

[30] *The NorduGrid User Guide,* 2004, [Online], http://www.nordugrid.org/documents/userguide.pdf

[31] NorduGrid middleware, the Advance Resource Connector, [Online], http://www.nordugrid.org/middleware/

[32] ORACLE Inc.: *Oracle8i Lite: The Internet Platform for Mobile Computing*
[33] M.T.Özsu, P.Valduriez: *Principles of Distributed Database Systems,* 2nd ed. Prentice Hall, 1999.
[34] D. Quass, A. Rajaraman, Y. Sagiv, J.Ullman, J. Widom: Querying Semistructured Heterogeneous Information. In *Deductive and Object-Oriented Databases, Proceedings of the DOOD'95 conference,* 1995, LNCS Vol. 1013, 319-344, Springer 1995.
[35] T.Risch, V.Josifovski: Distributed Data Integration by Object-Oriented Mediator Servers, *Concurrency and Computation: Practice and Experience J.* 13(11), John Wiley & Sons, September, 2001. Available at http://user.it.uu.se/~udbl/publ/ddiooms.pdf
[36] *ROOT: An Object-Oriented Data Analysis Framework – User's Guide,* 2004, [Online], http://root.cern.ch/root/doc/RootDoc.html
[37] Swegrid, [Online], www.swegrid.se
[38] SYBASE Inc.: *SQL Anywhere Studio,* http://www.sybase.com/mobile/
[39] O. Smirnova: *Extended Resource Specification Language,* 2003, [Online], http://www.nordugrid.org/documents/xrsl.pdf
[40] M.Stonebraker, P.Brown: *Object-Relational DBMSs: Tracking the Next Great Wave,* 2nd ed., Morgan Kaufmann Publishers, 1999
[41] O. Smirnova, P. Eerola, T. Ekelöf, M. Ellert, J.R. Hansen, A. Konstantinov, B. Kónya, J.L. Nielsen, F. Ould-Saada, A. Wäänänen: The NorduGrid Architecture and Middleware for Scientific Applications, *International Conference on Computational Science,* 264-273, 2003
[42] J. Smith, A. Gounaris, P. Watson, N.W. Paton, A.A.A. Fernandes, R. Sakellariou: Distributed Query Processing on the Grid, *Proc. Grid Computing 2002,* pages 279-290, Springer, LNCS 2536, 2002
[43] A. Tomasic, L. Raschid, P. Valduriez: Scaling Access to Heterogeneous Data Sources with DISCO, *IEEE Transactions on Knowledge and Date Engineering,* 10(5), 808-823, 1998
[44] G. Wiederhold: Mediators in the architecture of future information systems, *IEEE Computer,* 25(3), 38–49, 1992.

Improving Distributed Data Mining Techniques by Means of a Grid Infrastructure

Alberto Sánchez, José M. Peña, María S. Pérez, Víctor Robles, and Pilar Herrero

Facultad de Informática, Universidad Politécnica de Madrid, Madrid, Spain

Abstract. Nowadays, the process of data mining is one of the most important topics in scientific and business problems. There is a huge amount of data that can help to solve many of these problems. However, data is geographically distributed in various locations and belongs to several organizations. Furthermore, it is stored in different kind of systems and it is represented in many formats. In this paper, different techniques have been studied to make easier the data mining process in a distributed environment. Our approach proposes the use of grid to improve the data mining process due to the features of this kind of systems. In addition, we show a flexible architecture that allows data mining applications to be dynamically configured according to their needs. This architecture is made up of generic, data grid and specific data mining grid services.

Keywords: Data Mining, Grid Computing, Data Grid, Distributed Data Mining, Data Mining Grid.

1 Introduction

Data mining is characterized to be a complex process. There are two main characteristics that highlight this complexity. First, there are many non-trivial tasks involved in a standard data mining process. These tasks involve different activities like data preprocessing, rule induction, model validation and result presentation. A second determinant factor of data mining problems is the volume of the datasets they deal with.

Modern data mining systems are state-of-the-art applications that use advanced distributed technologies like CORBA, DCOM or Java-oriented platforms (EJB, Jini and RMI) to distribute data mining operations on a cluster of workstations or even all over the Internet. Distribution is a very important ally in the resolution of data mining problems. There are two main reasons to distribute data mining: (i) On the one hand, the efficient use of multiple processors to speed up the execution of heavy data mining tasks and (ii) On the other, there is originally distributed data that cannot be integrated into a single database due to technical or privacy restrictions.

The requirements of high performance data mining have been studied by some researchers. Maniatty, Zaki and others [26,38] collected the most important technological factors both hardware and software for data mining. Hardware support for redundant disks (RAID) and processor configurations (SMP computers and NUMA architectures) are mentioned. Within software contributions, parallel/distributed databases, parallel I/O and file systems are identified as appropriate data storages. Additional factors such as communication technologies like MPI (Message Passing Interface), CORBA, RMI or

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 111–122, 2004.
© Springer-Verlag Berlin Heidelberg 2004

RPC are also important features. Data Space Transfer Protocol (DSTP) [2] is an example of specific communication protocol for distributed data mining.
The studies of different algorithms approaches like Provost [29] analyzed two main parallel and distributed schemas (fine-grain and coarse-grain) and their application to distributed data mining problems. Krishnaswamy defined cost models for distributed data mining [25]. Joshi et al. [17] provided an overview of different parallel algorithms for both association and classification rules. Another overviews can be found on: Kamath and Musick [18], Kargupta and Chan [21,20], Provost and Kolluri [30] and Zaki [35]. These survey contributions are worthwhile to understand the different solutions patterns adopted when a distributed data mining algorithm is designed and also to evaluate the results obtained.

Nevertheless, all these needs are not met by traditional and homogeneous distributed systems. Grid computing has emerged as a new technology, whose main challenge is the complete integration of heterogeneous computing systems and data resources with the aim of providing a global computing space [8]. We consider that grid provides a new framework in which data mining applications can be sucessfully deployed.

The paper is organized as follows. First, we present a overview of distributed data mining. Then, we analyze the requirements of these systems. Section 4 describes the current state of grid technology and its relation with data management. In Section 5, a new flexible and vertical architecture is proposed for data mining grid. Finally, we conclude with some remarks and the ongoing and future work.

2 Distributed Data Mining

Data mining evolution has outlined the development of new contributions in any of the following two lines: (i) New algorithms, theoretical models or data mining techniques and (ii) Technological and design research for new data mining systems and architectures. The same can be asserted for distributed data mining topic. New distributed algorithms, derived from the original centralized versions, have been developed. Some examples are distributed or parallel algorithms for association rules [33], classification rules [34], sequence patterns [36] or clustering algorithms [19]. The second research line is also very important because it deals with the efficient use of computational resources as well as with technical issues related to communication, synchronization, scheduling and so on. The behavior of data mining processing has specific usage patterns in terms of these technical aspects. These patterns have to be studied to design efficient distributed data mining systems.
There are many important advances of commercial products for data mining like IBM's Intelligent Miner [15], Clementine [32] from ISL/SPSS or SGI's MineSet [24] and others[1]. But there are only few commercial data mining systems that provide distributed data analysis at the present moment. A distributed version of Clementine [16] is the most representative example.

Distributed Data Mining Systems (DDM) is an innovative topic and currently only experimental system implementations are under development.

[1] An extended list of free and commercial data mining system can be found at `http://www.kdnuggets.com/software`

JAM (Java Agent for Meta-learning) [28] is an architecture developed at University of Columbia. JAM has been developed to gather information from sparse data sources and induce a global classification model. JAM technology is based on the meta-learning technique. Meta-learning makes it possible to build partial models. These partial models are combined in a global model using the meta-learning formal support. JAM system induces local models from different locations called, datasites. These models are exchanged among the datasites in order to combine them in a common model. The basic elements of JAM architecture are learning agents and meta-classifiers that are placed at each of the datasites to perform local tasks and to communicate the models between nodes. JAM system is a very interesting approach for distributed classification but this approach is not easy to translate into other data mining tasks and queries.

PADMA system [22] is being developed in Los Alamos National Laboratory. PADMA is a document analysis tool working on a distributed environment. The basic elements of this architecture are cooperative agents that analyze the text, interact with other agents and show the results to the user. PADMA system works without any relational database underneath. Instead, there are PADMA agents that perform several relational operations (for example join or selection) with the information extracted from the documents. PADMA system has been developed by using C++ and MPI interface. Documents are stored on an PPFS [14] file system.
Database integration is the key feature of PADMA system. The internal implementation of basic relational operations works tightly-coupled with the data analysis functions.

Papyrus [11] is a distributed data mining system developed by the University of Chicago, Illinois and the National Data Mining Center. Papyrus is able to mine distributed data sources on a WAN network scenario. Papyrus system uses meta-clusters to generate local models which are exchanged to generate a global model. The idea is founded on a theory similar to JAM system. Although JAM and Papyrus have common features, the second is a more advanced architecture in two senses: (i) It uses a formalized model representation language (PMML) to exchange local models, (ii) Data is managed by an efficient multi-layer storage system called Osiris which is an evolution from a persistent object library named PTool. Papyrus has been programmed over an Agent environment called Best which has been implemented in Agent-Tcl, an extended version of the script language Tcl. Additional features, like VRML presentation of results are also supported. Papyrus is a high performance system for distributed data mining, but it is also restricted (like JAM) to a subset of all the possible data mining queries. Its main contribution is the inclusion of a specialized data access system optimized for data mining tasks.

Kensington [23] was originally developed as a research project at London Imperial College and it is currently supported as a commercial tool. Kensington architecture is based on a distributed component environment located at different nodes on a generic network, like the Internet. Kensington provides three kind of components: (i) User oriented components, (ii) Application servers and (iii) Third level servers. The first group interacts with the user getting commands from him and showing the results. The application servers handle persistent objects, control sessions, schedule tasks and, in general, they manage the overall data analysis processes. The third level servers perform massive computation tasks, by retrieving data from the databases and processing them in high performance computation clusters (HPCs).

Kensington system has been developed using open technologies like EJB (Enterprise Java Beans) and CORBA. For HPCs tasks and algorithm implementations C code and MPI (Message Passing Interface) have been used. Nevertheless, DDM Systems require a new scenario in order to fulfill their requirements. Next section talks about these needs.

3 What Actually Does DDM Need?

This contribution comes as a result of the analysis, mentioned above, of technological features highlighted by Zaki and Maniatty as well as the study of many different distributed algorithms patterns. Distributed data mining, although is not completely different from the traditional distributed processing problems, it has special characteristics to bear in mind. The support of many algorithm patterns is the key feature to ensure the extensibility of new open DDM systems. Every existing and new algorithm has a specific combination of the following features:

- Data partitioning schema: Vertical vs Horizontal.
- Communication primitive: One-to-one, one-to-many or different variations.
- Synchronization and control aspects: From centralized to distributed control.
- Granularity and load balancing criteria.
- Sub–algorithm scope: local vs global.
- Sub–algorithm distribution: independent vs centralized.

Many other characteristics may also be included in the list. These are not only features of association, classification and clustering algorithms. Pre-processing tasks like data cleaning, attribute discretization, concept generalization and others should also be performed in parallel in the same way knowledge extraction algorithms do.

DDM systems must keep the existing features of the current centralized versions. The most important characteristic is the support for the complete KDD process: preprocessing, data mining and post-processing. The additional features either inherited from the centralized version or distributed specific are the following:

- For data mining operations, general purpose DBMS are less efficient than flat files [27], but they don't provide any data management features (e.g. relational operations). Specialized database managers are required (specially on a distributed environment). Relational and mining operations must be integrated and defined at the same level.
- State-of-the-art trends and high-performance technologies are very useful tools and become necessary when interoperable and efficient systems are designed. Open architectures like CORBA are oriented towards system interoperability with different scenarios, most of the private LAN networks, but scalability is an important trend to be achieved. Grid Computing provides these extreme parallelization features.
- Factors like the inclusion of algorithms, changes on system topology, different user privileges or system load restrictions have a deep impact in the overall performance of the system. Important decisions like task scheduling, resource management or partition schema should be tuned again almost every time one of these factors is modified.

Nowadays, there is no a universal strategy to configure a distributed data mining environment to be optimal in the resolution of every query. As an alternative, our contribution presents an appropriate support for multiple optimization and configuration strategies. Control policies may be changed, even while the system is running, to configure and optimize the performance of the system.

Although much effort has been addressed on the development of efficient distributed algorithms for data mining this is not the only way to outperform existing data mining solutions. Although the specific distributed implementation of a rule induction algorithm plays an important role, distribution schema, task scheduling and resource management are also key factors.

We consider that there is no global mechanism to evaluate all these factors before the actual system is running in a real environment. Resource management strategies, even data mining-oriented (Bestavros [4]), are too restrictive to provide an effective performance in multiple different circumstances.

Grid Computing services constitute a flexible manner of tackling the data mining needs. These services are described in next section.

4 Grid Computing

Grid Computing takes advantage of the low-load periods of all the computers connected to a network, making possible resource sharing. Grid environments differ substantially from conventional and parallel computing systems in their performance, cost, availability and security features. Different grid services are used with the aim of managing these characteristics. This section describes some of the most important grid services related to the DDM requirements, previously defined.

4.1 Job Scheduling and Planning

Schopf in [31] defines Grid Scheduling like a process involving resources over multiple administrative domains. The concept of scheduling in a grid environment involves the acquiring resources, the matching jobs to resources, the managing data and the monitoring progress of a job in the environment. In short, Grid Scheduling determines, reserves and allocates the resources of a Grid that are required for a job execution.

In order to do this, it is important to know the job requirements, such as the time assignment, required resources, data and software tools, network use and costs. By knowing these requirements, it is possible to plan a job. This process involves parsing the submitted job request, checking static information about available resources, pre-selecting resources, querying dynamic information of the resources, generating the schedule and delegating such job for its running.

The Grid Scheduling Architecture Research Group (GSA-RG) [12] of the GGF (Global Grid Forum) defines the Grid Scheduler Architecture (GSA). GSA allows cooperation between Local Resource Management Systems (LRMS) and other Grid Services such as Index Services and Monitoring and Discovery Services to facilitate the brokering decisions based on more knowledge about the system.

There are several schedulers with different capabilities. Some of the most importants are the following: PBS (Portable Batch System) [3], which controls the beginning of the scheduling of batch jobs and gets routing these jobs between different hosts; SGE (Sun Grid Engine) [13] provides batch queueing, load balancing, job statistics, user-specifiable resources and suspending and resuming jobs; Condor [7] supplies job queueing mechanism, scheduling policy, priority scheme and resource monitoring and management.

4.2 Data Grids

According to Chervenak et al. in [6] the access to distributed data is as important as access to distributed computational resources. Above all since many distributed scientific applications and, in our case, data mining applications require access to huge amounts of data. The volume of this analized data is measured in terabytes and petabytes.

One of the major goals of grid technology is to provide efficient access to data. Grids provide access to distributed computing and data resources, allowing data-intensive applications to improve significantly data access, management and analysis. Grid systems responsible for tackling and managing large amounts of data in geographically distributed environments are usually named data grids.

Data grids require the use of data sources, which are facilities that may accept or provide data [1]. These sources are composed of four components: (i) Storage systems, which include several file systems, caches, databases, and directory services, (ii) Data types, which include files, relational databases, XML databases and others, (iii) Data models, that is, different databases schemas, and (iv) Access mechanisms, which include file-system operations, SQL, XQuery, or XPath. Data grids must manage all these component in a dynamic fashion. Furthermore, generic data management systems involve more challenges. The most important ones are two:

1. All the layers of the grid infrastructure are characterized by their heterogeneity. This includes storage systems, computing systems, data access mechanisms and policies. Heterogeneity does not only affect to the infrastructure, but always to the data itself. Different kind of data formats and data from heterogeneous sources contribute to make more difficult an efficient data management.
2. This infrastructure must be able to tackle huge volumes of data, from terabytes to petabytes.

4.3 Access Restrictions and Control Policies

Foster defines in [8] a grid like “coordinate resources that aren’t subject to centralized control”. If there isn’t a centralized control and the resources are distributed in a wide area network crossing organizational boundaries, resources could be accessed by a lot of different organizations and it will be necessary to control the accesses to the resources. This is the reason why security is one of the most important aspects in the Grid technology.

In [9] the concept of Virtual Organization (VO) is defined and shows the boundaries between the VOs. It must be made sure that only certain organizations or users can

access certain resources, and especially that they are really who they claim that they are. In addition, the Grid Security Infrastructure (GSI) must transparently interact with common remote access tools, such as remote login, remote file access, and programming libraries like MPI.

GSI is based on public-key cryptography, and therefore can be configured to guarantee privacy, integrity, and authentication. But, it is advisable to take into account that, depending of the application, it may be interesting to minimize the cost of the security disabling the integrity and privacy features. Furthermore, GSI supports authorization in both the server-side and the client-side with different authorization options.

5 DMG: Data Mining Grid

Data mining is often used in commercial and business applications. There are significant examples of this scenario: data mining is used in the financial field for detecting several kind of frauds, or in entreprises applications with the aim of detecting trends and patterns in purchasing behaviors.

Respect to data grids, they are more adapted to scientific applications, where the total volume of data tends to be higher than that of business applications. Data mining applications also demand new alternatives in the field of discovery, data placement, scheduling, resource management, and transactional systems, among others. This is due in part to the following reasons:

- It is required to access to multiple databases and data holders, in general, because no single database is able to hold all data required by an application.
- In a generic scenario, multiple databases do not belong to the same institution and are not situated at the same location, but geographically distributed.
- For increasing the performance of some steps of the data mining process, it is possible to use local copies of the whole dataset or subsets.
- Business databases or datasets may be updated frequently, which implies replication and coherency problems.

Additionally, generic data management systems, and particularly data mining grids, involve a great number of challenges, defined in Section 4.2.

Current proposals do not address all these requirements through a general and complete solution. Thus, it is fundamental to define an infrastructure that provides basic and advanced data services for data mining applications, taking into account their peculiarities. Following these guidelines, we propose to define an infrastructure that include these components at least:

- Coupling data sources, which can be dynamically installed and configured. This characteristic makes easier data movement and replication. Nevertheless, the most important problem to deal with here is the potentially large size of datasets as well as the poor performance of the communications in a wide-area network. Data filtering, data replication and use of local datasets help to enhance the efficiency of data mining applications on grid infrastructures.

- Replicated data must be kept consistent. However, depending on the concrete application, this constraint may be relaxed, in such way that solving the consistency do not make worse the application performance.
- Authorized access to data resources, providing a controlled sharing of data within a virtual organization. This implies the use of authentication and access control mechanisms, in the form of access policies.
- One important feature in these systems is data discovery, which is a process that consists of discovering data based on metadata attributes. In this sense, publication, indexing and updating mechanisms are required.
- Both data preprocessing and analysis are basic in data mining applications. Related to these phases, planning and scheduling issues arise. Planning allows the application to be enhanced and adapted to existing resources. Scheduling is used for allocating these resources to the data mining tasks.

In [5], Cannataro et al. define the *Knowledge Grid* as an architecture built on top of a computational grid. This architecture extends the basic grid services with services of knowledge discovery on geographically distributed infrastructures. The Knowledge Grid services are organized in two layers: the core K-grid layer, which contains services directly implemented on top of basic grid services, and the high-level K-grid layer, which includes services used to describe and execute data mining processes.

Another different architecture has been proposed by Giannadakis et al. in [10], named *InfoGrid.* InfoGrid is mainly focused on the data integration. This infrastructure includes a layer of Information Integration Services, which enables heterogeneous information resources to be queried effectively. InfoGrid provides data integration within the framework of a data analysis process, and this is its major concern.

We propose a generic and vertical architecture based on the main data mining phases: pre-processing, data mining and post-processing (see Figure 1).

All these data mining services use both basic data and generic grid services. Since the deployment of every phase is independent of the rest, this architecture has as main advantage its flexibility. Furthermore, whenever a data or generic grid service can be used for a purpose, they will be used primarily. Only if the typical behavior of a service must be modified, a specialized service must be placed at the Data Mining Grid Services level, but implementing the same interface. Thus, data mining applications will become portable to other grid infrastructures. Specialization Services are shown in the figure. These are:

- *SDFS,* Specific Data Filtering Service: Due to the huge amount of data involved in the process of data mining, filtering data is an important task, solved by means of this service.
- *SDRS,* Specific Data Replication Service: One important aspect related to distributed data is data replication. This service is responsible for managing it.
- *SDCS,* Specific Data Consistency Service: Its main purpose is maintaining the data consistency in the grid.
- *SDAS,* Specific Data Access Service: This service is an adaptation of the DAI (Data Access and Integration) Data Grid Service to data mining applications on grid.
- *SDDS,* Specific Data Discovery Service: This service improves the discovery phase in the grid for mining applications.

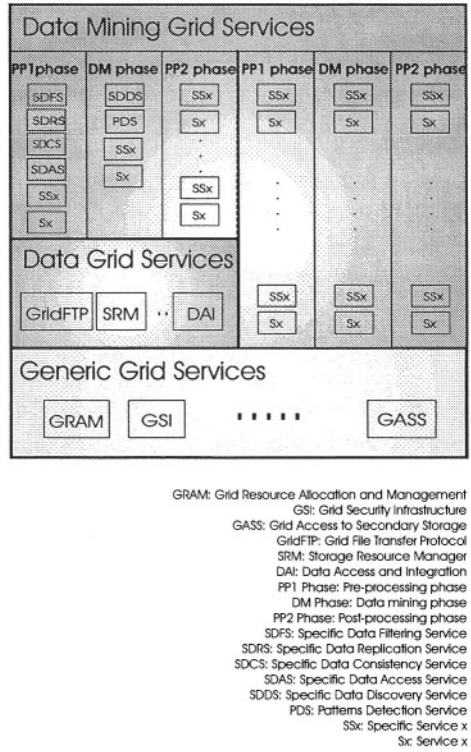


Fig. 1. Proposed Data Mining Grid Architecture

– *SSx:* We can define additional specific services for the management of other features provided by the generic or data grid services. The architecture allows these services be included without changes in the rest of the framework.

New services oriented to data mining application are included in the architecture. For instance, *PDS* (Pattern Detection Service) is responsible for detecting data patterns. *Sx* represents additional services, which are not provided by basic grid services.

Another advantage of defining a three-tier architecture is the possibility to modify any one of the layers for any other technology. For example, it is possible to substitute the set of services of the pre-processing phase by CORBA services, if for instance our legacy system use this kind of technology. This way, different solutions can be integrated for the resolution of the process of Data Mining.

6 Conclusions and Future Work

This paper has shown differents ways of making easier the data mining process in a distributed environment. Many methods have been evaluated and we propose the Data Mining Grid as an enhanced solution to this problem. The main advantage of Data Mining

Grid is its adaptation to the requirements of an expensive data mining process. Firstly, the huge amount of data that is necessary to analize in a data mining application can be managed by means of data grids. Secondly, analized data is usually geographicaly distributed and belongs to different organizations. Thus, it is very important that the access policies are followed by every resource providers. Therefore, each provider could select the dataview shown to each user. Different views cause different results in data mining processes. Grid environments provides all the security infrastructure and the management of the access policies. Finally, one of the most important point in a data mining process is the data discovery and what actions it is necessary to do with each kind of data, since they can be in different formats. Brokering and sheduling systems implemented in Grids make possible the search of data and resources and the selection of actions adapted to each data format.

In order to perform the Data Mining Grid, we have defined a vertical architecture based on the main phases of data mining: pre-processing, data mining and post-processing. This architecture is flexible and allows all the services to be adapted to the data mining process, improving its performance.

As future work we aim to deploy a prototype to test our approach, by following the guidelines of our proposed architecture.

References

1. Malcolm Atkinson, Ann L. Chervenak, Peter Kunszt, Inderpal Narang, Norman W. Paton, Dave Pearson, Arie Shashoni, and Paul Watson. Data access, integration and management. *Chapter 22, The Grid: Blueprint for a New Computing Infrastructure, Second Edition,* pages 391–429, dec 2003.
2. Stuart Bailey, Emory Creel, Robert L. Grossman, Srinath Gutti, and Harimath Sivakumar. *Large-Scale Parallel Data Mining,* chapter A High Performance Implementation of the Data Space Transfer Protocol (DSTP), pages 55–64. 1999.
3. A. Bayucan, R. L. Henderson, C. Lesiak, B. Mann, T. Proett, and D. Tweten. PBS Portable Batch System. External Reference Specification.
4. Azer Bestavros. Middleware support for data mining and knowledge discovery in large-scale distributed information systems. In *Proceedings of the SIGMOD'96 Data Mining Workshop,* Montreal, Canada, June 1996.
5. Mario Cannataro and Domenico Talia. The knowledge grid. *Commun. ACM,* 46(1):89–93, 2003.
6. A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The Data Grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications,* 23:187–200, 2001.
7. The Condor Project, *http://www.cs.wisc.edu/condor.*
8. I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann, 1999.
9. I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputer Applications,* 15(3), 2001.
   Descripci n detallada de la arquitectura Grid.
10. N. Giannadakis, A. Rowe, M. Ghanem, and Y. Guo. InfoGrid: providing information integration for knowledge discovery. *Information Sciences. Special Issue: Knowledge Discovery from Distributed Information Sources,* 155 (3–4): 199– 226, October 2003.

11. Robert L. Grossman, Stuart M. Bailey, Harinath Sivakumar, and Andrei L. Turinsky. Papyrus: A system for data mining over local and wide-area clusters and super-clusters. In ACM, editor, *SC'99*. ACM Press and IEEE Computer Society Press, 1999.
12. Grid Scheduling Architecture Research Group (GSA-RG). *http://ds.e-technik.uni-dortmund.de/yahya/ggf-sched/wg/arch-rg.htm.*
13. Grid Engine. *http://gridengine.sunsource.net.*
14. J. Huber. PPFS: An experimental filesystem for high performance parallel input/output. Master's thesis, Department of Computer Science, University of Illinois at Urbana-Champaing, 1995.
15. IBM. *Application programming interface and utility reference.* IBM DB2 Intelligent Miner for Data. IBM, September 1999.
16. ISL. Clementine servcer distributed architecture. White Paper, 1999. Integrates Solution Limited, SPSS Group.
17. Mahesh V. Joshi, Eui-Hong (Sam) Han, George Karypis, and Vipin Kumar. *CRPC Parallel Computing Handbook,* chapter Parallel Algorithms for Data Mining. Morgan Kaufmann, 2000.
18. C. Kamath and R. Musick. *Advanced in Distributed and Parallel Knowledge Discovery,* chapter Scalable data mining through fine grained parallesim: the present and the future, pages 29–77. AAAI Press / MIT Press, 2000.
19. H. Kargupta, W. Huang, S. Krishnamoorthy, and E. Johnson. Distributed clustering using collective principal component analysis. *Knoledge and Information Systems Journal Special Issue on Distributed and Parallel Knowledge Discovery,* 2000.
20. H. Kargupta, C. Kamath, and P. Chan. *Advanced in Distributed and Parallel Knowledge Discovery,* chapter Distributed and Parallel Data Mining: Emergence, Growth and Future Directions, pages 409–416. AAAI Press / MIT Press, 2000.
21. Hillol Kargupta and Philip Chan. *Advances in Distributed and Parallel Knowledge Discovery,* chapter Distributed and Parallel Data Mining: A Brief Introduction, pages xv–xxv. AAAI Press/MIT Press, 2000.
22. Hillol Kargupta, Ilker Hamzaoglu, and Brian Stafford. Scalable, distributed data mining-an agent architecture, page 211.
23. Kensingston, Enterprise Data Mining. Kensington: New generation enterprise data mining. White Paper, 1999. Parallel Computing Research Centre, Department of Computing Imperial College, (Contact Martin K hler).
24. Ron Kohavi. Data mining with mineset: What worked, what did not, and what might. In *Proceedings ofthe Workshop on Knowledge Discovery in Databases,* 1997. Workshop on the Commercial Success of Data Mining.
25. S. Krishnaswamy, S. W. Loke, and A. Zaslavsky. Cost models for distributed data mining. Technical Report 2000/59, School of Computer Science and Software Engineering, Monash University, Australia 3168, February 2000.
26. William A. Maniatty and Mohammed J. Zaki. A requirements analysis for parallel kdd systems. In Jose Rolim et al., editor, *3rd IPDPS Workshop on High Performance Data Mining,* pages 358–265, May 2000.
27. R. Musick. Supporting large-scale computational science. Technical Report UCRL–ID–129903, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA, 1998.
28. A. Prodromidis, P. Chan, and S. Stolfo. chapter Meta-learning in distributed data mining systems: Issues and approaches. AAAI/MIT Press, 2000.
29. Foster Provost. *Advances in Distributed and Parallel Knowledge Discovery,* chapter Distributed Data Mining: Scaling Up and Beyond, pages 3–28. AAAI Press/MIT Press, 2000.
30. Foster J. Provost and Venkateswarlu Kolluri. A survey of methods for scaling up inductive algorithms. *Data Mining and Knowledge Discovery,* 3(2):131–169, 1999.

31. J. M. Schopf. A General Architecture for Scheduling on the Grid. *Special Issue of JPDC on Grid Computing,* 2002.
Brokers y planificadores de tareas.
32. Colin Shearer. User driven data mining, 1996. Unicom Data Mining Conference. London.
33. T. Shintani and M. Kitsuregawa. Parallel algorithms for mining association rule mining on large scale PC cluster. In Zaki and Ho [37]. in conjunction with ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD99).
34. M. J. Zaki, C. T. Ho, and R. Agrawal. Parallel classification for data mining on shared-memory multiprocessors. In *Proceedings International Conference on Data Engineering,* March 1999.
35. Mohammed J. Zaki. *Scalable Data Mining for Rules.* PhD thesis, University of Rochester, July 1998. Published also as Technical Report #702.
36. Mohammed J. Zaki. Parallel sequence mining on SMP machines a data clustering algorithm on distributed memory machines a data clustering algorithm on distributed memory machines. In Zaki and Ho [37]. in conjunction with ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD99).
37. Mohammed J. Zaki and Ching-Tien Ho, editors. *Workshop on Large-Scale Parallel KDD Systems,* San Diego, CA, USA, August 1999. ACM. in conjunction with ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD99).
38. Mohammed J. Zaki and Ching-Tien Ho. Workshop report: Large-scale parallel KDD systems. In *SIGKDD Explorations* [37], pages 112–114. in conjunction with ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD99).

Improving Grid Services Security with Fine Grain Policies*

Fabrizio Baiardi[1], Fabio Martinelli[2], Paolo Mori[2], and Anna Vaccarelli[2]

[1] Dipartimento di Informatica, Universitá di Pisa
via Buonarroti, 2 - 56127 Pisa
baiardi@di.unipi.it
[2] Istituto di Informatica e Telematica, CNR - Pisa
via Moruzzi, 1 - 56125 Pisa
{fabio.martinelli, paolo.mori, anna.vaccarelli}@iit.cnr.it

Abstract. Grid computing is a continuously growing research field that concerns the implementation of a large scale resource sharing among different kind of institutions over the Internet. The sharing of resources among untrusted entities poses non trivial security problems. This paper proposes an approach to improve the security of computational services in the grid environment. For each grid service, this approach defines a fine grain security policy, that details the operations that are allowed on this service. This policy determines the secure environment where the grid job is executed.

1 Introduction

Grid computing is a technology addressed to the definition and implementation of a large scale resource sharing environment. A grid computing environment enables its participants to establish sharing relationships among them, to share resources of various kind. Each resource is shared by defining a grid service, i.e. a service that can be exploited by any other participants. Hence, the grid environment can be viewed as a dynamic set of grid services that may be aggregated in various ways to meet the needs of any participant. The aim of this environment is to support the collaboration among distinct organizations to solve problems. Examples of applications that may take advantage of the grid environment are the computational intensive ones, such as very large scale simulations (e.g. galaxy formation), or data intensive ones, such as image analysis (e.g. astronomy or climate study). The implementation and the management of these sharing relationships introduce a large set of problems to be faced, such as coordination, integration, interoperability and also security [9], [11].

Security is a big challenge in the grid environment. Since the grid environment is dynamic, the definition and the management of a trust policy is not trivial, because the number and the identities of the participants at a given

* Work partially supported by CNR project "Trusted e-services for Dynamic Coalitions" and by a CREATE-NET grant for the project "Quality of Protection (QoP)"

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 123–134, 2004.
© Springer-Verlag Berlin Heidelberg 2004

time cannot be predicted. Moreover, the heterogeneity of the participants and of their resources causes that different security mechanisms and local policies are adopted. The grid environment requires interoperability among these mechanisms. A basic set of security requirements, such as authentication, delegation, authorization, message confidentiality and integrity and so on, has been defined in [10] and [21]. Other security issues, instead, need further studies, such as the enforcing of fine grain local security policies to protect shared resources from applications that are executed on behalf of grid participants.

This paper proposes an approach to improve the security of the grid environment focused on computational services. A grid computational service provided by a participant q allows any other participant p to execute its application App_p exploiting the computational resource R_q shared by q. This poses non trivial problems from the security point of view because, in the most general case, even if p has been authenticated and authorized to execute its application on R_q, App_p is an unknown application and it could perform dangerous operations on R_q. In particular, this paper proposes to define, for each computational service, a fine grain security policy that describes in some details the operations that the applications are allowed to perform on the resource. This policy is exploited by a local resource monitor to confine in a restricted environment the applications that are executed on behalf of grid participants. Moreover, the grid service description could be extended by exploiting the information returned by this policy. The resulting extended grid service description can be exploited by the grid broker to choose the most appropriate service for each job request.

The paper is organized as follows. Section 2 describes the main features of the grid computing environment, Section 3 describes the problems connected with the sharing of computational resources, and Sections 4 and 5 propose an approach to solve the security problems. Section 6 describes the design of a tool, Gmon, to monitor the execution of Java applications in a grid environment.

2 Grid Computing

The grid environment is a distributed computing environment where each participant allows any other participant to exploit some of its local resources by providing a grid service. A fundamental concept is the one of Virtual Organization (VO). A VO is a set of individual and/or institutions, e.g. companies, universities, industries and so on, that want to share their resources. The management of a VO is very complex because the number of participant in a VO is typically very large, and VOs are dynamic and unpredictable. As a matter of fact, during the life of a VO, some participant can leave the VO, while new participant can be added. The sharing that the grid environment is concerned with, is not primarily file exchange, but rather direct access to computers, software, data and even other kind of resources [11]. These resources are heterogeneous and geographically distributed, and each of these is managed locally and independently from the others. The heterogeneity and the dynamicity of the resources involved in the grid environment requires considerable efforts for the integration

and the coordination that are necessary to create such a computing environment. The solution adopted to guarantee interoperability derives from the Web Service technology, and defines a grid services in terms of a set of standard protocols and a standard syntax to describe the service interface [8].

The Global Grid Forum (GGF) coordinates the efforts in the development of grid technologies, and maintains the Open Grid Service Architecture (OGSA) standard to define the implementation guidelines for grid services.

One of the most important project involved in grid computing is Globus [7]. This project defines a set of basic components that can be exploited to implement a grid infrastructures, and they allow the definition of OGSA compliant grid services. For instance, the Grid Resource Allocation Manager (GRAM) provides a set of standard mechanisms to allow the execution and monitoring of a job on a local resource, while the Grid Security Infrastructure (GSI) component provides a set of standard mechanisms to address security issues, such as grid identity, authentication, and others. Other projects concerning the definition and/or the implementation of a grid computing environment have been developed as well, such as Legion [4], Gridbus [3], WebOS [20] and some others.

Due to its collaborative and distributed nature, the grid environment requires a large support to guarantee security in its transactions. Security management in the grid environment is complicated by the need to establish secure relationships between a large number of dynamically created participants and across distinct administrative domains, each with its distinct resources and its own local security policy. The grid security model has to be integrated with existing local systems, because each participant often has a customized security infrastructure, and grid security mechanism has to inter-operate with the existing ones. Hence, the security model has to be implementation agnostic, i.e. it has to be defined in a way such that it can be implemented through any existing security mechanism. Moreover, the security model also has to guarantee the interoperability between different hosting environments.

The OGSA standard defines the grid security requirements, i.e. a security model to be applied to the grid environment [16]. This model address the following security disciplines: Authentication, Delegation, Single Logon, Credential Lifespan and Renewal, Authorization, Privacy, Confidentiality, Message Integrity, Policy Exchange, Secure Logging, Assurance, Manageability and Firewall traversal. This security model can be implemented through security services.

As an example, the solution proposed by the Globus project assigns a unique grid identity to each participant. This identity is represented by a X.509 certificate signed by the VO certification authority, that includes the owner identity string and its public key. This certificate is used by the owner to produce proxy certificates, that are used by the owner to authenticate himself on some services. Proxy certificates, in turn, are used by these services to authenticate themselves on other services on behalf of the owner with limited rights. Once the authentication is successfully performed, the grid participant is mapped into a local user with the proper set of right.

The strength of this mechanism is given by the secrecy of the private key that, in turn, mainly depends upon the owner local security policy. For instance, if the workstation of an imprudent VO participant p is violated and the private key of p is stolen, the attacker can impersonate p on the grid, and can execute malicious applications on the grid services provided by other participants. Hence, to detect this kind of attacks, the behavior of the applications executed on behalf of grid participant on the local resource has to be monitored.

3 Grid Computational Services

As previously described, each entity in a VO shares some of its local resources with all the other participants by providing a grid service. The nature of grid services may be various, depending upon the kind of resources shared by the entities. Some examples are: storage services, data bases services and computational services. This paper is focused on the last one: the grid computational services. In the rest of the paper, an individual that provides or exploits grid services will be denoted as *entity,* a computational resource that is shared on the grid will be denoted as *grid resource* and the related service *grid service.* Moreover, *grid application* will denote an application that is executed by the grid computational service. When an entity p defines a grid computational service S_p, each other entity q can execute its grid application exploiting S_p. To this aim, the grid environment provides mechanisms to describe the computational service features, to describe the application requirements, to request and to locate computational services, to remotely submit applications and so on.

A Resource Specification Language (RSL) has been proposed in the Grid Resource Allocation Manager (GRAM) architecture [6] to express job requests. In order to be Web Services compliant, this language has been updated by RSL2, that is XML based. Both RSL and RSL2 exploit a set of attributes to represent two type of information: the resource requirements, such as the main memory size or the CPU number, and the job configuration, such as the executable name, the input parameters, the input/output files and so on. As an example, the following RSL specification:

```
(&(executable=MyApp)(max_cpu_time=100)(max_memory=128)
(job_type=single)(count=5))
```

requests a computational resource with at least 128MB of main memory, to execute the application `MyApp` for a maximum CPU time of 100 seconds. This application starts as a single process, and this process will create other 4 other processes or threads. The Grid Service Description Language (GSDL) [19], instead, is a language derived from the Web Services Description Language (WSDL) [5], that is used to describe the features of a grid service, i.e. how entities and/or other grid services can interact with this service.

In the case of computational services, security plays a very important role. From the point of view of the entity that provides the computational service, a grid application is an unknown third party code, and its execution on the

local resource poses considerable problems from the security point of view. As a matter of fact, a grid application could include code meant to perform dangerous operations on the local resource. The identification, the authentication, and the authorization of the entities that exploit grid services are fundamental steps to prevent attacks. As a matter of fact, they are among the basic security requirements defined by the OGSA standard. However, these mechanisms provide coarse grained control over the applications, and they are not sufficient to guarantee the security of the computational resource. As a matter of fact, once the authentication service has proved the identity of q and the authorization service has stated that q is allowed to submit its application, these mechanisms do not provide further control over the subsequent actions of the application. A grid application could include dangerous code for, at least, the following reasons: *i)* application errors, *ii)* q is aware of the malicious code included its application, because q is the real attacker; *iii)* the grid identity of q has been stolen, because q has been successfully attacked by a third party. Hence, an improvement of the grid security infrastructure to provide fine grain control over the grid applications is required to address this issue.

The unauthorized operations that could be performed by grid applications can be various. In a simple case, an application could try to exploit the grid service overcoming the usage limitations that have been imposed. For instance, a grid application could try to exploit more CPU time than the one that has been declared in the job request. With reference to the previous example of job request, a violation of the job request occurs if `MyApp` is executed for more than 100 seconds, because the request states 100 seconds. In a more complicated case, instead, the grid application could perform malicious operations to gain unauthorized control of the grid computational resource. For instance, it could try to modify some system files or to exploit some known security holes of the system to guarantee an unauthorized access to the resource to the attacker.

4 Fine Grain Security Policy

The approach presented by this paper to improve the security level of the grid environment defines, for each grid service, a fine grain security policy that details the set of operations that the grid applications are allowed to perform on this service. Hence, the security policy defines the restricted environment that is created on the local resource to execute the grid applications. This approach enforces the security from the grid service provider side.

Moreover, the security policy can be used to extend the grid service description, and the extended grid service description can be exploited by a broker to determine the most suitable grid service to satisfy each job request. As a matter of fact, given a job and assumed that its behavior is represented in some way, the broker can choose a grid service where all the operations that compose the job behavior are permitted. For instance, if the grid application requires to communicate with other applications running on other computational services, the broker will choose a service that allows connections to other hosts.

According to our approach, two kinds of security constraints defines the environment where the grid application is executed. The first kind of constraints involves the resource usage limits that can be derived from the job request. As a matter of fact, an entity that wants to execute its application on the grid specifies, in the job request, the features of a computational service that meets the requirements of its application. This specification can be interpreted as a maximum limit to the resource exploitation imposed to the grid applications. For instance, in the previous example, the job request includes the field `(max_cpu_time=100)`. Consequently, the execution of the grid application cannot use more than 100 seconds of CPU time. Hence, if the grid application lasts more than 100 CPU seconds, this can be considered as a violation of the security constraints. A further example of violation occurs if `MyApp` creates more than other 4 processes/threads, because the requests specifies that `MyApp` is composed of 5 processes/threads. If the number of child processes created by a grid application is not monitored, a malicious user could set up a Denial Of Service attack by submitting an application that creates a very large number of processes, in order saturate the resource and to prevent further processes from being created.

The second kind of security constraints involves the recognition of dangerous or even malicious behavior of the grid application meant to gain unauthorized accesses to the grid resource. This task has been denoted in computer science as Intrusion Detection. Two main models of intrusion detection systems have been defined: misuse based and anomaly based.

The misuse based model requires the definition of the pattern of each possible intrusive behavior and, for this reason, the main disadvantage is that it prevents only those attacks that it knows about. The anomaly based model, instead, try to detect unusual behaviors of the applications. Since the applications that are executed by a grid computational service may be various, and may be executed on behalf of distinct entities, the determination of a normal behavior of a grid application is not trivial. However, this approach can be exploited by defining the permitted behavior of an application, instead of the normal one.

The permitted behavior is defined through a set of rules, denoted as security policy, that determines which operations and which sequences of operations are permitted on the grid service. The rules that compose the policy describe the only operations that are allowed by the service, while each other operation is denied by default. This strategy adopts the "default deny" approach.

The only way a grid application has to interact with the computational resource and to change its status is by invoking operative system calls. As a matter of fact, processes exploit system resources, e.g. memory or disk space, by issuing an appropriate request to the resource operative system through the system call interface. This implies that the application behavior can be monitored trough the system calls it issues, and the application environment can be restricted by preventing it from invoking some system call. Hence, in the following, we assume that the operations we are interested in correspond to system calls.

5 Policy Specification Language

This section describes the language to specify the security policy. The formal definition of this language is beyond the scope of this paper, hence an informal description is given. The security policy is composed of a set of rules. At first, we describe the simplest case, the *basic rules.* A basic rule refers to a system call *sc,* and defines: *i)* the allowed values for the parameters before the execution of *sc, ii)* the allowed values for the parameters after the execution of *sc*, *iii)* the allowed value for the result of *sc* and *iv)* the assignments to variables. The controls over the parameters after the syscall are necessary because some system calls store their results in variables whose references have been passed through the parameters. From the syntactical point of view, each basic rule, at first, reports the name of the system call, followed by the parameter controls and the variable assignment, included within braces. The first round brackets within braces include the controls over the parameters to be executed before the syscall, the second ones include those to be executed after the syscall, while the third ones include the controls to be performed on the result. Each control is implemented by applying an operator to some values. The operators are listed in the same order of the parameters they refer to. Obviously, the operator is implicitly applied also to the value of the syscall parameter it refers to. Hence, the first operator within round brackets is applied to the first parameter, and so on. For instance, the operator `eq(‘‘abc’’)` is satisfied if the value of the parameter it refers to is equal to the string “abc”. The square brackets, instead, include the variable assignments that are executed after the syscall. The first ones are used to assign the value of the parameters to variables. The assignments are executed in the same order in which the variable are listed. The second square brackets are used to assign the syscall result to a variable. Variables can be either local, i.e. visible only in the rule where are defined, or global, i.e. visible in all the rules of the policy. The following example shows a basic rule.

```
open{(eq(‘‘/tmp/*’’), eq(WRITE), -)()()[][V1]}
```

This rule allows the execution of the open syscall if, before the execution, the first parameter, i.e. the file name, is equal to `‘‘/tmp/*’’` and the second parameter, i.e. the open mode, is equal to the `WRITE` value. The symbol “-” is a placeholder that indicates that no controls are executed on the third parameter. As usual, the symbol “*” represents any string. The result of the syscall, i.e. the file descriptor handle, is assigned to the variable V1. Hence, this rule allows to open any file in the /tmp directory in write mode.

In the most general case, a rule defines a sequence of syscalls that represents an admitted behavior. The sequences are created by composing the basic rules in several ways, as described in the following. An example of composed rule is the following.

```
open{(eq(‘‘/tmp/*’’), eq(WRITE), -)()()[][V1]}.
write{(eq(V1), -)()()[][]}.close{(eq(V1))()()[][]}
```

This rule is defined in terms of three basic rules, that are composed through the sequence operator ".". The first basic rule involves the `open` syscall, and it is the same of the previous example. The result of the syscall, i.e. the file descriptor handle, is assigned to the variable V1. The variable V1 is local, and is visible only in this rule. Moreover, V1 is a set variable, i.e. it assumes a set of values. Hence, if more than one `open` syscall is executed, all the values assigned to V1 are stored, i.e. the last variable assignment does not overwrite the previous ones. The second basic rule involves the `write` syscall and is connected with the first one through the "." composition operator. This means that a `write` syscall is admitted only after an `open` one. The parameter controls of the `write` syscall exploit the values of the local variable V1 that have been instantiated by the `open` syscalls previously executed. Hence, a `write` syscall is admitted only if the file descriptor handle has been previously instantiated by an `open` syscall. This prevents the application from writing on file descriptors that have not been regularly initialized. In the same way, a `close` syscall is admitted only after a `write` one, on a file that has been previously opened and written according to the two previous basic rules. Notice that, in this example, since the rule uses the "." composition operator, one `write` operation only is allowed.

The other composition operators are described by the following grammar:

$$P ::= \alpha.P \quad \| \quad p.P \quad \| \quad PorP \quad \| \quad P\|P \quad \| \quad i(P) \quad \| \quad Z$$

Where P is a composed rule, α is a basic rule and p is a predicate. The informal semantics is:

- $\alpha.P$ represents the possibility of performing a syscall that satisfies the rule α and then behave as P;
- $p.P$ behaves as P in the case the predicate p is true;
- $P_1 or P_2$ represents the non deterministic choice between P_1 and P_2;
- $P_1 \| P_2$ represents the parallel execution of P_1 and P_2;
- $i(P)$ behaves as P x times, for any value of x;
- Z is the constant process. We assume that there is a specification for the process $Z = P$ and Z behaves as P.

6 Gmon

In the following, we focus on computational services that support the execution of Java applications. In this context, we describe the design and the implementation of an highly configurable security tool, Gmon, that implements the secure environment described in the previous sections to execute Java application. This tool exploits the job request and the local security policy to perform a very fine grained monitoring of the grid application executed by the Java Virtual Machine on the local computing resource.

The sharing of computational resources among a community of heterogeneous entities is a very hard task because it requires high interoperability among the involved resources. As a matter of fact, if an entity p wants to execute its grid

application on a computational resource R_q shared by another entity q, the format of this application has to be compatible with the environment installed onto R_q. We have focused our attention on Java language because it is very suitable to be adopted for developing grid applications. As a matter of fact, one of the major java features is portability: a Java program that has been compiled on a given machine with a given operating system, can be executed on any other machine with any operating system that supports Java. The platform independence of Java is a very interesting feature in the grid environment, because it addresses the interoperability problem. This means that each entity can develop its grid application using Java on its local system and it can execute the resulting bytecode on any other grid computational service supporting Java without any compatibility problem. The portability of the bytecode is a well known feature of Java, and it has been already exploited in some Java based distributed heterogeneous environment such as IceT [13], Javelin [17] and Bayanihan [18].

Several Java Virtual Machines (JVMs) that follow both the *Java Language Specification* [14] and the *Java Virtual Machine Specification* [15] are currently available. For instance, the IBM Jalapeno project of the IBM T.J. Watson Research Center has developed the Jikes Research Virtual Machine (RVM), that is a complete and open source JVM [1].

From the point of view of security, the standard Java security architecture provides a bytecode verifier, a class loader, and a security manager that implements an access control mechanism [2], [12]. Moreover, Java applications can report also origin and signature. In this context, Gmon can be considered as an extension of the Java security architecture because it permits to define and to manage complex security policies that include traces of the permitted behaviors.

Gmon confines the Java grid applications in a sandbox whose constraints are defined by both the job request and the security policy defined on the local resource. As described in the previous section, these constraints involve both the resource usage and the operations that the grid application performs on the local resource. To create a restricted environment, Gmon interacts with the JVM resource management. For instance, when a new object is created, the JVM memory support is invoked to allocate new space for this object. In this case, Gmon checks the current memory usage before the allocation request is actually issued to the operative system. Moreover, Gmon inspects the runtime behavior of a java application by tracing the system calls the application issues. When a syscall violates the environment constraints, an error code can be returned to the application, or the application can be stopped.

To insert the Gmon thread initialization and the proper invocation (hooks) to Gmon each time a system call is invoked, the JVM has been slightly modified. When the JVM starts, a further thread that executes Gmon is started. Each time the JVM performs a system call, this call is intercepted, the JVM is suspended and Gmon is activated to perform the security checks. The system call is executed and the JVM is resumed only if the security constraints are satisfied.

6.1 Implementation Details

Since Gmon closely interacts with the JVM it monitors, some implementation issues, such as the system call wrapping mechanism, depends upon the chosen JVM and operative system. Our tool has been developed under Linux (2.4.24) operative system, and we have adopted the IBM jikes RVM Java Virtual Machine [1]. This virtual machine exploits the gnu Classpath library to implement the Java core classes. This library too, is a widely used open source software.

When the JVM starts, a Posix thread that executes Gmon is created. After the initialization, Gmon is suspended on a semaphore. When a system call is issued by the JVM during the execution of a java application, this call is intercepted by a wrapper, that activates Gmon and suspends the JVM. The standard method to intercept system calls that is provided by Linux operating system, i.e. the ptrace( ) one, has not been adopted, because it is not configurable, and it does not allow to trace a subset of system calls only. In the Java case, the application does not interact directly with the operating system, but is interpreted by the JVM that, in turn, interacts with the operating system. Hence, for a matter of both flexibility and efficiency, the system call wrapper has been embedded in the JVM itself. This approach allows us to intercept only the syscalls we are interested in. Because of the design of the Jikes RVM, two distinct approaches have been adopted to intercept system calls. The first approach is adopted to intercept system calls issued by the bytecode of the application. Obviously, the bytecode of the application is not known a priori. However, the system calls are issued through the classes in the classpath library that, in turn, implements them exploiting the gnu libc functions through the Java native interface (JNI). Hence, the invocations to the libc functions we are interested in are wrapped by our functions, that activate the Gmon thread and suspend the JVM one through semaphores. Gmon checks whether the system call is allowed in the secure environment and, if the call is allowed, it is executed and the application is resumed.

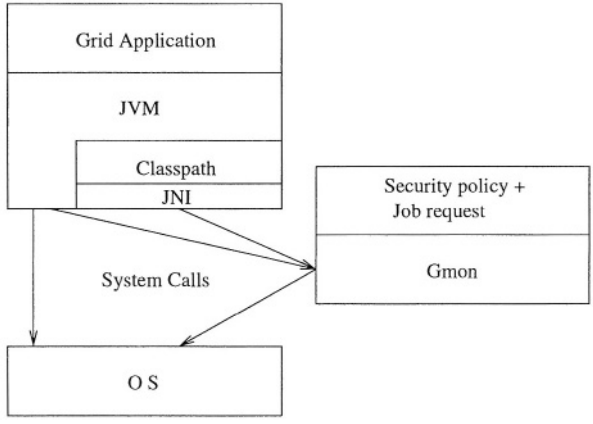


Fig. 1. Gmon architecture scheme

Otherwise, the application can be stopped or an error code can be returned to the application. Since the Gmon hooks are embedded in the JVM code, the application cannot prevent Gmon from being activated. Moreover, the consistency of the system call parameters that are checked by Gmon with respect to the parameters that are actually used when the system call is execute is guaranteed, because the JVM and, consequently, the grid application, is suspended before the parameter control and it is resumed only when the system call has been executed. As a matter of fact, the syscall is executed by Gmon. The second approach to intercept the system calls, instead, is adopted for system calls that are issued by the JVM itself. In general, these calls concern the resource management operations such as threads scheduling, memory management, and other runtime services. These calls are very simple to intercept, because the JVM source code is known a priori. Hence, instead of wrapping the call at a lower level, the appropriate hook can be inserted directly in the JVM. For instance, when a Java class is invoked for the first time in the application, it is loaded in main memory. Hence, the JVM opens the file that records this class, allocates new memory space, and reads this file into memory. These operations are executed through system calls issued by the JVM class loader. Since the behavior of the JVM is assumed to be trusted, not all the system calls issued by the JVM are intercepted and controlled, but only those that are exploited to monitor the system resources usage. Figure 1 describes the architecture of the proposed system.

Obviously, this architecture does not prevent from code invoked through the Java Native Interface (JNI). As a matter of fact, JNI is used to bypass the JVM and to execute arbitrary code. However, the use of JNI can be denied.

7 Conclusions

This paper has proposed an approach to improve the security of computational resources shared in the grid environment. This approach is based on the definition of a fine grain security policy that, for each computational resource, details the operations that the grid applications are allowed to perform on the resource.

We believe that the effectiveness of the security infrastructure of the grid computing environment is a fundamental issue to attract potential participants to a VO. Hence, from the potential participants point of view, the security improvement proposed in this paper is an incentive to participate in a VO. As a matter of fact, the more accurate are the security checks that are performed on the grid applications, the more potential participants are willing to share their computational resources in a grid environment.

References

1. B. Alpern, C. R. Attanasio, J. J. Barton, et al. The jalapeño virtual machine. *IBM System Journal,* 39(1), 2000.
2. A. Anderson. Java access control mechanisms. Technical report, Sun Microsystems, 2002.

3. M. Baker, R. Buyya, and D. Laforenza. Grids and grid technologies for wide-area distributed computing. *International Journal of Software: Practice and Experience (SPE),* 32(15):1437–1466, 2002.
4. S. J. Chapin, D. Katramatos, J. Karpovich, and A. Grimshaw. Resource management in Legion. *Future Generation Computer Systems,* 15(5–6):583–594, 1999.
5. E. Chrinstense, F. Curbera, G. Meredith, and S. Weerawarana. Web service description language. W3C, 2001.
6. K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A resource management architecture for metacomputing systems. *Lecture Notes in Computer Science,* 1459:62–92, 1998.
7. I. Foster and C. Kesselman. The globus project: A status report. In *Proc. of IPPS/SPDP '98 Heterogeneous Computing Workshop,* pages 4–18, 1998.
8. I. Foster, C. Kesselman, J.M. Nick, and S. Tuecke. Grid services for distributed system integration. *IEEE Computer,* 35(6):37–46, 2002.
9. I. Foster, C. Kesselman, J.M. Nick, and S. Tuecke. The physiology of the grid: An open grid service architecture for distributed system integration. Globus Project, 2002. http://www.globus.org/research/papers/ogsa.pdf.
10. I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In *Proc. 5th ACM Conference on Computer and Communications Security Conference,* pages 83–92, 1998.
11. I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications,* 15(3):200–222, 2001.
12. L. Gong. *Inside Java2 Platform Security.* Addison-Wesley, 2nd edition, 1999.
13. P.A. Gray and V.S. Sunderam. Icet: Distributed computing and java. *Concurrency: Practice and Experience,* 9(11):1139–1160, 1997.
14. J.Gosling, B. Joy, G. Steele, and G. Bracha. *The Java Language Specification.* Sun Microsystems, 2000.
15. T. Lindholm and F. Yellin. *The Java Virtual Machine Specification.* Sun Microsystems, 1999.
16. N. Nagaratnam, P. Janson, J. Dayka, F. Siebenlist, V. Welch, S. Tuecke, and I. Foster. Security architecture for open grid service. Global Grid Forum Recommendation Draft, 2004.
17. M.O. Neary, B. Christiansen, P. Cappello, and K.E. Schauser. Javelin: Parallel computing on the internet. *Future Generation Comp. Systems,* 15:659–674, 1999.
18. L.F.G. Sarmenta and S. Hirano. Bayanihan: building and studying Web-based volunteer computing systems using Java. *Future Generation Computer Systems,* 15(5–6):675–686, 1999.
19. S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, and C. Kesselman. Grid service specification. Global Grid Forum Recommendation Draft, 2002.
20. A. Vahdat, T. Anderson, M. Dahlin, E. Belani, D. Culler, P. Eastham, and C. Yoshikawa. WebOS: Operating system services for wide area applications. In *Proc. of the Seventh Symp. on High Performance Distributed Computing,* 1998.
21. V. Welch, F. Siebenlist, K. Czajkowski, J. Gawor, L. Pearlman, I. Foster, J. Bresnahan, C. Kesselman, S. Metier, and S. Tuecke. Security for grid services. In *12th IEEE International Symp. on High Performance Distributed Computing,* 2003.

Resource Dissemination Policies on Grids

Marta Beltrán, Jose L. Bosque, and Antonio Guzmán

ESCET, Universidad Rey Juan Carlos, 28933 Móstoles, Madrid, Spain
{mbeltran,jbosque,aguzman}@escet.urjc.es

Abstract. Grid computing allows large-scale sharing of resources. The resource management system (RMS) is one of the most important Grid components because it provides the mechanism to manage all the available resources in an efficient way. Due to the large size of current Grid systems, the RMS is usually a federation of several cooperating RMS's. This cooperation implies an information exchange between the set of heterogeneous RMS machines. In order to compare the different information exchange policies, this paper defines a new metric, the information efficiency. This metric allows to quantify the benefits obtained with a resource dissemination policy, considering the network traffic generated by the information exchange and its scalability. Conclusions about what policy is most efficient for Grid systems will be presented.

1 Introduction

Grid systems require components capable of managing the resources available for users and applications ([1]). The resource management system (RMS) is responsible for this function, that includes resources discovery (to locate a needed resource), registration (for resources joining or leaving the Grid) and monitoring (to have updated information about resources attributes and status) ([2]).

The RMS should be flexible, adaptable and scalable, capable of managing a large number of heterogeneous resources with a dynamic behavior and geographically distributed among multiple sites. In addition, systems with different administrative policies can be added to the Grid to share their resources but preserving their autonomy and constraints. And, finally, the RMS must support quality of service (QoS) and fault tolerance. In order to achieve all these requirements it is necessary to employ a group of cooperating RMS's instead of a single service ([3]).

This RMS group or federation must exchange information to efficiently cooperate. Resource dissemination determines the way in which this information is sent between the RMS machines, usually a very heterogeneous set of computer systems. This paper discusses and compares three different resource dissemination policies, batch or periodic, on line or on demand and event-driven or on-state-change. These strategies come from cluster computing techniques such as load balancing and scheduling algorithms and they have not been specifically developed for Grid systems. Therefore it is necessary to evaluate the best approach for resource dissemination on this kind of systems

The main contribution of this paper is the definition of a new metric to compare information strategies in terms of scalability, network overhead and effectiveness in the information updating. This metric, called information efficiency (ε) allows an exhaustive

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 135–143, 2004.
© Springer-Verlag Berlin Heidelberg 2004

analytical and experimental comparison of the three studied policies. The experimental comparison is performed on two different RMS federations and leads to very interesting conclusions about the convenience of the different resource dissemination approaches on Grid systems.

The rest of this paper is organized as follows. Section 2 presents different RMS resources dissemination strategies proposed in previous works. Section 3 defines the information efficiency necessary to compare these information policies. Section 4 reports an analytical and experimental comparison of the different approaches based on this efficiency. And finally, Section 5 with conclusions and future work.

2 Background

Traditionally, resource dissemination has been based on techniques that come from information services designed for load balancing and scheduling algorithms, performance diagnosis and monitoring tools, replica selection services, ... Therefore, the techniques proposed in previous works can be classified in periodic, on demand and event-driven rules.

Some examples of periodic resource dissemination are presented in [4] (RMS for Bond), [5] (RMS for Condor), [6] (European DataGrid), [7] (Globus), [8] (Javelin), [9] (Legion), [10] (MOL), [11] (NetSolve) or [12] (Nimrod/G). This approach is widely used due to its simplicity, because resources information is periodically disseminated through the Grid, regardless of whether this information is needed or not.

On the other hand, an example of the on demand policy is 2K ([13]). In the 2K system this kind of dissemination is implemented with mobile agents and the information is sent only when it is useful for other RMS machine and it is requested explicitly.

Finally, an example of event-driven policies is presented in [14]. The information protocol proposed in this paper supports this kind of dissemination, allowing asynchronous exchanges when some specified conditions are met (events). These events are typically related to machines state changes in a certain degree.

3 Information Efficiency

Although the periodic approach seems to be the most extended resource dissemination policy for RMS, there are still no justified reasons to accept this policy as the best for Grid systems. Many RMS federations adopt this approach considering only its simplicity but ignoring efficiency and scalability concerns.

Intuitively, it can be seen that the periodic approach for resources dissemination provides a simple solution for information exchange but one problem is how to fix the interval for this exchange. A long interval leads to work with obsolete information in the RMS, but a short interval, necessary to manage updated information, implies a heavy communication overhead. On the other hand, the on demand approach minimizes the number of generated messages: information is sent only when a RMS machine requests it. But this information policy always introduces an operation delay, because when some RMS machine needs information, it has to wait for the rest of RMS machines to

send it. This includes a synchronization point to perform information exchange between machines, and these kind of synchronizations should be avoided on Grid systems. The event-driven policy might obtain a compromise between the two previous solutions, but it has to be demonstrated.

In order to compare the different alternatives taking into consideration this kind of issues and to draw analytical conclusions, it is necessary to define the *information efficiency* (ε). This metric is defined to quantify the level of compromise between the benefits obtained with an information policy and the network overhead introduced in the system by this policy.

To take into account the network utilization using a certain resource dissemination, the number of generated messages (M) can be used. But it must be referred to the number of machines that compose the RMS, because a RMS that generate M messages is much more efficient if it is composed by great number of machines than if it is composed by only a few systems.

Therefore, the number of generated messages per RMS machine (m) is considered:

$$m = \frac{M}{N} \tag{1}$$

where N is the number of machines in the RMS.

The information efficiency is inversely related to m, because the resource dissemination must avoid causing a communication overhead. But this value is not sufficient to quantify the efficiency of a resource dissemination, the benefits obtained with this policy should be considered too.

Suppose that two information policies generate the same number of messages per RMS machine, but one of these policies does not exchange resources information with the required effectiveness. Thus, the information in the RMS machines will not be sufficiently updated and all the decisions will be based on obsolete information. This would lead to an overall system performance degradation and should be considered in the efficiency computation.

Therefore, the information efficiency can be defined as:

$$\varepsilon = \frac{S}{m} \tag{2}$$

where S is a speedup or profit measure capable of quantifying the different performances obtained in the Grid with different resources dissemination policies. In this paper, the speedup obtained executing an application on a Grid system instead of executing it on a single machine is used, due to the great influence that the resource information policy may have on this speedup.

4 Resource Dissemination Policies Comparison

4.1 Analytical Comparison

Once the information efficiency is defined, it can be evaluated for the three considered approaches. The number of total messages generated by the resource dissemination

Table 1. Efficiency values for the three studied resource dissemination policies

	Periodic	On Demand	Event-driven
m	$k \cdot N$	$2 \cdot R$	E
ε	$\frac{S}{k \cdot N}$	$\frac{S}{2 \cdot R}$	$\frac{S}{E}$

policy can be evaluated for a certain interval of time τ. To maintain the generality of the information efficiency definition and to allow its evaluation whatever network technology is used, only one to one communications has been considered, ignoring collective operations such as broadcasts.

- *Periodic dissemination:* Every RMS machine sends its information to the rest of machines in each information exchange interval, generating $N - 1$ messages. Let T denote this information interval (usually fixed by the administrator), and k denote the number of these intervals that occurs in the studied period τ. Therefore, $k = \tau/T$, and the total number of messages is:

$$M_P = k \cdot N \cdot (N - 1) \tag{3}$$

- *On demand dissemination:* When a RMS machine needs information, it sends a request message to the rest of RMS machines and they reply with the required information. Let R be the total number of information requests during the considered interval of time τ, therefore the total number of messages is:

$$M_{OD} = 2 \cdot R \cdot (N - 1) \tag{4}$$

- *Event-driven dissemination:* Every RMS machine sends its information to the rest of machines when its information values change by a specified amount, i.e., when an event takes place. Let E be the total number of events that take place in all the RMS machines during the considered interval τ. The total number of messages is:

$$M_E = E \cdot (N - 1) \tag{5}$$

The typical number of machines in a RMS system is large enough to allow approximating N by $N - 1$. With this approximation, the number of messages generated per RMS machine and the efficiency values are shown in table 1.

From this table it can be observed that the periodic approach is the only one inversely related to the number of machines in the RMS system. This dependence made the periodic policies bad solutions for resource dissemination in terms of extensibility and scalability. When the number of machines in the RMS increases (and this is directly related to the number of resources in the Grid), the periodic dissemination efficiency decreases due to the large communication overhead introduced in the system. And in most of the cases, the generated messages will include redundant information, not useful for any RMS machine.

The other two alternatives, the on demand and event-driven policies, are scalable because they are not dependent on the number of machines in the RMS. But can these two approaches achieve the same efficiency values?. Supposing that with these two

dissemination policies the same speedup could be obtained, the maximum efficiency values for these policies can be evaluated. The minimum values for R and E are 1, because with $R = E = 0$ it would not make sense to study resource dissemination (it would not be information exchange between machines). Therefore:

$$\varepsilon_{OD}^{max} = \frac{S}{2} \tag{6}$$

$$\varepsilon_{E}^{max} = S \tag{7}$$

From equations 6 and 7 it is clear that the event-driven dissemination is capable of obtaining the same speedup than an on demand dissemination with less messages (actually with half messages), therefore with more efficiency.

The conclusion of this analytical study allows rejecting the periodic dissemination due to its scalability limitations, specially for RMS systems with a large number of machines. And it allows rejecting the on demand dissemination too, in this case, due to the number of messages generated by this policy, always greater than for an event-driven policy capable of obtaining the same speedup. Therefore, the event-driven or on-state-change dissemination is recommended for Grid systems. This recommendation will be supported with the experimental results shown in the next section.

4.2 Experimental Comparison

In order to corroborate the conclusions that come from the previous analytical study, a RMS federation has been simulated in our laboratory. In fact, two different RMS federations have been used to compare the studied resource dissemination policies, one with four machines (small size) and other with sixteen machines (medium size).

The application used to evaluate the information efficiency of the information policies is a light load balancing algorithm executed to balance some simple CPU-bound tasks between the RMS machines. Therefore, the information exchanged between the RMS machines is related to their workload level, needed to execute the load balancing algorithm.

Due to space restrictions only some examples of all the performed experiments are shown, but all the obtained results are considered for conclusions. One example of the performed experiments is shown in table 2 for the small RMS and in table 3 for the medium one.

In all the experiments the RMS machines have to execute a workload composed by 10 CPU-bound tasks per RMS machine, i.e. 40 tasks for the small RMS and 160 tasks for the RMS. Therefore, the information volume necessary to balance the system load is equivalent in the two studied systems and fair comparisons can be established.

An initial tasks assignment is made, distributing 10 tasks per RMS machine. This allocation is not balanced because the RMS federation is a set of very heterogeneous machines, therefore, the load balancing algorithm begins balancing the load on the RMS. The load balancing process implies an information exchange between the RMS machines and allows comparing the three resource dissemination approaches. The interval of time considered for the experiments, τ, is the elapsed time of the whole application execution

Table 2. Experiments for the small RMS federation (N=4)

PERIODIC DISSEMINATION			
T (s)	k	S	ε
1	56	7.71	0.03
3	19	7.85	0.10
5	12	7.71	0.10
10	6	7.57	0.32
20	3	7.19	0.60
30	2	7.31	0.91
60	2	7.31	0.91
120	1	6.95	1.74
180	1	6.95	1.74
300	1	7.07	1.77
600	1	6.73	1.68
1000	1	6.42	1.61
ON DEMAND DISSEMINATION			
	R	S	ε
	8	7.71	0.48
EVENT-DRIVEN DISSEMINATION			
Neutral width	E	S	ε
0	36	6.06	0.17
0.1	53	6.84	0.13
0.2	43	6.06	0.14
0.25	37	8.00	0.22
0.3	33	6.42	0.19
0.4	18	5.81	0.32

on the RMS machines. And the executed tasks and the load balancing algorithm are always the same in all the experiments, the only difference is the resource dissemination policy used to exchange workload information between the RMS machines.

The experiments for the periodic approach have been performed with different information intervals values, because this parameter has a great influence on the information efficiency value. For each T value, the k and S values have been measured to evaluate the different alternatives efficiency. To evaluate the speedup, the elapsed time of the whole application on the RMS federation (with the load balancing algorithm) and on the slowest RMS machine, have been compared.

The on demand policy has been implemented to request information when a task has to be re-allocated. The R and the S values have been measured to evaluate the information efficiency.

And finally, the event-driven approach has been implemented with a three states system in which every RMS machine can be a receiver, a neutral or a sender machine depend-

Table 3. Experiments for the medium RMS federation (N=16)

PERIODIC DISSEMINATION			
T (s)	k	S	ε
1	181	16.22	0.01
5	38	16.04	0.03
10	20	15.73	0.06
20	11	14.89	0.10
30	8	14.53	0.13
60	5	12.43	0.18
120	3	10.17	0.24
240	2	6.59	0.24
300	2	4.49	0.16
ON DEMAND DISSEMINATION			
	R	S	ε
	26	6.82	0.13
EVENT-DRIVEN DISSEMINATION			
Neutral width	E	S	ε
0	58	15.87	0.27
0,1	140	17.18	0.12
0.2	162	13.64	0.08
0.25	150	17.38	0.12
0.3	105	14.90	0.14
0.4	71	13.15	0.18

ing on its workload level. The event that causes an information exchange is a state change on a RMS machine. The experiments have been performed for different state configurations, controlled with the neutral state width, varying from 0 (when there is not neutral state, all the RMS machines are receivers or senders) to 0.4 (when all the RMS machines will be probably neutral systems). This parameter has been studied due to its influence on the number of events occurred during the application execution, and therefore, on the information efficiency. In all the experiments, the E and S values have been measured.

With the obtained results, it can be seen that the best efficiency values for the small RMS federation have been obtained with the periodic dissemination.

In such a small RMS, the speedup values achieved by the three information policies are very similar because all these techniques are able to maintain the RMS information sufficiently updated. The differences come from the number of messages needed to maintain this information. If the information exchange interval is properly fixed for the periodic approach, its efficiency values are the best because this kind of policy is the one that produces less network overhead.

But the choice of this interval has a great influence on the obtained results. From table 2 it is clear that short interval values lead to unacceptable efficiencies due to the great number of messages generated by the periodic dissemination. But, on the other hand,

too large intervals, lead to an efficiency decrease caused by the lack of updated system information and therefore, by the subsequent speedup deterioration. This is specially critical on very dynamic systems, where the continuous state changes made the RMS information obsolete very fast.

But the efficiency results obtained for the medium RMS are quite different. In this case, the on demand policy obtains the worst speedup values due to the information delays introduced by this kind of dissemination. These delays are not very significant on small RMS federations but they become important when the number of machines exchanging their information increases: the RMS machine that requests the information must wait for more machines to reply and this finally leads to a larger application elapsed time.

The speedup worsening makes the on demand policy a bad choice for medium and large RMS federations. And comparing the speedup values obtained with the periodic and event-driven policies, they are still very similar, but the efficiency values of these two approaches are very different. This is due to the lack of scalability of the periodic approaches. While the event-driven dissemination maintains constant its efficiency values when the size of the RMS is increased, the periodic dissemination efficiency decreases dramatically and becomes this alternative inappropriate for medium and large RMS federations. This efficiency decrease can be explained with the dependence of the periodic information efficiency value on the number of RMS machines (table 1).

All these experiments have been performed on a plain RMS federation, where all the machines use the same resource dissemination policy and this dissemination is global, i.e., the machines always exchange their information with all the federation machines. For larger federations, information domains will be probably used. These domains imply that a machine sends its information only to certain RMS systems and not all. And the RMS machines may be organized in some kind of hierarchy. In these situations, the conclusions obtained in this study can be applied to the information exchanges between the machines inside a domain and to the exchange between the machines in different domains or hierarchy branches, if there are any.

5 Conclusions

This paper presents an information efficiency definition which is applicable in all the resources dissemination policies for RMS federations on Grid systems. The main contribution of this paper is the definition of this new metric that allows an exhaustive comparison of the traditional approaches for resources dissemination, periodic, on demand and event-driven. The efficiency definition includes two important issues, such as the benefit obtained using a certain information policy and the number of messages generated by this dissemination. Therefore, this comparison is made in terms of network overhead, scalability and performance improvement.

The implications of this comparison are clear and have been demonstrated with experimental results. The periodic approach is a very good choice for small RMS federations, obtaining the best efficiency values, but it is not an scalable solution and its efficiency decreases dramatically when the RMS federation size increases.

The on demand and event-driven approaches are both scalable (their efficiency is not dependent on the number of machines in the RMS federation). But the first alternative

introduces operation delays and is not able to obtain the efficiency values that can be achieved with the event-driven policy for medium and large RMS federations.

Therefore, the event-driven dissemination policy obtains the better compromise between the network utilization, the scalability and the effectiveness in the information exchange, all very important concerns on Grid systems, in medium or large RMS federations.

Further work will extend the resource dissemination policies comparison to larger RMS federations and will provide complementary conclusions evaluating these policies on a real Grid system.

References

1. Z. Nemeth and V. Sunderam. Characterizing grids: attributes, definitions and formalisms. *Journal of Grid Computing,* pages 9–23, 2003. Kluwer Academic Publishers.
2. I. Foster; C. Kesselman and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science,* 2150, 2001. Springer Verlag.
3. K. Krauter; R. Buyya and M. Maheswaran. A taxonomy and survey of Grid Resource Management Systems for distributed computing. *Software-Practice and Experience,* 2(12), 2001. Wiley InterScience.
4. K. Jun et al. Agent-based resource discovery. In *Proceedings of the IEEE Heterogeneous Computing Workshop,* 1999.
5. J. Basney and M. Livny. Deploying a high throughput computing cluster. In R. Buyya, editor, *High Performance Cluster Computing,* volume 1, 1999.
6. W. Hoscheck et al. Data management in an international Data Grid project. In *Proceedings of the IEEE/ACM International Workshop on Grid Computing,* 2000.
7. S. Fitzgerald et al. A directory service for configuring high-performance distributed computations. In *Proceedings of the IEEE Symposium on High Performance Distributed Computing,* pages 365–375, 1997.
8. M. Neary et al. Javelin 2.0: Java-based parallel computing on the Internet. In *Proceedings of the European Parallel Computing Conference,* 2000.
9. S. Chaplin; J. Karpovich and A. Grimshaw. The Legion resource management system. In *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processsing,* 1999.
10. J. Gehring and A. Streit. Robust resource management for metacomputers. In *Proceedings of the IEEE International Symposium on High Performance Distributed Computing,* 2000.
11. H. Casanova and J. Dongarra. Netsolve: A network-enabled server for solving computational science problems. *International Journal of Supercomputer Applications and High Performance Computing,* 11(3):212–223, 1997. MIT Press.
12. R.Buyya; D. Abramson and J. Giddy. Nimrod/G: An architecture for a resource management and scheduling system in a global computational Grid. In *Proceedings of the International Conference on High Performance Computing in Asia Pacific Region,* 2000.
13. F. Kon et al. 2K: A distributed operation system for dynamic heterogeneous environments. In *Proceedings of the IEEE International Symposium on High Performance Distributed Computing,* 2000.
14. Karl Czajkowski et al. Grid information services for distributed resource sharing. In *Proceedings of the IEEE International Symposium on High Performance Distributed Computing,* 2001.

A Grid Framework for Approximate Aggregate Query Answering on Summarized Sensor Network Readings*

Alfredo Cuzzocrea[1], Filippo Furfaro[1],
Giuseppe M. Mazzeo[1], and Domenico Saccà[1,2]

[1] DEIS Dept. – University of Calabria, Via P. Bucci, 87036 Rende (CS), Italy
{cuzzocrea, furfaro, mazzeo}@si.deis.unical.it
[2]ICAR – CNR, Via P. Bucci, 87036 Rende (CS), Italy
sacca@icar.cnr.it

Abstract. The problem of representing and querying sensor-network data issues new research challenges, as traditional techniques and architectures used for managing relational and object oriented databases are not suitable in this context. In this paper we present a Grid-based architecture that supports aggregate query answering on sensor network data, and uses a summarization technique to efficiently accomplish this task. In particular, grid nodes are used either to collect, compress and store sensor readings, and to extract information from stored data. Grid nodes can exchange information among each other, so that the same piece of information can be stored (with a different degree of accuracy) into several nodes. Queries are evaluated by locating the grid nodes containing the needed information, and choosing (among these nodes) the most convenient ones, according to a cost model.

1 Introduction

There are several application scenarios (such as control systems for environmental catastrophes prevention, traffic network monitoring systems, etc.) where huge amounts of data produced by sensor networks are collected and queried to support both punctual and trend analysis. In order to make sensor data analysis feasible, new data representation techniques and querying algorithms are required. In fact, traditional approaches, mainly coming from the Database research area, cannot efficiently manage sensor network data, as the stream of sensor readings is theoretically unbounded.

In order to support qualitative and statistical analysis, aggregate queries are issued on sensor data. As for OLAP and Data Mining contexts, aggregate queries are particularly "tailored" for data management on sensor networks since they allow us to perform a sort of *global analysis* on readings coming from a set of geographically distributed sensors. One of the most relevant research issues in this context is to make answering aggregate queries as much effective and efficient as possible. For instance,

* This work has been supported by a grant from the Italian Research Project FIRB "*Enabling ICT Platforms for Distributed High-Performance Computational Grids*", founded by MIUR and coordinated by the National Research Council (CNR).

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 144–153, 2004.
© Springer-Verlag Berlin Heidelberg 2004

environmental sensor networks supporting prevention of catastrophic natural events could widely beneficiate of fast answers of queries on environmental parameters of monitored geographic areas.

Following the classification proposed by Bonnet et al. in [2], there exist two approaches for processing queries on sensor network data: the *warehousing approach* and the *distributed approach.* According to the first one, the sensor network is used as data provider and the query processing task is hosted on the server where data is stored; according to the second one, queries can be performed either on a data aggregation server and directly on the sensor network ("in-network query processing"). The latter approach is based on the assumption that sensors have computational capability, so that they can support processing readings.

In our work, we address the sensor-network data querying issue under the assumption that sensors are not equipped by any embedded hardware or sophisticated communication component. Our approach can be classified as a warehousing approach, but we collect readings coming from sensor network in a Grid framework [6] (not on a single server), which has been shown to be particularly suitable for processing sensor data [12]. The use of a distributed system instead of a single host allows us to query data in a "multi-resolution" way, by maintaining several versions of aggregate data across the hosts belonging to the system. Furthermore, we adopt the compression paradigm in the data representation and querying; this allows us to efficiently manage sensor network readings. In fact, for the context of sensor network data analysis, it is more convenient to provide approximate answers rather than exact ones, as the goal is usually give bird's eye view of the monitored environment, provided that approximate answers can be evaluated more efficiently.

In this paper we present a Grid framework for approximate aggregate query answering on sensor network readings. This framework is based on a data exchanging protocol where each node maintains a succinct representation of summarized data located in other nodes. The general technique for supporting approximate query answering on summarized data streams presented in [5] is the core technique of the proposed framework. This technique allows approximate query answering in multiple ways, by using different degrees of resolution and approximation.

The remainder of the paper is organized as follows: in Section 2 we briefly outline related work; in Section 3 we present our technique for supporting data management for sensor networks via approximation. Finally, in Section 4 we draw conclusions and some directions for future work.

2 Related Work

Research has recently devoted a great deal of attention on *Approximate Query Answering* issue, whose aim is to provide approximate answers to queries on databases in order to reduce both disk I/Os needed for accessing data and time complexity of algorithms designed for computing answers. The overall assumption carried by techniques based on such an approach is that providing fast and approximate answers to queries is more convenient than computing exact answers since computing the latter could be resource-intensive in terms of time and space computational requirements. Further-

more, for many specific data-intensive contexts, such as OLAP and Data Mining applications, the use of approximation as the core paradigm for representing and managing data is becoming a critical requirement, mainly for the enormous size and the intrinsically distributed nature of data.

Besides this, aggregate queries support a wide number of perspectives of analysis by means of aggregation operators both of algebraic and holistic kinds. Thus, approximate answering to aggregate queries is a very important research direction in the context of the more general approximate query answering research topic. As a consequence, there exists in literature a wide set of proposals for obtaining fast and approximate answers to aggregate queries. We briefly outline them.

Ioannidis and Poosala [10] introduce the use of histograms for providing approximate answers to set-valued queries. Histograms are statistical-based data structures devoted to obtain a summarized representation of a given data domain. Then, Poosala and Ganti [11] argue to use histograms for estimating aggregate range queries in OLAP. Vitter and Wang [13] propose a wavelet-based technique for providing approximate answers to aggregate queries on multi-dimensional domains. Wavelet-based techniques, also proposed and improved by other authors, had a large following in the approximate query processing research community as they perform better than traditional histogram-based techniques. Hellerstein et al. [9] propose a system for effectively supporting online aggregate query answering based on random sampling-based methods; this system also provides theoretically-based probabilistic guarantees on the accuracy of the answers in terms of confidence intervals. Gibbons and Matias [8] exploit random sampling-based methods for supporting approximate query answering and demonstrate as these methods improve previous techniques, mainly for their computational requirements, which are very low. Ganti et al. [7] extend the general proposal of random sampling introducing the *weighted sampling* scheme that exploits query-workload information to continuously tune a representative sample of data thus improving the accuracy of the approximate answers. Database sampling approach has been more recently further investigated and improved by Chaudhuri et al. by designing more accurate sampling scheme taking into account *data skews* (i.e., asymmetric peak values in the data distributions) and *low selectivity* of queries [4]. So, they generalize the previous approach by generalizing this approach introducing the *dynamic sampling* method [1], that is devoted to dynamically build ad-hoc biased samples for each query, instead of always using the uniform ones, by combining samples selected from a family of non-uniform samples built during the pre-processing phase.

In order to make approximate query answering techniques useful for the context of data management for sensor networks a centralized representation of readings coming from sensors is strictly required. This approach is oriented to support knowledge extraction from sensor network data in terms of qualitative analysis, by means of aggregate queries. As a consequence, readings representation plays a leading role since a summarized and multi-dimensional representation must be obtained in each *Approximate Query Engine Server* devoted to support aggregate query answering on sensor network data. Then, the above-presented techniques can run directly on the summarized data structure. The approach presented in [5] follows such an organization and extends the previous work on approximate query answering on summarized data presented in [3] towards a general framework for managing sensor network data streams.

In more detail, the framework in [5] adopts the technique presented in [3] as core algorithm for obtaining approximate answers to aggregate queries on summarized data in a bi-dimensional fashion and defines a general technique for representing summarized data streams based on the compression paradigm. This allows representing with more detail "recent" data by compressing the "old" ones using synopses data structures. We better describe this approach in Section 3.1.

Bonnet et al. in [2] describe the designing guidelines for sensor database systems and recognize numerous advantages of the distributed approach over the warehousing approach, under the condition of having sensors with computational capability. In particular, they study the *in-network aggregation* and distributed query processing.

Yao and Gehrke in [14] extend the previous guidelines focusing the query processing issue and propose the definition of a *query layer* that improves the capabilities of a generic sensor network, and define a declarative language for efficiently in-network query processing.

Schlesinger et al. in [12] combine the Grid framework with sensor database systems. A query to the system is sent to one of the Grid nodes, where each node holds cached data of other Grid nodes (so, using a replication scheme). In particular, they propose a model describing inconsistencies in Grid organized- sensor database systems, and a technique for finding optimal distributed query plans by deciding on which node of the grid a query should be evaluated.

3 A Grid Framework for Storing and Querying Sensor Network Readings

In this section we describe our Grid-based architecture for storing sensor network data and supporting fast approximate (aggregate) query answering. The architecture is sketched in Fig.1, where the following entities are used:

- *Sensor*: is the basic data source for our system;
- *Sensor Domain* (*SD*): is a collection of sensors which are semantically related to one another (for instance, temperature sensors of the same region);
- *Stream Source* (*SS*): is the host collecting the readings coming from a certain sensor domain;
- *Data Dispatcher* (*DD*): is a service which receives sensor readings from *Stream Sources* and forward them to the *Sensor Stream Databases* which are "interested" in these readings;
- *Sensor Stream Database* (*SSD*): is a database that stores readings coming from a certain set of sensor domains (for instance, SSD_1 stores the readings of *Sensor Domain* SD_1, SSD_2 stores the readings of *Sensor Domains* SD_2 and SD_3, and so on). Furthermore, it contains compressed information (namely, *Snapshot*) on readings generated by sensors belonging to different sensor domains;
- *SSD Locator* (*SSDL*): is a service which accesses the catalog of *Sensor Stream Databases,* where the associations between *SSDs* and *Sensor Domains* are stored;

- *Query Dispatcher* (*QD*): is a service which processes range queries by locating the *SSDs* containing the data involved in the queries, and selecting the *SSDs* to be accessed in order to evaluate the queries effectively (i.e., with a reasonable accuracy) and efficiently;
- *Snapshot Catalog* (*SC*): is a service which accesses the catalog of snapshots; this catalog stores, for each *SSD,* information about the snapshots which are available in it;
- *Snapshot Agent* (*SA*): is an agent executing the snapshot policy, by updating (periodically) compressed data stored in *SSDs* (further details are given in the following).

Briefly, our system works as follows. Sensor readings are stored into the *Sensor Stream Databases.* Each *SSD* is associated to a set of *Sensor Domains,* and contains both an uncompressed and a compressed version of data coming from the related *Sensor Domains.* Data are compressed using the technique presented in [5], which is summarized in the following. Basically, compressed data can be used when answers having a dramatic accuracy are not needed, and fast answers are preferred.

In more detail, sensor readings coming from a given *Sensor Domain* SD_i are collected by the *Stream Source* SS_j associated to SD_i. Then, SS_j invokes a *Data Dispatcher* DD_k that forwards incoming readings to the right *SSDs* (i.e., the *SSDs* storing all data coming from SD_i). In particular, DD_k locates the *SSDs* by invoking the *SSD Locator.* As an *SSD* receives new information, it updates both the detailed and the compressed representation of its data.

Each *SSD* can also contain compressed information on sensor readings coming from *Sensor Domains* other than the ones associated to it. That is, information stored in an *SSD* is possibly replicated in other *SSDs,* but with a different degree of accuracy (i.e., different compression rate). As will be shown later, data replication will be used by *Query Services* to schedule query execution plans.

We now explain how compressed data are updated. Consider two *Sensor Stream Databases* SSD_1 and SSD_2, such that SSD_1 contains a compressed representation of the data stored in SSD_2. The *Snapshot Agent* periodically accesses SSD_2, extracts a compressed version of its up-to-date data, and sends it to SSD_1, which replaces its old compressed version of SSD_2 data with the new one. The *Snapshot Agent* accesses the *Snapshot Catalog* to check which pairs $<SSD_a,SSD_b>$ have a *Snapshot Contract.* Basically a *Snapshot Contract* between SSD_a and SSD_b says that SSD_a is authorized to extract compressed information from SSD_b, and contains parameters such as the frequency of updates (how often can SSD_a access SSD_b to retrieve its data?) and the compression ratio (how much "accurate" should be the snapshot to be extracted from SSD_b accorrding to SSD_a requirements?).

The range queries issued by clients are processed by the *Query Dispatcher,* which accesses both the *SSD Locator* and the *Snapshot Catalog* to locate the *SSDs* which contain either uncompressed or compressed data involved in the query. Then, the *Query Dispatcher* selects the most "convenient" *SSDs* to be accessed in order to evaluate the query. The choice of the most convenient *SSDs* is performed by using a cost model, which is not presented here due to space limitations (here we only focus our attention on architectural issues). Basically, the choice is made by considering

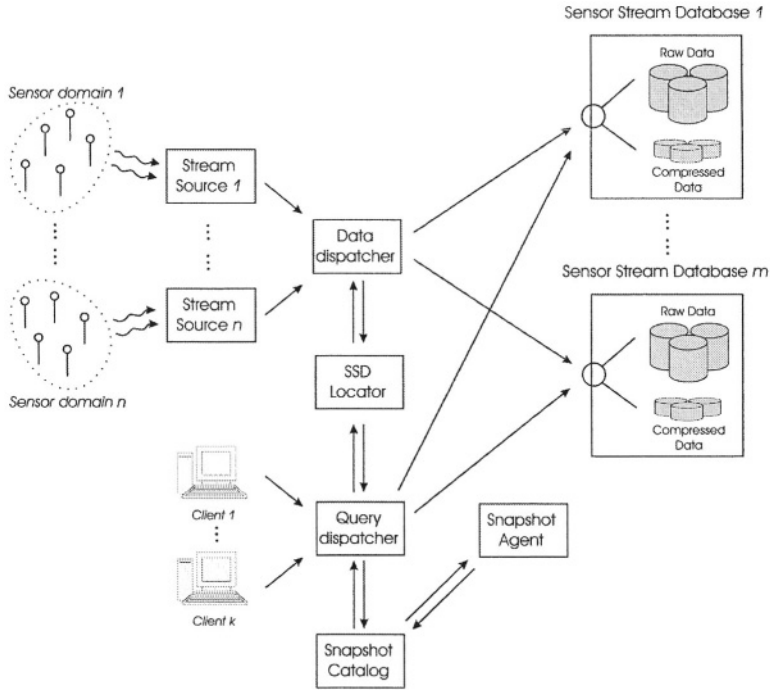


Fig. 1. System Architecture

several parameters, such as the degree of accuracy required by the client, the degree of accuracy of the compressed data available in different nodes, the reachability of nodes (some *SSDs* could be out of order), being all these parameters available in the *SSD Locator* and in the *Snapshot Catalog*.

This approach is intended to support the following distributed query engine based on the *approximation paradigm:* if a Grid node S_i on which the *QD* service is enabled receives a query Q concerning a set of *SSDs Y,* it can alternatively (1) re-direct Q towards each *SSD* belonging to *Y,* (2) answer to Q by using the compressed representation of the involved data distributed across the Grid (the *QD* service queries the *SSDL* service to know this), thus providing less detailed answers, or (3) decompose Q in a subset of queries $\{q_0,q_1,\ldots,q_{N-1}\}$ to be executed in a set of other Grid nodes. This addresses the definition of a *Grid-aware Distributed Query Engine* (*GaDQE*), which should be able to dynamically determine cost-based optimal distributed query execution plans taking into account both "traditional" parameters such as data size, transmission bandwidth, Grid nodes computational power, load balancing etc., and also the approximation paradigm, i.e. the amenity to query a certain *SSD* instead of another one by using the compressed "version" of the original raw data.

We argue that this direction allows us to significantly take advantage from the Grid features and to exploit and successfully use well-known similar techniques and algorithms coming from the Distributed and Parallel Databases research area. More generally, the overall aim of this approach is to define an *Open Grid Directory Service-based Environment* for accessing and querying summarized sensor network readings;

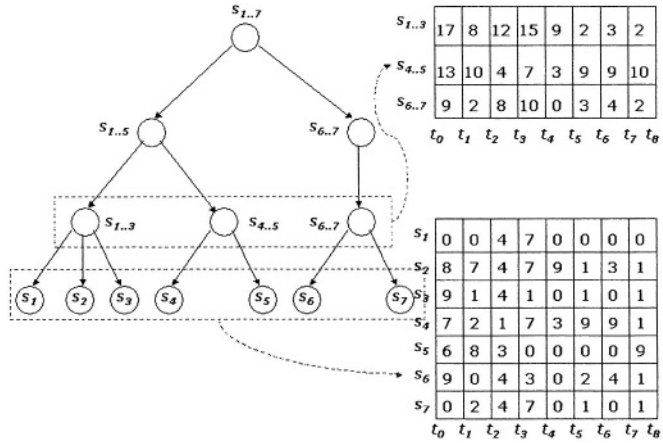


Fig. 2. Multi-Level Aggregation Scheme

this functionality can be implemented as a (Grid) service across the Grid, so that both a user or an application connected and identified on the Grid can discover the raw and the compressed data located in each *SSD* (in a similar way to a *Distributed File System*), browse them, extract from them meta-data concerning their supported approximation degree, and, at last, use the data compression paradigm as well as the core data-layer for supporting large-scale scientific applications based on the Grid infrastructure.

3.1 Representation of Sensor Network Readings

In order to efficiently support aggregate query answering, we adopt a bi-dimensional aggregation scheme for representing summarized sensor network readings, in which the first dimension represents the time and the other one represents the sensors. This scheme is used as a basis for the compression technique, which is described in Section 3.2. The temporal dimension can be aggregated by adopting several strategies, as time is naturally ordered; on the contrary, the sensor dimension is not naturally ordered. Indeed, for many real-life applications on sensor network data, defining an aggregation hierarchy is mandatory. As a consequence, we store a general tree representing the *Aggregation Hierarchy (AH)* defined on the sensor domain, thus obtaining a *Multi-Level Aggregation Scheme* (see Fig. 2).

Each level of the *AH* corresponds to a different aggregation degree on the summarized data, thus exploiting the multi-resolution way to represent and query the examined data.

3.2 Approximate Aggregate Query Answering on Summarized Data Streams

Aggregate querying summarized data streams (like as readings coming from a sensor network) has a leading role in our framework. More generally, the issue of defining

new query evaluation paradigms to provide fast answer to aggregate queries is very relevant in the context of sensor networks. In fact, the amount of data produced by sensors is very large and grows continuously; on the other hand, queries need to be evaluated very quickly in order to extract useful knowledge from sensor readings and perform a "timely" reaction to the physical world. Furthermore, the major benefit of such a query model on sensor readings is represented by the amenity of performing continuous queries, which retrieve an up-to-date "snapshot" of the monitored world continuously, as time passes and new readings are collected. For instance, a climate disaster prevention system would benefit form the availability of continuous information on atmospheric conditions in the last hour. Similarly, a network congestion detection system would be able to prevent network failures by exploiting the knowledge of network traffic during the last minutes.

In order to provide fast approximate answers to aggregate queries on sensor data streams we adopt the general framework proposed in [5]. Briefly, this proposal is based on a hierarchical summarization of data streams, called *Quad-Tree Window (QTW),* embedded into a flexible indexing structure. This structure is also compressed meaning that it is updated continuously, as new sensor readings arrive, and when the available storage space is not enough to host new data, some space is released by compressing the "oldest" data. Adopting this approach, the recent knowledge on the system is represented with more detail than the old one. Note that the recent knowledge is usually more relevant to extract for the context of applications on sensor data. Aggregate queries are the most useful kind of queries for such data. To obtain an efficient query engine on data streams, an "ad-hoc" data structure is needed for representing summarized data. In our proposal, data streams are collected and organized into a bi-dimensional array, where the first dimension corresponds to the set of sensors and the other dimension corresponds to the time. In more detail, the time is divided into intervals Δt-wide and each cell $(s_i,\Delta t_j)$ of the array represents the sum of all the readings generated by the source s_i during the time interval Δt_j. The time granularity is the critical parameter for such an aggregation data structure. That is, the coarser the granularity, the lower the accuracy of the representation. Thus, time granularity needs to be chosen accurately, depending on the particular application domain monitored by the sensor network. Using this data representation, an aggregate query Q on summarized data streams is defined by two ranges: R_S and R_T; the first one is that on the sensor dimension and the second one is that on the time dimension. The answer to Q, denoted by $A(Q)$, is obtained by applying the aggregation operator (like *Sum, Count, Avg* etc.) required by Q on the readings produced by the sensors belonging to the range R_S and during the time interval defined by R_T. Formally, we denote Q as the pair $<\{s_i,..,s_j\},[t_{start},..,t_{end}]>$. A particular class of range queries is that composed by the range-sum ones, which are devoted to retrieve the sum of the involved sensor readings. Note that the aggregation operator *Sum* is very useful for many interesting data-oriented applications on sensor networks like environmental data analysis and mining, distributed surveillance systems etc. Furthermore, other aggregation operators can be derived by the *Sum* one (for instance, the *Avg* operator can be obtained by the *Sum* and the *Count* ones).

The hierarchical nature of the *QTW* allows us to query data in a "multi-resolution" fashion, that is we can query summarized data using different levels of resolution with

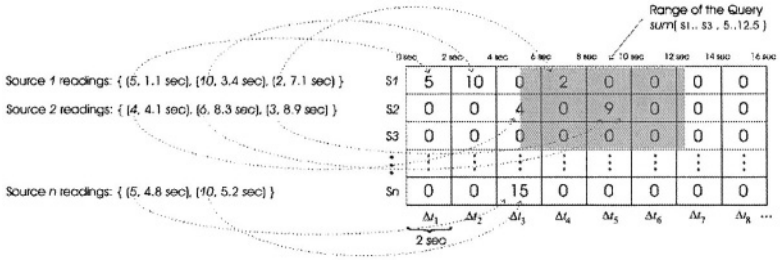


Fig. 3. Bi-Dimensional Representation and Querying of Summarized Data Streams

respect to the sensor domain. This amenity is ensured by the compact *QTW* representation based on the underlying quad-tree data structur; in fact, the quad-tree allows us to aggregate sensors according to a pre-fixed hierarchy and thus represent the summarized data in terms of different levels of this hierarchy. Using this particular data organization we are able to provide fast approximate answers to aggregate queries, as pre-computed sums can be used to speed-up the query evaluation task.

In more detail, the overall data structure representing the summarized data stream is an indexed list of *QTW*s that continuously grows in the time. We have designed a technique where "old" *QTW*s are compressed in order to release some storage space to add new *QTW*s.

Using this representation, an estimate of the answer of a range-sum query Q can be obtained by summing two contributions. The first one is given by the sum of those elements completely contained inside the range of the query. The second one is given by those elements partially contained. Note that the first of these contributions does not introduce any approximation, whereas the second one is approximate, as the use of the time granularity makes impossible to discriminate the distribution of readings generated by sensors within the same interval Δt_j. In more detail, the latter contribution can be evaluated by performing linear interpolation, assuming that data distributions inside each Δt_j are uniform (see Fig. 3).

In [5] we proposed a new technique that allows improving the efficiency of the evaluation task based on the linear interpolation by design an algorithm that takes a bi-dimensional data domain and builds a compressed representation of it, called Index. Depending on the nature of the data distribution of the domain, an Index allows providing approximate answer to range queries with a higher degree of accuracy than the simple linear interpolation. We designed different kinds of Indexes and a cost-based algorithm that dynamically discriminates if the interpolation technique or an Index must be used in the query evaluation. Furthermore, if an Index is needed, the algorithms select and builds the more appropriate one to be used for obtaining approximate answers. We remand to [5] for a detailed presentation of the overall technique.

4 Conclusions and Future Work

In this paper we presented a Grid framework for providing fast approximate answers to (aggregate) queries on sensor network data. It allows querying data in a multi-

resolution fashion, and provides a distributed environment making available multiple views (at different degrees of approximation) on summarized sensor network readings. Future work is mainly focused on the design and the implementation of the algorithm for the cost-based distributed query evaluation, and its test on real data sets.

References

[1] B. Babcock, S. Chaudhuri, and G. Das. "Dynamic Sample Selection for Approximate Query Answers", in *Proc. of the 2003 ACM Int. Conf. on Management of Data,* pp. 539-550, San Diego, CA, USA, June 2003.

[2] P. Bonnet, J.E. Gehrke, and P. Seshadri, "Towards Sensor Database Systems", in *Proc. of the 2nd Int. Conf. on Mobile Data Management,* pp. 3-14, Hong Kong, China, 2001.

[3] F. Buccafurri, F. Furfaro, D. Saccà, and C. Sirangelo, "A Quad-Tree Based Multiresolution Approach for Two-Dimensional Summary Data", in *Proc. of the 15th Int. Conf. on Statistical and Scientific Database Management,* pp. 127-140, Cambridge, MA, USA, July 2003.

[4] S. Chaudhuri, G. Das, M. Datar, R. Motwani, and R. Rastogi, "Overcoming Limitations of Sampling for Aggregation Queries", in *Proc. of the 17th IEEE Int. Conf. on Data Engineering,* pp. 534-542, Heidelberg, Germany, April 2001.

[5] A. Cuzzocrea, F. Furfaro, E. Masciari, D. Saccà, and C. Sirangelo, Approximate Query Answering on Sensor Network Data Streams", to appear in *A. Stefanidis, S. Nittel (eds.), Sensor-Based Distribution Geocomputing, Taylor & Francis Publishers,* 2004.

[6] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", in *Int. Jou. of High Performance Computing Applications,* Vol. 15 (3), pp. 200-222, 2001.

[7] V. Ganti, M. Lee, and R. Ramakrishnan, "ICICLES: Self-Tuning Samples for Approximate Query Answering", in *Proc. of the 26th Int. Conf. on Very Large Data Bases,* pp. 176-187, Cairo, Egypt, September 2000.

[8] P.B. Gibbons, and Y. Matias, "New Sampling-Based Summary Statistics for Improving Approximate Query Answers", in *Proc. of the 1998 ACM Int. Conf. on Management of Data,* pp. 331-342, Seattle, WA, USA, June 1998.

[9] J.M. Hellerstein, P.J. Haas, and H.J. Wang, Online Aggregation", in *Proc. of the 1997 ACM Int. Conf. on Management of Data,* pp. 171-182, Tucson, AZ, USA, May 1997.

[10] Y.E. Ioannidis, and V. Poosala, Histogram-based Approximation of Set-Valued Query Answers", in *Proc. of the 25th Int. Conf. on Very Large Data Bases,* pp. 174-185, Edinburgh, Scotland, September 1999.

[11] V. Poosala, and V. Ganti, "Fast Approximate Answers to Aggregate Queries on a Data Cube", in *Proc. of the 11th Int. Conf. on Statistical and Scientific Database Management,* pp. 24-33, Cleveland, OH, USA, July 1999.

[12] L. Schlesinger, and W. Lehner, "Querying Asynchronously Updated Sensor Data Sets under Quantified Constraints", to appear in *A. Stefanidis, S. Nittel (eds.), Sensor-Based Distribution Geocomputing, Taylor & Francis Publishers,* 2004.

[13] J.S. Vitter, and M. Wang, "Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets", in *Proc. of the 1999 ACM Int. Conf. on Management of Data,* pp. 194-204, Philadelphia, PA, USA, June 1999.

[14] Y. Yao, and J.E. Gehrke, "Query Processing in Sensor Networks", in *Proc. of the 1st Bie. Conf. on Innovative Data Systems Research,* Asilomar, CA, USA, 2003.

Large-Scale Resource Selection in Grids

Ali M. Roumani and David B. Skillicorn

School of Computing,
Queen's University, Kingston, Canada
{roumani, skill}@cs.queensu.ca

Abstract. Grid resource selection requires matching job requirements to available resources. This is a difficult problem when the number of attributes for each resource is large. We present an algorithm that uses the Singular Value Decomposition to encode each resource's properties by a single value. Jobs are matched by using the same encoding to produce a value that can be rapidly compared to those of the resources. We show that reasonable matches can be found in time O(m log n) where n is the number of resources and m the number of attributes for which a job might have requirements. This is in contrast to "approximate nearest neighbor" techniques which require either time or storage exponential in m.

Keywords: Computational grid, distributed resource allocation, nearest neighbor search, multidimensional search, high dimensional data space, Singular Value Decomposition.

1 Introduction

Grid resource selection requires matching jobs with resources that are able to execute them with reasonable dispatch. In most models of grids, this matching is performed by resource brokers who maintain lists of available grid computing nodes (clusters, supercomputers, possibly single processors). When a job is submitted to the grid, one of these resource brokers must find an appropriate location (or set of locations) on which the job will execute. The quality of the matching is important: on the one hand, using a first fit algorithm tends to lead to fragmentation of resources and hence underutilization; on the other hand, there is little point in trying for an absolute best-fit because the information available to the resource broker is stale.

What makes matching difficult as grids grow larger and more complex is that each resource is described by a potentially large tuple of properties, which we will call attributes; and a job may have requirements for any of these attributes. For example, a job may require minimum values for CPU cycles, memory, I/O bandwidth, disk storage, available bandwidth, etc. Let the number of resources be n and the number of attributes m. In a grid setting, it is not implausible for n to be in the thousands and m to be the tens (especially if, for example, the presence of particular software packages or specialized hardware is represented as attributes of a resource).

There is an obvious geometric interpretation of the problem in which each resource and each job are points in an m-dimensional space. The goal is to find the nearest

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 154–164, 2004.
© Springer-Verlag Berlin Heidelberg 2004

neighbor of the job – but with the extra difficulty that the value of each of the resource attributes must be no smaller than the value of the corresponding attribute for the job. Call this the Nearest Positive Neighbor problem.

Given a job, there is an obvious brute force algorithm for finding the best resource with complexity $O(nm)$. This is expensive, given that n could be large and resource allocation decisions are made for each job submitted to the grid.

The contribution of this paper is to show that a tuple of m properties can be encoded by a single value using Singular Value Decomposition (SVD) [1]. Matching a job to an appropriate resource requires encoding the job's requirements, and then searching a ranked list of these values. We show that, in practice, only a constant number of probes of the sorted list is required to find an appropriate resource, rendering the overall complexity of matching to $O(m \log n)$, where $\log n$ arises from binary search of the list. For comparison purposes, we compare the performance of the SVD-based ranking to ranking based on the sum of attributes and to random selection.

2 Related Work

The nearest neighbor problem is used in applications such as knowledge discovery and data mining [2], pattern recognition and classification [3], machine learning [4], data compression [5], multimedia databases [6], document retrieval [7], and statistics [8].

Answering nearest neighbor queries efficiently, especially in high dimensions, is a difficult problem. Many proposals have been made on using different data structures to represent the data and clever algorithms to search them. Herein, we do not attempt a comprehensive survey here.

For a small number of dimensions, simple solutions suffice: when $m = 1$, sorting the list of values and using binary search works effectively; when $m = 2$, computing the Voronoi diagram [9] for the point set and then using any fast planar point location algorithm to locate the cell containing the query point works just as well. However, for larger m, say $m > 10$ which is certainly plausible for pub/sub settings, the complexity of most methods grows exponentially as a function of m. Dobkin and Lipton [10] seem to be the first to give an upper bound for the time required to search for nearest neighbor. Their algorithm achieves a query time of $O(2^m \log n)$ and preprocessing time and storage space of $O(n\text{^}2^{m+1})$. Clarkson [11] improved on that and presented an algorithm with query time $O(f(m) \log n)$ and preprocessing time and space $O(n^{\lfloor m/2 \rfloor (1+\varepsilon)})$, for any $\varepsilon > 0$, where $f(m)$ denotes a function that grows at least as quickly as 2^m. Most of the subsequent approaches and extensions (e.g., [12, 13, 14]) require a query time of at least $\Omega(f(m) \log n)$. Although some of these algorithms claim $O(\log n)$ query time, the O-notation hides factors that are exponential in m.

One of the most widely used techniques is the *k-d* tree [15, 16]. A *k-d* tree is a data structure that partitions space using hyperplanes perpendicular to the coordinate axes, where partitions are arranged hierarchically to form a tree. Although many different versions of *k-d* trees have been devised, they share the basic idea of hierarchically decomposing space into a relatively small number of cells such that no cell contains too many objects, providing fast access to any point by position. Typically, *k-d* trees are

constructed by partitioning the point sets. Each node in the tree is defined by a plane through one of the dimensions that partitions the set of points into left/right (or up/down) sets, each with half the points of the parent node. This process is applied recursively on the children, using planes through different dimensions. Partitioning stops when only a small number of points remain in any cell (log n levels). To search for a nearest neighbor, the k coordinates of the query point are used to traverse the tree until the cell containing the point is found. An exhaustive search is then used to scan through the points inside the cell to identify the closest one. The average case analysis of heuristics using *k-d* trees for fixed dimension d and under restricted cases such as uniformly distributed point sets satisfying certain bounded-density assumptions, requires $O(n \log n)$ for preprocessing and $O(\log n)$ query time.

Although *k-d* trees are efficient in low dimensions, query time grows exponentially with higher dimensionality. The constant factors hidden in the asymptotic running time grow at least as fast as 2^m, depending on the distance metric used. Sproull [17] observed that the empirically measured running time of *k-d* trees increases quite rapidly with dimension. Arya, *et al.* [18] showed that if the set size, n, is not significantly larger that 2^m then boundary effects slightly decrease the exponential dependence on dimension. An exception to this behavior was shown by Meiser [19] with an algorithm that achieves $O(m^5 \log n)$ query time and $O(n^{m+\varepsilon})$ space, for any $\varepsilon > 0$. However, this improvement in query time was at the expense of storage space.

The apparent difficulty in obtaining algorithms that are efficient with respect to both space and query time for dimensions higher than 2 led to alternative approaches that look for approximate nearest neighbors of the query point – a point that may not be the nearest neighbor to the query point, but is not significantly further away from it than the true nearest neighbor. Such approximate solutions can achieve significant improvements in running time (e.g., [20, 21, 22]). However, most such heuristics use significantly larger storage space or have poor performance when the number of dimensions is greater than log n.

Hence, *k-d* trees and similar approaches do not seem viable, either in theory or in practice, for resource allocation. Furthermore, the Nearest Positive Neighbor requirement means that the performance of this class of algorithms becomes even worse. For example, the nearest positive neighbor may be quite far from the query point, with many closer but infeasible resources. After the point closest to the query has been found (requiring time logarithmic in n), searching out for subsequent candidate positive neighbors can require time logarithmic in n for each step. In practice, if not asymptotically, this makes current heuristics expensive for this problem.

2.1 Singular Value Decomposition (SVD)

We have already noted the natural geometric interpretation of a list of tuples describing resources. If we regard such a list as an n x m matrix, then the singular value decomposition can be regarded as transforming the original geometric space into a new one with the following useful property: the first axis of the new space points along the direction of maximal variation in the original data; the second axis along the direction of maximal variation remaining, and so on.

Let A be the n x m matrix representing the resources. Then the singular value decomposition of A is given by

$$A = U S V' \quad (1)$$

where dash indicates matrix transpose, U is an n x r (r is the rank of A) orthogonal matrix (i.e., $U' U = I$), S is an r x r positive diagonal matrix whose elements (called singular values) are non-increasing, $\sigma_1 \geq \sigma_2 \geq \sigma_r > 0$, and V is an orthogonal m x r matrix. Each row of U gives the coordinates of the corresponding row of A in the coordinate system of the new axes (which are defined by V).

The complexity of computing the SVD of a matrix is $O(nm^2 + m^3)$. Since the number of attributes, m, is much smaller than the number of resource tuples, n, in grid applications the complexity in practice is nm^2. The space required to store the data structure is $O(mr + r^2 + rn)$.

One of the most useful properties of an SVD is that the matrices on the right hand side can be truncated by choosing the k largest singular values and the corresponding k columns of U and k columns of V. In particular, the matrix U_k represents each resource in k dimensions – but these dimensions capture as much as possible of the variation in the original data and so are faithful representations of the high-dimensional data in fewer dimensions.

Note that these truncated matrices can be multiplied together as follows:

$$A_k = U_k S_k V_k' \quad (2)$$

and the matrix A_k is the best rank-k approximation to the original matrix.

3 SVD-Based Search (SVD)

SVDS algorithm is based on using SVD for resource ranking. We assume that resources availability and characteristics information can be provided by the information service used in the grid (as in Globus [23]). Using Lightweight Directory Access Protocol (LDAP) concepts, information service can define a hierarchical name space, uniform representation, and standard access methods for resources. The information service entries characterize resources in terms of their types, architecture, and current state (e.g., load, OS, installed software, availability). We also assume that information service contents are kept up-to-date using automated discovery and publication mechanisms. Applications or their agents use query mechanisms to locate resources that meet their resource requirements [24].

We have already commented that the complete set of resources and attributes is an awkward setting to find a nearest positive neighbor. Instead we will use SVD to transform the set of resources into a simpler representation. We can be sure that this representation is, in a strong sense, the best possible for a given structural complexity.

The question of what 'nearest' should mean in this context deserves further comment. The natural metric is Euclidean distance – a resource is a good match for a job if the Euclidean distance between them is small (and the resource has more capacity than the job requires, but not less). However, it is not clear that this simple metric is the most useful in practice – it seems likely that the fit for some attributes will always

be more important than for others. This could be handled by some kind of weighting of attributes (and equivalently dimensions) but the right way to do this is application dependent and will probably require more real-world experience. We will consider the practical problem of finding an approximate nearest positive neighbor (without formalizing what we mean by 'approximate').

Since the first feasible resource encountered in a list ranked by the sum is often almost optimal, we observed that the ranking achieved by using the sums of attributes is quite effective at arranging the feasible resources in order. In the case of real-valued attributes we improved the performance of SVDS by including the attributes sum as another pseudo-attribute, as we will show later. When normalizing the resource attribute values, the sum attribute is weighted as 25% of the rest of the attributes.

We preprocess the set of resources by computing the SVD of the matrix A, truncating the result to 1 dimension (i.e. $k = 1$), and sorting the list by increasing value of U_1. Each element of this list encodes the values of all of the attributes of the corresponding resource in a manner that maximizes the variation among resources – it 'spreads' them as far apart as possible along this newly created dimension.

Each job must now be mapped into the corresponding space, and a value comparable to the encoded resource values created. Rearranging the SVD decomposition equation we see that

$$U = A\,V'\,S^{-1} \tag{3}$$

In other words, points of A can be mapped into the transformed space by multiplying them by $V'S^{-1}$. This same multiplication can be applied to jobs to compute their coordinates in the transformed space. Since we have truncated the SVD at $k = 1$, this mapping requires only the first column of V and the first singular value, and therefore takes time $O(m)$.

This transformation maps a job to a single value that can be compared with the values corresponding to the resources. This is done using binary search to find the resource with the closest value. Since this resource may not be feasible (it is similar to the job but one or more of its attributes is smaller than the corresponding requirement of the job), the list is searched in a zigzag fashion from the original entry until a feasible resource is found.

Algorithms for updating an SVD space dynamically are also known; this is important for grid applications because we expect that resource availability will be updated at grid resource brokers fairly frequently. Hence, the matrix of transformed values needs to be changed as information about resource availability is updated. The computational complexity of updating SVD matrices is $O(nkm)$ (Berry [25]); in our case it is $O(nm)$.

For comparison purposes we also consider a similar algorithm, but using the sum of the attributes as the value used for ranking. The advantage of the sum is that any resource whose sum is smaller than the sum of the requirements of a job cannot possibly be feasible. We compute the sum of attributes for each resource and sort the list based on the sum. The sum is also computed for each job. Then binary search is used to find the resource closest in sum to it, and a feasible resource is searched for in the manner described above. This sum-based search algorithm (SUMS) will in practice perform best when the magnitudes of typical attributes are about the same – this can be arranged by normalizing if necessary.

Both, SVDS and SUM algorithms, have similar properties: both require $O(nm)$ storage for the ranking information (since the full set of attributes must be checked for feasibility); for both, the cost of binary search is $O(m \log n)$, and for both the cost of computing the fit between a job and a resource is $O(m)$. The preprocessing required for SVD is more expensive. However, this cost will be amortized over multiple matches of jobs to resources. The performance difference between the two rankings depends on how many probes are required to find a feasible match and the quality of such a match. In the next section we determine this empirically.

Finally we compare these algorithms with randomly selecting resources until a feasible one is found (call it RAND algorithm). This requires only constant time in n but we would not expect this match to be a good one.

4 Performance Evaluation

For the purpose of assessing the quality of a solution between the three different algorithms, we use the Euclidean squared distance from each solution to the optimal one (computed by running exhaustive search) as the metric for measuring proximity. We determine the number of probes required by simulation, varying the following parameters:

- Number of resources
- Number of attributes for each resource.

For each combination, the fraction of resources that are feasible for each job are held to approximately 5%. This, in our view, models the most plausible scenario to illustrate the performance challenges of the matching algorithm. If feasible resources are extremely scarce, then exhaustive search is probably the best matching technique, and users may in any case find such a grid too frustrating to use. On the other hand, if the fraction of feasible resources is large, then the grid is hugely underutilized which is also an unlikely scenario.

The feasibility fraction is forced by generating jobs in the following way: approximately 5% of the resources are selected at random and their pointwise (by resource) minimum is generated as a job. This job, by construction, is feasible for *at least 5%* of the resources but could, of course, be feasible for a much larger fraction of the resources. If this happens, a smaller set of initial resources is selected and the process repeated until the feasibility fraction is correct.

We model two different grid settings. In the first, the attributes are regarded as real values representing properties of execution nodes such as number of processors, amount of memory, communication bandwidth, processor speed and so on. Values for attributes are generated using Poisson distributions with different means for each attribute.

In the second setting, attributes are regarded as describing the presence or absence of particular properties of execution nodes such as particular software packages or specialized hardware that a job may require. In this case, binary values are used for the attributes, selected uniformly randomly. Experimental results reported are the averages of 120 runs.

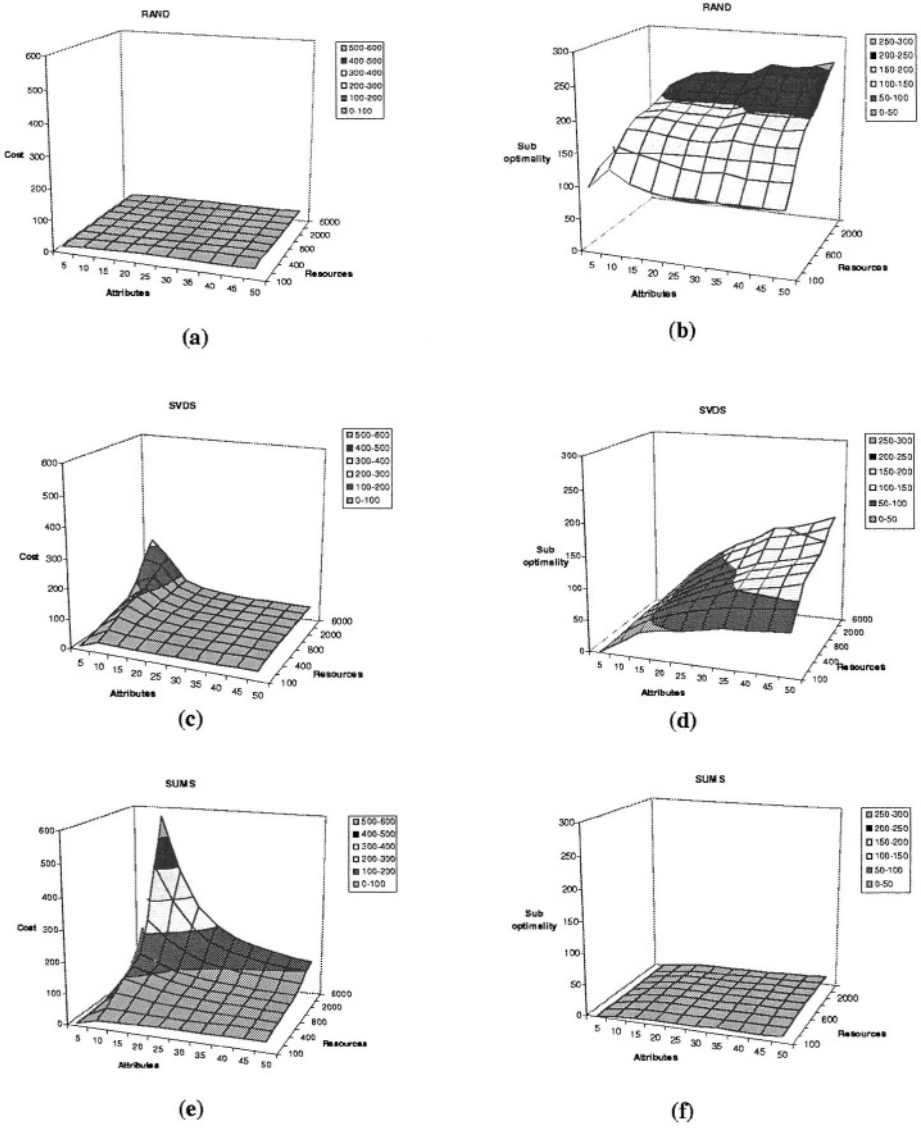


Fig. 1. Search cost and sub-optimality across resources and attributes in the real-valued attribute case. (a) and (b): RAND cost and sub-optimality. (c) and (d): SVDS cost and sub-optimality. (e) and (f): SUMS cost and sub-optimality.

Figure 1 plots the number of probes required to find a feasible resource for a job (left column), and the sub-optimality of this match (right column) when the attributes have real values. Sub-optimality is measured by exhaustively finding the feasible resources, computing which is optimal, then computing the difference in the Euclidean distance from the job to the best feasible resource and the selected feasible resource. Plots (a) and (b) show the number of probes required by RAND and the sub-

optimality for a range of resources and attribute, plots (c) and (d) show the same for SVDS, and plots (e) and (f) show those for SUMS.

As expected, RAND requires only a constant number of probes but finds matches that are less and less close to optimal as the number of resources and the number of attributes increase. SVDS requires slightly more probes than random matching (mainly due to the overhead of the initial binary search) but the number of probes increases only logarithmically as a function of the number of resources. That is, only a constant number of probes are required after the binary search. The quality of the matches found by SVDS is much better, half as expensive in most settings, than those found by random matching. SUMS requires much more probes than either of the other two techniques, and the number of probes increases much faster than SVDS as the number of resources and the number of attributes increases. On the other hand, SUMS finds nearly optimal matches in all settings.

The tradeoff between SVDS and SUMS is subtle when the number of resources and attributes is large. However, for what might be considered the most practical cases, i.e. those where the number of resources is moderately large and the number of attributes is relatively small, SVDS is a clear winner. In this case, SUMS requires a very large number of probes; while RAND produces very poor matches.

Figure 2 shows similar plots for the case when attribute values are binary. As before, the number of probes required by RAND is constant, but the quality of matches found shows a different pattern, with a peak around 15-20 attributes. SVDS requires fewer probes than RAND in most settings and produces better quality solutions in all settings. The sub-optimality of the match found by SVDS increases with increasing numbers of resources and attributes, but plateaus as these become large. SUMS finds matches of better quality than either of the other two techniques, but at expense of many more probes. Overall, SVDS is the preferred technique, especially when the number of attributes is small, since it requires moderate numbers of probes and finds almost optimal matches.

The theoretical properties of SVD suggest that it should always outperform other linear combinations of attributes, of which the sum is the simplest. The SVD approach can be naturally extended to more dimensions, setting $k = 2$ or 3. We have explored such extensions. However, at least on our datasets the overheads of the more expensive search procedure outweigh any gains from more sophisticated ranking.

5 Conclusion

There are several critical research challenges in the development of a computational grid infrastructure and the deployment of computational grid technologies. One of these challenges is the discovery of an effective resource allocation strategy that is scalable, reasonably effective in the presence of stale information, and provides good matches between jobs and available resources. Existing grid resource allocation techniques typically use first-fit matches and are, in any case, used with quite small sets of resources. As the number of resources grids make available increase, solutions that are cheaper to implement but which still provide reasonably good matches are required. To our knowledge, none of the existing matching algorithms stands as an attractive candidate.

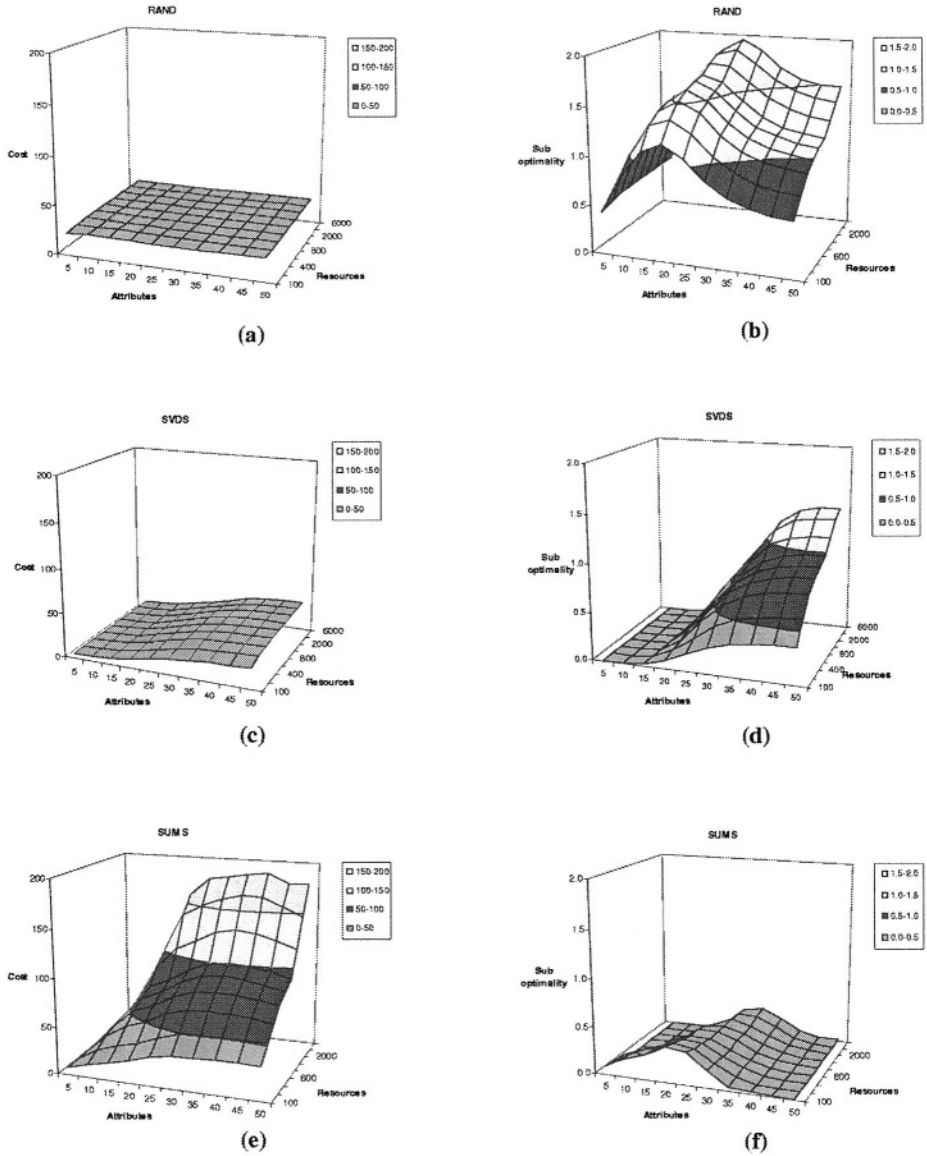


Fig. 2. Search cost and sub-optimality across resources and attributes in the binary-valued attribute case. (a) and (b): RAND cost and sub-optimality. (c) and (d): SVDS cost and sub-optimality. (e) and (f): SUMS cost and sub-optimality.

In this paper we have developed two different techniques for grid resource allocation, SVDS and SUMS. The SVD-based Search algorithm uses SVD in a preprocessing step for transforming the high dimensional space of grid resources into a low dimensional space that is easier to manipulate.

A detailed simulation model was developed to study the performance of the proposed scheme (SVDS) compared to the Sum-based Search (SUMS) scheme and random matching (RAND). Simulation experiments were conducted under different characteristics and grid parameters. The experiments results emphasized on the trade-off between efficiency and quality of the match. SVDS is the technique of choice when resources attributes are binary. When attributes are real-valued, SVDS outperforms RAND in all settings in terms of quality of the match while maintaining moderate cost. SVDS also outperforms SUMS for large numbers of resources and small numbers of attributes.

The theoretical analysis of SVD suggests that SVDS should perform even better than we showed in our experiments. Normalization of the attribute values, which all come from the same family of distributions, essentially arranges them in a hyper-sphere. Without the use of a sum pseudo-attribute, the first dimension of the SVD captures the direction of maximal variation, almost always a random radius of this hyper-sphere and, therefore, does not arrange the resources in an effective order. We hypothesize that SVDS will perform better on real data where the distributions in each dimension may be different. We are currently considering the effect of different distributions for different attributes.

References

[1] G. H Golub and C. F. Van Loan, *Matrix Computations,* The Johns Hopkins University Press, 1993.

[2] U. M. Fayyad, G. Piatetsky_Shapiro, P. Smyth and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining,* AAAI Press/Mit Press, 1996.

[3] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification", *IEEE Trans. Inform. Theory,* 13: pp.57-67, 1967.

[4] S. Cost and S. Salzberg, A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning,* 10: pp. 57-78, 1993.

[5] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression,* Kluwer Academic, Boston, MA, 1991.

[6] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B, Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: The QBIC system", *IEEE Computer,* 28: pp.23-32, 1995.

[7] S. Deerwester, S. T. Dumals, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis", *Journal of the American Society for Inform. Science,* 41(6): pp.391-407, 1990.

[8] L. Devroye and T. J. Wagner, *Nearest neighbor methods in discrimination,* In P. R. Krishnaiah and L. N. Kanal, editors, Handbook of Statistics, Vol. 2, North-Holland, 1982.

[9] M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications,* Springer-Verlag, Berlin, 1997.

[10] D. Dobkin, R. Lipton, Multidimensional search problems", *SIAM J. Computing,* 5: pp.181-186, 1976.

[11] K. Clarkson, "A randomized algorithm for closest-point queries", *SIAM J. Computing,* 17: pp. 830-847, 1988.

[12] P.K. Agarwal and J. Matousek, "Ray shooting and parametric search", *In Proc. 24th ACM Symp. On Theory of Computing,* pp. 517-526, 1992.

[13] A. C. Yao, F.F. Yao, "A general approach to d-dimensional geometric queries", *Proc. 17th Annu. Sympos. Theory Comput.,* pp.163-168, 1985.
[14] J. Matousek, "Reporting points in half-spaces", *In Proc. 32nd IEEE Sympos. on Foundations of Computer Science,* 1991.
[15] J. L. Bentley, B. W. Weide, and A. C. Yao, "Optimal expected-time algorithms for closest point problems", *ACM Trans. on Mathematical Software,* 6(4): pp. 563-580, 1980.
[16] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time", *ACM Trans. on Mathematical Software,* 3: pp.209-226, Sep. 1977.
[17] R. F. Sproull, "Refinements to nearest-neighbor searching in k dimensional trees", *Algorithmica,* 6: pp.579-589, 1991.
[18] S. Arya, D. M. Mount, and O. Narayan. "Accounting for boundary effects in nearest neighbor searching", *In Proc. 11th Annu. ACM Sympos. Comput. Geom.,* pp. 336-344, 1995.
[19] S. Meiser, "Point location in arrangements of hyperplanes", *Information and Computation,* 106: pp.286-303, 1993.
[20] S. Arya, D. Mount, N. Netanyahu, R. Silverman, and A. Wu, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions", *Journal of the ACM,* 45(6): pp. 891-923, 1998.
[21] K. L. Clarkson, "An algorithm for approximate closest-point queries", *In Proc. 10th Annu. ACM Sympos. Comput. Geom.,* pp. 160-164, 1994.
[22] J. Kleinberg, "Two algorithms for nearest-neighbor search in high dimensions", *In Proc. 29th ACM Sympos. on Theory of Computing,* 1997.
[23] Globus: A U.S. multi-institutional research effort that seeks to enable the construction of computational grids, Homepage - http://www.globus.org/, Oct., 2003.
[24] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy, "A distributed resource management architecture that supports advanced reservation and co-allocation", *International Workshop on Quality of Service,* 1999.
[25] M. W. Berry and R. D. Fierro, "Low-rank orthogonal decompositions for information retrieval applications", *Numerical Linear Algebra with Applications,* 3(4): pp.301-328, April 1996.

A Proposal of a CIM-Based Policy Management Model for the OGSA Security Architecture

Félix J. García Clemente[1], Gregorio Martínez Pérez[1], Óscar Cánovas Reverte[2], and Antonio F. Gómez Skarmeta[1]

[1] Department of Information and Communications Engineering
{fgarcia, gregorio, skarmeta}@dif.um.es
[2] Department of Computer Engineering
ocanovas@ditec.um.es
University of Murcia, Spain

Abstract. This paper describes an information model based on CIM (Common Information Model) for the policy-based management of the OGSA (Open Grid Services Architecture) security services, and shows a case of study for the authorization service. Currently, OGSA defines a set of security services and presents a management infrastructure based on policies, but there is no definition of a management model to be used over it. The lack of such a model makes difficult to build an efficient, well-integrated security management system. However, a model based on a well-recognized standard, such as CIM, enables the management of these security services in a uniform manner. CIM includes schemes for several areas such as *User* and *Security* for the identity and privilege management, and *Policy* for *if-then* rules and their groupings and applicability. Moreover, CIM is extensible and can be easily adapted to new OGSA security services and/or requirements that could be defined in the future.

Keywords: Management Model, OGSA Security Services, DMTF CIM Information Model, Policy-based Management, IETF Policy Architecture

1 Introduction and Rationale

During the past years, Grid Computing has emerged as an important research field with the purpose of providing the shared and coordinated use of diverse resources in dynamic, distributed virtual organizations [1]. On the other hand, Web Services introduce an emerging distributed computing paradigm and define a technique for describing the access to services, methods for accessing these services, and methods for discovering service providers. Recently, both technologies and concepts have started to merge and to benefit from the synergy of both paradigms.

The Global Grid Forum (GGF) [12] presented Open Grid Services Architecture (OGSA) as the fusion between Grid Computing and Web Services. The goal of OGSA is to take advantage of Web Services properties such as description and advertisement of services, binding of service descriptions and protocols, and compatibility with higher-level open standards, services and tools. To achieve it, OGSA uses the Web Service Description Language (WSDL) [2] to define a Grid Service as a Web Service. WSDL facilitates service lifetime management, service description, service discovery, interoperability with protocols, and other features.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 165–174, 2004.
© Springer-Verlag Berlin Heidelberg 2004

Moreover, Grid Services require security mechanisms, e.g. to guarantee the confidentiality and integrity of exchanged messages, or for authentication purposes. The OGSA Security Architecture [3] identifies the security requirements in a Grid environment, and based on them, defines a security model to secure Grid services.

Grids, as any computing environment, require some degree of system management, and specially security management. It is a potentially complex task given that resources are often heterogeneous, distributed, and cross multiple management domains.

Currently, the Common Management Model Work Group (CMM-WG) [7,17] is working in the specification of a management framework for OGSA. CMM-WG points to CIM as an interesting model for the management of the security services [5], but it does not include any further work in this line. This paper introduces a proposal of a policy-based management model based on CIM for OGSA security services, and shows a case of study for the authorization service. Our proposal makes use of the policy-based management architecture proposed by the IETF [8] and it is based on a previous design and implementation on Globus Toolkit 2.4 [6].

This document is structured as follows. Section 2 provides a overview of the different components of the OGSA Security Architecture, and section 3 presents the different levels of management in OGSA, focusing on security management. Then, section 4 presents DMTF CIM and the CIM schemes related to users, security and policies, which will be used in section 5 to describe our proposed CIM-based management model for OGSA security services. Then, section 6 shows a case of study for the authorization service. Finally, we conclude the paper with our remarks and some future directions derived from this work.

2 OGSA Security Architecture Overview

The OGSA security architecture is defined in [3]. This document presents a security model formed by the set of security components that are shown in figure 1.

In this layering, top components such as secure conversation, credential and identity translation, access control enforcement, and audit and non-repudiation, are application-specific components that depend on policies and rules for authorization, privacy, identity/credential mapping, and service/end-point. These Grid policies are specified and defined based on a language for policy expression and exchange. In the bottom layer, the security of the bindings is based on the security characteristics of the used transport protocol and message format. On the side, the trust model component defines and establishes trust relationships for Grid environment, i.e. defining VO membership. The secure logging component is a requirement for the auditing of any policy decision.

Finally, the left box groups all security management functions such as key management for cryptographic functions, user registry management, authorization, privacy and trust policy management and management of mapping rules, which enables federation. It may also include the management of anti-virus and intrusion detection services.

The Grid security model can use a variety of existing security technologies and standards, such as IPsec or SSL/TLS in the case of the network and transport layer, or HTTPS in the binding layer, or security standards based on use of XML and assertion

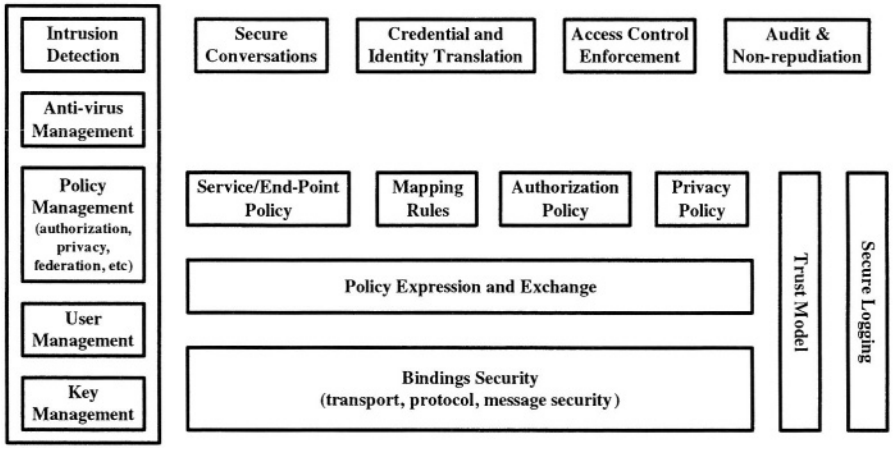


Fig. 1. Components of the Grid Security Model.

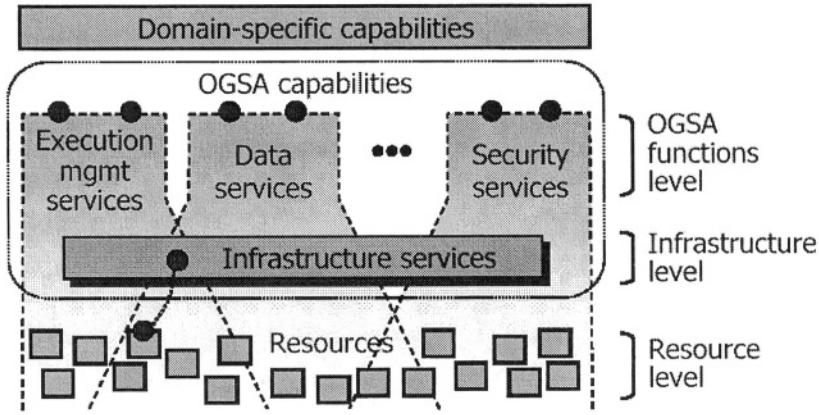


Fig. 2. Levels of management in OGSA.

languages (e.g., SAML [15]) in the message level. This variety can cause that similar security functions can be implemented at different levels. Therefore, to achieve integration and interoperability of the different security technologies, the security functions are defined and exposed as web services (i.e., with a WSDL definition).

3 Management in OGSA

Regarding management in OGSA [5], there are three levels and each level defines a particular management interface. These levels and interfaces, denoted by circles, are illustrated in figure 2.

At the resource level, the resources are managed directly through their native management interfaces (e.g., SNMP, CIM/WBEM, or proprietary interfaces). The infra-

structure level determines the base management behavior of resources, providing the basic management functionality that is common to the OGSA capabilities. Finally, the OGSA functions level provides the specific management interface for the OGSA capability. Note that the interfaces in both the infrastructure and OGSA functions levels are web services, whereas in the resource level is not usual.

At the OGSA functions level there are two types of management interfaces. The functional interface exposes the OGSA capabilities (e.g., create an authorization privilege for a Authorization service, or remove it), whereas the manageability interface provides the management of the capability (e.g., to start or to stop the service). The clear separation between those interfaces is necessary, since different users with different roles and access permissions use them.

4 CIM-Based Policy Canonical Representation

The Common Information Model (CIM) [9] is an approach from the DMTF that applies the basic structuring and conceptualization techniques of the object-oriented paradigm to provide a common definition of management-related information for systems, networks, user, and services. Moreover, it also provides mechanisms to extend those definitions.

CIM consists of a scheme that provides the model descriptions, and a specification that defines the details for integration with other management models. The schema is divided into the following conceptual layers:

- *Core model:* an information model that captures notions applicable to all areas of management.
- *Common model:* an information model that captures notions common to particular management areas, but independent of a particular technology or implementation. Examples of common areas are: application, database, device, network, policy, system, and user.
- *Extension schemas:* represent technology-specific extensions of the common model. These schemas are specific to environments, such as operating systems.

The latest version of the schema, CIM 2.8.1 [9], organizes the common model into thirteen components or models. Three models have been mainly considered to represent the management information for OGSA; they are:

- *System:* this model extends the management concepts that are related to a system, and specially a computer system; it includes software processes, threads, jobs, log information, help services, operating systems, file systems, virtual systems and clusters.
- *User:* this model extends the management concepts related to users and security, including organizations, persons, groups, roles, credentials, identities, levels of security, requirements for authentication, privileges, account management, access control, and Kerberos, public key and shared secret security services and credentials.

– *Policy:* this model provides a framework for specifying configuration and operational information in a scalable way using rules composed of conditions and actions. It includes, among other elements, policy rules, policy groups, and policy conditions and actions, both in generic and vendor-specific form.

An advantage of this approach to information management is that the model can be easily mapped to structured specifications such as XML, which can then be used for policy analysis as well as management of policies across networks. The mapping of CIM to XML is already undertaken within the DMTF [10] defining an XML grammar, written in DTD (Document Type Definition), which can be used both to represent CIM declarations and CIM messages for using the CIM mapping onto HTTP, but this mapping can not be used to represent CIM methods or CIM objects for web services. In the following section, we propose a way to perform this mapping of CIM to a valid representation for web services.

5 CIM-Based Policy Management for the OGSA Security Architecture

Our proposal defines the policy-based management of the OGSA security services as depicted in figure 3, which is based on the policy-based management architecture designed by the IETF [8] and a previous research work on Globus Toolkit 2.4 [6]. This architecture is composed of the four main functional elements now described:

– *Policy Management Tool* (PMT) that allows the administrator, or a user with similar permissions, to develop OGSA security policies making use of the Policy Console, and to monitor the status of the security service by means of a Management Console.

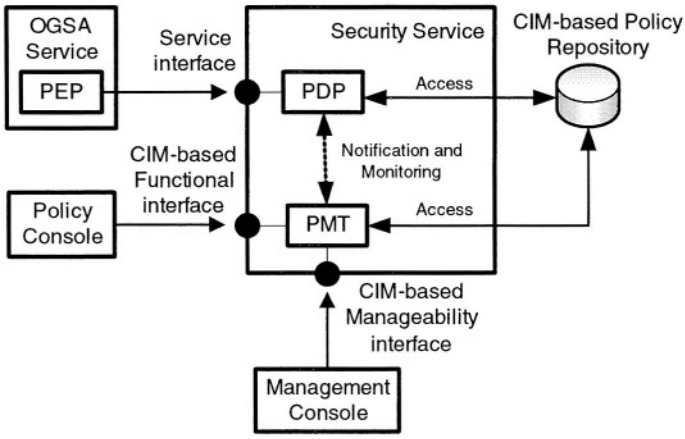


Fig. 3. CIM-based policy management interfaces and architecture.

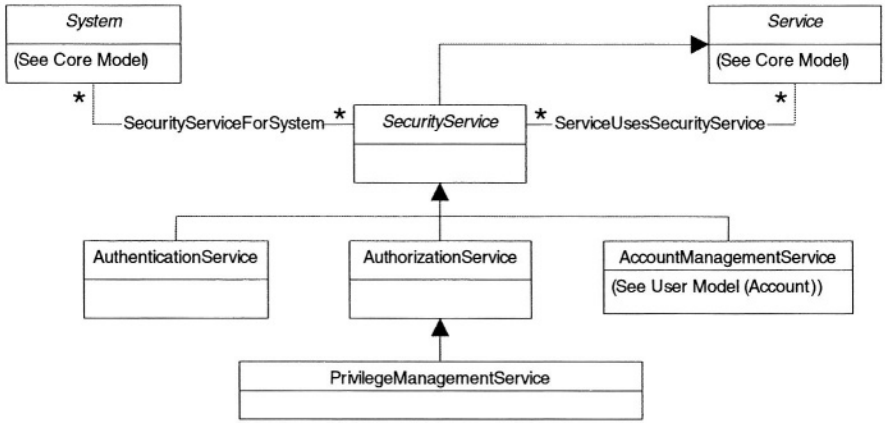


Fig. 4. UML diagram of the SecurityServices classes.

- *Policy Repository* is used by the management tool (PMT) to store the policies and by the decision points (PDPs) to get them. The IETF suggests the use of a Lightweight Directory Access Protocol (LDAP), but other possible solutions, such as other directories or databases are also available.
- *Policy Decision Point* (PDP) is responsible for interpreting the policies stored in the repository, recuperating the set of rules for a particular PEP, transforming them into a format that can be understood by PEP, and distributing them to the PEP. Usually the PDP is responsible for a specific set of policies, for example authorization policies, and then it is called Authorization PDP;
- *Policy Enforcement Point* (PEP) is a component running on an OGSA service that can apply and execute the different policies received from the PDP.

A security service provides three different interfaces: one is the service interface that provides the service specific functionality (out of the scope of this paper), and the two others are the management interfaces (the main target of this paper). We make use of CIM for the different reasons commented above, to model these two management interfaces.

For it, three different CIM classes (AuthenticationService, AuthorizationService and AccountManagementService) have been defined (see figure 4 for their interrelations), each one representing a different security service. The class SecurityService can be extended in the future to add new classes that can model other security services not yet supported (e.g., PrivacyService).

The proposed architecture is independent of any security service, so it could be used in the implementation of the Identity mapping service, the Privacy service, or other Security services. In the following section, a case of study focused on the Authorization management service is shown.

Also note that the mapping of CIM to a valid representation for web services is beneficial, since it permits to model a web service using the DMTF methodology and hence obtain its representation. Therefore, we propose the mapping of CIM objects to XML, which defines the internal representation of the policies that will be stored in the Policy Repository, and the mapping of CIM methods to WSDL, which defines the management interfaces provided by the PMT.

The following section shows how these specific mappings have been defined for the Authorization service.

6 Case of Study: OGSA Authorization Security Service Policy Management

CIM defines the classes showed in figure 5 to represent the management concepts that are related to an authorization security service. Privilege is the base class for all types of activities, which are granted or denied to a subject by a target. AuthorizedPrivilege is the specific subclass for the authorization activity.

Whether an individual Privilege is granted or denied is defined using the PrivilegeGranted boolean. The association of subjects to AuhorizedPrivileges is accomplished explicitly via the association AuthorizedSubject. The entities that are protected (targets) can be similarly defined via the association AuthorizedTarget. Note that AuthorizedPrivilege and its AuthorizedSubject/Target associations provide a short-hand, static mechanism to represent authorization policies.

On the other hand, the service PrivilegeManagementService, shown in figure 4, is responsible of creating, deleting, and associating AuthorizedPrivilege instances. PrivilegeManagementService has two methods defined as follows:

```
AssignAccess(
    [IN] Subject: ref ManagedElement {required},
    [IN] PrivilegeGranted : boolean,
    [IN] Activities: uint16[ ] {enum},
    [IN] ActivityQualifiers: string[ ],
    [IN] QualifierFormats: uint16[ ] {enum},
    [IN] Target: ref ManagedElement {required},
    [IN, OUT] Privilege: ref AuthorizedPrivilege):
    uint32 {enum}
RemoveAccess(
    [IN] Subject: ref ManagedElement,
    [IN] Privilege: ref AuthorizedPrivilege,
    [IN] Target: ref ManagedElment): uint32 {enum}
```

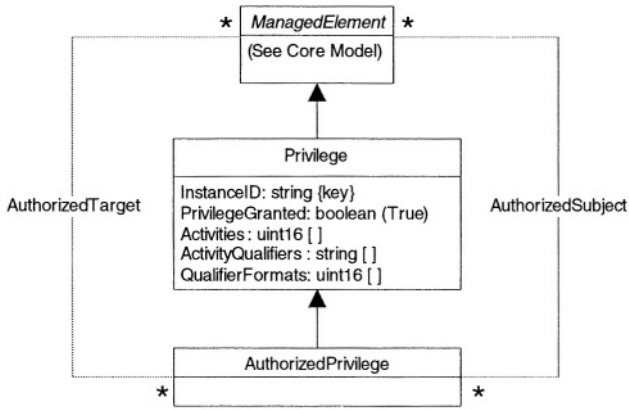


Fig. 5. UML diagram of User-Authentication classes.

The method AssignAccess updates the specified Subject rights to the Target according to the parameters of this call. The rights are modeled via an AuthorizedPrivilege instance. The parameter Activities specifies the activities such as Execute, Create, Delete, Read, Write, and others, to be granted or denied. And the parameters ActivityQualifiers and QualifierFormats specify the activity qualifiers and the qualifier formats used, respectively. The method RemoveAccess revokes a specific AuthorizedPrivilege for a particular target, subject, or subject/target pair.

Both the CIM objects and CIM methods can be mapped to structured specifications such as XML and WSDL, respectively. The mapping of the CIM object to XML defines the internal representation of the authorization policies that will be stored into Policy Repository.

For this case of study, the following example is the mapping of CIM class Privilege to the XML specification (i.e., a XML Scheme Definition (XSD) type [13]):

```
<xs:complexType name="CIM_Privilege">
<xs:complexContent><xs:extension base="CIM_ManagedElement">
    <xs:sequence>
        <xs:element name="InstanceID" type="xs:string"/>
        <xs:element name="PrivilegeGranted" type="xs:boolean"/>
        <xs:element name="Activities" type="xs:uint16"/>
        <xs:element name="ActivityQualifiers" type="xs:string"/>
        <xs:element name="QualifierFormats" type="xs:uint16"/>
   </xs:sequence>
</xs:extension></xs:complexContent>
</xs:complexType>
```

As you can see, the CIM class is mapped in a XS type extending the type ManagemendElement, and each class parameter is mapped into a different XS element. The ManagementElement and basic types (i.e., string, Boolean, and uint16) are defined in other XS documents.

On the other hand, the mapping the CIM methods to WSDL defines the functional interface for the service management. This is a simplified fraction of the WSDL document derived from the CIM methods:

```
<message name="getRemoveAccessRequest">
   <part name="Subject" type="xs:RefManagedElement"/>
   <part name="Privilege" type="xs:RefAuthorizedPrivilege"/>
   <part name="Target" type="xs:RefManagedElement"/>
</message>

<message name="getRemoveAccessResponse">
   <part name="value" type="xs:uint32"/>
</message>
<portType name="PrivilegeManagementService">
  <operation name="AssignAccess">
    <input message="getAssignAccessRequest"/>
    <output message=" getAssignAccessResponse"/>
  </operation>
  <operation name="RemoveAccess">
    <input message="getRemoveAccessRequest"/>
    <output message=" getRemoveAccessResponse"/>
  </operation>
</portType>
```

Note that the service interface in this case of study (i.e. the authorization security service) can be defined with a combination of various specifications that are addressing different aspects of this functionality, including SAML, XACML, and WS-Authorization. [14, 15, 16].

7 Conclusions and Future Work

This paper contributes to the definition of a policy model for the management of security services in OGSA (Open Grid Services Architecture), and to the integration of the DMTF standard management initiative to provide a common definition of management information in the global standardization effort currently undertaken by the Grid community.

CIM can be used for the definition of the management interfaces and the policy internal representation for a Security service. Furthermore, the use of CIM and an IETF-based policy-based management architecture facilitates the implementation of the management service.

As a statement of direction, we are currently working on the definition of new Security services in CIM (e.g. Privacy) and finishing the implementation of the proposed policy-based management architecture.

Acknowledgments. This work has been partially funded by the EU POSITIF (Policy-based Security Tools and Framework) IST project (IST-2002-002314).

References

1. I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In *Proceedings of Euro-Par,* 2001.
2. I. Foster, C. Kesselman, J. M. Nick and S. Tuecke. *The Physiology of the Grid.* Open Grid Service Infrastructure Working Group, Global Grid Forum, 2002.
3. N. Nagaratnam, P. Janson, J. Dayka, A. Nadalin, F. Siebenlist, V. Welch, I. Foster, and S. Tuecke. *The Security Architecture for Open Grid Services.* OGSA Security WG, Global Grid Forum, 2003.
4. F. Siebenlist, V. Welch, S. Tuecke, I. Foster, N. Nagaratnam, P. Janson, J. Dayka, and A. Nadalin. *OGSA Security Roadmap.* OGSA Security WG, Global Grid Forum, 2002.
5. F. Buchholz. *Resource Management in OGSA.* Common Management Model WG, Global Grid Forum, 2004.
6. F. J. Garcia, O. Cánovas, G. Martínez, and A. F. Gómez-Skarmeta, "Self-configuration of grid nodes using a policy-based management architecture", Active and Programmable Grids Architectures and Components (APGAC'04) Workshop, June 2004.
7. OASIS WSDM TC, Web Services Distributed Management: Management Using Web Services. *Committee Draft,* 2004.
8. IETF – Policy Framework (policy) Working Group. *http://www.ietf.org/html.charters/policy-charter.html.* WWW, 2004.
9. DMTF – Common Information Model (CIM) Standards. *http://www.dmtf.org/standards/cim.* WWW, 2004.
10. DMTF – Web-Based Enterprise Management (WBEM) Initiative Standards. *http://www.dmtf.org/standards/wbem.* WWW, 2004.

11. HP – Web Services Management Framework. http://devresource.hp.com/drc/specifications/wsmf. WWW, 2004.
12. GGF – The Global Grid Forum. *http://www.gridforum.org.* WWW, 2004.
13. W3C – XML *Scheme.http://www.w3.org/XML/Schema.* WWW, 2004.
14. OASIS – eXtensible Access Control Markup Language TC. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml. WWW, 2004.
15. OASIS – Security Services TC. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security. WWW, 2004.
16. GGF – Grid Security Area. https://forge.gridforum.org/projects/sec. WWW, 2004.
17. GGF – Common Management Model WG. http://forge.gridforum.org/projects/cmm-wg/. WWW, 2004.

A Grid-Based Approach for Processing Group Activity Log Files

Fatos Xhafa[1], Santi Caballé[2], Thanasis Daradoumis[2], and Nan Zhou[3]

[1] Dept. of Languages and Informatics Systems, Polytechnic University of Catalonia
Jordi Girona Salgado 1-3, 08034 Barcelona, Spain
fatos@lsi.upc.es

[2] Open University of Catalonia, Department of Information Sciences
Av. Tibidabo, 39-43, 08035 Barcelona, Spain
{scaballe,adaradoumis}@uoc.edu

[3] College of Information Science & Technology, Drexel University
3141 Chestnut Street, Philadelphia, PA 19104-2875
nan.zhou@cis.drexel.edu

Abstract. The information collected regarding group activity in a collaborative learning environment requires classifying, structuring and processing. The aim is to process this information in order to extract, reveal and provide students and tutors with valuable knowledge, awareness and feedback in order to successfully perform the collaborative learning activity. However, the large amount of information generated during online group activity may be time-consuming to process and, hence, can hinder the real-time delivery of the information. In this study we show how a Grid-based paradigm can be used to effectively process and present the information regarding group activity gathered in the log files under a collaborative environment. The computational power of the Grid makes it possible to process a huge amount of event information, compute statistical results and present them, when needed, to the members of the online group and the tutors, who are geographically distributed.

1 Introduction

In the online collaborative learning teams, monitoring, awareness and feedback during the group activity are key factors in determining group functioning and task performance and, hence, the success of the learning outcome. Indeed, it is crucial to keep the group members informed of the progress of their peers in performing the learning exercise both as individuals and as a group. It is also important for group members to be aware of the extent to which other members are participating in the collaborative process as this will influence their decision making [1]. Collaborative learning also involves a tutor, who is responsible for acquiring information about students' problem-solving behavior, group processing [2] and performance analysis [3]. To this end, researchers have tried to provide learning teams and tutors with tools and approaches that facilitate monitoring and providing awareness and feedback to support the group

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 175–186, 2004.
© Springer-Verlag Berlin Heidelberg 2004

activity. Such approaches [4], [5] usually rely on processing group activity data from different sources.

Computer Supported Collaborative Learning (CSCL) applications are characterized by a high degree of user-user and user-system interaction and hence generate a huge amount of information usually maintained in the form of event information. In order to make this information useful to the group activity, it must be appropriately collected, classified and structured for later automatic processing by computers as part of a process of embedding information and knowledge into CSCL applications (Fig. 1). The aim is to extract essential knowledge about the collaboration and to make it available to users as awareness and feedback.

Asynchronous collaboration is an important source of group activity data. Data collected during the online collaborative learning activity is then classified and structured into group activity data log files. In order to constantly provide group participants with as much awareness and feedback as possible, it is necessary to efficiently process these log files so that the extracted data can be used for computing statistical results, which can be presented to group members whenever needed.

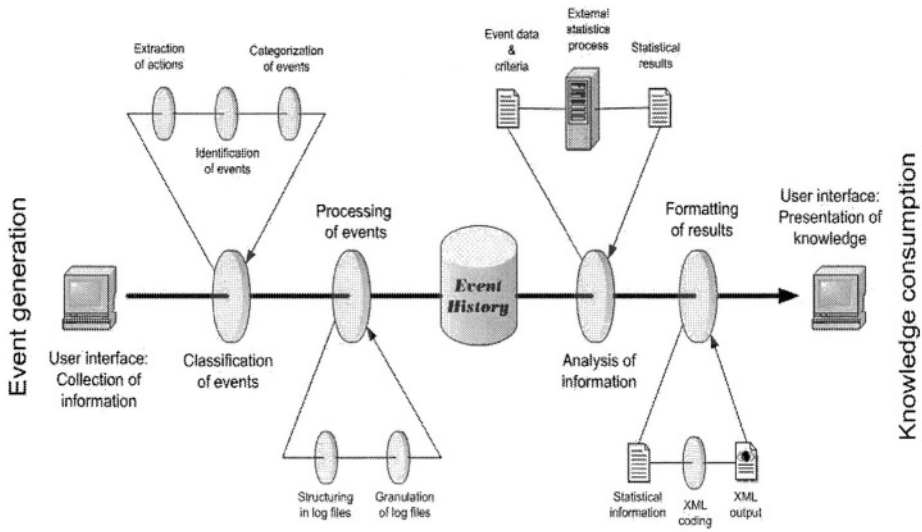


Fig. 1. The process of embedding information and knowledge into CSCL applications

Due to the huge amount of event information gathered in data log files, the processing requires considerable computational resources beyond those of a single computer. In order to make the extracted information available in real time it is necessary to reduce the computational time to acceptable levels. As there may be hundreds or even thousands of students distributed in teams as well as many tutors conducting the learning exercises, the amount of data produced will place great demands on the computational resources given that, in such situations, we need to process the information related to particular classrooms, specific teams within a classroom and even to certain phases of the learning exercise. Our experience at the Open University of Catalonia

[6] has shown the need to monitor and evaluate real, long-term, complex, collaborative problem-solving situations through data-intensive applications that provide efficient data access, management and analysis. The Virtual Math Teams (VMT) Project[1] at Drexel University [7] aims to develop the first application of digital libraries to small group collaborative learning, which requires the processing of a large volume of collaborative activity log files.

The lack of sufficient computational resources is the main obstacle to processing data log files in real time and in real situations this processing tends to be done later, which as it takes place after the completion of the learning activity has less impact on it. With the emerging Grid technology such a handicap can be overcome by using its computational power. The concept of a computational Grid [8] has emerged as a way of capturing the vision of a network computing system that provides broad access not only to massive information resources, but to massive computational resources as well. There is currently a lot of research being conducted on how to use the Grid computing paradigm for complex problem solving [9], processing huge amount of data in biology and medicine, simulations, and collaborative systems. For such problems, putting together distributed computing and storage resources is clearly of great value. Moreover, different technologies such as Globus [10], MPI-Grid2 [11], Condor-G [12], NetSolve [13] and frameworks such as Master-Worker Framework on computational Grid [14] as well as infrastructures for data-intensive Grid applications [15] have been proposed to support the development of Grid-based applications.

In this paper we propose a Grid-based approach for processing group activity log files in order to make the processed information available to the group members in an efficient manner, to compute statistical results and to present the results to the group members and tutors, who are in different locations, as a means of facilitating the group activity, decision making, task accomplishment, and assessment of the progress of the group etc. Our starting point is the definition of an appropriate structure for the log files designed as a part of a more generic platform for supporting CSCL applications [17]. The purpose is to define the structure of event information to be stored in order to permit the structuring of the event information in log files of different degrees of granularity, e.g. corresponding to a single group or to a whole classroom made up of several groups and/or for a given period of time in the group activity. Later, we show how to use Grid infrastructure through the Master-Worker paradigm for processing the log files resulting in a database ready to be used for statistical computations. Furthermore, the natural parallelism inherent in our data log files, as well as in our analysis procedures, makes it feasible to use distributed resources efficiently. Indeed, a Grid computing environment includes computing and storage resources with diverse capabilities and hence different degrees of granularities in our log files, allowing an efficient use of available resources.

The rest of the paper is organized as follows. We give in Section 2 the context that motivated this research and make reference to other studies in the field. In Section 3 we show the structure of data log file used for gathering the event information generated during group activity. In Section 4 we present the Master-Worker approach for

[1] For more information, see http://www.cis.drexel.edu/faculty/gerry/vmt/index.html

processing log files using the Grid infrastructure and, finally, in Section 5 we draw conclusions and outline ongoing work.

2 Context and Related Work

Our real context refers to group activity at the Open University of Catalonia (Spain) and the Math Forum at Drexel University (USA). The former results from applying the Project-Based Collaborative Learning paradigm to model several online courses, such as "Software Development Techniques". These courses involve hundreds of students, a dozen of tutors and are characterized by intensive collaboration activity due to the complexity of the learning practices.

To implement the collaborative learning activities and capture the group interaction we use the Basic Support for Cooperative Work (BSCW) system, a groupware tool that enables asynchronous collaboration over the web [16]. BSCW records the interaction data into log files, which can be used for interaction analysis and knowledge extraction. However, its centralized architecture does not allow data access, management and the analysis of BSCW log files.

In particular, BSCW does not incorporate functionalities to process the log files nor provides the means to calculate and present statistics results. Moreover, BSCW generates a unique log file at the end of the day, which includes a large volume of data describing the activity of all virtual groups. Given that log files do not classify or structure information in any way, there is no possibility of scaling them up. As a result of this there is no way to access data related to separated workspaces, specific groups, or phases of the learning practice.

In recent years the popularity of distant collaborative learning among students has increased enormously. This semester, for example, we have had more than 500 students distributed in more than 100 virtual groups composed of 4 to 6 members. All the groups worked, mainly asynchronously, during 4 months. Due to the large volume of interaction data generated, the wide geographical distribution of the students, and the limitations of BSCW, a Grid solution for data-intensive applications and data analysis becomes imperative to overcome the above-mentioned problems and provide a more effective service to our students and tutors.

Similarly, the VMT Project has been investigating how small groups of students meet online and solve mathematical problems collaboratively using Synergeia [18] (an extension of BSCW) and other similar systems. This study gives great importance to the processing of transaction logs of the collaboration activities, which is currently done manually. As the size of transaction logs increases, it will become even more necessary to develop a means for the automatic processing of data.

In the context of our research, Grid computing [19], [20] has been used to support the real-time requirements imposed by human perceptual capabilities as well as the wide range of many different interactions that can take place as one of the most challenging issues of collaborative computing support. For instance, Grid computing offers high-throughput and data-intensive computing [17], which greatly facilitate the process of embedding information and knowledge into CSCL applications making it

possible to provide users with real-time awareness and constant feedback. In the literature, however, there has been, to the best of our knowledge, little study aimed at achieving these objectives. As an initial approach, the OCGSA framework [21] proposes an event archiving service, which logs the messages or events communicated between online users of a group instance into a persistent database. However, in proposing the implementation of the functionality, this framework does not offer any methodology which takes advantage of the distributed nature of Grid computing to partition the generated event information for efficient parallel processing.

3 The Structure of Group Activity Log Files

In collaborative learning systems, usual group activity results in a lot of interaction which generates a huge amount of events. Therefore, CSCL applications have to be designed to permit the pre-structuring, classification and partitioning of these large amounts of event information into multiple log files to meet different criteria (e.g. group or time) in order to correctly capture the group activity and increase the efficiency of data processing.

The existing CSCL applications have several drawbacks in structuring the log files that prevent efficient processing. To overcome this, we firstly propose a definition and classification of event information generated in a CSCL system and, secondly, we explain how to store this information in log files according to different criteria with the aim of facilitating its later processing in a Grid infrastructure (see also Fig. 1).

3.1 Definition and Classification of Event Information

The most important issue while monitoring group activity in CSCL applications is the collection and storage of a large amount of event information generated by the high degree of interaction among the group participants. Such a large amount of informational data may need a long time to be processed. Therefore, collaborative learning systems have to be designed in a way that pre-structures and classifies information in order, on the one hand, to correctly measure the group activity and, on the other hand, to increase the efficiency during data processing in terms of analysis techniques and interpretations.

The classification of information in CSCL environments is achieved by distinguishing three generic group activity parameters: task performance (i.e. collaborative learning product), group functioning and scaffolding [17]. Furthermore, in a collaborative learning experience, the group activity is driven by the actions of the participants on the collaborative learning resources, which are aggregated to the user events to form another taxonomy in which we can differentiate, at a high level of abstraction, between active, passive and support user actions (see Fig. 2). Therefore, in CSCL applications there is a strong need for the classification of all types of events generated by user actions according to the three generic parameters mentioned. To this end, a complete and tight hierarchy of events (Fig. 2) is provided to collect and

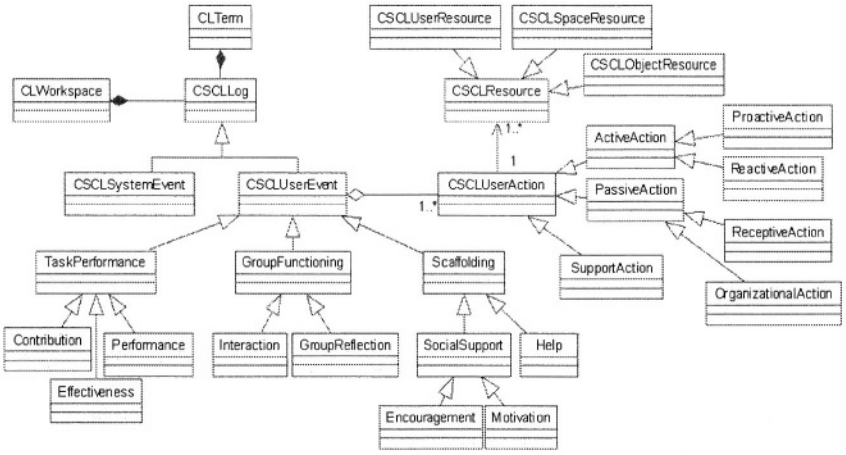


Fig. 2. A hierarchy to collect and classify all events generated during the group activity

categorize the identified events generated by user actions during the collaborative learning activity.

In order to implement this classification, we developed a generic, reusable, component-based library for the construction of specific CSCL applications [17] in which a specific component called *CSCL Knowledge Management*[2] was designed representing the formalization of this hierarchy of events. Note that *CLWorkspace* in Fig. 2 refers to the log file aggregating event information that is generated in a given workspace. Such a workspace may correspond to a whole group or to a phase within a group activity.

3.2 The Structure of the Log Files

In order to prepare the event information for efficient processing, as soon as we classified and turned it into persistent data, we store it in the system as log files, which will contain all the information collected in specified fields. Next, we intend to predefine two generic types of log files according to the two basic criteria, time and workspace, that characterize group collaboration. These log files will represent as great a degree of granularity as possible regarding both criteria and they will be parameterized so that the administrator can set them up in accordance with the specific analysis needs. Thus, the finest grain or the smallest log file should be set up to store all events occurring in each group for the shortest time interval. Therefore, every single workspace will have its own log file made up of all the events occurring within the workspace for a given period of time.

During data processing it will be possible to concatenate several log files so as to obtain the appropriate degree of granularity thus making it possible for a distributed

[2] CSCL Knowledge Management component is found at: http://cv.uoc.edu/~scaballe/clpl/api/

system to efficiently parallelize the data processing according to the characteristics of the computational resources. The aim is to efficiently process large amounts of information enabling the constant presentation of real-time awareness and constant feedback to users during the group activity.

Thus, concatenating several log files and processing them in a parallel way, it would be possible to constantly show each group member's absolute and relative amount of contribution, which would provide participants with essential feedback about the contribution of others as a quantitative parameter supporting the production function. In a similar way, real-time awareness is possible by continuously parallelizing and processing each and every single fine-grained log file of each workspace involved at the same time in order to permanently notify all workspace members of what is going on in their groups. Finally, showing the results of complex statistics after longer periods of time (e.g. at 12 hour intervals) is very important for the group's tutor to be able to monitor and assess the group activity as a qualitative parameter supporting acquisition of information about students' problem-solving behavior, group processing and performance analysis.

4 A Master-Worker Approach for Processing Log Files

The Master-Worker (MW) model (also known as Master-Slave or Task Farming model) has been widely used for developing parallel applications. In the MW model there are two distinct types of processors: master and workers. The master processor performs the control and coordination and assigns tasks to the workers. It also decides what data will be sent to the workers. The workers typically perform most of the computational work. The MW model has proved to be efficient in developing applications using different degrees of granularity of parallelism. Indeed, it has several advantages such as flexibility and scalability (the worker processors can be implemented in many different ways and they can be easily added if needed) as well as separation of concerns (the master performs coordination tasks and the worker processors carry out specific tasks). This paradigm is particularly useful when the definition of the tasks to be completed by the workers can be done easily and the communication load between the master and workers is low.

4.1 Master-Worker Paradigm on the Computational Grid

The MW paradigm has been used in developing parallel applications in traditional supercomputing environments such as parallel machines and clusters of machines. Over the last few years, Grid computing has become a real alternative for developing parallel applications that employ its great computational power. However, due to the complexity of the Computational Grid, the difficulty encountered in developing parallel applications is higher than in traditional parallel computing environments. Thus, in order to simplify the development of Grid-aware applications several high-level programming frameworks have been proposed, among which is the Master-Worker Framework (MWF) [14].

MWF allows users to easily parallelize scientific computations through the master-worker paradigm on the computational grid. On the one hand, MWF provides a top level interface that helps the programming tasks to distribute large computations in a Grid computing environment; on the other hand, it offers a bottom level interface to existing grid computing toolkits, for instance, using the Condor system to provide Grid services. The target applications of MWF are parallel applications with weak synchronization and reasonably large granularity. As we show next, this framework is appropriate for processing log files of group activity since we have different degrees of granularity available so as to guarantee efficiency and, furthermore, there is no need for synchronization or communication between the worker processors. Moreover, in our application, the communication load between the master and workers is very low.

4.2 The Architecture of the Application

The architecture of the application (Fig. 3) is made up of three parts: (1) the Collaborative Learning Application Server, which is in charge of maintaining the log files and storing them in specified locations; (2) the MW application for processing log files and, (3) the application that uses the resulting information in the data bases to compute statistical results and present them to the final user.

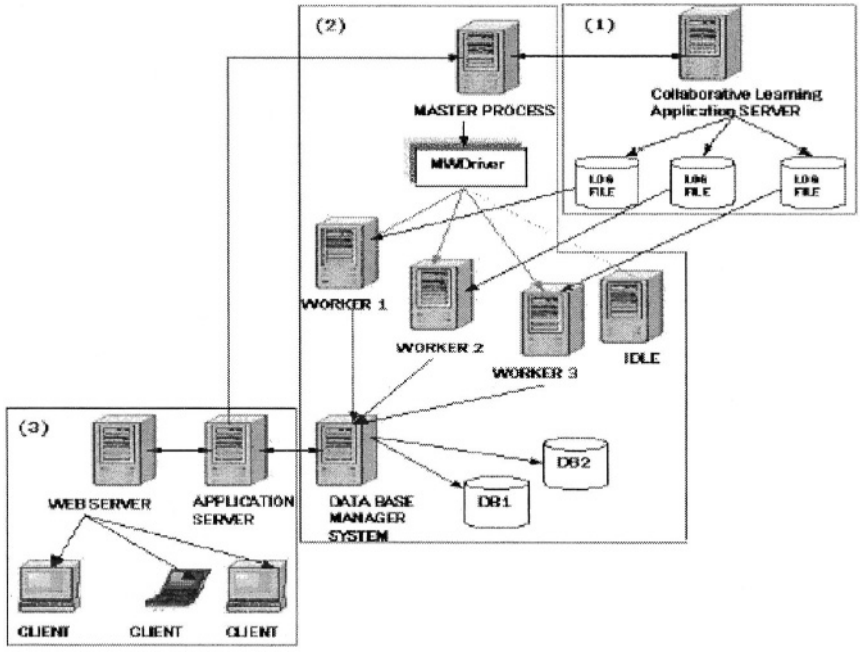


Fig. 3. The architecture of the application for processing log files

The Master-Worker Application for Processing Log Files. We proceed now to present more details of the MW application, basically how the master and worker processors are programmed. The master is in charge of generating new tasks and submitting them to the MWDriver for distributing them to the worker processors while the worker processors run in a simple cycle: receiving the message describing the task from the master, processing the task according to a specified routine and sending the result back to the master. The MW framework, which schedules the tasks, manages the lists of workers and of tasks to be performed by the MWDriver. Tasks are assigned to workers by giving the first task on the list to the first idle worker on the worker list. We take advantage of the fact that the MWDriver's interface allows the task list to be ordered according to a user's criteria and the list of workers to be ordered according to their computational power. Thus, we order the task list in decreasing order of log file size and the machines in decreasing order of processing capacity so that "good" machines have priority in receiving the largest log files.

Furthermore, we have a unique type of task to be performed by the workers that consists in processing a log file. We assume that the workers have the processing routine available; otherwise, the worker would take a copy of the routine on receiving a task for the first time and then use a flag to indicate whether it must receive a copy of the routine or not. The task is described as follows:

Task description:

```
address of the location of the log file;
name of the log file;
size of the log file;
address of the location where the processing routine
 is found.
url of the database where the processed information
 will be stored;
```

The master processor is programmed as follows:

```
while (true) do
  check for new log files generated from the
    Collaborative Learning Application Server;
  update the list of the <log file description>
    for the new incoming log files;
  for each new log file generate a task;
```

We note that the log files generated by the Collaborative Learning Application Server can be stored either in disk spaces of the same server or at different locations (machines) available in the Grid. Furthermore, the processed information by the workers can be stored either in unique or different databases that can be found at different machines as specified in the tasks to be realized by the worker processors.

The worker processor is programmed as follows:

```
receive the task;
receive the specified log file from the specified
 location in the task description;
run the processing routine on the log file
send to the master the task's statistics (execution
 time, number of events processed...) upon
 completion of the task;
```

Efficiency issues of the MW Application. It should be observed that the communication takes place between master and the workers at the beginning and the end of the processing of each task. Therefore, our application has weak synchronization between the master and the workers, which ensures that it can run without loss of performance in a Grid environment. Moreover, the number of workers can be adapted dynamically so that if new resources appear they can be incorporated as new workers in the application; in addition, if a worker in the Grid becomes unavailable while processing a task, the task can be reallocated to another worker. Finally, by having different degrees of granularity of the log files it is possible to efficiently distribute the load balance among workers and minimize the transmission of the data log files from their original locations to the worker machine.

4.3 The Design of the Resulting Database

Once the event information from the log files has been processed, the workers (see Fig. 3) send back the task reports (e.g. processing time, number of processed events, etc.) to the collaborative learning application server through the master so as to verify the results achieved. The results of data processing, which workers send to the database manager system, should have correctly represented all the information contained in the log files so as to make it possible to consult both the desired data from the database directly (e.g. number of connected users, type of documents in a certain workspace, etc.) and the computed complex statistical results from the database. These statistical results should be obtained by the application server as fast as possible and presented to group members and tutors in different formats.

Thus, based on the premises argued in [22], we provide[3] a logic design of the database, which is generic, efficient and independent from any specific database manager. We have designed the database in a way to satisfy all of these requirements and to allow users to consult data regarding the basic entities that take place in any CSCL environment (users, objects, workspaces, connections, etc.).

4.4 XML Representation of the Statistical Results

The third part of our application uses the resulting information in the databases to compute statistical results and present them to the members of the online collabora-

[3] See the design of the database at: http://cv.uoc.edu/~scaballe/GADA04/DBDesign.pdf

tive group and the tutors In this context we are studying an XML coding of the statistical results in order to make it possible to present this information to final users in different forms (see Fig. 1). Considering the fact that the data is highly structured and the design of the relational database [23], we propose that application be designed as a middleware [24], which performs the following functions: to extract necessary information from the databases, to compute statistical measurements as desired, and to convert the results into XML output. This design will provide sufficient flexibility as to allow ad hoc statistical measurements to be obtained as well as permitting the creation of user-specified document type definitions (DTD) to accommodate the different needs of information representation.

5 Conclusions and Ongoing Work

The efficient embedding of information and knowledge about the ongoing group activity into collaborative learning environments is crucial to the success of the online collaborative learning activity. Moreover, disposing of such information as fast as possible makes the use of more computational resources indispensable in order to process a huge amount of event information generated during the ongoing group activity. In this study we have shown a Grid-aware approach for processing log files of group activity in an efficient yet simple manner. Our approach is based on the Master-Worker paradigm on the Computational Grid and shows its feasibility by satisfying several conditions such as weak synchronization, dynamic adaptation of resources, efficient load balancing etc., which ensures that our application can run without loss of performance in a Grid environment. We have achieved this through a careful definition of the event information generated during group activity and an adequate structure for log files that collect the event information. This allows us to dispose of log files according to different criteria as well as different degrees of granularity.

We are currently implementing this application that we will test under a real environment using a grid infrastructure formed by machines at three Spanish Universities (Open University of Catalonia, Polytechnic University of Catalonia and University of Valladolid) within the scope of a joint Spanish Research Project (CICYT). Doing so, we are using real data coming from workspaces of online collaborative learning groups at the Open University of Catalonia and those from the VMT Project at Drexel University.

Acknowledgements. This work has been partially supported by the Spanish MCYT project TIC2002-04258-C03-03 and NSF VMT project (IERI grant #0325447).

References

1. Dillenbourg, P. (ed.) (1999): Collaborative Learning. Cognitive and Computational Approaches. Elsevier Science Ltd. 1-19.
2. Kiesler, S. and Sproull, L.S. (Eds.) (1987). Computing and change on campus. New York: Cambridge Press

3. Daradoumis, T., Xhafa, F. and Marquès, J.M. (2003) Exploring Interaction Behaviour and Performance of Online Collaborative Learning Teams, 9th Int. Workshop on Groupware, CRIWG'03, France. Lecture Notes in Computer Science, Vol. 2806, pp. 126-134.
4. Martínez, A., de la Fuente, P., Dimitriadis, Y. (2003) Towards an XML-Based Representation of Collaborative Action. In: Proc. of the CSCL 2003, Bergen, Norway.
5. Zumbach, J., Hillers, A. & Reimann, P. (2003). Supporting Distributed Problem-Based Learning: The Use of Feedback in Online Learning. In T. Roberts (Ed.), Online Collaborative Learning: Theory and Practice pp. 86-103. Hershey, PA: Idea
6. Open University of Catalonia http://www.uoc.edu (web page as of August 2004)
7. Virtual Math Team Project http://mathforum.org (web page as of July 2004)
8. Foster, I. and Kesselman, C. (Eds) (1999) The Grid 2e, 2nd Edition Blueprint for a New Computing Infrastructure. Morgan-Kaufman
9. Arnold, D., Agrawal, S., Blackford, S., Dongarra, J., Miller, M., Seymour, K., Sagi, K., Shi, Z. and Vadhiyar, S. (2002) Users' Guide to NetSolve V 1.4.1, Univ. of Tennessee, Technical Report, ICL-UT-02-05.
10. Globus: http://www.globus.org (web page as of July 2004)
11. MPICH-G2: A Grid-enabled MPI. http://www3.niu.edu/mpi/ (web page as of July 2004)
12. Condor-G: http://www.cs.wisc.edu/condor/condorg/ (web page as of July 2004)
13. Casanova, H. and Dongarra, J. (1998): NetSolve: Network enabled solvers, IEEE Computational Science and Engineering, 5(3) pp. 57-67.
14. Goux, J.P., Kulkarni, S., Linderoth, J. and Yoder, M. (2000): An enabling framework for master-worker applications on the computational grid. In 9th IEEE International Symposium on High Performance Distributed Computing (HPDC'00). IEEE Computer Society.
15. Pérez, M., Carretero, J., García, F., Peña, J.M., and Robles, V. (2003) MAPFS: A Flexible Infrastructure for Data-Intensive Grid Applications, In 2003 Annual CrossGrid Project Workshop and 1st European Across Grids Conference, Santiago de Compostela, Spain
16. Bentley, R., Appelt, W., Busbach. U., Hinrichs, E., Kerr, D., Sikkel, S., Trevor, J. and Woetzel, G. (1997) Basic Support for Cooperative Work on the World Wide Web. Int. J. of Human-Computer Studies 46(6) 827-846.
17. Caballé S., Xhafa, F., Daradoumis, T. and Marquès, J.M. (2004) Towards a Generic Platform for Developing CSCL Applications Using Grid Infrastructure. In: Proc. of the CLAG/CCGRID'04, Chicago, USA
18. Stahl, G. (2002) Groupware Goes to School, 8th Int. Workshop on Groupware. CRIWG'02, La Sirena Chile. LNCS, Vol. 2440, pp. 7-24. ISBN: 3-540-44112-3.
19. Foster, I. and Kesselman, C. The Grid: Blueprint for a Future Computing Infrastructure. pp. 15-52. Morgan Kaufmann, San Francisco, CA, 1998.
20. Bote-Lorenzo, M. L., Dimitriadis, Y. A., Gómez-Sánchez, E.: Grid Characteristics and Uses: a Grid Definition. In: Proc. of the 1st European Across Grids Conference (CD), Santiago de Compostela, Spain, 2003.
21. Amin, K., Nijsure, S., and von Laszevski, G.: Open Collaborative Grid Services Architecture (OCGSA), In: Proc. of the W3C EuroWeb 2002 Conference, Oxford, UK, 2002.
22. Watson, P. (2003) Databases and the Grid In: Grid Computing: Making The Global Infrastructure a Reality, Berman, F. et al. (eds.) Wiley
23. Fernández, M., Kadiyska, Y., Suciu, D., Morishima, A., and Tan, W. (2002) SilkRoute: A framework for publishing relational data in XML. In ACM Transactions on Database Systems (TODS), Vol. 27, Issue 4.
24. Kyung-Soo, J. A design of middleware components for the connection between XML and RDB. Proceedings of IEEE ISIE 2001, Vol. 3, pp. 1753 - 1756.

Design, Implementation, and Evaluation of a Resource Management Multiagent System for a Multimedia Processing Grid

Isaac Chao, Ramon Sangüesa, and Oscar Ardaiz

Politecnic University of Catalunya, Spain
{ichao, sanguesa}@lsi.upc.es, oardaiz@ac.upc.es

Abstract. This paper proposes marketplace-based agent architecture for Grid Resource Management that works on the application level. Relaying in FIPA as agent systems standard and Globus Toolkit as "de facto" middleware solution for Grid computing, we achieve the desired flexibility and modularity in order to provide a pluggable agent layer for a broader generic Grid Computing framework. The agents in the system use a utility table built previously to the system operation as information source for improving their negotiation abilities. It allows them to, given a state of the resources on the Grid, check the predicted performance of the possible configurations and select the best-rated values for some task execution parameters. We have implemented the architecture using JADE and have performed preliminary experiments consisting on multimedia processing for the conversion between video formats.

1 Introduction

Grid Computing promises to provide high-end computing resources to Grid users regardless of their physical location or the architectural or administrative details of the shared resources [1]. The Grid is conceived to be used by a set of dynamic virtual organizations (VOs) in a secure and coordinated manner, trough the Grid middleware that provides the required infrastructure. The last steps on Grid computing are on converging with Web Services, in order to leverage their characteristics of interoperability and independency from transport protocols, programming languages and system software when describing software components. The Open Grid Services Architecture (OGSA) [2] defines a standard architecture where Grid Services are interoperated to meet the need of VOs. The WS-Resource Framework [3] is the newest effort on meeting Web Services standards.

Despite of the success of Grid Computing on providing solutions for a number of large-scale science and engineering problems, the problem of Grid Resource Management (GRM) still remains an open challenge. GRM problem, as presented in [4], covers the management of user requests, the matching of them with suitable resources through service discovery and the scheduling of the tasks on the matched resources. The high complexity arises from the various issues GRM has to address: resource heterogeneity, site autonomy and security mechanisms, the need of negotiation between resource users and resource providers, handle the addition of new policies, extensibility, fault tolerance and QoS to cite most relevant.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 187–197, 2004.
© Springer-Verlag Berlin Heidelberg 2004

Multi-Agent Systems (MAS) applications to Distributed Computing first devoted on system and network management [5], and actually focuses in other complex issues, like the Semantic Web [6]. The advantages of the application of Software Agents and MAS to Grid computing have been also considered in some papers [7, 8]. Agents are well known for their excellent negotiation abilities, and as Grid systems embrace service-oriented computing, a bigger emphasis is put on trading services between users and providers.

In this paper, we propose an environment based on agents on a marketplace, where agents representing Grid users can detect and meet agents representing Grid Services, negotiate with them for the conditions of the service, and execute the tasks on the Grid node corresponding to the Service. Agents allow flexible negotiation, exchanging messages on behalf of the entities they represent. In our system, agents can use an extra information source for negotiations, consisting on a table built previously to the system execution, which contains the possible grid configurations rated on utility.

The rest of the paper is organized as follows. Section 2 overviews previous research on agents as applied to Grid computing. Section 3 describes system design and architecture. Section 4 details the implementation and system setup. Section 5 describes experiments and section 6 concludes the paper.

2 Related Work

A few systems present agent architectures for the Grid. In [9] we find an adaptive framework for agent negotiations in a Grid environment, providing various negotiation models. Another approach is AgentScape [10], a distributed scalable middleware platform for software agents and services. It can be used to support the integration and coordination of distributed resources in a Grid environment, but still looks like in prototype in this specific application. Perhaps the more evolved system is ARMS [11], where a hierarchy of identical agents acting both as users and providers perform resource discovery and scheduling, and the internal-resource scheduling is performed using PACE toolkit, a module for performance prediction.

Also mobile agents architectures have been proposed for the Grid [12], dealing with resource discovery and performance management in the context of service-oriented Grids, and presenting the mobile agent based programming and execution framework (MAGDA) system. Another example where agents are applied to Web Services is presented in [13], where authors describe the integration of agent technologies with Web Services. They cover the technical aspects of creating, deploying, and publishing agents as Web Services.

The system we present in this paper differs from the previous ones in the following aspects. First we rely on widely adopted standards both on Grid computing and MAS fields. The agent layer is developed using JADE [14] and thus FIPA-compliant [15], ensuring compatibility with any other agent platform that complies with FIPA specifications. The agent layer works on top of Globus toolkit [16], the “de facto” middleware on Grid computing, ensuring OGSI-compliancy [17]. Our focus on standards arises from the firm intention of a future integration of this agent layer in a generic Grid computing framework that is being developed in our group GridCat [18]. Modularity and reusability were the first issues on mind in the system design phase. A

second difference is the fact that this agent system operates at the application level, and specifically designed for Grid environments, as opposed to other presented works in which a generic agent middleware is suggested to be applied to Grid computing. Regarding this second aspect, we find [19] as the nearest work to what we present here.

3 System Design and Architecture

3.1 Overall Architecture

The overall architecture (figure 1) is a heterogeneous MAS, with agents exchanging messages and negotiating in a Marketplace. It is composed basically by three types of agents (excluding administrative agents provided by the agent platform). Broker agent (BA) acting on behalf of Grid users, Seller agents (SA) acting on behalf of Grid resource providers and Marketplace agent (MA), which registers BA and SA in the Marketplace, track negotiations and stores the resulting agreements between traders.

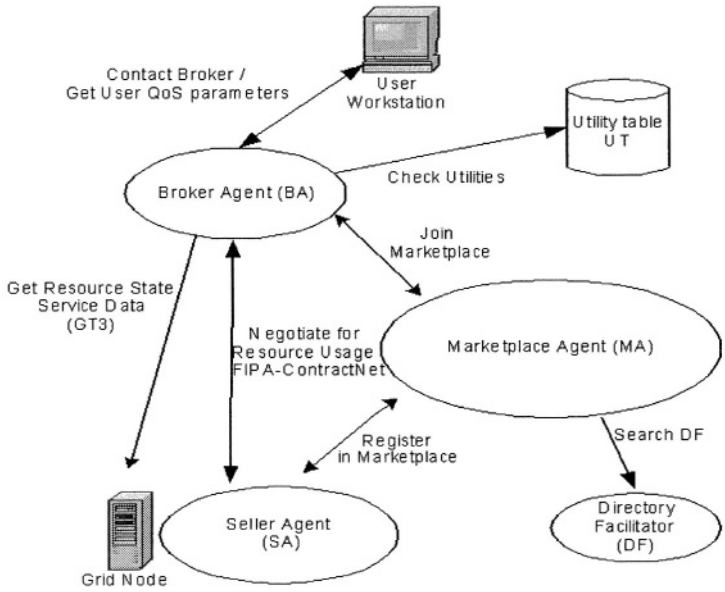


Fig. 1. The Overall Architecture

Apart from the messages exchanged between BA an SA, extra information sources can be used during negotiations. A table (UT) which is built previously to the system operation and which rates the utility of different experiment configurations can be checked by the BA. It gives an estimation on how long is going to take the task execution in the scheduled resources, as well as information on which is the most accurate manner of preparing tasks.

3.2 Broker Agent

The BA is the responsible of the interaction with users on the Grid. It gets from the user the details on the task to execute on the Grid as well as the QoS parameters. Then it contacts the Marketplace to join in and looks for suitable SA looking for jobs to be scheduled in their resources. Next, the Marketplace allows the Broker to negotiate with a number N of registered SAs, and 1 to N negotiation starts between BA and SAs, following the FIPA-Contract-Net Protocol (Figure 2). During negotiations the BA will check the UT for the best configurations on the resources in order to perform the required tasks. Once negotiations are completed, the Broker will check again the information extracted from the UT for the selected resources to decide the values of some task dependent parameters and schedule the Job using the Grid middleware Globus.

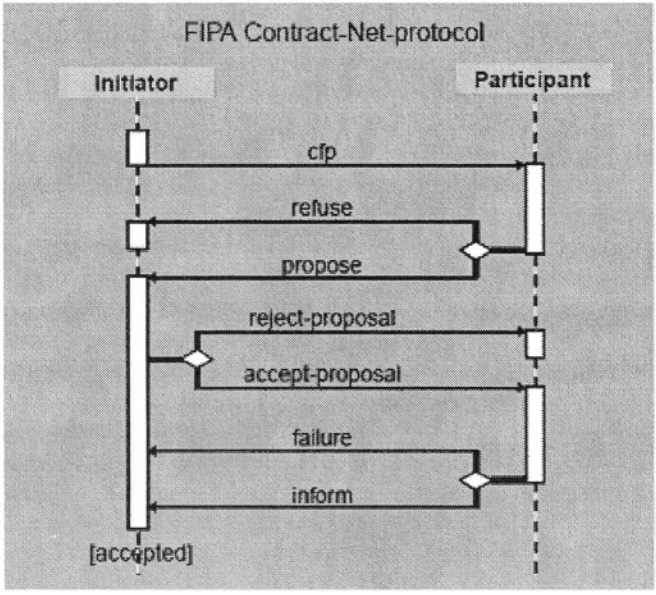


Fig. 2. The FIPA-Contract-Net Protocol Sequence Diagram

3.3 Seller Agent

The SA first registers in the Marketplace to get the permission to register itself in the Directory Facilitator (DF), a kind of yellow pages service provided by the agent framework JADE. Once this operation is completed, the SAs wait to be contacted by a BA interested in starting a negotiation for the resource usage. If the BA conditions on job executions in the resource meet the resource provider requirements, then the SA allows the BA to schedule the agreed jobs in the resource. If for any reason (resource provider specific requirement or node failure) a SA wants to keep off from the Market the resource it represents for a while, it only has to deregister from Marketplace which in turn will allow the SA to delete itself from the DF.

3.4 Marketplace Agent

The role of the MA is the intermediation between BAs and SAs. We could have used a different approach for the Marketplace design, with the MA actually performing the negotiation between agents, much in a centralized manner, but we think that a decentralized approach, with traders negotiating on their own and using Marketplace for contact purposes, suits better to flexibility and independency requirements of a Grid system. With the decentralized approach, BA and SA need to be more intelligent in order to achieve negotiations by themselves, but also the agents gain more autonomy to accomplish the objectives of the users and resource providers on behalf they act to.

The Marketplace has also a role of yellow pages, since any interaction with the DF of the rest of agents has to be requested to the MA, which can accept or deny the request. When a BA and a SA get to an agreement for resource usage at the end of a negotiation, the MA tracks the result of the negotiation to compose a kind of historical record of the negotiations for the Grid usage.

3.5 Utility Table

This UT is built before any agent operation. It is a compendium of the result of the execution of tasks in the Grid for a variety of possible configurations of resources states (activity, load, net traffic) and job parameters (granularity, time between jobs submission) that can be used by the agents to evaluate the performance of the possible executions given a state of the resources on a Grid. The BA checks this UT to get and idea of the utility of the possible executions on jobs in the registered nodes.

3.6 System Operation Example

A possible scenario is the following. A user wants to get some task done before a fixed deadline, and the sooner the best. It contacts a BA, passing it the deadline for the task completion. From here on, all the rest of GRM operations are on charge of the BA till the returning of the result to the user. The BA will contact the MA and check for the registered SAs on the Market. After getting a list of the candidate resources, the BA checks the current resource states, by calling the appropriate GT3 service on the candidate the resources. The service exposes this information in its associated service data. Once it knows the state of the registered resources, the BA checks the UT to know which between the possible configurations is best rated on utility for the user purposes. Then it starts negotiating with the selected SAs, which can in turn accept or reject the user offers, depending on the resource provider requirements (for example if a resource provider don't want its resource to be used by others between 8 a.m. and 22 p.m., the SA wont allow to execute a job in its machine which starts at 6 a.m. and is supposed to take 3 hours long). Once the BA has reached to an agreement with a set of suitable set of SAs for the resources usage, it checks again the UT records to decide which will be the granularity of jobs and the time between the submissions of each of them to the remote resource in order to minimize the time to completion.

4 Implementation and System Setup

4.1 Agents Implementation

The agents are implemented using JADE Framework (check [14] for details), and they behave in a coordinated manner, by means of the sequential execution of their behaviours (fig 3). Each line in the sequence diagram represents a JADE sequential behaviour, where the behaviours are scheduled and completed one after the other in order to achieve the final collective behaviour. The behaviours in which a message is sent to another agent are represented by arrows, and the behaviours in which the agent waits listening for incoming messages from other agents or call internal methods are represented by dotted lines. We can describe with words what is happening. Supposing the User has called a Broker service on the Grid and a BA has been

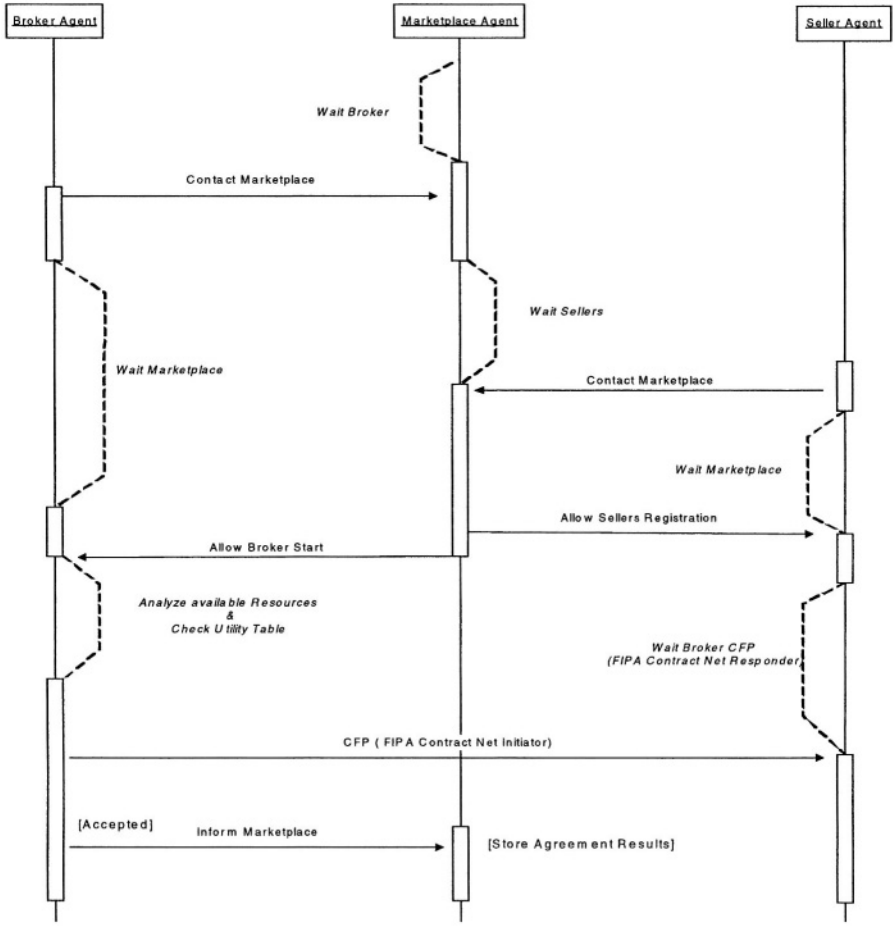


Fig. 3. The Agents operation Sequence Diagram

instantiated, the following occurs: First the MA waits Broker to contact it, and once a Broker contacts the MA, and thus the latter knows that someone is interested on using Grid Services, the MA automatically start waiting for SAs interested in register the Marketplace. After a SA is registered, it starts listening for incoming BA sending CFP. When a number of Sellers is registered on the Marketplace, MA allows the BA to start negotiating with SAs. The BA first analyze the state of available resources by inspecting the resources Service Data, exposed by active GT3 services on Grid resources (each grid resource provider activates on start-up its corresponding SA and Grid Service). After getting an idea of what the actual state of resources is, the BA checks the UT to select the best-rated configurations given the current state of the resources. Once a set of target resources has been selected, the BA starts a FIPA ContractNetInitiatior behaviour in order to negotiate with the SAs for the resource usage. If the FIPA-Contract-Net steps resolve on agreement, the BA informs the MA that records the details of the agreement. If the SA refuses to negotiate in the terms proposed by the BA, a new SA will be contacted by the BA, trying always to keep the utility given by the UT as high as possible. Once an agreement has been achieved with the targeted SAs, the only step remaining is the scheduling of the tasks on the resources. The granularity of the task (considered to be the number of jobs in which we divide it) and the time we wait between each job submission (depending on the available bandwidth on the network that parameter can have strong influence) are fixed by looking at the best-rated combination on the UT. After all this process, and checking that the deadline specified by the user is not being exceed, the scheduling of jobs is performed using the Globus middleware.

4.2 Grid Services Implementation

The Grid Service we deploy on the resources is currently used just for extracting a variety of system information, including CPU rating, memory, system load and network traffic between the most interesting (and we call it Meta Service). We rely on UNIX standard commands such as top, netstat and the information recorded in the directory /proc of any UNIX system to get the system information. The command top gives the average load of the system by means of the number of processes running on the machine. The command netstat gives (among other information) the number of total packets received by the machine. If we evaluate this on to consecutive (say separated by 2 seconds) time stamps, we get an estimation of the network traffic on the resource. Other information as the CPU ratings, the total and free memory and the OS can be checked directly on the /proc files.

The service is called by the BA each time it wants to have detailed information on the resource state, and the system information exposed on the service data gets updated. This happens normally just before checking the UT for the utility of configurations. When the resource provider creates his SA, the Meta Service is launched on the resource to serve the BA.

4.3 System Setup

In order to properly operate with the system, we need first to build an UT as extensive as possible. Since the size of this table (and so the overhead introduced by its

calculation and checking) grows exponentially with the number of resources, we are considering using Genetic Algorithms [19] in order calculate the parameters of the optimal configurations, much in the same way as in [20], and thus saving time. This way, the agents may act first on a training period to build the UT and, once this step is completed, the normal system operation can start till the next significative changes are introduced on the system and the UT has to be recalculated. By the moment we are performing preliminary testing of the system in a very small grid test bed (only 4 computers) and thus those scalability problems still don't hold. The table is generated by a handy Perl script, which allows for a programmable execution of jobs on the machines, forcing the required conditions of each experiment by executing via Globus process and pings that both increase load and network traffic on the targeted machines up to the required level. In those created conditions, jobs are executed via Globus on the machines and the total time spent to completion in a given configuration is stored in a configuration line of the UT.

Once the UT is built, we only need to provide Globus Toolkit (GT 3.2 in this case) installed on the resources and JADE jars on the CLASSPATH in order to start the MAS execution.

5 Experiments

We have performed preliminary experiments on the commented small grid test bed. Those experiments are intended to serve as a system execution proof and a test for the implementation.

Full tests contrasted with a plain Globus based grid execution performance will be performed in the near future. We believe anyway that the strong contribution of this system compared with the manual approach are the levels of automatization and the flexibility introduced by agents negotiation capabilities. We have tested the system capabilities for the conversion between multimedia formats AVI and DivX.

The experiments consisted on a conversion from small AVI files (24 MB) to DivX format using the software menconder bundled in the MPlayer_1.04prea version. We first built an UT, on a controlled size due to the small size of the test bed.
The following resources were used in the experiments:

kanaima.lsi.upc.es	*Intel Pentium4 CPU 1.70GHz, 1 GB RAM, Linux RedHat 9.0*
khafre.lsi.upc.es	*Intel Pentium3 CPU 1.2GHz, 2 GB RAM, Linux RedHat 9.0*
pcmartino.upc.es	*Intel Pentium4 CPU 1.6GHz, 512 MB RAM, Linux RedHat 9.0*
mannkell.lsi.upc.es	*Intel Pentium4 CPU 3.00GHz, 1 GB RAM, Linux RedHat 9.0*
davinci.lsi.upc.es	*(used to perform sized pings to other machines)*

We used the following values for the parameters:

Resources: 1 machine launching jobs in up to 3 resources
System load: 3 values (Low: less than 33% Medium: between 33% and 66%, High: more than 66%).
Network Traffic: 3 values (Low: none or a few pings. Medium: regular pinging with 200K or 300K of packet size. High: regular pinging of more than 300K of packet size).

Granularity: 2 different: 5 jobs of size of ~5 MB each, and 3 jobs of 8 MB each.
Time between job launch: 2 values: 1 second and 2 seconds.

A snapshot of a few lines of the generated UT for the performed experiments is given on figure 4.

```
#Configs
#MACHINES|LOAD|NET_TRAFF|GRANULARITY|TIME_B_SEND|T_TIME

kanaima khafre pcmartino | L L L | L L L | 5 | 1 | 203
kanaima khafre pcmartino | L L L | L L L | 5 | 2 | 205
kanaima khafre pcmartino | L L L | L L L | 3 | 1 | 204
kanaima khafre pcmartino | L L L | L L L | 3 | 2 | 209
kanaima khafre pcmartino | M L L | L M L | 3 | 1 | 205
kanaima khafre pcmartino | M L M | L L L | 5 | 1 | 219
kanaima khafre pcmartino | M L L | M L L | 5 | 1 | 222
kanaima khafre pcmartino | M L L | M L L | 5 | 2 | 228
kanaima khafre pcmartino | M L L | M L L | 3 | 1 | 225
kanaima khafre pcmartino | M L L | L M L | 5 | 1 | 223
kanaima khafre pcmartino | M L L | L M L | 3 | 1 | 225
kanaima khafre pcmartino | M L L | L M H | 5 | 1 | 233
kanaima khafre pcmartino | M L M | H M L | 5 | 2 | 235
kanaima khafre pcmartino | M L M | H L L | 3 | 1 | 236
kanaima khafre pcmartino | M L H | H L L | 5 | 2 | 238
kanaima khafre pcmartino | M L M | H L H | 5 | 1 | 240
kanaima khafre pcmartino | M L L | M H M | 3 | 1 | 242
kanaima khafre pcmartino | M L H | L L H | 5 | 2 | 244
kanaima khafre pcmartino | M L L | M M M | 5 | 1 | 245
kanaima khafre pcmartino | M L M | L H H | 3 | 2 | 248
kanaima khafre pcmartino | M H H | L M M | 5 | 1 | 250
kanaima khafre pcmartino | M H L | L H L | 3 | 1 | 250
```

Fig. 4. Snapshot of a few lines of the generated Utility Table

Once the UT is built, we perform the system operation experiment with the MA, BA and the 3 SAs corresponding to the three resource machines. The broker on the client was given a deadline of 225 seconds. During the system execution, the BA uses the Meta Grid Service to obtain the following values for the selected resources states:

kanaima khafre pcmartino | M L L | M L L |

Then, when checked the UT from fig.4, the BA had to make a choice between the following configurations for the given state:

| granularity | time between sends | total time

| 5 | 1 | 222 or | 5 | 2 | 228 or | 3 | 1 | 225

In consequence, the BA selected a granularity of 5 and a time between job submissions of 1 second to achieve the completion of the task in the predicted smallest time (222 seconds) for the considered state of the targeted resources.

We believe that this kind of operations can be performed in real scale grids, specially for those where resources are reasonably persistent on the grid (at least enough to keep a UT valuable for some time in order to amortise the time spent on the generation) and we can expect Grid users to perform repetitive tasks on the resources under different dynamically changing states of resources.

6 Conclusion and Future Plans

The paper described the ongoing work on the realization of FIPA-compliant marketplace-based MAS for Grid Resource Management, layered on top of the Globus Toolkit Grid middleware. We believe that, being developed at the applications layer and relaying on what we consider the most adopted standards both in Grid Computing and Agent Technology, our system achieves the desired levels flexibility and modularity that allows for a proper integration on a broader generic Grid computing framework.

The presented architecture and implementation based on agents marketplace for trading Grid resources, opens the perspective for further developments introducing pricing schemas and economical models. As pointed out in [22], the growing involvement of industry and commerce in Grid activity brings the opportunity to introduce economic paradigm to drive effective resource utilisation on Grids. Our next step is to use the well-proven strong negotiation capabilities of agent technologies to incorporate those economic models, leveraging the works presented on [22, 23] among others. On the short term, we plan to test scalability of the system in a medium sized test bed, and the integration on the mentioned generic Grid framework.

Acknowledgements. We want to thank the members of GridCat that are not cited in the authorship of this paper for their help and support on the effective implementation and evaluation of the architecture presented in this paper.

References

[1] I. Foster, C. Kesselman, S. Tuecke: The Anatomy of the Grid: Enabling Scalable Virtual Organizations**.** International J. Supercomputer Applications, *15(3),* 2001

[2] I. Foster, C. Kesselman, J. Nick, S. Tuecke: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002

[3] Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, T. Maguire, D. Snelling, S. Tuecke: From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution. K., March 5, 2004

[4] Chaitanya Kandagatla: Survey and Taxonomy of Grid Resource Management Systems, University of Texas, Austin Technical Report

[5] Peter Stone and Manuela Veloso: Multiagent Systems: A Survey from a Machine Learning Perspective. December 4, 1997 School of Computer Science, Carnegie Mellon University, Pittsburgh. Technical Report CMU-CS-97-193

[6] THE SEMANTIC WEB webpage: http://www.w3.org/2001/sw/

[7] I. Foster, N. R. Jennings and C. Kesselman (2004): Brain meets brawn: Why Grid and agents need each other. Proc. 3rd Int. Conf. on Autonomous Agents and Multi-Agent Systems, New York, USA (to appear)

[8] Omer F. Rana: Software Agents and Grid Computing: Friends or Foes?, Research Statement/Interests.

[9] Shen, W., Li, Y., Ghenniwa, H. and Wang: C. Adaptive Negotiation for Agent-Based Grid Computing, Proceedings of AAMAS2002 Workshop on Agentcities: Challenges in Open Agent Environments, Bologna, Italy, July 15-19, 2002, pp. 32-36

[10] Niek J. E. Wijngaards, B. J. Overeinder, Maarten van Steen, and Frances M. T.: Brazier. Supporting internet-scale multi-agent systems. Data Knowledge Engineering, 41(2-3):229-245, 2002

[11] Junwei Cao, Daniel P. Spooner, James D. Turner, Stephen A. Jarvis, Darren J. Kerbyson, Subhash Saini, Graham R. Nudd: Agent-Based Resource Management for Grid Computing. CCGRID 2002: 350-351

[12] B. Di Martino, O.F. Rana: Chapter in book "Performance Analysis and Grid Computing", edited by V. Getov, M. Gerndt, A. Hoisie, A. Malony, and B. Miller, Kluwer. Grid Performance and Resource Management using Mobile Agents, November 2003

[13] Moreau, L., Avila-Rosas, A., Dialani, V., Miles, S. and Liu, X. (2002): Agents for the Grid: A Comparison with Web Services (part II: Service Discovery). In Proceedings of Workshop on Challenges in Open Agent Systems - (-), pages pp. 52-56, Bologna, Italy

[14] JADE framework webpage: <http://sharon.cselt.it/projects/jade/>

[15] FIPA webpage: http://www.fipa.org/

[16] GLOBUS TOOLKIT webpage: <http://www.globus.org/>

[17] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, P. Vanderbilt, D. Snelling: Open Grid Services Infrastructure (OGSI) Version 1.0.Global Grid Forum Draft Recommendation, 6/27/2003.

[18] GRIDCAT webpage: http://www.gridcat.org/

[19] Majid Ali Khan, Shankar K. Vaithianathan, Kresimir Sivoncik and Ladislau Bölöni: Towards an Agent Framework For Grid Computing. CIPC-03 Second International Advanced Research workshop on Concurrent Information Processing and Computing, Sinaia, Romania, 2003.

[20] Robert E. Smith and Nick Taylor: A Framework for Evolutionary Computation in Agent-Based Systems . Proceedings of the 1998 International Confrence on Intelligent Systems

[21] M.L. Green and R. Miller: Molecular structure determination on a computational & data grid, BioGrid'04: Second International Workshop on Biomedical Computations on the Grid, 4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2004), April 19-22, 2004, Chicago, Illinois.

[22] S. Newhouse, J. Darlington, M. Asaria, A. Mayer, W. Lee, J. MacLaren, S. Cox, K. Nammuni, and K. Keahey: Trading Grid Services Within the UK e-Science Grid. In *UK e-Science All Hands Meeting,* p. 13–20, Nottingham, UK, Sep. 2003

[23] Rajkumar Buyya, David Abramson, Jonathan Giddy, and Heinz Stockinger: Economic Models for Resource Management and Scheduling in Grid Computing, Special Issue on Grid Computing Environments, The Journal of Concurrency and Computation: Practice and Experience (CCPE), Wiley Press, USA, May 2002

Towards a Peer-to-Peer Event Broker Grid in a Hybrid Network Environment

Eiko Yoneki and Jean Bacon

University of Cambridge Computer Laboratory
Cambridge CB3 0FD, United Kingdom
{Eiko.Yoneki, Jean.Bacon}@cl.cam.ac.uk

Abstract. Peer-to-peer networks and grids offer promising paradigms for developing efficient distributed systems and applications. Event-based middleware is becoming a core architectural element in such systems, providing asynchronous and multipoint communication. There is a diversity of large scale network environments, and events flow from tiny sensor networks to Internet-scale peer-to-peer (P2P) systems. Event broker grids need to communicate over wired P2P networks, wireless mobile ad hoc networks and even Web Services. Thus,a good architecture of an event data model and subscription model with integrated semantics is important. In this paper, We propose an ontology-based event model for such systems and events are defined in RDF. The system includes an event correlation service for composite event detection and propagation.

1 Introduction

Event-based middleware which is based on the publish/subscribe communication paradigm became popular, because asynchronous and multipoint communication is well suited for distributed computing applications. The Grid community has recognized the essential nature of event-based middleware such as the Grid Monitoring Architecture, the Grid Notification Framework and peer-to-peer (P2P) high performance messaging systems like NaradaBrokering [9]. They are core architectural elements for constructing Grid systems. Most distributed event-based middleware contains three main elements: a producer who publishes events (messages), a consumer who subscribes his interests to the system, and an event broker with responsibility to deliver the matching events to the corresponding consumers. While the mechanisms for publish/subscribe communications are well understood and robust implementations can be found, some issues still remain open. For example, how to get consensus of interests, the mechanism to find mutual interests between publishers and subscribers, how to provide the quality and accuracy of information, etc. This paper addresses the issues on subscriptions in event-base middleware, which provides the backbone of communications for grid systems such as efficient service advertisement, or knowledge management. Most early event-based middleware systems were based on the concepts of group (channel) or subject (topic) communication. These systems categorize events into pre-defined groups. In an attempt to overcome the

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 198–210, 2004.
© Springer-Verlag Berlin Heidelberg 2004

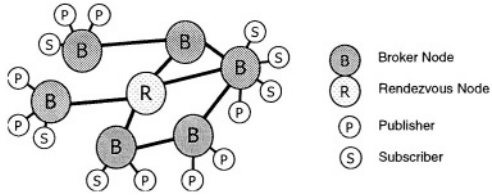


Fig. 1. Hermes: Event Broker Grid over Wide Area Networks

limitation on subscription declarations, the content-based model has been introduced [1],[4] which allows subscribers to use flexible querying languages to declare their interests with respect to event contents. Events are routed based on their content and consumer interests. Most event-based middleware for wide area network architectures employ an overlay network of event brokers. These brokers then perform content-based routing of events at the application-level. Fig. 1 shows the broker grid used by Hermes (see [10] for the details).

A mobile ad hoc network (MANET) is a dynamic collection of nodes with rapidly changing multi-hop topologies that are composed of wireless links. The combination of mobile devices and ad-hoc networks is best managed by the creation of highly dynamic, self-organizing, mobile peer-to-peer (MP2P) systems. In such systems, mobile hosts continuously change their physical location and establish peering relationships with each other based on proximity. There is a diversity of large scale network environments, and events flow from tiny sensor networks to Internet-scale P2P systems. The architecture requires further consideration of the event data model, and subscription model. It is important to provide efficient communication to support such event-based systems especially over mixed and hybrid network environments. We propose an ontology-based event-based middleware that integrates information flows over various event-based middlewares and can combine Web Services over distributed systems.

This paper continues as follows: section 2 describes the background and related work, section 3 describes the integration of an event broker grid over P2P and Mobile P2P networks, section 4 reports an experimental ontology-based event model, and it concludes with section 5.

2 Background and Related Work

Peer-to-peer networks and grids offer promising paradigms for developing efficient distributed systems and applications. In analyzing both models grids are essentially P2P systems. The grid community recently initiated a development effort to align grid technologies with Web Services: the Open Grid Services Architecture (OGSA) lets developers integrate services and resources across distributed, heterogeneous, dynamic environments and communities. The OGSA model adopts the Web Services Description Language (WSDL) to define the concept of a grid service using principles and technologies from both the grid and Web Services. The architecture defines standard mechanisms for creating, naming, and discovering persistent and transient grid-service instances.

The convergence of P2P and Grid computing is a natural outcome of the recent evolution of distributed systems, because many of the challenging standards issues are quite closely related. This creates best practice that enable interoperability between computing and networking systems for the P2P community at-large. Multicast has traditionally been used as transport for group communication. However today's IP based multicast schemes are not scalable to support large groups, and an infrastructure of multicast is not supported well over wide area networks. Thus, Application-Level Multicast Routing Protocols (ALMRPs) are replacing group communication over wide area networks. ALMRP is a promising candidate for P2P communication. Narada and many subsequent designs can best be understood as two layered protocols: a protocol that maintains a mesh of hosts, and a multicast routing protocol on top of this mesh. Other examples are Bayeux/Tapestry [17] and CAN [12]. The multicast service model is less powerful than that of a content-based network, and there is currently no optimal way of using or adapting the multicast routing infrastructure to provide a content-based service.

In distributed network environments such as P2P networks, it is attractive to construct event-based middleware over overlay networks to provide scalability. Distributed hash tables (DHT) have been adopted to create P2P overlay networks that can be used easily to construct a topic-based publish/subscribe system in distributed environments. For the past several years, many prototype publish/subscribe systems have been reported, including Gryphon [6], SCRIBE [13], Hermes [10], and SIENA [4]. SCRIBE is a topic-centric publish/subscribe messaging system using DHT over Pastry [14]. Pastry uses routing mechanisms to achieve great scalability. SCRIBE depends on Pastry to route messages to the destinations. Hermes [10] takes a similar approach and offers a more scalable event broker system by adding advanced rendezvous nodes. Several publish/subscribe systems implement some form of content-based routing such as SIENA. SIENA is a notification service scalable to wide-area networks. Routing strategies in SIENA use two classes of algorithm: advertisement forwarding and subscription forwarding. They prune the propagation tree by propagating only those paths that have not been covered by previous requests.

In mobile ad-hoc networks, much research currently focuses on general datagram routing in both unicast and multicast routing, but no definite solution to define publish/subscribe semantics using these protocols has been provided. To achieve improved asynchronous and one-to-many communication systems in MANET environments, we introduced the semantics of publish/subscribe over mobile ad hoc networks [15].

3 Integration of an Event Broker Grid over P2P and MP2P

The proposed approach addresses an ontology-based unified event model in event-based middleware that integrates Grid services, Web Services, P2P interactions and traditional middleware operations. Note that complete event-

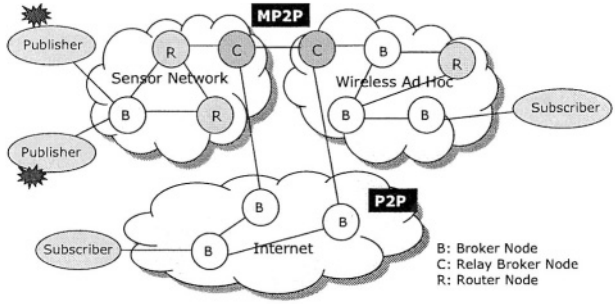


Fig. 2. Event Broker Grids over Mixed Peer-to-Peer Networks

based middleware require not only a unified event model over an asynchronous communication paradigm but also additional functions such as programming language integration, reliability, fault-tolerance, security, and transaction processing. Moreover, with event typing, composite events, and event persistence, it becomes more complete event-based middleware. In this paper, an event correlation is also expressed in section 3.3. Establishing an interoperable event broker grid especially between classic P2P and mobile P2P environments is the ultimate goal in this paper. Fig. 2 shows the broker grid over three different networks. The relay nodes in mobile ad hoc networks have usually more resource, and they are the good candidates among members of event broker grids which propagate the published events to the subscribers. One important aspect to be considered when constructing event broker grids is understanding the purpose of the grid, so that it can provide the appropriate functions.

3.1 Subscription Models

One of the key issues in messaging systems is designing subscription models. Topic-based addressing is an abstraction of numeric network addressing schemes. With the content-based subscriptions used in SIENA and Gryphon, delivery depends only on message content, extending the capability of event notification with more expressive subscription filters. In Pronto [16], besides topic-based routing, a filtering function that selects messages based on their content is supported. The content filter language is based on the WHERE syntax of SQL92 over XML messages. Content-based publish/subscribe is essential for better (filtered) data flow in mobile computing. The most advanced and expressive form of subscription language is content-based with pattern matching; such a language is important for event notification. Common topic-based systems arrange topics in hierarchies, but a topic cannot have several super topics. Type-based subscription [5] provides a natural approach to this if the language offers multiple sub-typing, thus avoiding explicit message classification through topics. This works well with typed languages, but it is complex to deploy this degree of serialization of objects. Moreover, mobile applications may not have the concept

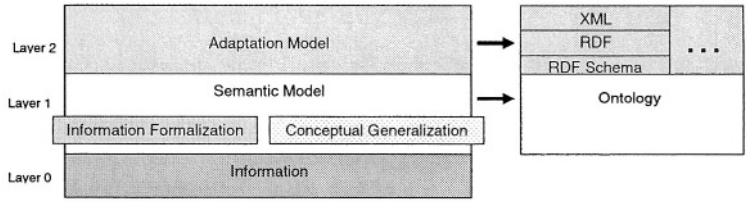


Fig. 3. Ontology Based Event

of objects or typing. Topic-based messaging systems can easily be built with application level multicast using DHT in P2P networks. However, there is no optimal way of adapting multicast routing to provide content-based messaging. Research is also ongoing to structure complex content-based data models [7] and reflection-based filters [5].

A key challenge when building event-based middleware is efficient propagation of subscriptions among brokers. Brokers require this information to forward incoming events exclusively to interested consumers. Filtering out unrelated events can save significant overhead. The proposed approach is a combination of hierarchical topics and high speed content filtering and will be a more flexible approach for both P2P and MP2P environments.

3.2 Data Model

When designing distributed systems, the information flowing should itself be made more abstract (to provide compaction), and the data should be evaluated whenever necessary. In Pronto [5], semantic transcoding is attempted. The concept of semantic transcoding is more than reducing the data size. The data are linked to an annotation and annotations can be corresponding text for a video clip, a summary of a document, greyscale/downsized/low -resolution image data or a linguistic annotation of the content for voice synthesis.

An ontology seems to be a good approach to event structuring and modeling by providing a formal conceptualization of a particular domain that is shared by a group of people; the members of the group can establish specific group communication among the different participants. Fig. 3 shows the ontology-based event structure attempted in this paper.

Note that in wireless ad hoc networks, the high computational process should be avoided for data manipulation and data themselves have to be described in a compact format with efficient encoding schema. Resource Definition Framework (RDF) is the most often used example of an ontology. In our experimental project, RDF is used (see Section 4.3).

RDF: Resource Definition Framework (RDF) is a foundation for processing metadata in order to provide interoperability between applications, and enables automated processing of web resources. RDF with digital signatures will be key to building the trusted collaboration mechanism. The syntax of RDF uses

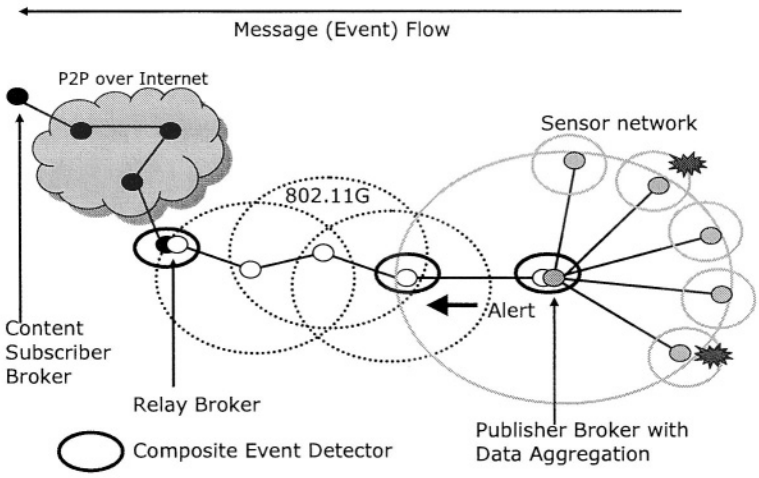


Fig. 4. Mixed Network Environments

XML: one of the goals of RDF is to make it possible to specify semantics for the data based on XML in a standardized, interoperable manner. RDF and XML are complementary: RDF is a model of metadata and only addresses by reference many of the encoding issues that transportation and file storage require. For these issues, RDF relies on the support of XML. It is also important to understand that this XML syntax is only one possible syntax for RDF and that alternate ways to represent the same RDF data model may emerge.

3.3 Event Correlation

A subscriber might only be interested in a certain combination of messages (composite event) that produces the new information during the propagation of messages over the networks. Composite events represent complex patterns of activity from distributed sources. In monitoring systems such as detecting direction and speed of certain phenomena (e.g. a moving glacier or migrating birds), mobile computing devices with sensors and clocks can record the event occurrence time and communicate this information to other devices as they pass by. Event correlation combines the information collected by the individual devices into higher level information or knowledge. In data centric network environments such as sensor networks, data (message) aggregation over the network is important. Within the sensor networks, for example, the Tiny AGgregation (TAG) service (see [8]) allows users to express simple, declarative queries and have them distributed and executed efficiently in networks of low-power, wireless sensors. TAG processes aggregates in the network by computing over data as it flows through the sensors, discarding irrelevant data and combining relevant readings into more compact records when possible. In Fig. 4, the alert event from sensor networks is propagated to the subscribers via event brokers over mixed

networks. Publisher brokers, which can provide data aggregation services by using composite event detection. They can also make decisions either to discard incoming events or to distribute them depending on subscriptions. In order to express correlation of events, the composite event language introduced in our group's earlier work (see [11] for details) is supported as an option along with the composite event detector.

4 An Experimental Ontology-Based Event Model

An experimental construction of an ontology-based event model over our two event-based middlewares is described. The first, Hermes, is for wide area networks and the second, ECCO Mobile, is for wireless ad hoc networks. They have originally different event models, however common characteristics on the subscription model are both supporting topic and content-based subscriptions. Most content-based publish/subscribe systems support a subscription language that allows a subscriber to express its information need. The resulting filtering expression is then used by the brokers to determine whether a particular event notification is of interest to a subscriber. If the language is expressive, efficient matching of notifications will be difficult. However, if the language does not support rich constructs, its applicability is limited. Both Hermes and ECCO Mobile use XPath [2] as a subscription language in prototype implementations.

4.1 Hermes

The Hermes event-based middleware [10] is our group's recent work, which uses peer-to-peer techniques to build and maintain a scalable grid of event brokers for event dissemination. Hermes supports event typing: an event type has an event type name, a list of typed event attributes, and an event type owner so that, at runtime, published events and subscriptions can be type-checked. Event types are organized into inheritance hierarchies. These hierarchies can be application specific and thus help facilitate large-scale distributed system design. Before a publisher can publish an event instance, it must submit an advertisement to its local event broker, containing the event type that it is willing to publish. Subscribers express their interest in the form of subscriptions that specify the desired event type and a conjunction of (content-based) filter expressions over the attributes of this event type. To ensure that an event dissemination tree is created between publishers and subscribers, some broker in the system must become the rendezvous node for each particular event type. A rendezvous node is selected by hashing the event type name to a broker identifier - an operation that is supported by the P2P routing substrate (see Fig. 1). Advertisements and subscriptions are routed towards the rendezvous node, and brokers along the path set up filtering state for them. Event brokers communicate with four major kinds of messages: (1) Type messages set up rendezvous nodes. (2) Advertisements denote a publisher's desire to publish events of a certain type. (3) Subscriptions are used by subscribers to express

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:t="http://www.cl.cam.ac.uk/~ey204/type"
     targetNamespace="http://www.cl.cam.ac.uk/~ey204/type">

  <xsd:complexType name="BaseEventType">
    <xsd:sequence>
      <xsd:element name="id" type="xsd:long" minOccurs="0"/>
      <xsd:element name="publisher" type="xsd:string" minOccurs="0"/>
      <xsd:element name="timestamp" type="xsd:dateTime" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="BaseEvent" type="t:BaseEventType"/>
</xsd:schema>
```

Fig. 5. An event type schema defined in XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<h:hermes xmlns:h="http://www.cl.cam.ac.uk/~ey204">
  <type>
    <addtype typename="CDEvent" extends="BaseEvent">

      <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns:t="http://www.cl.cam.ac.uk/~ey204"
          targetNamespace="http://www.cl.cam.ac.uk/~ey204/type">

      <include schemaLocation="BaseEvent"/>

      <complexType name="CDEventType">
        <complexContent>
          <extension base="t:BaseEventType">
            <all> <element name="classification" type="xsd:string"/> </
          </extension>
       </complexContent>
      </complexType>

      <element name="CDEvent" type="t:CDEventType"/>
    </xsd:schema>
    </addtype>
  </type>
</h:hermes>
```

Fig. 6. An XML definition of a type message

```
<?xml version="1.0" encoding="UTF-8"?>
<h:hermes xmlns:h="http://www.cl.cam.ac.uk/~ey204">
  <publication routing="typeAttr">
    <publish typename="CDEvent">
      <t:LocationEvent xmlns:t="http://www.cl.cam.ac.uk/~ey204/type">
        <composer>Evans</composer>
        <id>99999</id>
        <timestamp>2003-06-20T12:00:00.000-00:00</timestamp>
        <classification>jazz</classification>
      </t:CDEvent>
    </publish>
  </publication>
</h:hermes>
```

Fig. 7. An XML definition of a publication message

their interest in publications. Finally, (4) Publications contain event instances published by publishers. Fig. 5-7 show examples of the event definition in XML.

Subscription Language: A subscription message is first sent by a subscriber-hosting broker and contains a type-based or type- and attribute-based subscription. An example of a type- and attribute-based subscription (see Fig. 8).

```
<?xml version="1.0" encoding="UTF-8"?>
<h:hermes xmlns:h="http://www.cl.cam.ac.uk/~ey204">
<subscription>
<subscribe typename="CDEvent">
<typeattr>
<xpath>child::*[child::id>99999 and child::category="jazz"]</xpath>
</typeattr>
</subscribe>
</subscription>
```

Fig. 8. An XML definition of a subscription message

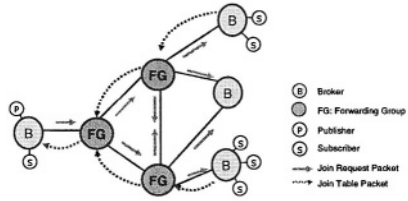


Fig. 9. ECCO Mobile

4.2 ECCO Mobile

ECCO Mobile [15] is an on-demand multicast routing protocol in MANET which integrates publish/subscribe schema, and supports content-based subscriptions. ECCO Mobile is a self organizing protocol for mobile P2P. The topology of a mobile P2P system has to constantly adjust itself, by discovering new communication links, and also needs to be fully decentralized due to the lack of a central access point. Content-based subscriptions at a broker node are aggregated and summarized into a compact data format in Bloom filters [3]. The digest of published events and advertisements are transformed to the Bloom filters which travel within a multicast packet header. When event digests reach the subscriber broker, a matching operation between the event digest and subscription decides either to join the particular multicast group or not. The publishing broker determines the multicast group from the propagated subscriptions. Routing uses the forwarding group concept (See [15] for details) to reduce packet flooding (see Fig. 9).

XML-based Typed Event (Message): ECCO Mobile defines events in XML format in XML schema.The XML schema for the event consists of a set of typed elements. Each element contains a type and a name. The element's name is a simple string, while the value can be in any range defined by the corresponding type. Fig. 10 shows the schema example named 'CD' and Fig. 11 shows example messages.

Subscription Language: A subset of XPath is used as a filter-specification language (see examples in Fig. 12). Complex XPath expressions will be transformed to simplified and unified formats. The subscriptions are tightly linked to the corresponding event data structure.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema attributeFormDefault="unqualified"
    elementFormDefault="qualified"
    xmlns="http://www.cl.cam.ac.uk/~ey204/lib/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:simpleType name="categoryType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="jazz"/>
            <xs:enumeration value="classic"/>
            <xs:enumeration value="pop"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="priceType">
        <xs:restriction base="xs:float">
            <xs:maxInclusive value="25.00"/>
            <xs:minInclusive value="10.00"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:element name="category" type="categoryType"/>
    <xs:element name="composer" type="xs:string"/>
    <xs:element name="price" type="priceType"/>
    <xs:attribute name="id" type="xs:int"/>
    <xs:attribute name="timestamp" type="xs:date"/>
    <xs:element name="CD">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="category"/>
                <xs:element ref="composer"/>
                <xs:element ref="price"/>
            </xs:sequence>
            <xs:attribute ref="id"/>
            <xs:attribute ref="timestamp"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Fig. 10. XML Schema Definition for Type CD

```
<?xml version="1.0" encoding="UTF-8"?>
<CD id="001" timestamp="1999-02-27T12:00:00.000-00:00"
      xmlns="http://www.cl.cam.ac.uk/~ey204/lib/">
    <category>jazz</category>
    <composer>Evans</composer>
    <price>18.00</price>
</CD>
```

Fig. 11. Event Instance for Type CD

```
1. /CD[category=jazz and price<20 and price>15]
2. /CD[category=classic][composer=Bach]
3. /TAPE[category=jazz][composer=Davis]
4. /CD[category!=jazz] will be transformed to
                /CD[category=pop and classic]
5. /CD[composer=Davis]|/CD[category=pop]
    will be transformed to two subscriptions
```

Fig. 12. Subscription Filter Examples

Compact Data Structures in Bloom Filters: Subscriptions are aggregated at the brokers and represented in compact data format. Also the event advertisement and notification are transformed to the compact data format, which uses XPath as an intermediate expression during the transformation. For encoding

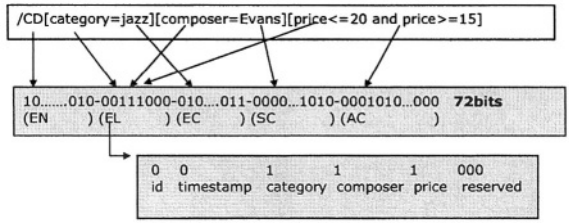


Fig. 13. An Example of Encoding Subscriptions in Bloom Filters

data structures, Bloom filters are used (see examples in Fig. 13). Bloom Filters are compact data structures for probabilistic representation of a set in order to support membership queries. (See [15] for more details) Each filter provides a constant-space approximate representation of the set. Errors between the actual set and the Bloom filter representation always take the form of false positives to inclusion tests. False positive probability decreases exponentially with linear increase in the number of hash functions and vector size.

4.3 Event Representation with RDF

Fig. 14 shows the ontology-based event structure defined in RDF schema, and Fig. 15 shows examples for Hermes and Ecco Mobile. RDF and RDFS give the capability to provide some semantics, however OWL (Web Ontology Language) ontology will be the next level, which has a stronger language with a larger vocabulary and stronger syntax than RDF. The two event brokering systems, Hermes for wire area networks environments, ECCO Mobile for wireless ad hoc networks are integrated with the RDF based event data model.

5 Conclusions and Future Work

In this paper, We presented an approach to integrate publish/subscribe semantics with an ontology based event model. Event-based middleware is becoming a core architectural element in P2P networks and grids for developing distributed systems and applications. The focus in this paper is the event data model and subscription model with integrated semantics to provide efficient group communication. The proposed approach is implemented over Hermes for wide area networks and ECCO Mobile for wireless ad hoc networks, and the prototype shows the proof of concept for the effective use of an RDF-based ontology. The project is evolving using OWL in combination with an event correlation service. Combining complex event composition and detection with event-based systems will provide a sound active information framework. Our ultimate goal is to create an active event broker grid over mixed P2P networks in a multi event broker model and to integrate other event-based systems in a unified interface and, in particular, to adapt to mobile computing environments and Web Services. The work described in this paper will be part of this endeavor.

```
<rdf:RDF       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
               xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
Class Definitions:
<rdfs:Class rdf:about="http://www.cl.cam.ac.uk/~ey204/schema.rdf#BaseEvent">
<rdfs:label>BaseEvent</rdfs:label>
<rdfs:comment>Class for the general category part of Event</rdfs:comment>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.cl.cam.ac.uk/~ey204/schema.rdf#CD">
<rdfs:label>CD</rdfs:label>
<rdfs:comment>Class for CD</rdfs:comment>
<rdfs:subClassOf
  rdfs:resource="http://www.cl.cam.ac.uk/~ey204/schema.rdf#BaseEvent"/>
</rdfs:Class>

Property Definition:
<rdfs:Property rdf:about="http://www.cl.cam.ac.uk/~ey204/schema.rdf#id">
<rdfs:domain
  rdf:resource="http://www.cl.cam.ac.uk/schema.rdf#BaseEvent"/>
<rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
  ..
<rdfs:Property rdf:about="http://www.cl.cam.ac.uk/~ey204/schema.rdf#timestamp">
<rdfs:domain
  rdf:resource="http://www.cl.cam.ac.uk/schema.rdf#BaseEvent"/>
  ..
<rdfs:Property rdf:about="http://www.cl.cam.ac.uk/~ey204/schema.rdf#Composer">
<rdfs:domain
  rdf:resource="http://www.cl.cam.ac.uk/schema.rdf#CD"/>
  ..
<rdfs:Property rdf:about="http://www.cl.cam.ac.uk/~ey204/schema.rdf#Category">
<rdfs:domain
  rdf:resource="http://www.cl.cam.ac.uk/schema.rdf#CD"/>
  ..
<rdfs:Property rdf:about="http://www.cl.cam.ac.uk/~ey204/schema.rdf#Price">
<rdfs:domain
  rdf:resource="http://www.cl.cam.ac.uk/schema.rdf#CD"/>
  ..
</rdf:RDF>
```

Fig. 14. Event Definition in RDF Schema

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns="http://www.cl.cam.ac.uk/~ey204/schema.rdf#">

<CD rdf:ID="Hermes001">
    <id rdf:value="99999"/>
    <timestamp rdf:value="2003-06-20T12:00:00.000-00:00"/>
    <composer rdf:value="Evans"/>
    <category rdf:value="jazz"/>
</CD>

<CD rdf:ID="ECCO001">
    <id rdf:value="001"/>
    <timestamp rdf:value="2003-06-20T12:00:00.000-00:00"/>
    <composer rdf:value="Evans"/>
    <category rdf:value="jazz"/>
    <price rdf:value="18.00"/>

</CD>
```

Fig. 15. Event Example in RDF

Acknowledgment. This research is funded by EPSRC (Engineering and Physical Sciences Research Council) under grant GR/557303.

References

1. Bacon, J. et al. Using Events to build Distributed Applications. In *Proc. IEEE SDNE Services in Distributed and Networked Environments,* 1995.
2. Berglund, A. et al. XML Path Language (XPath) 2.0. Working Draft. In *W3C, http://www.w3c.org/TR/xpath20/,* 2001.
3. Bloom, B. Space/time Trade-offs in Hash Coding with Allowable Errors. In *CACM, 13(7),* 1970.
4. Carzaniga, A. et al. Achieving scalability and expressiveness in an Internet-scale event notification service. In *Proc. 19th ACM Symp. on Principles of Distributed Computing,* 2000.
5. Eugster, P.Th. et al. The Many Faces of Publish/Subscribe. *ACM Computing Surveys, 35(2), pages 114-131,* 2003.
6. IBM. Gryphon: Publish/Subscribe over Public Networks. In *http://researchweb.watson.ibm.com/gryphon/gryphon.html.*
7. Muhl, G. et al. Filter Similarities in Content-Based Publish/Subscribe Systems. In *Proc. ARCS,* 2002.
8. Madden, S. et al. TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *Proc. OSDI,* 2002.
9. The Narada Event Brokering System. *http://grids.ucs.narada.edu/ptliupages/projects/narada.*
10. Pietzuch, P. and Bacon, J. Hermes: A distributed event-based middleware architecture. *Proc. DEBS,* 2002.
11. Pietzuch, P. Shand, B. and Bacon, J. Composite Event Detection as a Generic Middleware Extension. *IEEE Network Magazine, Special Issue on Middleware Technologies for Future Communication Networks,* 2004.
12. Ratnasamy, S. et al. Application-Level Multicast using Content-Addressable Networks. In *Proc. NGC,* 2001.
13. Rowstron, A. et al. SCRIBE: The design of a large-scale event notification infrastructure. In *Int'l. Workshop of NGC,* 2001.
14. Rowstron, A. et al. Pastry: Scalable Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Middleware,* 2001.
15. Yoneki, E. and Bacon, J. Content-Based Routing with On-Demand Multicast. *Proc. 23rd ICDCS Workshop - WWAN,* March 2004.
16. Yoneki, E. and Bacon, J. Gateway: a Message Hub with Store-and-forward Messaging in Mobile Networks. *Proc. 24th ICDCS Workshop on Mobile Computing Middleware (MCM03),* 2003.
17. Zhuang, S. et al. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proc. NOSSDAV,* 2001.

Bioinformatics Data Access Service in the ProGenGrid System

Giovanni Aloisio, Massimo Cafaro, Sandro Fiore, and Maria Mirto

Center for Advanced Computational Technologies/ISUFI, Dept. of Innovation Engineering University of Lecce, via per Monteroni, 73100 Lecce, Italy
{giovanni.aloisio, massimo.cafaro, sandro.fiore, maria.mirto}@unile.it
&
SPACI Consortium

Abstract. Current bioinformatics workflows require the collection of results coming from different tools on several Web sites. High-throughput services integrated through Web Services allow researchers to access a virtual organization by providing large computational and storage resources. There are considerable costs associated with running a high-throughput application including hardware, storage, maintenance, and bandwidth. Moreover, often such tools use biological data banks heterogeneous in the format and semantic, so the task of enabling their composition and cooperation is even more difficult. Researchers are now taking advantage of economies of scale by building large shared systems for bioinformatics processing. Integrating Computational Grids and Web Services technologies can be a key solution to simplify interaction between bioinformatics tools and biological databases. This paper presents a data access service for retrieving and transferring input data coming from heterogeneous data banks to high throughput applications, wrapped as Web Services.

Keywords: Bioinformatics, Data Management, Workflow, Grid Computing, Computational Grid, Web Services.

1 Introduction

Bioinformatics is the study of the contents and information flow in biological processes and systems. The main challenge faced by bioinformatics is the development and maintenance of an infrastructure for the storage, access, transfer and simulation of biomedical information and processes. The maturity of genomics and proteomics opens a door to the problem for decoding the functions of genes, in order to relate them with the fundamental biologic problems. Structural genomics also requires the solution of the problem of efficiently determining the 3D structure of biological macro-molecules in order to better understand their functions. Often, many processes like these are combined together, so that obtaining a valid result implies the integration of numerous applications and algorithms, combined in complex "in silico" experiments.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 211–221, 2004.
© Springer-Verlag Berlin Heidelberg 2004

Workflow techniques [1] are needed to assist the scientists in the design, execution and monitoring of their experiments. Involved high performance computing and collaboration applications require the use of large computing power and suitable hardware and software resources to provide the results shortly.
Grid technologies are now emerging in the bioinformatics field. Computational Grids [2] allow leveraging heterogeneous and independent computational resources regardless of location or platform to perform a comprehensive task and to manage huge volumes of data. Moreover, the convergence of several trends, including grid technologies and Web Services [3], has made possible a new model for bioinformatics. Processing power, storage, and network bandwidth have all advanced to the point where it is now feasible to provide high-throughput applications as Web Services.
Current bioinformatics applications often access large databases (e.g. protein data banks) without any data access service taking into account features of either applications or data type. We think that such applications could improve their performances and quality of results by providing specialized data access services. As an example, we consider the problem of executing in parallel the same bioinformatics application, e.g. the Basic Local Alignment Search Tool (BLAST) [4] sequence alignment of a target protein against different partitions/subsets of a protein database, forming a so called *parameter sweep* computation. The main problem addressed here is how the result set, obtained querying the input database, can be dynamically partitioned, its partitions assigned to the *i-th* BLAST process, and how resulting data can be suitably composed.
In particular,

- the **features of biological data** (e.g. the fact that a protein can be studied considering different representations, such as primary, secondary or ternary structure, and different annotation data),
- the **peculiar organization and heterogeneity of biological databases** (e.g., typical databases such as Swiss–Prot [5] and PDB [6] are flat text files organized by a large set of compressed files), and over all,
- the **data requirements of bioinformatics tools** (e.g. the BLAST computation requires access to the protein sequences, whereas the Rasmol visualization tool requires the secondary structures, and other text mining tools require the text annotation data),

need a specialized Data Access Services (DAS) taking into account those requirements.
Although some data access services are appearing on the Grid, such as that defined within the **DAIS-WG** (Data Access and Integration Services Working Group) of the Global Grid Forum [7], the GridDB project described in [8] or OGSA–DAI project [9], these do not (yet) offer enhanced mechanisms for specialized data access services for bioinformatics data.
On the basis of these considerations, we present a high level data access architecture offering a set of specialized services for accessing biological databases. In particular, bioinformatics–specific services will be entirely developed taking into account the previous requirements, whereas basic services will leverage GRelC

implementation [10]. Moreover, an experiment, modeled as a workflow in which data access is involved, is also described.
Both Data Access Service and Workflow are components of a distributed and ubiquitous grid environment, accessible through the web, developed at the University of Lecce in the ProGenGrid (Proteomics and Genomics Grid) project [11], for supporting "in silico" experiments in bioinformatics. In particular the DAS constitutes the data layer of our ProGenGrid bioinformatics platform, whereas the workflow component allows application modeling and scheduling.
The paper is organized as follows. Section 2 describes the ProGenGrid architecture. Section 3 presents a distributed data access service and the underlying protocol. Section 4 discusses a simple case study. Section 5 concludes the paper and discusses future work.

2 ProGenGrid System

ProGenGrid (see Fig. 1) is a software platform exploiting a Service Oriented Architecture (SOA) that wraps programs and data as Web Services and offers tools for their composition to provide ease of use, re–use and better quality of results. Services are divided in two classes:

Application-level services
- Composition of complex activities using Workflow technology for designing, scheduling and controlling bioinformatics services;
- Collaborative working for the sharing of experimental results.

Middleware-level services
- Biological database access: interaction with distributed biological data sources accessible through a uniform and standard front-end;
- Discovery and use of existing analysis tools available as Web Services for their sharing;
- Access control list to carry out the authorization process for a specific data bank.

Such services will be used by the developers to build enhanced services and will be available in a first prototype, through a web portal. As shown in Fig. 1, the main components of our system are:

- Data Access Service, for accessing and integrating biological and biomedical databases;
- Ontology, for semantic modeling of bioinformatics resources (data, tools, usage rules);
- Workflow, for composition of "in silico" experiments, i.e. application design;
- Web Portal, the interface for designing, developing, executing and monitoring applications on the Grid.

These services are built on top of the Globus Toolkit [12] and are based on Web Services technology allowing independence from platforms/programming languages and reusability of the code.

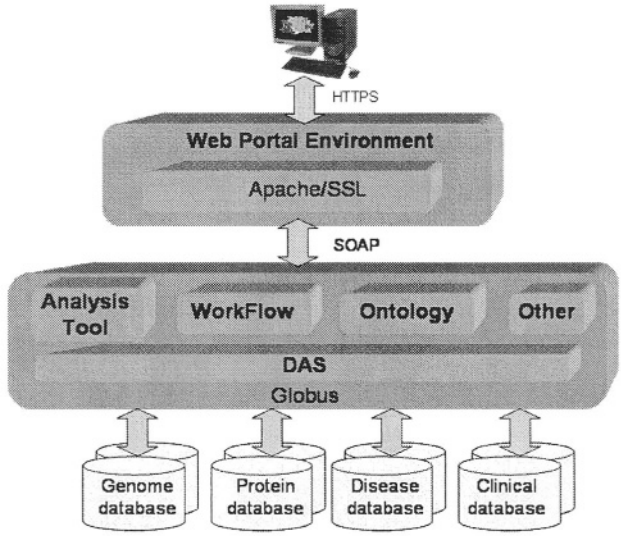


Fig. 1. ProGenGrid Architecture

3 Bioinformatics Enhanced Data Access Service

Most of the bioinformatics applications utilize databases to search for similarities between different species and their genomes. The National Center for Biotechnology Information (NCBI) [13] maintains a separate database for each of the species' genomes while the SIB (Swiss Institute of Bioinformatics) [14] together with EMBL [15], have made available a data bank of protein sequences, contained in a single flat file. Moreover, some applications such as Blast or PatSearch [16] use only the amino acids or nucleotide sequence for their computing and need specific tools (the formatdb command of the Blast algorithm or EMBOSS [17] and SRS [18] in the PatSearch case) to index them.
Based on bioinformaticians requirements, the proper database must be installed and made available to all of the computational nodes that are participating to the computation. A data grid could provide a virtual ubiquitous dataset, so that each participating node can easily access a portion or the entire required database, if needed.
At run-time, a single computational node requires data to our service (see Fig. 2). It is composed by one Broker which collects user data/application requests, and one or more low level Data Access Services (DASs) which directly, securely and transparently interact with data sources.
In the following, the main functions of these modules are described:

- **Broker:** it manages the user requests, selecting the applications that can satisfy the request (on the basis of an opportune choice of the best resource), starting the execution of one or more applications and gathering the output data to return the result in a single report. It submits several queries to

the DASs retrieving the desired information. Moreover, the Broker controls the access to a set of authorized users exploiting the GSI (Globus Security Infrastructure) protocol [19] using a specific plugin [20];
- **DAS:** exploits the GRelC Service based on a SplitQuery mechanism that firstly partitions/decomposes the result of a SELECT query in several fragments and then returns one or more fragments at each data request coming from the applications running on the computational grid nodes. This service is a component of the GRelC (Grid Relational Catalog) project [10], developed at University of Lecce, which aims at a grid-enabled access service for relational and not relational repositories. More details about the SplitQuery are available in section 3.1.

The execution of simple tasks, such as sequence comparison or complex ones designed as workflows, is demanded to the broker that schedules all of the jobs on the Grid and, if the process requires it, retrieves a data set contacting one or more DAS which translates the request to one/more queries, where queries can have different abstraction levels (initially, we thought to use the SQL standard language but we are considering other hypotheses, providing a request virtualization layer). Then, DAS directly interacts with data sources, hiding their heterogeneity and other physical details.
A query submitted to DAS may require filtering the data set. As an example, querying a protein database containing N proteins can involve different selection criteria, such as:

- **simple partitioning** (e.g., M database partitions each one containing N/M protein sequences, or structures, etc., selected in a generic or specific order);
- **length-based partitioning** (e.g., 2L database partitions, each one containing sequences whose length is equal to l, $l \in [a-L, a+L]$;
- **pattern-based partitioning/fragmentation** (e.g., all of the protein sequence partitions, each one including all of the proteins containing a given pattern Pi, i=l,..,k);
- **semantic partitioning/fragmentation** (e.g., all of the human proteins).

In the following section, we describe the SplitQuery mechanism talking about the motivations, the operation and the protocol.

3.1 SplitQuery

Several grid applications need to retrieve a lot of information coming from huge data repositories. Sometimes the computation on a retrieved dataset (obtained as a result of a SELECT query on a data repository) can be divided among several computational grid nodes simply by distributing the entire record set among them and processing the subsets locally.
In these cases, it does not matter how the data is partitioned or which subset is computed by any clients, because there are not dependencies among data stored in the same or in different subsets. This represents a key factor for this kind of query.

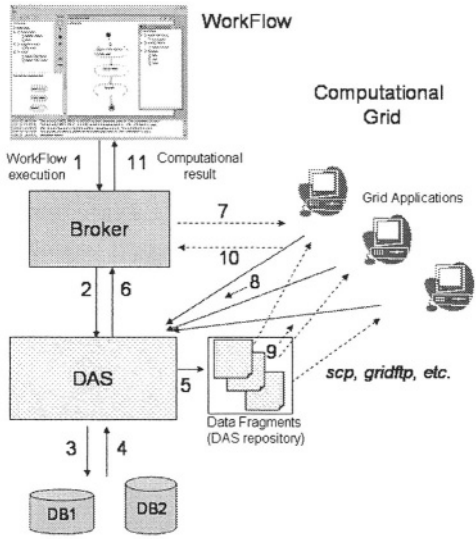


Fig. 2. DAS Architecture integrated in a Workflow

Our aim is to allocate at run-time these subsets on the idlest machines taking into account some dynamic parameters, as for instance the CPU availability of the computational grid nodes.

3.2 SplitQuery Mechanism

The SplitQuery is a two-phase delivery mechanism (Fig. 3) which allows the user (in the first phase) to submit a SELECT query to the GRelC Service, to store the result set into several Fasta or XML fragments and to immediately obtain a QueryIdentifier (QI) related to the performed query (each QI is guaranteed to be unique, no matter when it is generated).
Later (second phase), fragments can be retrieved by client applications, simply by presenting to the GRelC Service the correct QI.
As shown in Fig. 2, the broker requires (line 2) and obtains a QI (line 6) to be transferred to the application (line 7), and, using the same QI, it requires the data fragments (line 8). Upon termination, the application will send the results to the broker (line 10) that will merge them in a single report. Moreover, the broker in the request can specify the format - Fasta, to allow a rapid search of the sequences or XML - and dimension - number of tuples for each fragment - depending on several parameters of the applications.

3.3 SplitQuery Phases

The SplitQuery protocol is based of two phases:

- SplitQuery Submission;
- SplitQuery Retrieval.

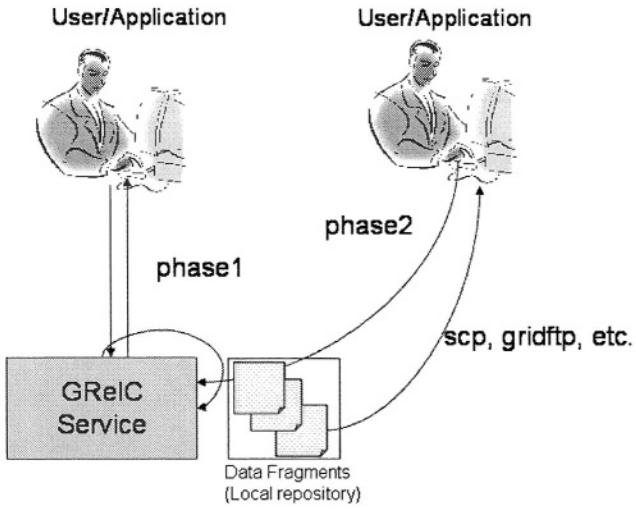


Fig. 3. Basic SplitQuery mechanism

SplitQuery Submission. In this phase the user submits the SplitQuery by using the GRelC Service and specifying the following information:
GRelC_DB: the database name on the GRelC Service side;
Query: the SELECT query to be submitted to the target database;
RecordsXFragment: the number of records to store in each XML/Fasta fragment;
timetolive: the TTL associated with this query. When this time has expired the fragments not yet retrieved will be removed on the GRelC Service side;
TransferProtocol: it represents the protocol used to transfer the fragments (i.e., GridFTP, scp, etc.);
NumberOfActiveFragments: this is the number of fragments available on the GRelC Service side as cache for the incoming requests. When some fragments are required by client applications the GRelC Service refills the local queue providing a number of available fragments equal to *NumberOfActiveFragments;*
FasterResponse: if set to on it allows to obtain the QI before the query submission; otherwise the QI will be sent back to the client after the query submission and the creation of *NumberOfFragments* fragments;
MaxNumberOfFragments: it represents the maximum number of fragments that can be retrieved each time by a single client;
ActivationTimestamp: it represents "when" the SELECT query will be submitted.

After query submission the client (for instance a broker) obtains a response containing the following information:
QI: the Query Identifier useful in the second phase to get fragments;
ResultRecordsetCreation: the response for the SELECT query submitted. If *FasterResponse* is set to on, this parameter is not available in the first phase.

SplitQuery Retrieval. In this phase the user retrieves fragments by using the SplitQuery Retrieval mechanism and specifying the following information:

- *QI:* to specify the query;
- *Fragments:* the number of fragments to be retrieved.

After this request the client (for instance a Grid application) obtains a reply containing the following information:

- *Fragments:* the data requested;
- *RetrievalResult:* return value related to the data transfer and the SplitQuery Retrieval result.

3.4 Applying SplitQuery to the Blast Application

Currently, we are implementing a web service version of the Blast algorithm (written in C language to improve performance), according to our architecture, which exploits the GSI protocol for granting secure access to trusted users and wraps the NCBI's Blast version. Our SplitQuery service will provide the input for this service, according to the policy described in the previous section. So the Broker Service, should call more Blast web service instances and merge the results. It is worth noting here that a simple merge will result in a faulty output. Indeed, to have a correct Blast result about some statistical parameters, the effective database size and the effective query length are needed (both are calculated using the length adjustment) but are normally not available when we run the blastall command for just a subset of the database. A solution could be the use of a patch available in the NCBI BLAST, as in the dBlast approach [21]. Applying this patch, we can get these critical (statistical) values, on the basis of the options specified by the user. Subsequently, the "sub-jobs" are started with these values and will have the same statistics as if these were run as one BLAST job on the entire database. At the end of the computation a simple text merge will return the correct results.
To provide approximate e-value statistics (measure of the likelihood that the alignment is a random match between the query and database) some approaches use a linear-regression model (i.e. in the Blast.pm [22]) whereas other ones exploit only the effective database size (i.e. in the parallelBlast [23] and mpiBlast [24]). So, we would like to investigate these methods and other ones for obtaining accuracy requirement in our experiments.

4 Using DAS in a Workflow: Case Study

Recently many workflow languages have been defined such as Web Services Flow Language (WSFL) [25], Business Process Execution Language (BPEL) [26], and also some UML extensions. We use the UML (Unified Modeling Language, [27]) activity diagram as a workflow language specification. UML (as well as all of its extensions) is the most widely accepted notation for designing and understanding complex systems; it has an intuitive graphical notation, and UML activity

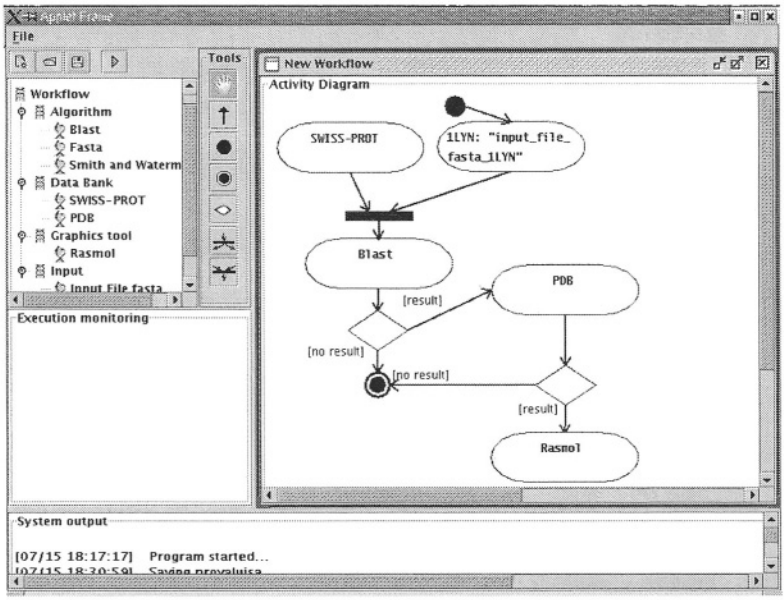


Fig. 4. Workflow of a bioinformatics experiment of sequence comparison

diagrams support [28] most of the control flow constructs suitable to model workflow execution.

As an application of ProGenGrid, we present a workflow modeling the process of searching similarity matching among proteins. Figure 4 depicts our ProGenGrid prototype showing an activity diagram specification of the similarity search process. This process starts by supplying a target protein < IDProtein > or its FASTA format (in this example, the protein target is 1LYN), and the search procedure accesses the database, i.e., all of the information about target protein are recovered from the Swiss-Prot database. The results are stored in a file used as input for the similarity analysis task using the Blast algorithm that computes a list of proteins sorted according to a similarity measure taking into account the protein sequence with respect to the target protein. Blast returns a list of similar proteins, such proteins are given in input to the PDB database to obtain their secondary structure that can be visualized using Rasmol [29]. Moreover, the secondary structure of the target protein can also be retrieved by the PDB database and visualized through Rasmol, and its graphical representation can be compared with respect to each similar protein produced by Blast.

In a simple experiment, such as the one describe above, our data access service is fundamental to access the Swiss-Prot and PDB data banks to retrieve the data.

5 Conclusions and Future Work

In this work an enhanced data access service for accessing biological databases in the ProGenGrid platform has been presented. It takes into account the features

of biological data and bioinformatics tools requirements. In particular the new SplitQuery approach to access the biological data banks has been also described discussing its protocol and retrieval mechanisms. At present this service is partly implemented and future work will provide the full implementation of the entire architecture, measuring the performance with respect to other approaches of distributed high throughput applications such as Grid Blast [30]. Finally, after implementing all of the components as Web Services, we will move toward a Grid Services architecture (i.e., the Open Grid Service Architecture [31], based on the emerging WSRF [32]) in order to offer more functionalities and deploy the Bionformatics infrastructure needed to support "in silico" experiments.

References

1. WfMC. Workflow management coalition reference model. Site address, http://www.wfmc.org/.
2. I., Foster, C., Kesselman. The Grid: Blueprint for a New Computing Infrastructure, Published by Morgan Kaufmann (1998).
3. Kreger, H. "Web Services Conceptual Architecture.", WSCA 1.0. IBM, 2001.
4. Altschul, Stephen F., Gish Warren, Webb Miller, Eugene W. Myers, and David J. Lipman (1990). Basic local alignment search tool. J. Mol. Biol. 215:403–410.
5. B., Boeckmann, A., Bairoch, R., Apweiler, M., Blatter, A., Estreicher, E., E., Gasteiger, M. J., Martin, K., Michoud, C., O'Donovan, I., Phan, S., Pilbout, and M., Schneider. The Swiss-Prot protein knowledgebase and its supplement TrEMBL. Nucleic Acids Research 31: 365-370 (2003). Site address: http://www.ebi.ac.uk/swissprot/.
6. H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne. The Protein Data Bank. Nucleic Acids Research, 28 pp. 235-242 (2000). Site address: http://www.rcsb.org/pdb/.
7. Global Grid Forum (GGF). Site address: www.gridforum.org.
8. D. T. Liu, M. J. Franklin. GridDB: A Data-Centric Overlay for Scientific Grids. Technical Report UCB//CSD-04-1311, 3/16/04. Site address: http://www.cs.berkeley.edu/ dtliu/pubs/griddb_tr.pdf.
9. Open Grid Services Architecture Data Access and Integration OGSA-DAI, Site Address: http://www.ogsadai.org.uk/.
10. G. Aloisio, M. Cafaro, S. Fiore, M. Mirto. The GRelC Project: Towards GRID-DBMS, Proceedings of Parallel and Distributed Computing and Networks (PDCN) IASTED, Innsbruck (Austria) February 17-19 (2004). Site address: http://gandalf.unile.it.
11. G., Aloisio, M. Cafaro, S. Fiore, M. Mirto. ProGenGrid: A Grid Framework for Bioinformatics. To appear in the Proceeding of International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics (CIBB 2004), September 14-15 2004, Perugia, Italy.
12. I., Foster, C., Kesselman. Globus: A Metacomputing Infrastructure Toolkit, Intl J. Supercomputer Applications, Vol. 11, 1997, No. 2, pp. 115-128.
13. National Center for Biotechnology Information. Site address: http://www.ncbi.nlm.nih.gov/.
14. Swiss Institute of Bioinformatics. Site address: http://www.isb-sib.ch/.
15. EBML-EBI European Bioinformatics Institute. Site address: http://www.ebi.ac.uk/swissprot/.

16. G. Grille, F. Licciulli, S. Liuni, E. Sbisa' and G. Pesole (2003). PatSearch: a program for the detection of patterns and structural motifs in nucleotide sequences. Nucleic Acid Research Vol.31 no.13 pp. 3608-3612.
17. Rice, P. Longden, I. and Bleasby, A. EMBOSS: The European Molecular Biology Open Software Suite. Trends in Genetics June 2000, vol 16, No 6. pp. 276-277. Site address: http://www.ch.embnet.org/EMBOSS/.
18. SRS Network Browser. Site address: http://www.ebi.ac.uk/srs/srsc/.
19. S. Tuecke. Grid Security Infrastructure (GSI) Roadmap. Internet Draft 2001. Site address: www.gridforum.org/security/ggf1_2001-03/drafts/draft-ggf-gsi-roadmap-02.pdf.
20. G. Aloisio, M. Cafaro, D. Lezzi, R. Van Engelen, "Secure Web Services with Globus GSI and gSOAP", Proceedings of Euro-Par 2003, 26th - 29th August 2003, Klagenfurt, Austria, Lecture Notes in Computer Science, Springer-Verlag, N. 2790, pp. 421-426, 2003.
21. van Driel, Marc A., Hekkelman, Maarten L., and Rodriguez R. "dBlast. A wrapper to run NCBI BLAST parallel/distributed". Submitted. Site address: http://www.cmbi.kun.nl/software/dBlast/.
22. BioPerl Project. S. A. Chervitz, G. Fuellen, C. Dagdigian, R. Resnick & S. E. Brenner. Bioperl: Object-Oriented Perl Modules for Bioinformatics. Objects in Bioinformatics Meeting, 1997. Site address: http://bio.perl.org/Projects/Blast/.
23. David R. Mathog. Parallel BLAST on split databases. Bioinformatics. 2003 Sep 22;19(14):1865-6.
24. A. Darling, L. Carey, and W. Feng. The Design, Implementation, and Evaluation of mpiBLAST. ClusterWorld Conference & Expo in conjunction with the 4th International Conference on Linux Clusters: The HPC Revolution 2003, San Jose, CA, June 2003. Site address: http://mpiblast.lanl.gov/.
25. IBM. Web services flow language - wsfl. Site address: http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.
26. IBM. Business process execution language for web services - bpel4ws. Site address: http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/.
27. OMG. Uml - unified modeling language: Extensions for workflow process definition. Site address: http://www.omg.org/uml/.
28. R. Eshuis and R. Wieringa. Verification support for workflow design with UML activity graphs. In CSE02. Springer Verlag, 2002.
29. Roger A. Sayle and E. J. Milner-White. RasMol: Biomolecular graphics for all. Trends in Biochemical Science (TIBS), September 1995, Vol. 20, No. 9, p.374. Site address: http://www.umass.edu/microbio/rasmol/.
30. North Carolina BioGrid Project. Site address: http://www.ncbiogrid.org/index.html
31. Foster, I, Kesselman, C., Nick, J., & Tuecke, S. (2002). The Physiology of the Grid: An Open Grid Services Architecture for Distributed System Integration. Technical Report for the Globus project. Site address: http://www.globus.org/research/papers/ogsa.pdf.
32. WS - Resource Framework (WSRF). Site address: http://www.globus.org/wsrf/.

MSF: A Workflow Service Infrastructure for Computational Grid Environments

Seogchan Hwang[1] and Jaeyoung Choi[2]

[1] Supercomputing Center, Korea Institute of Science and Technology Information,
52 Eoeun-dong, Yusung-gu, Daejun 305-306, Korea
seogchan@kisti.re.kr
[2] School of Computing, Soongsil University,
1-1 Sangdo-5dong, Dongjak-gu, Seoul 156-743, Korea
choi@ssu.ac.kr

Abstract. Globus has become a standard in the construction of Grid computing environments. However, it still needs more work and research to satisfy requirements from various grid applications such as workflow services. We propose a Meta Scheduling Framework (MSF) for workflow service management based on the Globus toolkit. The MSF provides an XML-based Job Control Markup Language (JCML) for describing information and procedures of applications including dependencies of jobs, a workflow management engine for scheduling complicated job flow, and an execution environment without the need for code modification or wrapping.

1 Introduction

Grid computing [1] is a new infrastructure which provides computing environments for grand challenge problems by sharing large-scale resources. The Globus toolkit is a standard in constructing a Grid and provides essential grid services such as security, resource management, data transfer, information service, and so on. However, it still needs more work and research to satisfy the requirements of various grid applications. Workflow management is emerging as one of the most important grid services, yet it is difficult to use the grid resources for general applications because of various characteristics such as heterogeneity and dynamic organization. Numerous research groups have been working on workflow related projects.

GridFlow [2] is a workflow management system, which uses agent-based resource management and a local resource scheduling system, Titan. It focuses on the scheduling of time-critical grid applications in a cross-domain and highly dynamic grid environment by using a fuzzy timing technique and performance prediction of application. MyGrid [3] provides services such as resource discovery, workflow enactment, and distributed query processing for integration. It is a research project middleware to support biological environments on a Grid. And Condor [4] provides a workload management system for compute-intensive jobs and a scheduling of dependencies

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 222–231, 2004.
© Springer-Verlag Berlin Heidelberg 2004

between jobs using DAGMan. This project provides similar functionalities but requires its own specific infrastructure.

In this paper, we introduce a system, called Meta Scheduling Framework (MSF), for grid computational environments. MSF provides a Job Control Markup Language that is able to specify the job flow for general applications which are not developed for grid environments. It also provides a workflow management service, based on Globus, to execute the job, and a graphical user interface to facilitate the composition of grid workflow elements and access to additional grid resources

In section 2, we illustrate the structure of the MSF and its components in detail. Section 3 describes the implementation of the system and examines a sample application, AutoDock. Finally, we discuss the conclusion of the research in Section 4.

2 Meta Scheduling Framework

Usually grid applications require complicated processing steps, for instance, when a task requires an input file which is the output of another task, and the two tasks are simultaneously running in separate computing nodes. Therefore, a job description language is essential to describe dependencies among jobs, and a management system is required to control the flow of tasks.

The goal of this research is to develop a framework to provide a workflow service to applications using the Globus. To accomplish this, we designed a workflow de

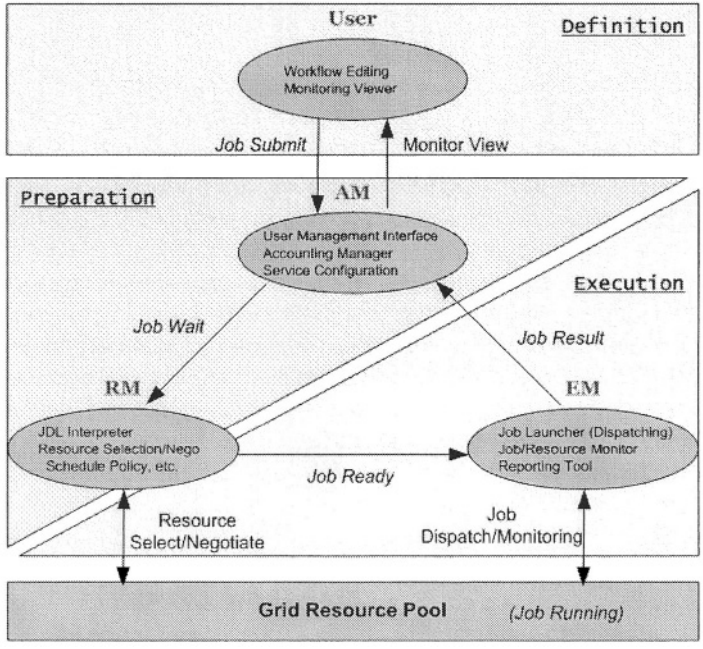


Fig. 1. Meta Scheduling Framework Architecture

scription language, called Job Control Markup Language (JCML), and a workflow management system. The JCML is designed to describe a process of tasks. And the workflow management system provides services to control the flow of tasks.

The architecture of MSF is depicted in Figure 1. A user describes a job flow using the MSF console. The Access Manager (AM) provides services which include user authentication, environment setup, and a job submission service. The Resource Manager (RM) provides resource discovery and matching. The Execution Manager (EM) provides job launching, monitoring, and reporting. More details are described in Section 2.3.

MSF consists of three phases: definition, preparation, and execution. During the definition phase, jobs are defined to specify a Job Definition List (JDL), which describes a task flow using JCML. In this phase, users connect to the AM for authentication of the MSF and then the AM creates a user proxy for globus. In the preparation phase, resources are searched and assigned to the matched tasks. Next, the AM creates an agent to provide a proxy service to the user. The agent then passes the JDL and traces the status of the job. The RM receives the JDL from the AM and analyzes it. After finding appropriate grid resources for the job, the RM then assigns them to tasks and generates a worklist that includes information on activities and their executing order. Finally, during the execution phase, the tasks on the worklist are executed, the status is monitored, and the results are reported.

2.1 Job Control Markup Language

A job description language to specify the flow of an application task must provide a way to describe various grid environments and task information including execution environments, arguments, sequence, data dependency, prefetching, and so on. The JCML is a workflow description language based on the Graph eXchange Language (GXL) [5], which defines a graph-based XML representation for specifying the dependencies among components. The JCML consists of four major elements: Info, Resource, Component, and Dependency. Figure 2 shows the JCML structure.

Info: This element lists the document name, scope, target namespaces, authoring date, and so on.
Resource: This element describes the resources of hardware and software required to execute a job. The hardware includes architecture, CPU, memory, disk, and network bandwidth. The software includes an operating system, installed application, and local scheduler. The Time represents the deadline to be executed.
Component: This element lists all of the task-related information. A Node is an executing program or computer in the workflow. A node is classified into a task and a resource. A task node is an executing program in workflow, while a resource node is an assistant, which supports the task node, and represents the physical computing resources, such as storage and database. While a task node includes execution file, input, output, arguments, and resource configuration, a resource node includes data location and the access method. If it is necessary to refer to a series of nodes as a

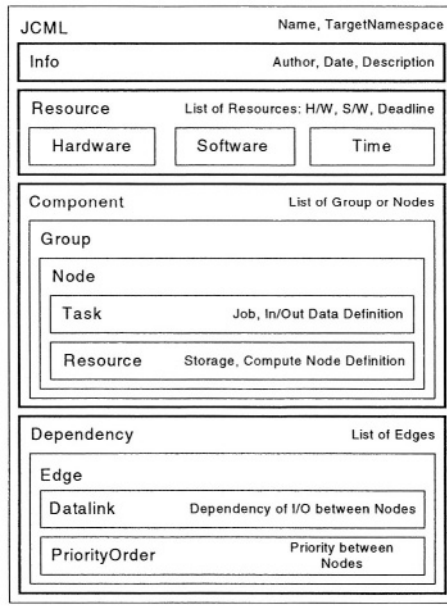


Fig. 2. JCML Structure

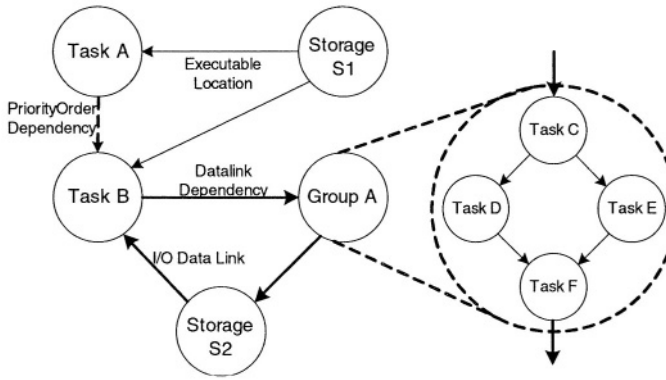


Fig. 3. Example model of JCML

single node according to a job flow logic, a group is logically a node. One group can include another group(s). Figure 3 illustrates an example model of JCML. A circle and a dotted circle represent a node and a group, respectively. The example is composed of two storage nodes, Storages S1 and S2; two task nodes, Tasks A and B; and a group, Group A, which includes four task nodes: Tasks A, B, C, and D.

Dependency: This element describes the dependencies of a workflow. Each line is an edge which represents an execution order and a dependency between two objects (node or group). It has two types of links, PriorityOrder and Datalink, which have a

direction that expresses a starting point and an end point of linked nodes. The PriorityOrder just represents an execution order between the two linked nodes. The Datalink displays a flow of data which is used for the input or output file of each task.

As shown in Figure 3, each line represents an edge between the linked nodes, and all edges have directions. A PriorityOrder expressed in a dotted line is used for task nodes, but does not include resource nodes. Tasks A and B are linked to Storage S1 with solid lines, which imply the location of execution files. When the task is running, the execution file is transferred to a computing node selected by the scheduler. The edge which links Task B and Group A represents a dependency between two nodes, transferring the output of Task B to the input of Group A. The edge linking with Storage S2 shows that Task B reads the input from S2 and Group A writes the output to S2. Next, Task C of Group A receives the input from Task B and sends the output to Tasks D and E. Task F then receives two input files from Tasks D and E and finally writes the output to S2.

2.2 Workflow Management Systems

A workflow management system guarantees that a flow of tasks will be executed exactly. MSF provides a workflow service, which is a scheduler-based paradigm [6] based on a state transition machine.

Figure 4 illustrates the processing steps of a job in the workflow engine. A job is processed by a workflow management system as follows: analyzing the job, mapping resources, generating a worklist, and scheduling tasks. A user specifies a JDL to execute a grid application. After analyzing the JDL, the JDL Analyzer searches resources and assigns resources to the job, selected by the Resource Discovery and Selector, which uses an external information service such as MDS [7] and NWS [8], and then generates the worklist. The worklist has the execution information of the task

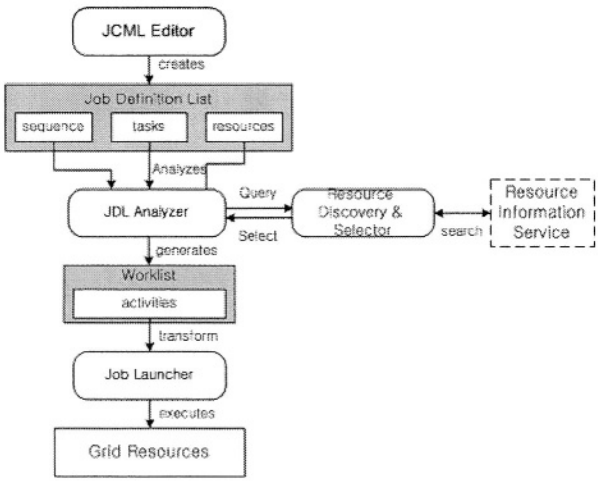


Fig. 4. Workflow Management Architecture

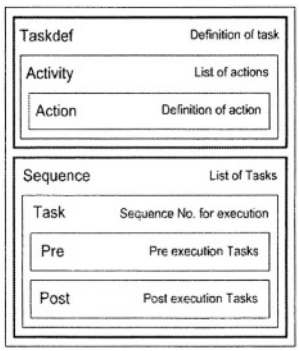


Fig. 5. Worklist Structure

and its execution order. Each task consists of activities. Figure 5 shows the worklist structure.

The Taskdef describes the behavior of a task, or a list of activities (actions) such as file copy or running programs, which all have an execution order. The Sequence describes the sequence information of Taskdef. The tasks, each with their own sequence numbers, are executed according to their sequence number and their causal ordering specified in Pre and Post in Task.

```
<node id="d" type="applicationJob">
 <excutable name="mkdpf3" location="/usr/local/autodock/share/" arguments=""/>
 <application name="autodock" version="3.0.1" />
 <configuration>
  <env name="workDir" value="/home/seventy9/queue/job"/>
  <envname="PATH" value="/bin:/user/bin:/bin/usr/local/autodock/bin:/usr/local/
autodock/share"/>
 </configuration>
 <data>
  <in id="dia" name="l.pdbq" location="" cmdArg="" link="yes"/>
  <in id="dib" name="p.pdbqs" location="" cmdArg="" link="yes"/>
 </data>
</node>
<edge id="bd" from="b" to="d" direction="directed" type="datalink"></edge>
<edge id="de" from="d" to="e" direction="directed" type="datalink"></edge>
```

(a) Job Definition List

```
<action id="3">
 <execution>
  <resource>
   <host name="ironfly.ssu.ac.kr" port="2119"/>
   <workDir path="/home/seventy9/queue/job"/>
   <env name="PATH" value="/bin:/usr/bin:
      /bin/usr/local/autodock/bin:/usr/local/autodock/share"/>
  </resource>
  <command>
   /usr/local/autodock/share/mkdpf3 l.pdbq p.pdbqs
  </command>
 </execution>
</action>
```

(b) Worklist

```
&(directory="/home/seventy9/queue/job")
(arguments="l.pdbq""p.pdbqs")
(executable = "/usr/local/autodock/share/mkdpf3")
(environment=("PATH" "/bin:/user/bin:/bin/usr/local/autodock/bin:
 /usr/local/autodock/share"))
```

(c) Globus RSL

Fig. 6. Three Job Description Formats in MSF

The Job Launcher executes activities according to the worklist. At the workflow engine, it is required to transform activities in the worklist to RSL [9] type in order to execute the real task in a local grid scheduler. Figure 6, for example, shows the three job description formats in MSF: (a) a partial definition of a node and an edge in JDL, (b) an action of a Taskdef in worklist, and (c) RSL command.

2.3 Components Architecture

The MSF, as shown in Figure 7, has four major components: (a) Access Manager (AM) manages the user connection and requests, (b) Resource Manager (RM) is in charge of resource discovery and assignment, (c) Execution Manager (EM) controls workflow, and (d) Information Manager (IM) provides all event data to other system components.

The main purpose of AM is to manage the connection between clients and the MSF. If a client connects to the MSF, then the AM accepts it and performs authentication and authorization of the user. If the client is verified, the Master Agent creates a new User Agent. The User Agent manages the client session and performs some functions: submitting a job, tracking the status of a client request, and providing a service configuration.

RM provides two functions, resource discovery and allocation. The JDL Interpreter analyzes the JDL received from AM and forwards the resource information to the Resource Collector, which searches resources using an external Grid Information

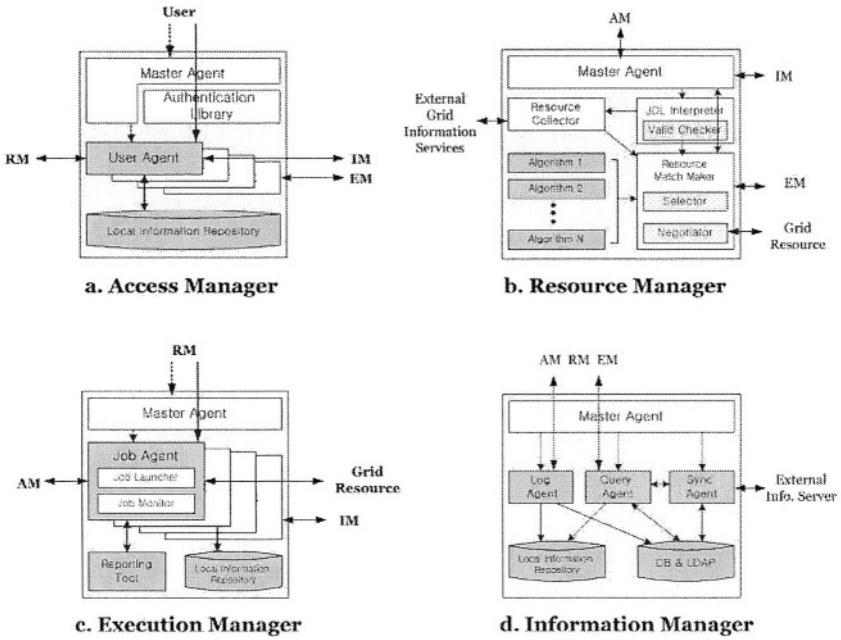


Fig. 7. MSF Components Architecture

Service. The Resource Match Maker selects appropriative resources for the job using a matchmaking algorithm, which can be replaced with another in a plug-in manner. The selected resources are then negotiated by the Negotiator. Finally, the RM generates a worklist, and passes it to the EM.

The EM carries out job launching, monitoring, and reporting. A Master Agent creates a Job Agent corresponding to the job. The Job Agent then assigns the activities of the tasks to selected resources according to the order of the worklist, which monitors the status of running tasks and reports of the result of each task.

The IM controls the event log data from the other components and consists of three agents: the Log Agent which records all events from other components; the Query Agent which searches for the requested query; and the Sync Agent which synchronizes the data with other IM, if necessary.

3 Implementation and Examination

We developed a prototype of MSF using pure Java (JDK 1.4.0) and Java CoG Kit (version 0.9.13) on Globus 2. We also implemented and executed a Virtual Screening on MSF. A Virtual Screening is one of the design methods, called *docking* in Bioinformatics, which combines one protein with many, sometimes hundreds of thousands of ligands to discover candidates for new drugs. In order to execute docking, the format of the material must be changed. We chose the AutoDock [10] application for this experiment. Figure 8(a) shows the processing steps involved in transforming the format of the protein and the ligand for docking in AutoDock. The dotted square represents a process for protein transformation, which is executed just once during the virtual screening. Figure 8(b) illustrates the JCML process to make a workflow used by MSF for AutoDock processing. A circle represents a program which transforms each format according to the AutoDock processing steps. The programs of babel, mol2topdbq, mkgpf3, and mkpd3, are commands which transform to the format of mol2, PDBQ, GPF, and DPF, respectively. And finally autogrid3 and autodock3 calculate the docking energy of the two materials, the protein and the ligand.

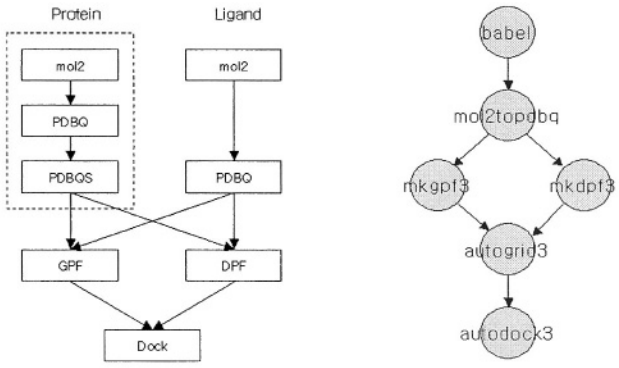


Fig. 8. (a) AutoDock Procedure Steps (b) JCML Process Model

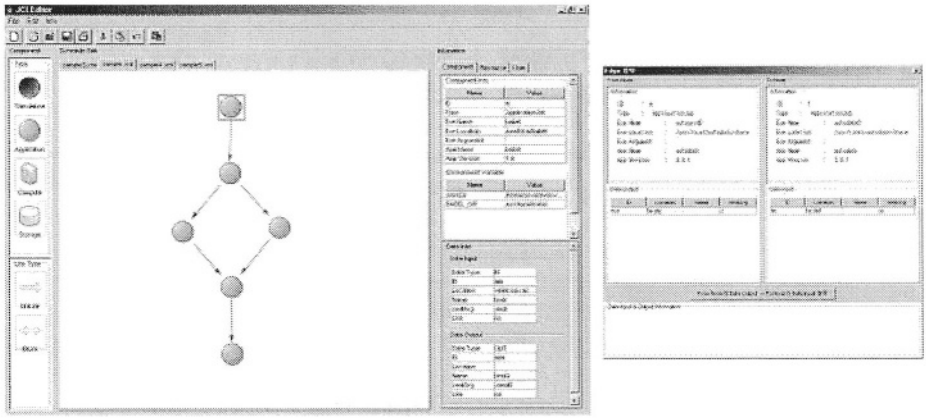


Fig. 9. JCML Editing Windows: (a) Main Window, (b) Edge Window

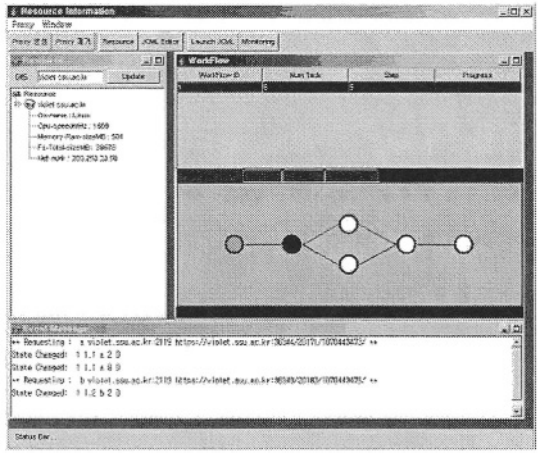


Fig. 10. MSF main console

After JCML modeling, it is required that the job flow be specified. Figure 9 displays an editing window for the AutoDock. We drew a workflow for AutoDock using the task and edge icons, and specified the task information in the right side of the main window. We also set up the dependencies among nodes in the edge window.

Figure 10 displays the main console which monitors the status of the job flow. The main console has three windows: a resource window which shows the current state of the selected resource, a workflow window which illustrates the status of each task in the workflow, and an event window which displays the event from the task.

4 Conclusion

In this paper, we described the architecture of a Meta Scheduling Framework. The main purpose of a MSF is to provide a workflow service for general applications in a Grid environment. Therefore, we designed and implemented a workflow description language, JCML, to describe the flow of application, including complexity and dependencies, and a workflow management system to execute and monitor this flow.

Currently, we are working to extend the architecture in order to enhance efficiency and availability and to describe jobs in greater detail with JCML. In addition, a new version of Globus 3.0, OGSA [12], has been released, which integrates scientific and enterprise environments based on web service. The OGSA platform will be supported by MSF in the near future.

References

1. I. Foster and C. Kesselman, ed., The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, (1998).
2. J. Cao, S. A. Jarvis, S. Saini and G. R. Nudd, GridFlow: Workflow Management for Grid Computing, 3rd International Symposium on Cluster Computing and the Grid, (2003), 12-15.
3. R. Stevens, A. Robinson and C. Goble, myGrid: personalized bioinformatics on the information grid, Bioinformatics, 19(1), (2003), 302-304.
4. M. Litzkow, M. Livny and M. Mutka, Condor - A Hunter of Idle Workstations, 8th International Conference of Distributed Computing Systems, (1998), 13-17.
5. A. Winter, B. Kullbach and V. Riediger, An Overview of the GXL Graph Exchange Language, Software Visualization. Lecture Notes in Computer Science, Vol. 2269, (2002), 324-336.
6. A. Cichocki, A. S. Helal, M. Rusinkiewicz and D. Woelk, eds., Workflow and Process Automation: Concepts and Technology, Kluwer, (1998).
7. K. Czajkowski, S. Fitzgerald, I. Foster and C. Kesselman, Grid Information Services for Distributed Resource Sharing, 10th International Symposium on High-Performance Distributed Computing, (2001), 181-184.
8. R. Wolski, N. Spring and J. Hayes, The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing, Future Generation Computing Systems, 15(5/6), (1999), 757-768.
9. The Globus Resource Specification Language, http://www.globus.org/gram/rsl_spec1.html.
10. AutoDock, http://www.scripps.edu/pub/olson-web/autodock/.
11. Python Molecule Viewer, http://www.scripps.edu/~sanner/python/pmv/webpmv.html.
12. I. Foster, C. Kesselman, J. Nick and S.Tuecke, The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, Open Grid Service Infrastructure WG. Global Grid Forum, (2002).

Using PROTEUS for Modeling Data Mining Analysis of Proteomics Experiments on the Grid

Mario Cannataro, Pietro Hiram Guzzi, Tommaso Mazza, and Pierangelo Veltri

University Magna Græcia of Catanzaro, 88100 Catanzaro, Italy
{cannataro, hguzzi, t.mazza, veltri}@unicz.it

Abstract. Novel experiments in bio-medical domain involve different technologies such as mass spectrometry, bio-molecular profiling, nanotechnology, drug design, and of course bioinformatics. Bioinformatics platforms should support both modelling of experiments, and collection, storing and analysis of the produced data. The advent of proteomics has brought with it the hope of discovering novel biomarkers that can be used for early detection, prognosis and treatment of diseases. Mass Spectrometry (MS) is widely used for the mass spectral identification of the thousands of proteins that populate complex biosystems such as serum and tissue. Data Mining (DM) is the semi-automated extraction of patterns representing knowledge implicitly stored in large databases. The combined use of MS with DM is a novel approach in proteomic pattern analysis and is emerging as an effective method for the early diagnosis of diseases. However, it involves large data storage and computing power so it is natural to consider Grid as a reference environment. The paper presents how PROTEUS, a Grid-based Problem Solving Environment for bioinformatics applications, allows formulating, modelling and designing of proteomics experiments involving DM analysis of MS data.

Keywords: Grid-based Problem Solving Environment, Data Mining, Workflow, Ontology, Mass Spectrometry, Biomarker discovery.

1 Introduction

Currently, complex experiments in the bio-medical domain comprise different tasks involving many, and possibly heterogeneous, technological platforms such as mass spectrometry and bio-molecular profiling, nanotechnology, computational chemistry and drug design, and so on. Each of these platforms uses and produces data in different formats having different meanings, and the main role of a bioinformatics platform is to glue those different tasks of complex experiments, collecting raw data and discovering new knowledge by applying advanced algorithms (e.g. sequence alignment, structure prediction, docking, data mining, etc.). We think that bioinformatics platforms should support the modelling and building of such complex experiments, that can be considered as a composition of 'in vivo' (e.g. experiment on a live rat, smart drug delivery through

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 232–243, 2004.
© Springer-Verlag Berlin Heidelberg 2004

nano-devices), 'in vitro' (e.g. experiment on a cell line), and 'in silico' (i.e. bioinformatics analysis of biological data) tasks.

In particular, after decoding human *genome,* today the new challenge is studying the *proteome,* i.e. the set of proteins encoded by the genome, to define models representing and analyzing the structure of the proteins contained in each cell and, eventually, to prevent and cure any possible cell-mutation generating human diseases. Proteomic analysis is a widely used powerful technique, in order to identify different molecular targets in different pathological conditions. For example, innovative methodologies for discovering novel therapeutic approaches in cancer could obtain a molecular profiling of cancerous cells by using mass spectrometry and proteomic analysis, this profile could be enhanced combining microarray experiments, new drugs could be designed by using both proteomic (docking) techniques and computational chemistry methodologies, and finally, better effectiveness of therapeutic effect could be obtained employing smart drug delivery through nano-devices [7].

Bioinformatics platforms have been designed to support application in biomedicine [11]. Their main characteristics are the following.

Naturally distributed. Different data sets are stored in different web servers [1].

Resource Consuming. Data sets have large sizes and complexity of computations is not negligible. In fact data formats are often heterogeneous and a lot of time is required to interface these different formats. [14]

Privacy aware. Data exchanged between organizations are often relative to patients, for this a secure transport layer is crucial.

A grid-based Problem Solving Environment (PSE) [16], can help researchers to define and compose complex applications, hiding software details and easing composition. Grid infrastructure [8], [10], because of its security, distribution, service orientation and computational power is thus the natural implementation for bioinformatics Problem Solving Environments.

We developed PROTEUS [4], a grid-based Problem Solving Environment running on top of Grid middleware that permits to compose and run bioinformatics application. Main ideas of our PSE are:

- Modeling bioinformatics processes through ontologies and metadata;
- Implementing a component-based programming model;
- Ontology-based composition of Open Source Software components and heterogeneous data sources (biological databases);
- Designing and scheduling distributed bioinformatics applications using workflow techniques;
- Deploymnent on Grid infrastructure.

The paper is structured as follows. Section 2 introduces the structure of Mass Spectrometry data with reference to Matrix Assisted Laser Desorbtion/Ionisation-Time Of Flight (MALDI-TOF) spectrometer, and describes a general model of a *Mass Spectrometry Experiment.* Section 3 presents the platform for mass spectrometry data analysis and its features. Section 4 describes the use of ontologies in PROTEUS with particular attention about PROTON

and DAMON ontologies. Finally, Section 5 concludes the paper and describes future work.

2 Mass Spectrometry Data Analysis: Modelling with PEDRo

In this section we introduce the structure of MS data and then present a model of a typical bioinformatic process: the *Mass Spectrometry Experiment.* Since all phases of MS experiments could impact the data analysis phase, we briefly explain how to prepare a sample, then we deal with data preprocessing and data mining with reference to MS data analysis [5].

2.1 Using of Mass Spectrometer and MALDI-TOF Data

The mass spectrometer is an instrument designed to separate gas phase ions according to their m/Z (mass to charge ratio) value. The *heart* of the mass spectrometer is the analyzer. This element separates the gas phase ions. The analyzer uses electrical or magnetic fields, or combination of both, to move the ions from the region where they are produced, to a detector, where they produce a signal which is amplified. Since the motion and separation of ions is based on electrical or magnetic fields, it is the mass to charge ratio, and not only the mass, which is of importance.

Output of mass spectrometer is a spectrum (see Fig. 1) whose correct interpretation is crucial in proteomic analysis. Historically this phase has been performed by humans in different field. Proteomic mass spectra, diversely, have a major complexity and a human analysis is unfeasible. Automatic interpretation need to be driven with empirical rules in order to obtain knowledge from data. In fact each phase described in Fig. 2 introduces a modification in spectrum obtained.

In our research we are interested to mine Matrix Assisted Laser Desorbtion/Ionisation-Time Of Flight (MALDI-TOF) spectrometry data. In this approach, ionisation is obtained by:

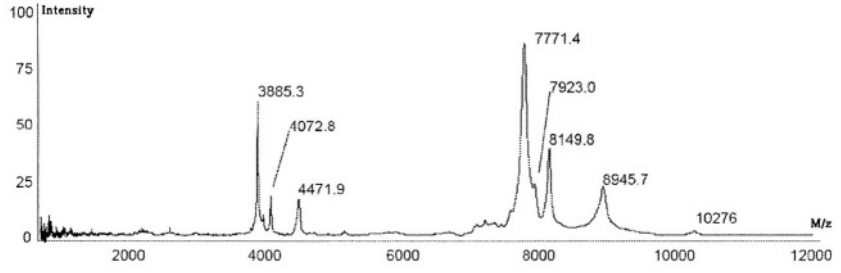


Fig. 1. Peptide/Protein profile of a biological sample, Low Mw window: 1000-12000 *m/z*

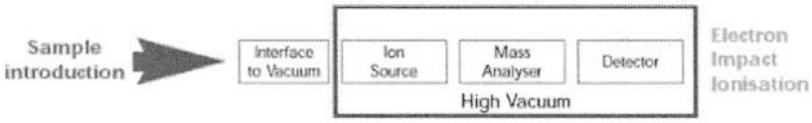


Fig. 2. Mass Spectrometer step by step

1. Formation of a solid solution;
2. Matrix Excitation by laser excitation;
3. Analyte Ionization.

Ions formed in a MALDI ion source are extracted and accelerated by an electric field into an analyzer consisting of a long straight *drift tube*. The ions pass along the tube until they reach a detector. After the initial acceleration phase, the speed reached by a ion is inversely proportional to its mass (strictly, inversely proportional to the square root of its m/Z value). Since the distance from the ion origin to the detector is fixed, the time taken for a ion to traverse the analyzer in a straight line is inversely proportional to its speed and hence proportional to its mass. Thus in a MALDI-TOF MS spectrum each m/Z value of horizontal coordinates has its characteristic time-of-flight from the source to the detector expressed by an electrical intensity.

When obtaining a MALDI spectra we have to consider some imperfection causes: (i) noise, (ii) peak broadening, (iii) instrument distortion, (iv) carbon-13, (v) saturation, (vi) miscalibration, (vii) contaminants of various kinds, (viii) cations other than protons. Data cleaning is performed in different times using: (i) Best-practices sample preparation; (ii) Mass-spectrometer software; (iii) Appropriates algorithms in data pre-processing phase. In some applications, e.g. identification of proteins, research has focused on obtaining a list of peaks, each of which represents a peptide. In our application we stores the raw data consisting in a large collection of (m/z, intensity) couples, as in Fig. 3. In this representation of MS data and, in particular, in MALDI-TOF output raw data, the m/Z ratio is normalized. Usually, the charge detected is unitary therefore the value of the m/Z ratio is indicative of the real mass detected.

2.2 Modelling and Representation of Experiment Schema

The proteomics field is moving towards high-throughput technologies. To take advantage of this, it is necessary that standards are developed. *PEDRo* (Proteomics Experiments Data Repository) is described as a *modular, extensible model* [15], and since we are trying to build a modular system this is very convenient.

There exists a need for public repositories for proteomic data that allows for rigorous comparisons between experiments based on gels, arrays and so on. These repositories must capture information such as where the data comes from, who made it and how, as well as annotations and identification. PEDRo could

5274.807	4.000
5274.839	3.000
5274.872	4.000
5274.904	2.000
5274.936	6.000
5274.968	4.000
5275.001	3.000

Fig. 3. {m/z, intensity} Mass Spectrometry raw data

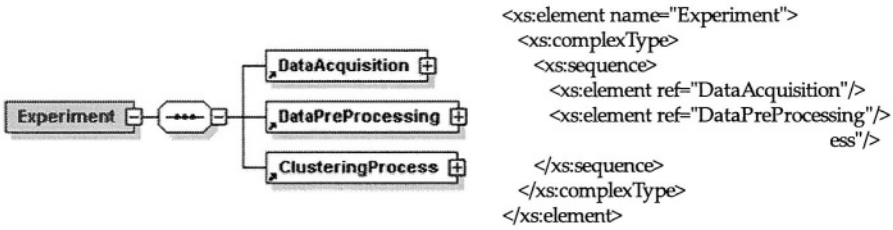


Fig. 4. Root element of the experiment modelled through XML Schema

become that standard. We have used a schema called PEDRo as a basis for the sample preparation process, but we have made some modifications to this model which concern data preprocessing and data clustering. The schema used is a common XML schema, as in Fig. 4, where the three main process are: (i) Data Acquisition, (ii) Data Pre-Processing, (iii) Data Mining.

Data Acquisition is the process that describes the preliminary phases of the proteomic data analysis. It is possible to insert complementary information besides the sample description, the sample preparation process and the instrumental characteristics as in Fig. 5.

Data Pre-Processing is the process that consists in spectrum noise and contaminants (described above) *cleaning up.* The steps of this phase are showed in Fig. 6. They are performed, automatically, by software enclosed in our MALDI-TOF instrument except for the *binning* phase. This is a method for data reduction. We have implemented it to compare analysis conducted on overall spectra with respect to partial spectra, and to discover if the power of expression embedded in the spectra decreased after its application. The *binning* algorithm is simple. It consists of a linear and iterative function which calculates, for each mass interval, the sum of the peaks intensity detected in the same interval and substitutes all peaks present in this interval with only one peak with intensity equal to the sum of the intensities. Binning segment grows, linearly, according to a 400 ppms factor.

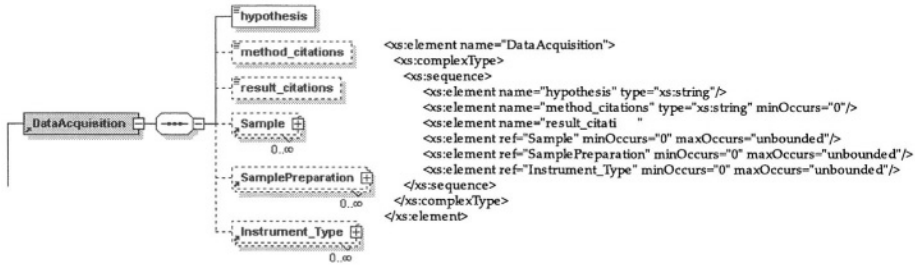


Fig. 5. Data Acquisition Steps

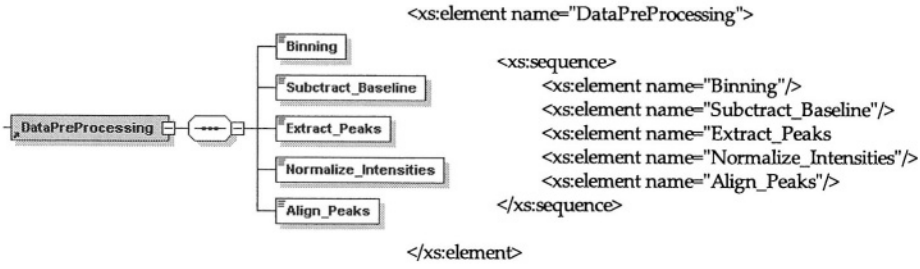


Fig. 6. Pre-Processing Steps

Data Mining. Data Clustering is is a way to perform a descriptive modeling of data that partitions input data, in this case input spectra, in a set of different clusters, where spectra in the same cluster are similar (w.r.t. a given similarity measure), whereas spectra in different clusters are different. In our experiment, Data Clustering is applied to MS data with the goal to find the following clusters: (i) Diseased Patients ($BRCA^+$), (ii) Diseased Patients ($BRCA^-$), (iii) Healthy Patients($BRCA^+$), also said *(Carriers),* (iv) Healthy Patients. BRCA1 is a gene that has a role in breast cancer, more information can be found in [7]. With classification algorithm, a new unknown sample can be classified after the method has been tuned using a training set of samples. We currently implemented only a classification procedure. In our work, the training set for platform tuning will exclusively be composed by public available SELDI-TOF mass spectrometry data[1], made available by a research group at the U.S. National Cancer Institute (NCI). After, we will apply the bioinformatics platform on MS data produced in our University. As in Fig. 7, it is possible to choose performing analysis of MS data with different methods and algorithms. Currently we have used and tested

[1] NCI Web Site. [http://ncifdaproteomics.com/ppatterns.php]

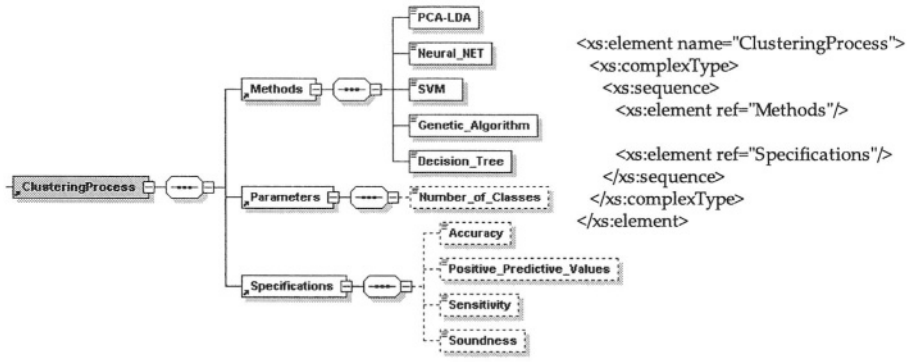


Fig. 7. Data Mining steps

the Q5 [12] algorithm, an implementation of PCA-LDA [17] twin processes. Just now, there is only one tunable parameter and it is possible to set the overall number of classes for the classification. Besides, other technical specifications, inherent the optimization and the quality of the calculated results, are tunable. Moreover we are implementing more tools based on neural networks.

The back-end user interface of the overall system for MS data modeling (PEDRo) allows to export the instance of the basic model inserted by the user in a XML file format. The modelling and the analysis phases are strictly joined. In fact our platform PROTEUS takes its configuration from the XML file built by the previous modelling process conducted through PEDRo. In the next Section, we will show the architecture and the features of Proteus.

3 Proteus: Architecture and Features

PROTEUS is a Grid-based Problem Solving Environment [16] for bioinformatics applications. To fulfill bioinformatics application requirements and to help biologists application requirements [4], we propose a framework based on:

- Grids, with their security, distribution, service orientation, and computational power;
- Problem Solving Environment approach, useful to define, describe and execute (i.e. control) such applications;
- Ontologies, Web (Grid) Services, and Workflows technologies, at an inner level, to describe, respectively, the semantics of data sources, software components with their interfaces, and performances and bioinformatics tasks.

With the first item PROTEUS satisfies the high powerful computational requirements of bioinformatics applications. Moreover Grid environment is composed of distributed computational nodes, and fulfill the distributed nature of bioinformatics applications and data management. Fig. 8 shows main components of PROTEUS (see [3] for further details).

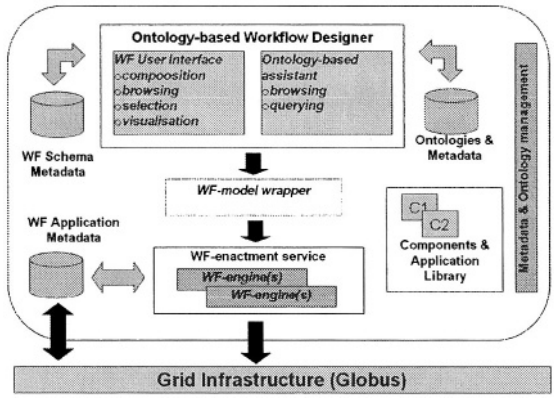


Fig. 8. Proteus Architecture

PROTEUS can be used to assist users in:

- *formulating problems,* allowing to compare different available applications (and choosing among them) to solve a given problem, or to define a new application as composition of available software components;
- *running an application on the Grid,* using the resources available at run time thus leveraging the Grid scheduling and load balancing services;
- *viewing and analyzing results ,* by using high level graphic libraries, steering interfaces (that allow to interactively change the way a computation is conducted), and accessing the past history of executions, i.e. the past results, that form a knowledge base.

4 Using Domain Ontologies in PROTEUS

This Section describes the role of domain ontologies in Proteus. Currently we have developed two domain ontologies: DAMON (Data Mining Ontology) represents the set of tools and methodologies to conduct data mining analysis on data [2], whereas PROTON (Proteomic Ontology) describes specific features and details of the domain under investigation: i.e. proteomics and mass spectrometry experiments. In PROTON we have both a classification of biological and non biological concepts as shown in Fig. 9.

Analysis. With this concept we model theory of proteins. Introducing this concept is important in our context. In medical research the literature citations of a method are a criterium of evaluation. This class can be specialized in the following subclasses: (i) Mass-Fingerprinting Analysis, (ii) Primary Structure Analysis, (iii) Secondary Structure Analysis, (iv) Tertiary Structure Analysis, (v) Quaternary Structure Analysis.

Task. A task is a concrete problem that the researcher has to solve. Specialisation of this concept in sub-classes belongs the principals area of proteomic study.

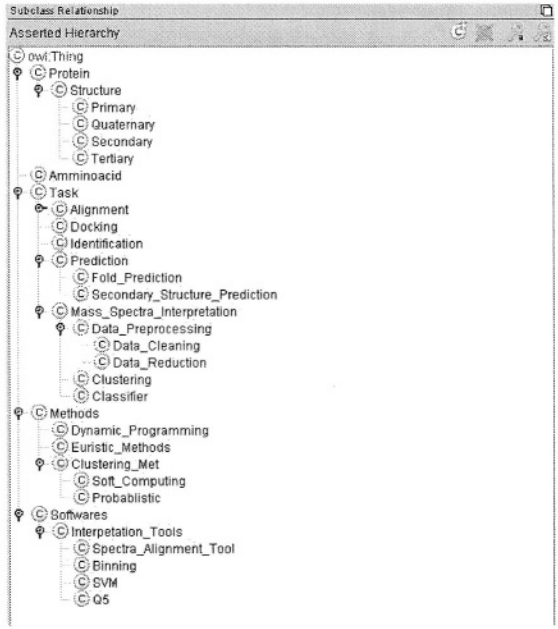


Fig. 9. Fragment of PROTON ontology (Protege screenshot)

Interpretation of MS data. Interpretation of MS data has several different aspects that will be underlined in sub-classes. This data can be used to identify a protein, or to recognize a disease.

Alignment. Alignment [9] is a classical task in proteomic inherited from genomic. Particularly we can recognize Sequence Alignment when protein primary structures are compared, and Structural Alignment if the spatial conformations are compared.

Prediction. Prediction [13] is equally a classical problem in proteomic. A researcher is often interested in prediction of secondary or tertiary structure of a protein starting by primary sequence.

Method. A method is a way to perform a task. Method Taxonomy is similar to Task Taxonomy.

Software. A software is an implementation of a method in a web-server, standalone application or grid-node.

Currently ontology integration is made through OnBrowser, a tool for managing ontologies on the Grid [6]. In order to perform a complete 'in silico' experiment, it is necessary to cooperate between the biologic and bioinformatics group. It will be possible if a unified semantic is accepted. Ontology can realize this union bringing experimental research and Bioinformatics. Suppose that a researcher need to classify data in order to recognize healthy-diseased condition over serum fingerprinting data. He/she can browse PROTON and find in the Analysis Section the literature related in order to recognize already used meth-

ods. Then he/she can explore Methods taxonomy finding methods like e.g. Soft Computing techniques, Probabilistic techniques, Preprocessing, etc., and finally he/she will follow the links to software (e.g. Q5 software). At this point, he/she can find in DAMON ontology details about the Q5 software [12] and all related information. This integration permits the collaboration between two different groups: biologists and bioinformaticians. In effect once the appropriate software has been found by a biologist from biological consideration, bioinformatic people can discover all that is required for the experiment in DAMON ontology.

4.1 Ontology-Based Design of Proteomics Experiments

The design and execution of an application on PROTEUS comprises the following steps:

Ontology-based component selection. Browsing PROTON a user can at first find the articles related to its application in order to select method that is already tested. Once this phase is ended, user select the resources (i.e. methods and servers in which they are available). In our experiment a user can expand and visit individuals of Data Interpretation Tools and find Data-Preprocessing, Data-Clustering and Data-Classification. When a methods for Classification (e.g. Q5) is selected, the researcher can browse DAMON ontology and find all information and resources needed to execute such software. Fig. 10 shows a fragment of DAMON ontology describing Q5.

Workflow design. Selected components are combined producing a workflow schema that can be translated into a standard executable workflow language.

Application execution on the Grid. The workflow is scheduled by one or more workflow engines on the Grid.

Results visualization and storing. After application execution and result collection, the user can enrich and extend the PROTEUS knowledge base adding results and experience in PROTEUS ontologies.

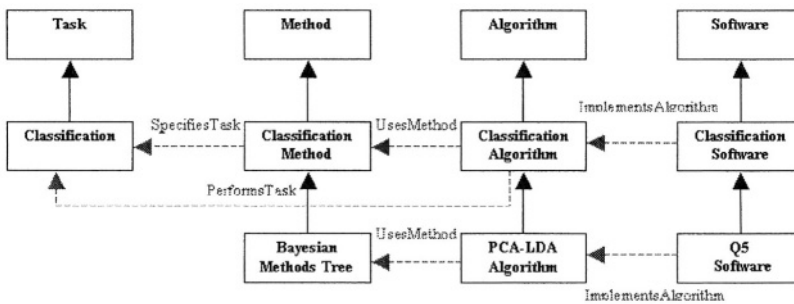


Fig. 10. Modelling of Q5 in DAMON

5 Conclusions and Future Work

Bioinformatics applications involve many scientific platforms, producing heterogeneous data that need to be processed by different kind of tools. To face complexity of such applications powerful modelling techniques have to be used. Ontologies and workflows play a significant role enhancing application formulation and definition in PROTEUS, a Grid-based Problem Solving Environment. We described a methodology for the design of bioinformatics applications where basic components are selected through domain ontologies and composed through workflows.

As case study, Data Mining of Mass Spectrometry data has been considered. Currently we are improving our MS experiment model and we are creating a communication protocol between PEDRo (used to model the Mass Spectrometry side of the experiment) and PROTEUS (used to conduct Data Mining). We are designing a method in order to transform a XML document in JAVA classes through the JAXB API (Java Architecture XML Binding) to improve the configuration of PROTEUS. A second step will be the complete integration of other data mining tools inside PROTEUS. In fact, grid-based integration of such tools could help us to manage, in a very effective way, the very large amount of input data and the complexity of data mining computations, that require a lot of computational resources.

References

1. P.G. Baker, A. Brass, S. Bechhofer, C. Goble, N. Paton, and R. Stevens, *TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources. An Overview,* Proceedings of the Sixth International Conference on Intelligent Systems for Molecular Biology (ISMB'98) (Menlow Park, California), AAAI Press, June 28-July 1 1998, pp. 25–34.
2. M. Cannataro and C. Comito, *A DataMining Ontology for Grid Programming,* Workshop on Semantics in Peer-to-Peer and Grid Computing (in conj. with WWW2003) (Budapest-Hungary), 2003.
3. M. Cannataro, C. Comito, A. Guzzo, and P. Veltri, *Integrating Ontology and Workflow in PROTEUS, a Grid-Based Problem Solving Environment for Bioinformatics,* Proc. ITCC2004 (Las Vegas, Nevada), 2004.
4. M. Cannataro, C. Comito, F. Lo Schiavo, and P. Veltri, *Proteus, a Grid based Problem Solving Environment for Bioinformatics: Architecture and Experiments,* IEEE Computational Intelligence Bulletin **3** (2004), no. 1, 7–18.
5. M. Cannataro, G. Cuda, M. Gaspari, P. H. Guzzi, T. Mazza, P. Tagliaferri, G. Tradigo, P. Veltri, F. Costanzo, and S. Venuta, *Mass Spectrometry Data Analysis for Early Detection of Inherited Breast Cancer,* WIRN 2004, CIBB Workshop, submitted paper.
6. M. Cannataro, A. Massara, and P. Veltri, *The OnBrowser Ontology Manager: Managing Ontologies on the Grid,* Int. Workshop on Semantic Intelligent Middleware for the Web and the Grid, in conjuntion with ECAI-2004, Springer Verlag, August 22-26 2004.

7. G. Cuda, M.Cannataro, B. Quaresima, F. Baudi, R. Casadonte, M.C. Faniello, P. Tagliaferri, P. Veltri, F.Costanzo, and S. Venuta, *Proteomic Profiling of Inherited Breast Cancer: Identification of Molecular Targets for Early Detection, Prognosis and Treatment, and Related Bioinformatics Tools,* WIRN 2003, LNCS (Vietri sul Mare), Neural Nets, vol. 2859, Springer Verlag, 2003.
8. I. Foster and C. Kesselman, *The Grid: Blueprint for a Future Computing Infrastructure,* Morgan Kaufmann Publishers, 1999.
9. George Fuellen, *A gentle guide to multiple alignment,* (1996).
10. W. E. Johnston, *Computational and Data Grids in Large-Scale Science and Engineering,* Future Generation Computer Systems **18** (2002).
11. P. Kearney and P. Thibault, *Bioinformatics meets proteomics bridging the gap between mass spectrometry data analysis and cell biology,* Journal of Bioinformatics and Computational Biology **1** (2003), no. 1, 183–200.
12. Ryan H. Lilien, Hany Farid, and Bruce R. Donald, *Probabilistic disease classification of expression-dependent proteomic data from mass spectrometry of human serum,* Journal of computational biology **10** (2003), no. 6, 925–946.
13. Kabsch Sender, *Dssp:dictionary of secondary structure prediction,* 1970.
14. S Schweigert P Herde R Sibbald, *Issues in incorporation semantic integrity in molecular biological object-oriented databases,* CABIOS **11** (1995), no. 4, 339–347.
15. E.W. Selway L. Walker J. Riba-Garcia I. Mohammed S. Deery M. Howard J. A. Dunkley T. Aebersold R. Kell D.B. Lilley K.S. Roepstorff P. Yates J.R. Brass A. Brown A.J. Cach P. Gaskell S.J. Hubbard S.J. Oliver Taylor C.F. Paton N.W. Garwood K.L. Kirby P.D. Stead D.A. Yin Z. Deutsch, *A systematic approach to modeling, capturing, and disseminating proteomics experimental data,* WWW Document, 2003, http://pedro.man.ac.uk/files/PEDRo_PowerPoint_talk.zip.
16. D. Walker, O. F. Rana, M. Li, M. S. Shields, and Y. Huang, *The Software Architecture of a Distributed Problem-Solving Environment,* Concurrency: Practice and Experience **12** (2000), no. 15.
17. Ka Yee Yeung and Walter L. Russo, *An empirical study on principal component analysis for clustering gene expression data,* Uw-cse-2000-11-03, University of Washington, November 2000.

Workshop on Java Technologies for Real-Time and Embedded Systems (JTRES) PC Co-chairs' Message

Over 90 percent of all microprocessors are now used for real-time and embedded applications, and the behavior of many of these is constrained by the physical world. Higher-level programming languages and middleware are needed to robustly and productively design, implement and enforce real-time constraints along with functional requirements.

Designing real-time and embedded systems that implement their required capabilities, are dependable and predictable, and are parsimonious in their use of limited computing resources is hard; building them on time and within budget is even harder. It is therefore essential that the production of real-time embedded systems can take advantage of languages, tools, and methods that enable higher software productivity. For these reasons, ideally, developers should use programming languages that shield them from many accidental complexities, such as type errors, memory management, and steep learning curves. The Java programming language is an attractive choice because of its safety, productivity, its relatively low maintenance costs, and the availability of well trained developers. However, Java is unsuitable for developing real-time embedded systems, mainly due to under-specification of thread scheduling and the presence of garbage collection. To address these problems, a number of extension to Java have been proposed, the two most representative being the Experts Group Real-Time Specification (RTSJ) for Java and the J-Consortium Real-Time Core Extension (RTCore). In such a context, the goal of the Second Workshop on Java Technologies for Real-Time and Embedded Systems is to gather researchers working on real-time and embedded Java to identify the challenging problem that still need to be properly solved, in order to assure the success of the of Real-Time Java as a technology, and to report results and experiences.

JTRES 2004 Papers

This edition of the JTRES workshop attracted many researchers and good quality papers have been submitted. Various topics have been covered by the contributions. Many papers focus on the memory model for real-time Java specified by RTSJ, proposing modifications, extensions for distributed environments, support tools, abstractions and design patterns able to facilitate programming with RTSJ scoped memories. Other papers deal with real-time extensions able to include timing aspects in non-real-time applications, and with RTSJ modifications required to support applications and systems belonging to particular profiles (high integrity, limited memory, etc.). Experience papers focus on embedded Java for mobile applications and on the use of RTSJ in standard distributed

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 244–245, 2004.
© Springer-Verlag Berlin Heidelberg 2004

process control environment. The embedded Java topic is covered by a paper proposing a new Java-based operating system, and another paper dealing with a time-predictable cache for a Java processor. Finally, a paper reports authors' experience in teaching real-time Java.

August 2004

Angelo Corsaro, Washington University
Corrado Santoro, University of Catania
(JTRES 2004 Program Committee Co-chairs)

Real–Time Teaching with Java: JPR^{3}*

Diego Alonso, Juan A. Pastor, and Bárbara Álvarez

Universidad Politécnica de Cartagena (Spain)
diego.alonso@upct.es

Abstract. This paper describes a development platform built around a digital railroad scale–model: JPR^{3} (Java Platform for Realtime Railway Research). The laboratory equipment and software aims to achieve two goals: help and motivate students of real–time systems and as support for postgraduate students. Students find the scale–model really challenging and are very motivated by it; thus it's easy for them to really learn and practice all the concepts of real–time systems. But it's not only for students use: it also serves as a research platform for postgraduate students, thanks to the possibilities offered by the scale–model. Java has been chosen as the programming language codify the platform and the implementation of the system is described in this work.

1 Introduction

Although many undergraduate courses in computer engineering acquaint students with the fundamental topics in real–time computing, many do not provide adequate laboratory platforms to exercise the software skills necessary to build physical real–time systems. The undergraduate real–time systems course at Technical University of Cartagena is practical and carries out in a laboratory with a Digital Model Railroad Platform, where students can apply the real–time concepts explained at class [1] and see them work in a real enviorment.

Thanks to the Real–Time Extension [2], Java now offers a wonderful API for real–time systems' teaching, because there is a clear relation between real–time concepts and Java objects. Plus the fact that students can see it work in a real-time operating system, the result is a complete "real–time experience" for them. Also, Java has been chosen because students are already familiar to it, since Java is studied during the first courses. This way, they can focus on applying real–time techniques rather than in learning a new programming language.

But in the process of real–time learning, the railway scale–model is also of great help, not only because its a real, physical system, but also because it motivates the students, who find it very challenging. The total length of the track plus the complexity added by the possibility of using the turnouts, results

* This work was supported in part by the Spanish Ministry of Education (with reference ACF2000–0037–IN) and the Regional Government of Murcia (Séneca Programmes with reference PB/8/FS/02)

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 246–255, 2004.
© Springer-Verlag Berlin Heidelberg 2004

in a complex circuit to control in which problems with several levels of difficulty (and risk) can be simulated.

This paper is organised in five more sections. Section 2 gives a complete description of the laboratory equipment, both hardware (Sect. 2.1) and software (Sect. 2.2). In Sect. 3 the real–time problems related to the mock–up are outlined, and the Java implementation of the server–side of JPR^3 is described in Sect. 4. An example of a real practice is presented in Sect. 5. Finally, Sect. 6 summarised the content of the paper and outlines future plans for JPR^3.

2 The Platform at the Laboratory

This section presents a brief description of the equipment present at the laboratory, focusing on the railroad scale–model and the software control platform. The railroad scale–model has been developed around a commercial system designed and built by Märklin[3]. Specifically, it is based on the Märklin Digital System. Figure 1 shows a panoramic view of the railroad platform once finished.

On the other side, the control architecture is run on a normal Intel Pentium computer, placed next to the scale–model and connected to it by an RS–232 serial wire.

2.1 The Digital Model Railroad

Märklin commercialises beautiful all–time locomotives and all kind of accessories to simulate a real railway network. The railroad scale–model placed at the laboratory is formed by the following elements and accessories:

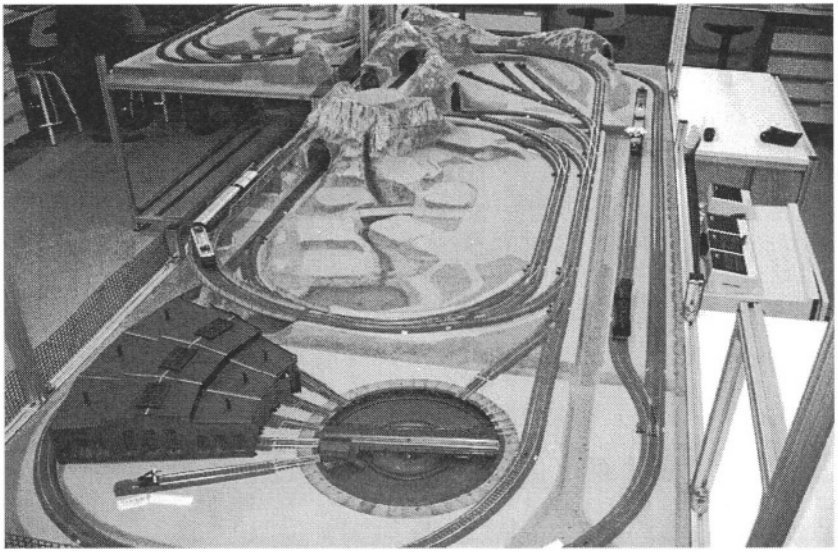


Fig. 1. Overview photograph of the scale–model

- ⋆ Five digital locomotives, capable of moving in both directions and with special functions, such as play the bell, turn–on the lights, or even throw smoke.
- ⋆ Sixteen digital turnout switches (where three tracks join) with manual and automatic control.
- ⋆ Six digital semaphores, which are only passive elements, i.e., the locomotive doesn't stop by itself if the semaphore is in red.
- ⋆ Twenty one double reed contact sensors to manage and control the position of the different locomotives in the mock–up.
- ⋆ Around a hundred railway tracks, both straight and curved, that make up our particular railway network.

The Märklin Digital System uses the tracks as power and control lines for all elements present in the scale–model, so the number of wires is minimum and new elements can be easily added. Moreover, it uses the centre of the tracks as the main conductor line, so the polarity of the signal is independent of the direction of the movement of the locomotives. But this kind of communication, based on friction, has a great drawback: it's very noisy. So the transmission speed is set to a very low value and every command is sent several times to ensure its correctly received.

All the active elements of the scale–model (turnouts, semaphores and locomotives) have a unique identification number and carry a device to decode the control commands that travel by the tracks. Because of the noise in the system, each decoder needs to decode, at least, three times the same command for it to proceed with it. This non-desired feature greatly increases the latency of the system.

The reed contacts are placed before and after every turnout around the mock–up, in order to monitor the traffic on the railroad. Each reed contact is really composed by two switches, which are activated depending on the direction of the locomotive that is stepping through it. To get the state of the reed contacts, three Märklin S88 Decoders are used. Each one provides a reading of the status of up to eight reed contacts, resulting in a total of sixteen sensors.

The scale–model can be manually operated by means of the Märklin 6021 Control Unit, the core of the Märklin Digital System. This module is in charge of both converting the control orders to electric signals, that are transmitted through the rails, and of reading the state of the Märklin S88 Decoders. Finally, to be able to control the scale–model with a computer, a module that provides a RS–232 serial interface with the Control Unit is used (see Fig. 2).

2.2 The Software Platform: JPR^3

The software design of the JPR^3 was started following a four view design approach [4] [5]. The initial development of the platform was guided by three objectives:

1. The application has to run in a host system that doesn't make use of the Real–Time Java extension. The user has to be able to configure whether the Real–Time API has to be used or not.

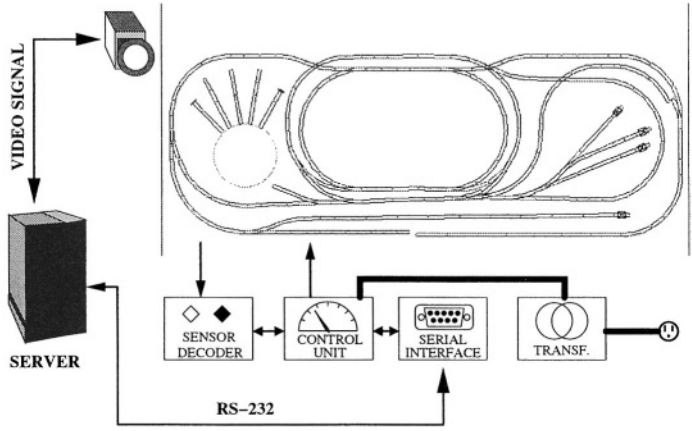


Fig. 2. Diagram of the configuration of the scale–model

2. The architecture has to be modular and easily extendable, so new features could be added (such as the use of some video camera or a simulator of the mock–up).
3. The architecture *should* be distributed, so different clients (such as an automatic control module or a web client) could make use and monitor the mock–up.

With all these objectives in mind, the application was developed following the schema shown in Fig. 3. This paper presents only the, what is called, *server–side* of the application, which is, after all, the only that really has real–time constraints.

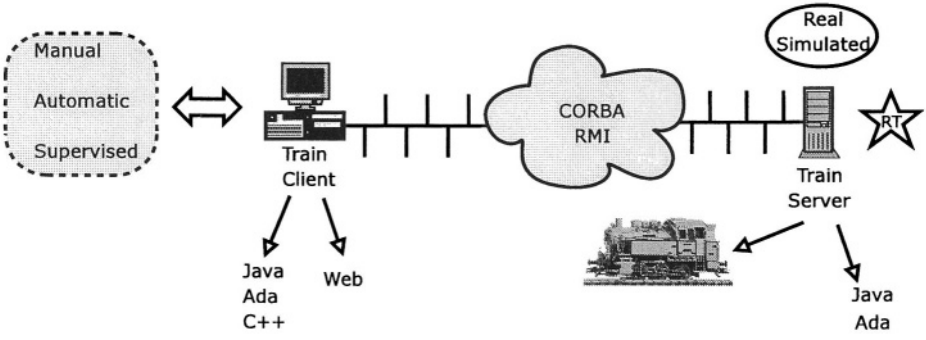


Fig. 3. Architecture overview

To control the elements of the mock–up, the Märklin Serial Interface provides several commands to send to the Control Unit. Section 3 presents all the issues related to the implementation of the communication with the scale–model, which, as we will see, is not trivial. The available implemented control commands are:

- ◇ Stop the mock–up
- ◇ Start the mock–up
- ◇ Change turnout track
- ◇ Read reed contacts
- ◇ Manage semaphores
- ◇ Use locomotives functions
- ◇ Change locomotive speed
- ◇ Change locomotive direction

3 Real–Time Characterisation of the Scale–Model

The greatest time constraint imposed by the scale–model is due to the communication system. As said before, the Märklin Digital System provides a great advantage (there are practically no wires) but at a great cost (the communication is noisy, commands are sent several times and it works at a very low speed). Moreover, a small *unknown* delay has to be introduced between two consecutive commands, because, otherwise, the last command could be *lost in its way* and thus completely ignored by the Control Unit.

Although the available set of commands is reduced, as seen in last section, it is obvious that not all commands have the same priority. Commands such as *stop* and *start* have a greater priority over the rest. Indeed, the emergency-stop or stop-all order has to executed at fast as possible, to avoid collision between locomotives. Also the scale–model ignores all commands until the start one has been received, making the sending of other commands useless.

The other important time constraint is imposed by a simple fact: *locomotives do not have to crush!*. This desirable objective means in practice that there has to be a free track between two locomotives. In this case, the word *track* groups all tracks between two reed contacts. As said in Sect. 2, there's a reed contact sensor before and after every turnout element of the mock–up, and they are the only available source of information to know where a locomotive may be. We say *may be* to mean that we only know that some locomotive has stepped through one reed contact in a given direction, thus it is only known that the locomotive is somewhere over the track.

Having said that, the safety condition for the system is the following: the frequency of the reading of the state of the reed contacts has to ensure that no locomotive could have activated two different reed contacts between two consecutive readings. So, given the maximum locomotive speed and the shortest track (group of tracks between two consecutive reed contacts) the minimum period between reads is around 500 msec.

4 Java Implementation Issues

Once the architecture of the application was finally defined and the server–side finished, it was time to test it. At that time, we chose the solution proposed

by Timesys[6]: a modification of the Linux Kernel and an implementation of the Virtual Machine over it. So, by now we are using version LEK-X86BSP-4.1-155.3 of Timesys' Linux Kernel and version RTJ-X86BSP-1.0-547.1 of the Java Virtual Machine. Everything has been installed, as said before, in the Intel Pentium computer present at the laboratory.

As said in Sect. 2.2, the most important objective seeked by the development of the application was to make it as highly configurable as possible. The goal was to be able to configure the kind of threads it would use (normal Java threads or RealtimeThreads [7]). This choice, as well as many more, are made during the application start–up by means of configuration files. Figure 4 shows an UML diagram of the execution layer. Among these configurable characteristics, two will be named:

- The use of the Java Real–Time extension.
- The time constraints of the different threads, such as the deadline, period, priority, … Some values are obviously ignored when non real–time execution is selected.

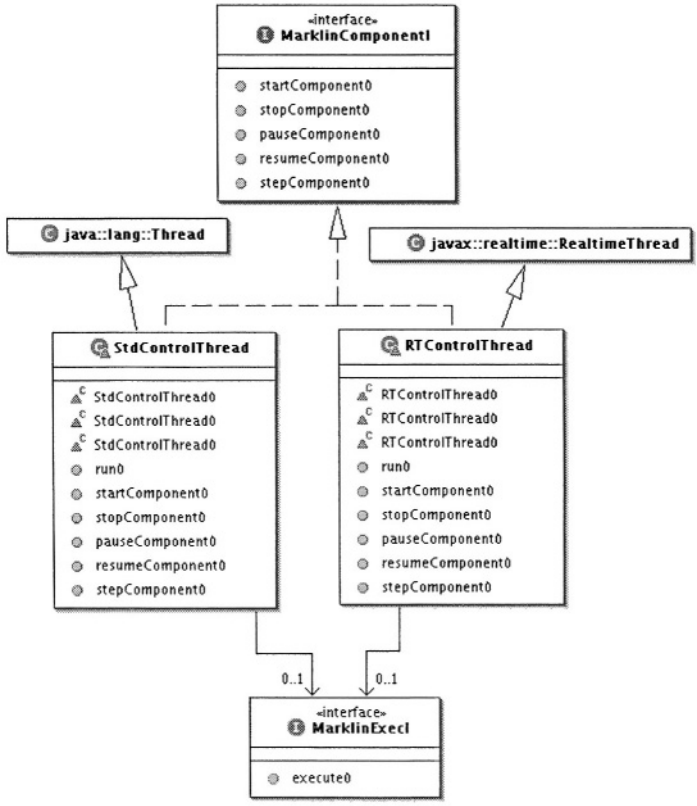


Fig. 4. Execution layer class diagram

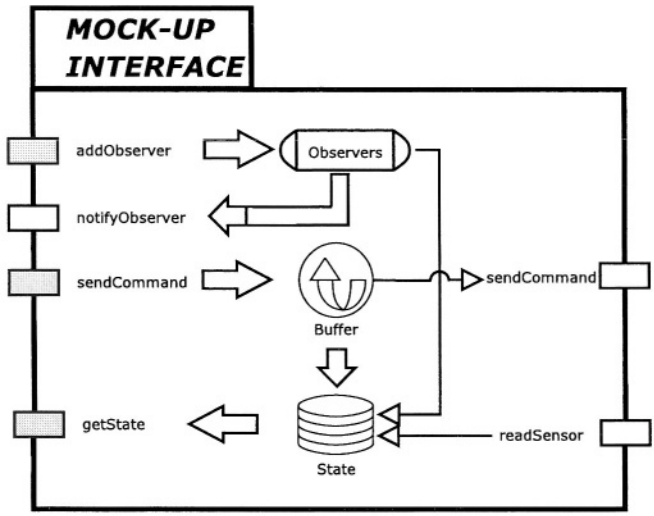


Fig. 5. Inner configuration of the mock–up interface

When the initial creation and configuration step is over, every component shown in Fig. 3 runs in it's own thread (whether real or non real–time one). Each one of these threads supervises and controls every object and thread created in the component it's in charge of. This way, the platform can be easily and safely stopped.

Figure 5 shows the configuration of the component we have named "Mock–Up Interface". This component is the core of the server–side of the application, because is in charge of managing the communication between the mock–up and the different possible clients.

As can be seen in the figure, the constraint imposed by the latency of the communication is solved by the use of two circular buffer with different priorities. This was necessary to ensure that high–priority commands are executed nearly when they are send to the component. The application consider only two such commands: *emergency stop* and *release of the system,* whose importance can be derived by their names. Besides the circular buffer, the server also follows an observer pattern [8] to notify every client of the state of mock–up.

When the initital configuration and creation of objects is finished, three threads are kept running in the server component:

- One thread, with the highest priority of all, to send commands to the mock–up and read the sensors.
- Another thread to notify the observers of the state of the mock–up that it has changed.
- One thread to execute different control strategies, developed by the students, when in automatic control mode.

5 A Practical Exercise

The laboratory has been used in courses such as Concurrent Programming and Real-Time Systems. These courses are placed in the last year of the Industrial Electronics and Control Engineer graduate curriculum, and they cover a good percent of ACM/IEEE-CS recommendations in computing technologies [9]. In particular, the course Real-Time Systems has seven lectures which concentrated on the following topics:

1. Characteristics of real–time systems and introduction to the Real–Time Specification for Java.
2. Concurrent programming.
3. Scheduling schemes (cyclic executive and priority–based models).
4. Reliability, fault tolerance and low–level programming.

The practical exercises must provide opportunities for these theoretical concepts testing. With the basic infrastructure described in the above sections, a highly comprehensive set of programming practices were developed. One of them is briefly described in this section, simply to give an overall idea of the laboratory's possibilities.

The railroad platform simulator focuses on modularity. Students should be able to design an application module using simulator module interfaces. An example of practical exercise is carried out by the students starting from the following requirements specification. Once the students know the railroad platform, we fix the turnout switches in order to have a lineal problem, as can be seen in Fig. 6.

It is necessary to avoid that a train collides with other train. Evidently, because of the turnout switches are fixed, we have a one–way railroad and it is not possible that a train meets to other train in a turnout switch or two

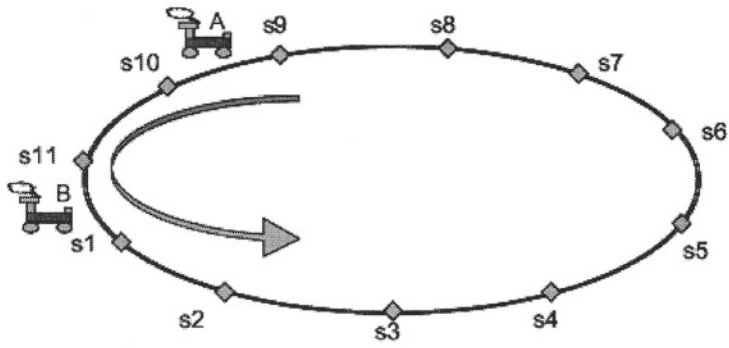


Fig. 6. A practical scheme

trains collide. In order to avoid accidents, it has always to have a "free stretch"[1] between two trains. In accordance with this idea, if a train `(train A)` has gone over a reed contact sensor which represents the beginning of a track and other train `(train B)` is on the following track then we have to stop the first train `(train A)` until the second train `(train B)` leaves the occupied track. In this moment, the stopped train must recovery its initial speed.

As Fig. 6 describes, the `train A` is over a stretch between s9–s10 reed contacts and can go on towards the stretch delimited by s10–s11 reed contacts. However, this train must be stopped when standing on s11 reed contact if the `train B` has not stood on s1 reed contact.

To avoid the constant execution of stopping and starting procedures, it is recommended to increase the speed of the forward train (in our example `train A`) around 20% until having a "free stretch" between the trains again. From this moment, the forward train travels with its initial speed[2].

The students must implement a program for monitoring the state of the different sensors and modifying the speed of the trains in order to avoid collisions between them as the above specification describes. Each train must be implemented as RealtimeThreads. In this way, the control is distributed and other manager task will centralize the occupation of the tracks. Some real–time characteristics of the railroad platform are: critical safety operations (to stop a train, to stop the whole system, to change the state of a turnout switch, ...) need to have a very high priority; the time needed to stop a train must be bounded; the detection of a possible collision between trains must be in certain limits, and so on. In addition, the times of standing on each reed contact sensor must be registered.

Finally, the student must take into account possible fails and manage them using the mechanism of exceptions. In this way, the exercise allows to practice the different topics reviewed during the course as concurrent programming, scheduling schemes, fault tolerance, etc ...

6 Conclusion and Future Work

JPR^3 aims to help students exercise their knowledge of concurrent programming, real–time systems, control application, etc ... as well as research tool for postgraduate students. The platform was designed to be easy to use and configure for students, but also extensible so new features could be added in a painless way.

Obviously, the research and development of JPR^3 does not end here. We've already started up a number of activities to extend its functionalities.emphasiseese, we wish to emphasize the following ones: design of a simulator of the mock–up (so students could test their control strategies without using the real mock–up), use of video cameras to carry out visual supervision and control of

[1] We define a "free stretch" as the space of tracks between two reed contacts

[2] We suppose in this example that the students know the simulator and its interface to reading all sensorial information accurately

the locomotives (cameras can also be placed *inside* the locomotives) and, finally, make it a real distributed application by using a communication middleware, such as CORBA and/or RMI.

Although we have chosen a particular implementation of the Virtual Machine, we also plan to test JPR^3 with other implementations of the JVM and real–time operating systems.

References

1. Burns, A. Wellings, A.: *"Real–Time Systems and Programming Languages"*. Addison–Wesley (2001)
2. Real–Time Specification for Java v1.0 *http://www.rtj.org/rtsj-V1.0.pdf*
3. Märklin Trains homepage *http://www.marklin.com*
4. Hofmeister, C. et als.: *"Applied Software Architecture"*. Addison–Wesley (2000)
5. Powel, B.: *"Real–Time Design Patterns"*. Addison–Wesley (2003)
6. TimeSys Corporation homepage *http://www.timesys.com*
7. Dibble, P.: *"Real–Time Java Platform Programming"*. Prentice Hall (2002)
8. Gamma, E. et als.: *"Design Patterns: Elements of Reusable Object Oriented Software"*. Addison–Wesley (1995)
9. CM and IEEE-CS. Year 2001 model curricula for computing (Computing curricula 2001). The joint IEEE computer society/ACM task force. [Online]. Available: http://www.acm.org/education/curricula.html.

Toward a Component-Based Embedded Java-Oriented Operating System*

Frédéric Dang Tran, Jean-Philippe Fassino, Olivier Lobry, Jacques Pulou, and Nicolas Rivierre

France Telecom R&D MAPS/AMS
38-40 rue du Général Leclerc
92794 Issy Moulineaux Cedex 9 FRANCE
{frederic.dangtran, olivier.lobry, jacques.pulou, nicolas.rivierre, jeanphilippe.fassino}@francetelecom.com

Abstract. This papers presents a Java-oriented operating system framework targeted at resource-constrained embedded environments. This framework relies on a component-based architecture that allows a specific OS profile to be constructed by assembling fine-grained system components. Such a modular approach yields several advantages: the ability to customize the operating system according to the specific requirements of the execution environment or target applications, a strong integration of the Java virtual machinery with the OS layer with no abstraction mismatch between the two and the ability to monitor the system in a non-intrusive way. Our framework resorts to a component-aware ahead-of-time compilation of Java bytecode that performs global optimization techniques in order to keep the overhead of the component framework minimal.

1 Introduction

Embedded technology is evolving at a fast pace with more powerful CPUs, reduced energy consumption and new radio communication standards promising ever increasing bandwidth over the air (GPRS, UMTS). The cycle of innovation and obsolescence is getting shorter with a constant pressure on service providers to deploy new services and applications over a vast range of devices while reducing time to market and development costs. In this regard, the use of a high-level object-oriented programming language such as Java in embedded environments, as an alternative to traditional native languages such as C or C++, is attractive and could provide a partial solution to this challenge. Java is gaining momentum for consumer devices such as mobile phones or PDAs thanks to the standardized embedded versions of the Java Platform (MIDP). However such platforms relegate Java to the application level while system and time-critical tasks are handled by native code and an underlying RTOS. Initiatives to extend Java

* This work is supported in part by the European IST E2R (End To End Reconfigurability) project as part of the 6th PCRD.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 256–267, 2004.
© Springer-Verlag Berlin Heidelberg 2004

with real-time capabilities (RTSJ [15]) assume more or less implicitly as well that the Java machinery is supported by a host operating system (e.g. typically a POSIX compliant OS) and have yet to prove their feasibility on small devices. Relying on a host operating system raises several issues: a possible mismatch between the system abstractions visible from Java and the ones of the underlying OS, a large memory footprint and poor performance due to the number of protection layers to cross for accessing system resources.

One alternative is to avoid an operating system altogether and to implement the embedded Java VM on the bare hardware. In other words, a custom-built OS kernel is designed with exactly the set of kernel features required to implement the Java VM. Note that this set of required kernel features (that together define an OS "profile") can vary depending on the target application domain and the targeted Java platform. For example an application-specific Java operating system with no dynamic object creation might run without a garbage collector. Another profile might be content with a cooperative thread scheduler rather than a preemptive one. A key challenge for an embedded software infrastructure is to provide a framework with a high degree of configurability to support a vast range of embedded systems each with different required functionality. On top of that, a given embedded system, once deployed, may benefit from dynamic reconfigurability over its life time in order to replace or add new functionality.

Our proposed system aims to address these issues by resorting to a component-based architecture in which the embedded software infrastructure (OS proper + Java runtime) is constructed out of a set of reusable components which are the basic building blocks of the system. These components can be developed either in C or Java (using ahead-of-time compilation) with well-defined communication mechanisms between native components and Java components. We currently target the J2ME/CLDC profile with the restriction that dynamic loading of Java bytecode is not allowed.

This paper is organized as follows. Section 2 provides an overview of the component model underpinning our operating system framework. The development tools and in particular the ahead-of-time Java bytecode to C code compiler is described in the next section. Preliminary performance results are provided in section 4. We cover related work in section 5 before concluding.

2 System Architecture

2.1 Overview

Our system component model is based on the Think component model [1] which is itself a specialization of the Fractal component model [4]. The main structural features of this model are as follows (see figure 1 (a)):

- System code is decomposed into components. A component is the unit of (re)configuration. A component interacts with its environment (other components) through typed interfaces. For a given component, we distinguish the interfaces *provided* by the components (that correspond to a service offered

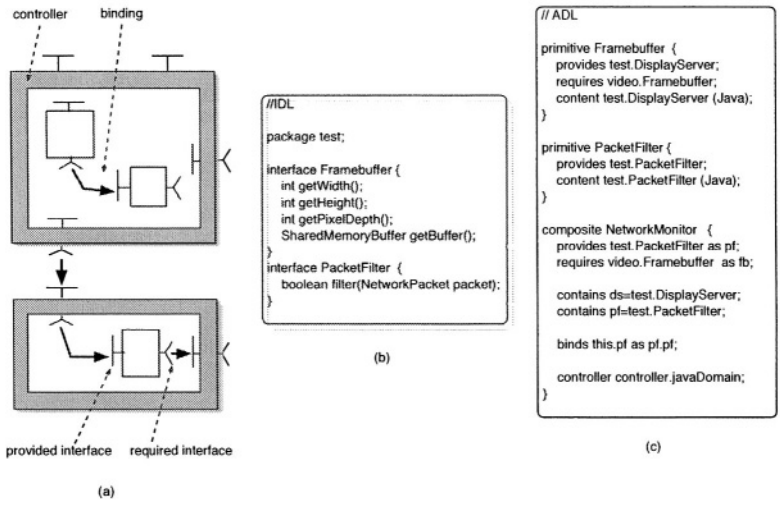


Fig. 1. Component model, (a) main abstractions, (b) IDL example, (c) ADL example

by the component such as a memory manager or thread scheduler) from the interfaces that are *required* by the components (and that corresponds to services needed by the component). Components are recursive: a component can act as a container of sub-components. The recursion ends with *primitive* components which encapsulate base components implemented in a programming language (Java or C) and beyond the control of the component model. Components encapsulate all operating system services (such as CPU scheduling, resource allocation and device drivers). The hardware platform itself is also reified as a set of components that provide a thin wrapper over the hardware interface (e.g. MMU, CPU cache etc.).

- A component interface is a set of functions that can be invoked by another component while guaranteeing a strict separation between the interface specification and the interface implementation.
- a composite component is made up of two parts: the contents of the component which corresponds to the functional part of the component and the *controller* part which acts as a "membrane" around the component. This membrane serves several purposes: (i) it hides the interface offered by the inner components unless explicitly exported by the composite component, (ii) it can embed interceptors that insert arbitrary functionality into the invocation path between the component content and the outside world and (iii) it provides introspection capabilities at run-time.
- Components are connected together (via their interfaces) through bindings. A binding is a communication channel between components. The simplest form of binding within the same protection domain is a direct pointer in C or an object reference in Java. Cross-domain bindings include system calls, (local)RPC or interceptors that perform an access control.

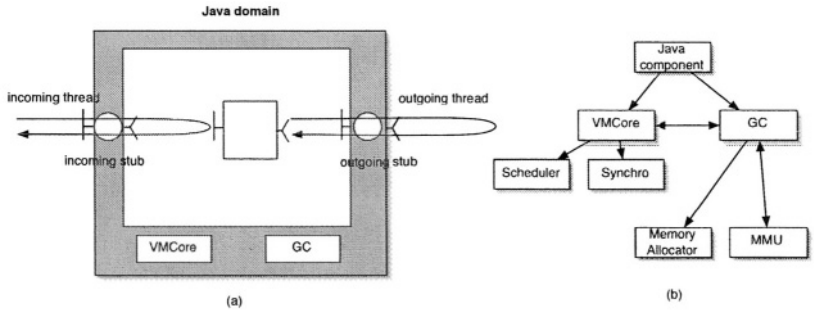


Fig. 2. Java domain: (a) cross domain invocation (b) VM components

Component interfaces are specified using a language-neutral Interface-Description-Language (IDL). This IDL language, though very Java-like, is not Java and can be mapped to a specific programming language, the mapping to Java being straightforward. Figure 1 (b) shows a sample IDL declaration that defines two interfaces, the first one provides access to a framebuffer while the second one abstracts a packet filter that intercepts network packets. Either interface can be implemented by a C component or a Java component.

Components and bindings between component interfaces are specified in ADL (Architecture Description Language). The ADL sample shown in figure 1 (c) defines two primitive components (FrameBuffer and PacketFilter) and a composite component (NetworkMonitor) that encapsulates them.

Each component declaration specifies the interface required or provided by the component. A primitive component declaration specifies the Java class (or C file) that implements the component. A composite component declaration contains binding statements that connect its internal components with one another as well as a controller statement that indicates the type of controller required for the component.

2.2 Java Domain

A Java domain is a specific type of composite component that acts as an isolation domain for the Java components it hosts. It provides to these components this illusion of being under the control of a dedicated Java VM. Thus each domain is associated with:

- its own dedicated heap
- its own garbage collector
- its own view of classes static fields

As shown in figure 2 (a), each Java domain is a composite component whose controller part consists of (i) a Garbage Collector and a VMCore base component (covered in more detail later) and (ii) interceptors. As any component, a

Java domain can export and import interfaces which are its means of communication with external components (other Java domains or native C components). Parameters are always passed by value through these interfaces except for provided/required interfaces that are passed by reference. Two Java domains cannot hold a direct reference to the same Java object allowing an independent garbage collection of each Java domain heap. Automatically-generated interceptors are interposed for each provided/required interface of the Java domain:

- For each provided interface exported by a Java domain, an incoming stub is generated that receives the incoming request. The stub associates a Java context with the incoming thread on the fly making it a Java thread for the time of the invocation. It also unmarshals reference interface parameters creating (or reusing) outgoing stubs.
- For each required interface imported by the Java domain, an outgoing stub acts as a proxy to the external target component. As an optimization, the creation of a stub is avoided if the target object happens to be co-located with the caller.

As hinted above, a cross-domain invocation does not involve a thread switch in either direction (incoming or outgoing calls). An outgoing invocation from a Java domain toward a system component can be considered as a system call and would be implemented in a traditional VM through a JNI mechanism. The ability to define arbitrary communication channels between system components and Java components has no direct counterpart in the standard Java model in which the Java libraries define a fixed interface with the underlying system. Making Java components talk to native C components and vice-versa become transparent and the component architecture allows to keep track of the dependencies between components. On top of that the fact that Java components are statically compiled to native code before deployment allows to move further down the borderline between Java code and C code.

2.3 Java VM Architecture

The Java minimal runtime is integrated within the component architecture as follows (see figure 2 (b)). The VMCore component encapsulates the core of the Java VM including thread management and synchronization functions. The VMCore component interacts with the system Scheduler component (that acts as a thread factory) and the system Synchro component (that acts as a synchronization primitive factory). Memory management is architectured with a strict separation between a policy-independent mechanism layer encapsulated within the VMCore component and a specific garbage collection policy encapsulated by the GC component. The GC component makes use of the Memory Allocator system component for allocating (or possibly extending the heap) and possibly the MMU component for optimizing certain operations (read/write barriers).

The current framework aims to support type-exact stop-the-world or incremental GC algorithms. A Java component (not the functional Java code of the

component but the native code generated from the Java bytecode) invokes some of these methods at appropriate places. For example, a GC safe point is inserted before each method invocation and backward jump, a memory barrier is inserted for every field access when the field type is a reference.

3 Development Tool Chain

The component-aware development tool chain is depicted in figure 3. The construction of a specific OS profile requires as input:

- the ADL specification of the OS at boot time
- IDL specifications for all interfaces used by the target OS
- Component implementations of all required primitive components either in C or in Java

From the IDL declaration, an IDL compiler generated meta-data and appropriate glue code. In particular, a virtual dispatch table is generated for every IDL interface. The IDL compiler can also produce a skeleton implementation for C native components. Component implementations are compiled either using a native C compiler for the target machine architecture or by passing through a Java-bytecode-to-C translation phase (covered in more detail further on). A component assembler takes as input the hierarchical ADL component specification of the OS and computes from it a transitive closure of the graph of components required to build the target OS and generates a list of object files that is handed down to a native linker that produces the binary OS image.

This development tool chain is opened to the extent that it can be complemented with arbitrary controller-specific tools. In the case of Java domain controllers, an IDL compiler generates interceptors for all the IDL interfaces involved in Java domains.

3.1 Ahead-of-Time Java Compiler

Since our current focus is on deterministic Java execution, our system resorts to ahead-of-time compilation of Java bytecode on a systematic basis and does not support bytecode interpretation or JIT compilation. Instead of generating native code directly, the compiler generates C code which allows to reuse an existing C compiler as the final compiler back-end alleviating the task of porting the compiler to each target machine architecture. On top of that, the generated C code needs not to be heavily optimized since the C compiler will take care of producing optimized binary code. The compiler proceeds in two passes. An abstract interpretation of the Java bytecode is first performed in order to determine the type of every stack slot for every instruction. A second pass generates the C code in which the Java stack is simulated (and in some cases bypassed) with local C variables.

The compiler takes advantage of the fact that the Java runtime environment is statically linked with no way to discover and download code at runtime. With this “closed world” approach, the generated code can be optimized as follows:

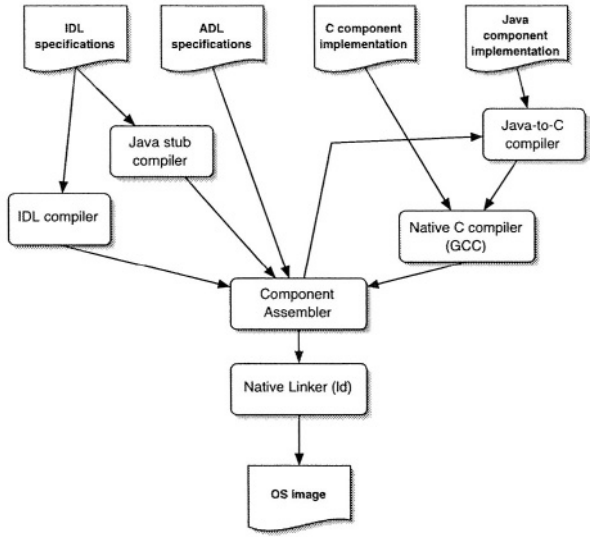


Fig. 3. Development tool chain

- accesses to class fields are compiled as a memory access using a constant offset with no dynamic field resolution
- method call invocations are devirtualized when possible using a Class Hierarchy Analysis (CHA) approach.
- interface method invocations are handled through a constant-time lookup scheme similar to the one proposed in [14] in which every interface method is assigned a global unique ID which is used to index a fixed-sized interface method dispatch table. Moreover, interface calls are virtualized when possible using CHA if the target class can be determined (combined with the previous optimization, this may finally yield a static call).

In addition, the compiler features a certain number of options that allow to produce "unsafe" code that bypass certain run-time checks required by the Java semantics, namely null pointer checks and array bounds checks. Direct Access to physical memory from Java unsafe code is provided through dedicated Java classes. The bytecode-to-C translator is configured to spot the use of such classes and to directly inline the method implementation without the overhead of a method invocation.

With regard to the component architecture, the Java-bytecode-to-C compiler can be used in different ways:

- each Java domain can be considered as a separate "closed world" with optimizations only applying within its boundary. This can lead to code duplication if, say, some invocations can be devirtualized in one domain only.
- all Java code to be deployed is handled globally.

The bytecode-to-C translator is written in Java and follows a component-based design using a Java only implementation of the Fractal component framework [4] and the Jabyce bytecode transformation library [5].

3.2 Component Assembling and Global Optimizations

By default a cross-component invocation (that always goes through a well-identified interface) involves a level of indirection. If both components are written in Java, the bytecode-to-C translator can optimize the invocation using CHA. In the case of invocations between C components, the IDL compiler generates a virtual-method-dispatch table for each IDL interface that is used at run-time to reach the target implementation function. Using specific hints in the ADL specification of the OS profile can direct the component assembler tool to optimize cross-component invocations provided that some programming conventions are followed. The programmer can implement the functions of a particular native component as inline functions (using GCC extension) making it possible to gain the same speed advantage as a macro. Such an optimization is typically used for the Java VM components covered in section 2.3 for which it would unrealistic to implement time-critical functions such as memory barriers through a level of indirection.

Such an optimization is triggered by the component assembler through a simple whole-program analysis when a particular IDL interface (say GCPolicy) is implemented by only one type of component.

3.3 Component Instrumentation

A key benefit of the component architecture is that it allows to instrument the system in a non-intrusive way (i.e. with no modification to the functional code of the components). A component "membrane" (the gray area in figures 2 and 1) can be used to intercept all incoming or outgoing invocations passing through it. Such an interception mechanism can be used to implement monitoring and logging functions that trace cross-component calls at an appropriate level of granularity. Such interceptors are generated by a controller-specific IDL compiler and included in the final "debug-mode" OS image by the component assembler. We are currently conducting an experiment aiming at monitoring the garbage collector component for two different purposes:

- in order to obtain statistics about memory usage (ratio of object reference write access vs read access, locality of the access, lifetime of the objects etc.) that can be used to tune the GC algorithm accordingly. As an example, memory barrier function specified in the abstract interface of the GC policy can be intercepted by a logger.
- in order to validate its implementation subject to many potentially synchronization bugs. Incremental policies are tricky to design and implement [3]. We use the approach described in [2] for producing an oracle from the temporal specifications of the system. Such oracles are useful to check the trace observed at runtime against a verified behavioral model.

4 Performance Evaluation

Our Java-based componentized operating system has to cope with two sources of memory overhead: (i) the use of pre-compiled Java code instead of bytecode and (ii) the overhead of the component framework proper.

Regarding the former point, the C code produced by the bytecode-to-C compiler is inevitably larger that the original bytecode (between 2x and 4x). However the use of static linking allows to remove all the meta-data associated with Java .class files and can compensate for the increase in code size. As an illustration, the DeltaBlue Java benchmark program [17], once statically compiled, yields a binary code of 160KB, roughly the same size as the original .class files.

By default, a Java or native component (either primitive or composite) incurs a memory footprint overhead due to the meta-data associated with the controller of the component that allows to introspect and reconfigure at run-time the component network[1]. The table below shows the memory footprint overhead for default components and optimized "frozen" components. In the latter case, the overhead is drastically reduced by choosing to directly and statically bind sub-components together precluding run-time reconfigurability.

	Standard component	Optimized component
Exported interfaces	4+12*n	8*n
Imported interfaces	4+8*n	0
Sub-components	4+8*n	0
Sub-bindings	4+12*n	0

The cost of a cross-component invocation must be assessed in two different cases: (i) when the two components are located within the same domain (either two Java components within the same Java domain or two native components) and (ii) when the two components are located in different domains. By default an intra-domain cross-component invocation involves one virtual dispatch through a fixed-sized interface method table. On a a RISC processor such as ARM, the overhead of an interface invocation is two instructions compared to a static method call. A first optimization option (that precludes run-time reconfiguration) is to statically linked the caller with the target component implementation thus removing the indirection level. The last drastic option is to inline the target method within the caller. The last two options can be turned on as global optimization options as explained in section 3.2.

The following table summarizes the cost of a intra-domain cross-component call with these options on an ARM processor.

Call type	ARM instructions
default cross-component invocation	4
statically linked cross-component call	2
inlined cross-component call	0 (but memory overhead)

[1] Note that we do not support Java reflexive API but only introspection capabilities at the boundaries between components.

An inter-domain invocation incurs the following overheads:

- a fixed overhead for an incoming call to a Java domain in order to make the incoming thread behave as a Java thread for the duration of the call. This involves creating a Java thread object and saving the current processor state (setjmp operation) to catch any uncaught exception raised in the Java component.
- a variable overhead due to parameters marshalling.

Note that outgoing calls from a Java domain are much cheaper that incoming calls since they do not incur the former overhead. Some optimizations and programming guidelines can help to reduce the communication cost of inter-domain calls: the use of shared memory primitives and the ability to inline the target function in the outgoing stub (e.g. for method such as System.currentTimeMillis).

5 Related Work

A number of operating systems have explored the use of component architectures though very few in the context of embedded systems and with Java in mind. Knit [12] is a component model targeting operating systems (written in C) that relies on a component definition and linking language which is, in first approximation, similar to our model. However Knit does not explicitly model interconnection between components and it is not clear how non-functional code can be inserted in a non-intrusive way. Knit targets workstation operating systems and assumes components of a much larger grain than our model. TinyOS [13] is a component-based operating system targeted at deeply embedded operating systems (e.g. sensor networks) that relies on its own component-aware language (an extension of C) and virtual machine. TinyOS resorts to whole-program-optimization to optimize cross-component calls. Contrary to TinyOS, we argue that with proper extensions (or restrictions) to the Java model, Java can be an adequate system programming language with no need to define an ad-hoc programming environment.

Several academic projects have aimed at constructing a Java-oriented OS complementing the Java architecture with system-level abstraction. KaffeOS [9] is a multi-process Java virtual machine (and as such not an OS strictly speaking). The process abstraction common in classical operating system such as UNIX is introduced within a single JVM as the unit of isolation and resource control. KaffeOS architecture inherits the monolithic characteristic of Unix with a fixed boundary between user code and kernel code. The JX OS [8] comes closer to our approach. JX is a Java-oriented OS based on a component architecture in which Java code is partitioned into domains that can communicate with one another via so-called portals. JX compiles Java code ahead-of-time after download. Contrary to our system, a cross-domain invocation is more costly since it implies a thread switch. The component model underpinning JX is not recursive and makes it difficult to intercept cross-component calls in a not intrusive way. It is not clear

how the VM machinery itself (e.g. garbage collector) can be reconfigured. Despite the claims of their authors, JX is currently targeted at servers with ample amount of resources. As far as commercial systems are concerned, SavajeOS [10] or JBed [11] aim to offer Java-oriented operating systems for mobile phones (compliant with the J2ME/CLDC/MIDP standard) with a strong integration of the Java runtime with the OS layer. There is little information on the internal architecture of these systems and their level of modularity.

Regarding Java-to-C translations, Harissa [6] and Toba [7] are two Java-to-C translators that output C code in a way similar to our approach (abstract bytecode interpretation, optimization based on CHA). However our compiler tool features an open component-based architecture making it easy to customize or extend it. Jepes [18] proposes a compiler specifically targeted at embedded systems along with an "interface-directed" configuration framework which associates Java interfaces with configuration directives allowing optimizations such as inlined assembly macros or interrupt handler wrapping. We believe that Jepes could gain advantage of our component framework with a cleaner separation of functional and non-functional code and on the other hand a less Java-centric approach that does not preclude the mixing of Java and native components.

6 Conclusion

We have presented a component-based operating system framework that allows to construct minimal Java-oriented OS profiles in a modular fashion by assembling and composing fine-grained system-level Java or C components together. The use of a unique formalism to describe the Java virtual machine, the system layer and the application layer yields a clean design that greatly facilitates the iterative design of the system and the reuse of code. The overhead of the component architecture on performance and memory footprint can be kept minimal through a component-aware development tool chain that resorts to global optimization techniques before "freezing" the part of the system that does not require dynamic reconfigurability.

We have developped an OS profile that implements the J2ME/CLDC API along with a subset of the MIDP API with support for statically-compiled code only. This OS runs on an ARM-based platform (iPaQ PDA) with a total memory footprint of 200Mbytes. We are currently extending our prototype toward a more complete implementation of a Java virtual machine including a JIT compiler and support for downloadable Java bytecode. We are also in the process of designing a synchronous Java operating system profile compliant with the STIP Java profile [16] targeting secure terminals with electronic payment support. Such an OS would provide a synchronous automaton-like Java programming model that avoids all the pitfalls associated with asynchronous threads (indeterminism, potential deadlock...). Java domains would act as synchronous "islands" that insulate the Java code from external asynchronous events, the asynchronous-synchronous transition being handled by the component controller transparently from user code.

References

1. J.P. Fassino, J.B. Stefani, J. Lawall, G. Muller: "Think: a software framework for component-based operating system kernels" Usenix 2002, Monterey, California, USA.
2. N. Rivierre, T. Coupaye: "Observing component behaviours with temporal logic" ECOOP 2003 Workshop Correctness of Model-based Software Composition, July 2003 Darmstadt, Germany.
3. K. Havelund: "Mechanical verification of a garbage collector", in 4th International Workshop on Formal Methods for Parallel Programming: Theory and Applications, 12-16 April 1999.
4. E. Bruneton, T. Coupaye, J.B. Stefani: "Recursive and Dynamic Software Composition with Sharing" Seventh International Workshop on Component-Oriented Programming (WCOP02)-ECOOP 2002, Malaga, Spain June, 2002.
5. R. Lenglet, T. Coupaye, E. Bruneton: "Composing Transformations of Compiled Java programs with Jabyce", 2004, submitted for publication.
6. G. Muller, U. Shultz: "Harissa: A Hybrid Approach to Java Execution", IEEE Software, March 1999.
7. T.A. Proebstin et. al., "Toba: Java for Applications, a Way Ahead of Time (WAT) Compiler", COOTS 1997.
8. Michael Golm, Jürgen Kleinöder, Frank Bellosa: "Beyond Address Spaces - Flexibility, Performance, Protection, and Resource Management in the Type-Safe JX Operating System", Eighth IEEE Workshop on Hot Topics in Operating Systems, May 2001, Elmau, Germany.
9. Godmar Back, Wilson C. Hsieh, Jay Lepreau: "Processes in KaffeOS: Isolation, Resource Management, and Sharing in Java", Proceedings of the 4th USENIX Symposium on Operating Systems Design & Implementation, October 2000.
10. Savaje OS: http://www.savaje.com
11. Jbed: http://www.esmertec.com/solutions/mobile_multimedia/jbed_cldc/
12. Alastair Reid, Matthew Flatt, Leigh Stoller, Jay Lepreau, Eric Eide. "Knit: Component Composition for Systems Software" In Proc. of the Fourth Symposium on Operating Systems Design and Implementation (OSDI 2000), October 2000.
13. David Gay, Phil Levis, Rob von Behren, Matt Welsh, Eric Brewer, and David Culler "The nesC Language: A Holistic Approach to Networked Embedded Systems", Proceedings of Programming Language Design and Implementation (PLDI) 2003, June 2003.
14. B. Alpern et al: "Efficient Implementation of Java Interfaces: Invokeinterface Considered Harmless", ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), Tampa, FL, USA, Oct 14-18, 2001.
15. Bollella, Gosling, Brosgol, Dibble, Furr, Hardin, and Turnbull. The Real-Time Specification for Java. Addison-Wesley, 2000.
16. http://www.stip.org/
17. http://research.sun.com/people/mario/java_benchmarking/
18. U.P. Schultz, K. Burgaard, F.G. Christensen, J.L. Kristensen: "Compiling Java for Low-End Embedded Systems", LCTES'03, San Diego, June 2003.

Studying the Behaviour of the Single Parent Rule in Real-Time Java*

M. Teresa Higuera-Toledano

Facultad Informática, Universidad Complutense de Madrid, Ciudad Universitaria, 28040 Madrid Spain
mthiguer@dacya.ucm.es

Abstract. This paper addresses the issue of improving the performance of memory management for real-time Java applications, building upon the Real-Time Specification for Java (RTSJ). This specification imposes strict assignment rules to or from memory areas preventing the creation of dangling pointers, and thus maintaining the pointer safety of Java. The dynamic issues that Java presents, requires for some cases ensuring the checking of these rules at run-time, which adversely affects both the performance and predictability of the RTSJ application. This paper presents an efficient algorithm for managing scoped areas which requires some modifications in the current RTSJ specification.

1 Introduction

The *Real-time Specification for Java* (RTSJ) [6] introduces the concept of scoped memory to Java by extending the Java memory model to provide several kinds of memory areas: the *garbage-collected heap, immortal memory* areas that are never garbage collected, and *scoped memory* areas that are collected when there is not a thread using the memory area. The lifetime of objects allocated in scoped areas is governed by the control flow. Because scoped areas can be reclaimed at any time, objects within an area with a longer lifetime are not allowed to create a reference to an object within another area with a potentially shorter lifetime. Strict assignment rules placed on assignments to or from memory areas prevent the creation of dangling pointers. An RTSJ implementation must enforce these scope checks before executing an assignment.

This paper focuses on data structure and algorithms supporting the RTSJ assignment rules. In order to enforce the safety assignment rules in a more efficient way than the implementation proposed in [3], we study the behaviour of the single parent rule, and propose an alternative implementation based on the current edition of RTSJ [6]. We present an in depth description of the RTSJ memory model (Section 2). We

* Founded by the Ministerio de Ciencia y Tecnología of Spain (CICYT); Grant Number TIC2003-1321.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 268–277, 2004.
© Springer-Verlag Berlin Heidelberg 2004

suggest a solution to improve the RTSJ memory model, which can be implemented efficiently by using simple data structures and algorithms (Section 3). We provide an outline of related work (Section 4). Finally a summary of our contribution conclude this paper (Section 5).

2 The Scoped Memory Model Behavior

RTSJ makes distinction between tree main kinds of tasks: *(i) low-priority* that are tolerant with the GC, *(ii) high-priority* that cannot tolerate unbounded preemption latencies, and *(iii) critical* that cannot tolerate preemption latencies. Since immortal and scoped areas are not garbage collected, they may be exploited by critical tasks. Several related threads, possibly real-time, can share a memory area, and the area must be active until at least the last thread has exited. The way that threads access objects within memory areas in the current RTSJ edition is governed by the following rules:

1. A traditional thread can allocate memory only on the traditional heap.
2. High-priority tasks may allocate memory from the heap, or from a memory area other than the heap by making that area the current allocation context.
3. Critical tasks must allocate memory from a memory area other than the heap by making that area the current allocation context.
4. A new allocation context is entered by calling the `enter()` method or by starting a real-time thread (i.e. a task or an event handler). Once an area is entered, all subsequent uses of the new keyword, within the program logic, will allocate objects from the memory context associated to the entered area. When the area is exited, all subsequent uses of the new operation will allocate memory from the area associated with the enclosing scope.
5. Each real-time thread is associated with a scoped stack containing all the areas that the thread has entered but not yet exited.

Since assignment rules cannot be fully enforced by the compiler, some dangling pointers must be detected at runtime [3]. The more basic approach is to take the advice given in the current edition of the RTSJ specification [6], to scan the scoped area stack associated to the current task, verifying that the scoped area from which the reference is created was pushed in the stack than the area to which the referenced object belongs. This approach requires the introduction of *write barriers* [11]; that is to introduce a code exploring the scoped area stack when creating an assignment. Note that the complexity of an algorithm which explores a stack is $O(n)$, where n is the depth of the stack. Since real-time applications require putting boundaries on the time execution of some piece of code, and the depth of the scoped area stack associated with the task of an application are only known at runtime; the overhead introduced by write barriers is unpredictable. In order to fix a maximum boundary or to estimate the average overhead introduced by write barriers, we must limit the number of nested scoped levels that an application can hold [5].

Scoped areas can be nested and each scope can have multiple sub-scopes, in this case the scoped memory hierarchy forms a *tree*. Consider two scoped memory areas, A and B, where the A scoped area is parent of the B area. In such a case, a reference to the A scoped area can be stored in a field of an object allocated in B. But a reference from a field of an object within A to another object allocated in B raises the `IllegalAssignment()` exception. When a thread *enters* a scoped area, all subsequent object allocations come from the entered scoped area. When the thread *exits* the scoped area, and there are no more active threads within the scoped area, the entire memory assigned to the area can be reclaimed along with all objects allocated within it. For the scoped area model behavior, the current edition of the RTSJ specification [6] adds the following rules[1]:

1. The *parent* of a scoped area is the area in which the object representing the scoped area is allocated[2].
2. The *single parent rule* requires that a scope area has exactly zero or one parent.
3. Scope areas that are made current by entering them or passing them as the initial memory area for a new task must satisfy the single parent rule.

In the current RTSJ, when a task or an event handler tries to enter a scoped area S, we must check if the corresponding thread has entered every ancestor of the area S in the scoped area tree. Then, safety of scoped areas requires checking both the set of rules imposed on their entrance and the aforementioned assignment rules. Both tests require algorithms, the cost of which is linear in the number of memory areas that the task can hold. We suppose that the most common RTSJ application uses a scope area to repeatedly perform the same computation in a periodic task. Then, to optimize the RTSJ memory subsystem, we suggest simplifying data structures and algorithms. In order to do that, we propose to change the RTSJ suggested implementation of the parentage relation for scoped areas.

3 The RTSJ Single Parent Rule

The single parent rule guarantees that a parent scope will have a lifetime that is not shorter than of any of its child scopes, which makes safe references from objects in a given scope to objects in an ancestor scope, and forces each scoped area to be at most once in the tree containing all area stacks associated with the tasks that have entered the areas supported by the tree. The single-parent rule also enforces every task that uses an area to have exactly the same scoped area parentage. The implementation of the single-parent rule as suggested by the current RTSJ edition makes the behavior of the application non-deterministic. In the guidelines given to implement the algorithms affecting the scope stack (e.g. the `enter()` method), the single parent rule guarantees

[1] In the former RTSJ edition these rules do not appear.

[2] The suggested implementation of this rule in RTSJ is not correct nor consistent.

that once a thread has entered a set of scoped areas in a given order, any other thread is enforced to enter the set of areas in the same order. Notice that determinism is an important requirement for real-time applications.

Consider tree scoped areas: A, B, and C, and two task $\tau 1$ and $\tau 2$. Let us suppose that task $\tau 1$ has entered areas A and B, and task $\tau 2$ has entered areas A and C. If task $\tau 1$ tries to enter the area C or task $\tau 2$ tries to enter the area B, which violates the single parent rule, the `ScopedCycleException()` is thrown. If, for example, $\tau 1$ enters the area C before $\tau 2$ tries to enter it, $\tau 2$ violates the single parent rule raising the `ScopedCycleException()` exception. But, if $\tau 2$ enters the area B before $\tau 2$ tries to, then it is $\tau 1$ which violates the single parent rule and raises the `ScopedCycleException()` exception.

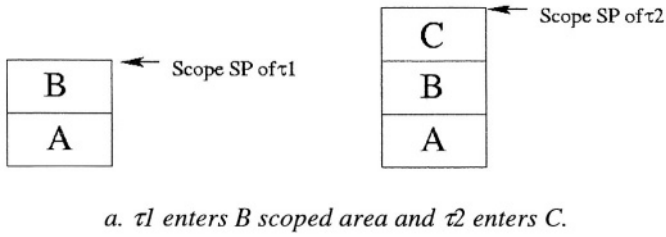
3.1 The Proposed Parentage Relation

In order to maintain the single-parent rule of the current RTSJ edition, we consider that the parent of a scoped area is the area within which the area is created [6], and we add the following rules:

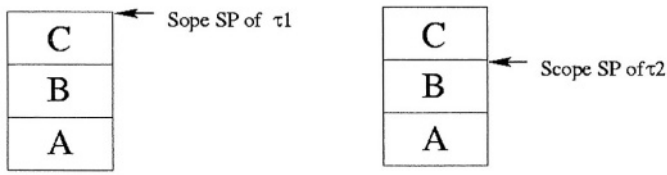
1. The parentage relation of scoped areas implies an scoped area tree structure, where a memory area is added when creating it and deleted when collecting it (instead of when entering and exiting the memory area).
2. Each scoped area must maintain a reference count of the number of scoped areas within which have been created *(child-counter),* in addition to the reference count of the number of threads having it as current area *(task-counter).* The child-counter allows us to maintain alive a scoped area which task-counter values 0 (i.e. it have not yet entered by a task, or it have been entered and exited), but it is the father of the current area of a task.
3. When both reference counters the child-counter and the task-counter for a scoped area reach zero, the scoped area is a candidate to collection.

In this way, as in the current RTSJ edition, we obtain an area tree based on a hierarchy relation. But the parent relation is based on the way that scoped areas are created, instead of the order in which scoped areas have been entered by threads. Consider three scoped areas: A, B, and C, which have been created in the following way: the A area has been created within the heap, the B area has been created within the A area,and the C area has been created within the B area, which gives the following parentage relation: the heap is the parent of A, A is the parent of B, and B is the parent of C. Then, the child-counter for A and B has been incremented to 1, whereas for C it is 0.

Let us further consider the two tasks $\tau 1$ and $\tau 2$ of our previous example, where we have supposed that task $\tau 1$ has entered areas A and B, which increases by 1 the task-counter for A and B. And task $\tau 2$ has entered areas A and C, which increases by 1 the task-counter for A and C (see Figure 1.*a*). In this situation, the task-counter for A values 2, whereas for B and C values 1. If task $\tau 1$ enters the area C and task $\tau 2$ the



a. τ1 enters B scoped area and τ2 enters C.



b. τ1 enters C scoped area and τ2 enters B.

Fig. 1. The scope stack and the single parent rule.



a. τ1 enters B scoped area. *b. τ1 enters C scoped area.*

Fig. 2. Two diferents stated for the coped stack of task τ1.

area B, at different than those that occur in the suggested implementation of RTSJ [6][3], the single parent rule is not violated. Then, instead of throwing the `ScopedCycleException()`, we have the situation shown in Figure 1.*b*. At this moment, the task-counter for scoped memory areas A, B, and C value 2.

Note that the scoped stack associated to task τ2 includes only the A and B scoped regions. Then, even if the task τ2 has entered the scoped memory C before to enter B, pointers from objects allocated in B to objects allocated in C are dangling pointers, as consequence they are not allowed. We consider another situation: task τ1 enters into scoped area A creates B and C, which increases both the task-counter of A by 1 and its child-counter by 2, whereas both the task-counter and the child-counter for B and C value 0. Then, task τ1 enters into scoped areas B (Figure 2.*a*) and C (Figure 2.*b*), which increases by 1 the task-counter of both B and C. Only references from objects allocated within B or C to objects within A are allowed. Note that it is not possible for task τ1 create a reference from an object within B to an object within C, and vice-versa from an object within B to an object within C, even if task τ1 must exit the area C before to exit the area B. Then, if a task τ2 enters into scoped area C and stays there

for a while, task τ1 leaves C and leaves B, the scoped area B can be collected and there are not dangling pointers.

Non-scoped areas (i.e. the heap and immortal areas) are not supported in the scoped tree. Moreover, the heap and immortal areas are considered as the *primordial scope,* which is considered to be the root of the area tree [3]. Notice that, for the heap and immortal memory areas, there is no need to maintain the reference-counters because these areas exist outside the scope of the application.

Then, we propose to change the RTSJ specification, so that scoped memory areas are parented at creation time. This new parentage relation introduces great advantages because *i)* simplifies the semantic of scoped memory as the single parent rule becomes trivially true, *ii)* scope cycle exceptions does not occur, *iii)* each thread requires only one scoped stack, *iv)* and the parentage relation does not change during the scoped memory life.

3.2 Checking the Assignment Rules

We next show how to extend area tree data structures to perform all required checks in constant time. Our approach is inspired in the suggested parentage relation of scoped memory areas. As stated the RTSJ imposed assignment rules, references can always be made from objects in a scoped memory to objects in the heap or immortal memory; the opposite is never allowed. Also the ancestor relation among scoped memory areas is defined by the nesting areas themselves, and this parentage is supported by the area tree. Since area tree changes occur only at determined moments, i.e. when creating or collecting a scoped area, we can apply the technique based on displays that has been presented in [2]. Each scoped area has associated a display containing the type identification codes of its ancestors and its depth in the area tree (see Figure 3).

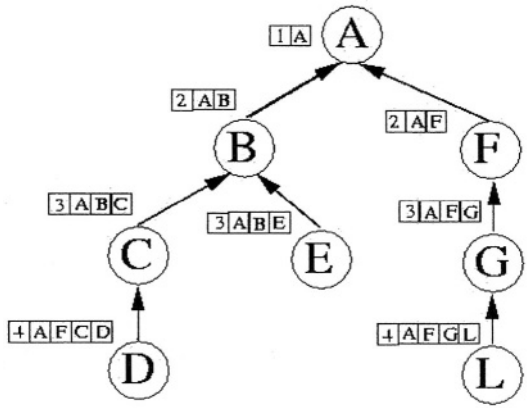


Fig. 3. Display-based area tree structure.

In order to use the display-based technique, we suggest including the following rules in RTSJ, instead of the rule which associates a scope stack to each task:

1. The heap and immortal memory areas are always assigned the minimum depth.
2. When creating a scoped memory area, its corresponding depth is the depth of the father area plus 1.
3. Figure 4 shows the pseudo-code that we must introduce in the execution of each assignment statement (e.g. x.a=y) to perform the assignment checks in constant- time.

```
X = area to which the x object belongs;
Y = area to which the y object belongs;
if( (Y.depth <> 0) and (X.display[Y.depth]<>Y.display[Y.depth])) IllegalAssignment:();
```

Fig. 4. Checking the assignment rules.

3.3 Maintaining the Display Structure

In the current RTSJ specification edition, the `enter()` method can throw the `scopeCycleException()` whenever entering in a scoped region that would violate the single parent rule. The current RTSJ edition also advises to push/pop the entered region on the scope stack belonging to the current task and to increase/decrease the reference counter of the region when the task enters/exits the `enter()` method (see Figure 5).

```
if entering ma would violate the single parent rule throw ScopedCycleException;
push ma on the scope stack belonging to the current thread;
increase the ma reference count;
execute logic.run method;
decrease the ma reference counter;
pop ma from the scope stack.
```

Fig. 5. Current pseudo-code for ma.enter(logic).

Figure 6 shows the pseudo-code of another operation affecting the scope stack, the construction of a new task. In order to maintain the reference counter collector of scoped regions, we must increase/decrease the reference counter of all regions on the scope stack when creating/destroying the task.

```
cma = curent memory region;
ima = initial memory region;
if cma is heap or immortal
        create a new scope stack containing cma
else
        start a new scope stack containing the entire current scope stack;
for every scoped memory area in the new scope stack increase the reference count;
if ima != current allocation context push ima on new scope stack;
run the new thread with the new scope stack;
when the thread terminates every memory area pushed by the thread will have been popped;
for every scoped memory area in the scope stack decrease the reference count;
free the scope stack.
```

Fig. 6. RTSJ pseudo-code to construct a task.

Whereas in the suggested RTSJ implementation solution, actions are required each time a task is created or enters an area having a *O(n)* complexity, where *n* is the number of nested scoped areas. In our proposed solution, when creating a thread or entering an area the display tree structure is not affected (see Figures 7 and 8). Then, both operations, which associate a scoped memory to a task, become constant execution time. Even if these operations are potentially infrequent, we consider that it is an interesting collateral effect obtaining by changing the parentage relation of scoped memory regions.

```
make  ma the current area;
increase the task reference count of ma;
execute logic.run method;
decrease the task reference count of ma;
restore the previous current area.
```

Fig. 7. The proposed enter() method using displays.

```
ima = initial memory area;
make  ima the current area;
increase the task reference count of ima;
run the new thread ;
when the thread terminates decrease the task reference count of ima
```

Fig. 8. Constructing a task by using displays.

The current RTSJ edition also presents another method allowing a task to change the allocation context; the `executeInArea()` method, which checks the current scope stack in order to find the area to which the message associated with the method is sent. Since these methods require an exploration of the stack, they have an *O(n)* complexity. Notice that in our proposed solution entering an area older than the current one that is in the same branch of the area tree (i.e. in the same scope stack), has the same consequences as the `executeInArea()` method. Therefore, this method is not strictly necessary, and actually it does not appear in the former edition of the RTSJ specification.

3.4 Estimating the Write Barrier Overhead

We have modified the KVM [7] to implement three types of memory areas: *(i)* the heap that is collected by the KVM GC, *(ii)* immortal that is never collected and can not be nested, and *(iii)* scoped that have limited live-time and can be nested. To obtain the introduced write barrier overhead, two measures are combined: the number of events, and the cost of the event. We use an artificial collector benchmark which is an adaptation made by Hans Boehm from the John Ellis and Kodak benchmark. This benchmark executes $262*10^6$ bytecodes, where $15*10^6$ performs a store operation (i.e., aastore: $1*10^6$, putfield: $6*10^6$, putfield_fast: $7*10^6$, putstatic: $19*10^6$, and putstatic_fast: 0). That is, the 5% of executed bytecodes perform a write barrier test, as already obtained with SPECjvm98 in [8]. The write barrier cost is proportional to the number of executed evaluations. With this implementation, the average write barrier cost is only 1.6%. Note that this low overhead is not realistic because the KVM is a very small and very slow JVM.

4 Related Works

The enforcement of the RTSJ assignment rules includes the possibility of static analysis of the application logic [6]. Escape analysis techniques could be used in order to remove run-time checks. But the dynamic issues that Java presents, requires for some cases to check the assignment rules at run-time. However, static and dynamic techniques can be combined to provide more robustness and predictability of RTSJ applications. The idea of using both write barrier and a stack of scoped areas ordered by life-times to detect illegal inter-area assignments was first introduced in [4]. The most common approach to implement read/write barriers is by inline code, consisting in generating the instructions executing barrier events for every load/store operation. This solution requires compiler cooperation and presents a serious drawback because it increases the size of the application object code [9]. This approach is taken in [1] where the implementation uses five runtime heap checks (e.g. CALL, METHOD, NATIVECALL, READ, and WRITE). Alternatively, our solution instruments the bytecode interpreter, avoiding space problems, but this still requires a complementary solution to handle native code. The display-based technique was firstly used to support RTSJ scoped area in [2]. The main difference between both techniques is that encoding of the type hierarchy in [5] is known at compile time, whereas in [2] the area tree changes at runtime. This technique has been extended to perform memory access checks in constant-time. The main contribution of our approach is to avoid the single parent checks by changing the parentage relation of scoped area within the area tree, which ensures all algorithms managing scoped areas to be executed in constant-time.

5 Conclusions

To enforce the RTSJ imposed rules, a compliant JVM must check both the single parent rule on every attempt to enter a scoped memory area, and the assignment rules on every attempt to create a reference between objects belonging to different memory areas. Since objects references occur frequently, it is important to implement checks for assignment rules efficiently and predictably. The parentage relation of areas is based on the way they are created/collected, instead of the way they are entered/exited by tasks such as the RTSJ suggests. Then, we avoid checking on every attempt to enter a scoped memory area. Also, changes on the proposed stack based structure are less frequent, which allows us use more simple algorithms. The scope stack can be coded as a display, which allows us to use subtype test based techniques making the enforcement of memory references time-predictable, and does not depend of the nested level of the area to which the two objects of the memory reference belong.

In order to do the implementation of required algorithms more efficient, every scoped area has two reference counters and a scoped stack associated to it, which allows us a more efficient management of areas, making it time predictable. Note that by collecting areas, problems associated with reference-counting collectors are solved: the space and time to maintain two reference-counts per scoped area is minimal, and there are no cyclic scoped area references. Note that the introduction of this change in the parentage relation simplifies the complex semantics for scoped memory region adopted by RTSJ.

References

1. W.S. Beebe and M. Rinard. “An Implementation of Scoped Memory for Real-Time Java”. In Proc of 1st International Workshop of Embedded Software (EMSOFT), 2001.
2. A. Corsaro and R.K. Cytron. “Efficient Reference Checks for Real-time Java. “ACM SIGPLAN Conference On Languages, Compilers, and Tools for Embedded Systems”, LCTES 2003.
3. P.C. Dibble. “Real-Time Java Platform Programming”. Prentice Hall 2002.
4. M.T. Higuera, V. Issarny, M. Banatre, G. Cabillic, J.P. Lesot, and F. Parain. “Area-based Memory Management for Real-time Java”. In Proc. of the 4th International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC). IEEE 2001.
5. M.T. Higuera and, V. Issarny “Analyzing the Performance of Memory Management in RTSJ”. In Proc. of the 5th International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC). IEEE 2002.
6. The Real-Time for Java Expert Group. “Real-Time Specification for Java”. RTJEG 2002. http://www.rtj.org
7. Sun Microsystems. “KVM Technical Specification”. Technical Report. Java Community Process, May 2000. http://java.sun.com.
8. Standard Performance Evaluation Coorporation: SPEC Java Virtual Machine Benchmark Suite. http://www.spec.org/osg/jvm98, 1998.
9. Zorn B. “Barrier Methods for Garbage Collection”. Technical Report CU.CS. Department of Computer Science. University of Colorado at Boulder. http://www.cs.colorado.edu. November 1990.

Supporting Mixed Criticality Applications in a Ravenscar-Java Environment

Hao Cai and Andy Wellings

Department of Computer Science, University of York, UK
{haocai, andy}@cs.york.ac.uk

Abstract. Ravenscar-Java is a subset of Java augmented by a subset of the Real-Time Specification for Java. It is targeted at high integrity real-time systems, however, currently only a single integrity level is supported. This paper proposes extensions to Ravenscar-Java to allow it to support multiple criticality applications within the same virtual machine on a single processor. A real-time isolate is defined which supports both temporal and spatial firewalling. Communication mechanisms are provided to allow controlled interaction between high and low-level integrity applications. The implementation in a Ravenscar-Java environment is discussed. Byte code verification and analysis is performed offline to ensure the robust, predictable, scalable, efficient and safe execution of Ravenscar-Java applications. A temporal deterministic runtime architecture of the Ravenscar-Java is proposed to achieve temporal and spatial isolation between applications, and also improve the scalability by safely sharing the runtime data structures as much as possible with the help of the offline analyzer.

1 Introduction

For many people, Java is the antithesis of a high integrity programming language [6,7]. Ironically, many of the features that have led to its success as a programming language for general applications are problematic when used in high integrity systems. Its combination of object-oriented programming features, garbage collection and its poor support for real-time multithreading are all seen as particular drawbacks. The Real-Time Specification for Java [3] has introduced many new features that help in the real-time domain. However, the expressive power of these features means that very complex programming models can be created, necessitating complexity in the supporting real-time virtual machine. Consequently, Java, with the real-time extensions as they stand, is too complex for confident use in high integrity systems. A high integrity real-time Java profile, called Ravenscar-Java, has been defined [6,7] to address these problems and work is underway to implement the required real-time virtual machine [4].

Executing multiple computations (called “isolates” [8]) in a single instance of a safe language virtual machine can improve performance and overall platform scalability, and realize the full potential of language-provided safety and protection [8,18]. It is also desirable for high integrity systems where multiple applications of different integrity levels may exist on the same platform without having to

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 278-291, 2004.
© Springer-Verlag Berlin Heidelberg 2004

engineering everything to the highest integrity level. The overall objective of this paper is to propose an extension to Ravenscar-Java so that it supports real-time isolates. The paper is organized as follows. Section 2 introduces Ravenscar-Java Isolates. Section 3 proposes an architecture for their implementation. Section 4 discusses related work. Section 5 gives the current status and future work.

2 Real-Time Isolates in Ravenscar-Java

Ravenscar-Java Isolates [15] are defined to improve the overall system scalability, and allow a mixture of different integrity level applications to share the same computational platform. In this section, the overall model of a Ravenscar-Java Isolate is introduced; the approach to spatial and temporal isolates is discussed; the communications between isolates along with the communication integrity policy are also specified.

2.1 Overall Model

A Ravenscar-Java system consists of a fixed number of mixed integrity level isolates. An isolate consists of a fixed number of schedulable objects (real-time threads and asynchronous event handlers) that are created immediately after the program begins its execution. There are two execution phases: an initialization phase and a mission phase, illustrated in Figure 1:

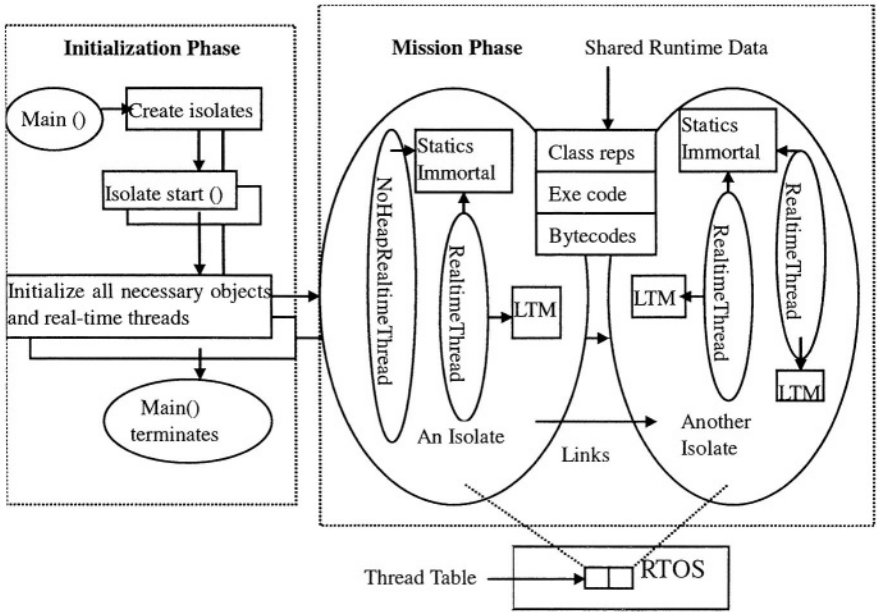


Fig. 1. Ravenscar-Java Isolates.

- Initialization phase — The initialization phase includes the initialization of all the isolates. For each isolate, all classes of the isolate are loaded and all the schedulable objects (and all other permanent objects) are created, all the static variables of classes are initialized. All the isolates must finish their individual initialization phases before any isolate can enter its mission phase. All errors occurring during initialization phase will prevent the mission of all isolates.
- Execution phase — After the initialization of all isolates, all the isolates enter their mission phase at the same time. For each isolate, all its schedulable objects execute under time constraints. To facilitate global predictability and schedule feasibility, no nested isolate creation is allowed during the mission of any isolates. Two-level scheduling is supported. Each isolate is allocated a priority and a fixed amount of CPU time in a specified period by the top-level isolate scheduler. When an isolate is eligible for execution, its thread are scheduled according to the Ravenscar-Java scheduling and release parameters using the RTSJ's default fixed priority scheduler. Each isolate has its own immortal memory and linear time scoped memory (LTM). Isolates do not directly share any objects (but can communicate via a form of message passing). The communications between isolates are allowed as long as their behaviors satisfy the temporal and data integrity of isolates. All the objects for communication are created in their own isolate immortal memory space.

2.2 Approach to Spatial and Temporal Isolates

To allow the running of applications with mixed integrity levels on a single platform, applications must be isolated from one another, both *spatially* and *temporally*. Spatial isolation of applications is usually achieved by executing applications in different memory segments protected by hardware memory management mechanism. However, this approach suffers from communication and context switch overhead, and the approach also reduces the scalability of the overall system. Furthermore, some platforms may not provide this facility. Java is a strongly typed language. This combined with the safety and security mechanisms provided allow a level of trust to be held that unintentional behaviour cannot compromise the integrity of the virtual machine. Each isolate has its own memory areas and does not directly shares any objects with other isolates. Assuming that the extended Ravenscar Virtual Machine (RJVM) has been engineered to the highest integrity level, there is no way for an isolate to access objects of any other isolates. Hence, isolates can be mapped to threads of the underlying real-time operating systems running in a single memory segment. (We assume here a layered approach to fault tolerance. Replication techniques and design diversity may be used at an architectural level to tolerate errors that occur as a result of hardware faults, design faults in the virtual machine or Java compiler etc.)

Temporal isolation is usually achieved by scheduling and resource reservation techniques. Basically, there are four approaches that could be taken when scheduling between isolates. The first approach is the single-level priority based scheduling with CPU budgets enforced. Here, the scheduler schedules all the schedulable objects of all isolates according to a global priority. This approach has the advantage that the scheduling of threads is responsive, but it requires global analysis for all isolates. The

second is the APEX approach [1]: time slicing between isolates and fixed priority scheduling within isolate. Here, a cyclic schedule is used to schedule isolates. Each isolate is guaranteed an amount of time within a period, so no global analysis is required. The main disadvantage is that a high priority thread that becomes runnable outside its isolate time slice must wait for the next time slice. The scheduling of schedulable objects in the isolate is not responsive, and large release jitter can occur. The third treats each isolate as an RTSJ's processing group [3,2] and uses global fixed priority scheduling. This approach groups a set of schedulable objects together with the CPU budgets that is recharged periodically. The thread of the isolate is eligible for execution as long as its processing group has some CPU budget left. The schedulable objects are more responsive especially for sporadic threads, however more complex scheduling analysis is needed for the system. The fourth approach allocates a priority to each isolate and is the one adopted by our extended Ravenscar Java VM. The scheduling between isolates is priority based scheduling but the CPU consumption is bound by using processing group parameter with CPU budgets recharged periodically. Large release jitter can be avoided for the schedulable objects of higher priority isolates. Schedulable objects of a high priority isolate can be scheduled for execution when released as long as the isolate has CPU budget left. Ravenscar-Java targets hard real-time systems. All the schedulable objects must meet their deadline. The priority assignment for isolates is deadline monotonic priority assignment scheme based on the shortest deadline schedulable object of each isolates. To minimize the release jitter of schedulable objects within isolates, the period of an isolate is constrained to the least common multiple of all the periodic real-time threads of the isolate. As long as the first level scheduling parameter for isolates are specified, the feasibility analysis of an isolates can be performed within the isolate without knowing the detailed scheduling of other isolates.

2.3 Communication Between Isolates

Each isolate has several real-time links that connect it with other isolates and allow messages to be sent. To guarantee the integrity level of isolates when communications occur, an integrity policy is defined. Communication between isolates is supported by unidirectional real-time links. The integrity policy controls information flow between isolates and prevents low integrity isolate from corrupting a high integrity isolate. The corruption not only consists of data but also of the temporal behavior of isolates, so both a temporal integrity policy and a data integrity policy should be defined. The communication mechanism between isolates is defined based on the multilevel integrity mechanisms of the GUARDS architecture [9].

Data Integrity Policy
Four integrity levels in the range L0-L3 are defined, where integrity level L3 is the highest level. The following definition and notations are used to describe the integrity policy of the communication mechanism.

- Isolates are partitioned into Single Level Isolates and Multi Level Isolates.
- Single Level Isolates (SLIs) are assigned a single constant integrity level. Their integrity level will not be changed during their execution. They should not receive data items from lower integrity levels than themselves, and should not provide information to higher integrity levels.
- Multi Level Isolates (MLIs) allow more flexibility by being able to provide data to several integrity levels. They can dynamically modify their current integrity level to be able to either provide or receive data from the required integrity level isolates as long as they are validated at a given integrity level in the same way as SLIs.
- Real-time Links are defined as the communication channels between isolates. They are bounded wait free queues holding object references, and allow reading and writing to be performed at the same time without blocking. This preserves the temporal integrity of isolates during communications.

Formally, we define *I* as the set of isolates of the system, which are either *SLIs* or *MLIs,* defining thus a partitioning of *I:{SLI, MLI}.* The current integrity level of isolate $i \in I$ is defined as $pl(i)$. An MLI is assigned an intrinsic level and is able to run at any level less than or equal to its intrinsic level. We describe an MLI using a tuple, { max $pl(i)$, $pl(i)$}, where: the max $pl(i)$ is the intrinsic level which can be reached by the MLI, and $pl(i)$ the current integrity level which reflects the level of information used so far during the execution. $L < i_1, i_2, il >$ is defined as the communication channel from isolate i_1 to isolate i_2 with a static integrity level *il* , which is set at its creation time; Four types of links are defined, along with their associated integrity policy:

1. $\forall (i_1, i_2) \in SLI \times SLI; L < i_1, i_2, il > \Rightarrow pl(i_1) \geq pl(i_2); il = pl(i_1)$

These links require that $pl(i_1) \geq pl(i_2)$ at their creation time. Because the SLI cannot dynamically change their integrity level, so the SLIs can send and receive data using these links without checking for integrity level violation.

2. $\forall (i_1, i_2) \in SLI \times MLI; L < i_1, i_2, il > \Rightarrow il = pl(i_1)$

These links require that $pl(i_1) \geq \max pl(i_2)$ at their creation time. The i_1 sending operations can be performed without checking for integrity level violation. The i_2 cannot change its integrity level higher than $\max pl(i_2)$, so the i_2 can receive messages from the link without checking for integrity level violation.

3. $\forall (i_1, i_2) \in MLI \times SLI; L < i_1, i_2, il > \Rightarrow \max pl(i_1) \geq pl(i_2); il = \max pl(i_1)$

These links require that $\max pl(i_1) \geq pl(i_2)$ at their creation time. The i_1 sending operations are constrained with $pl(i_1) = \max pl(i_1)$. The i_2 receiving operations can be performed without checking for integrity level violation.

4. $\forall(i_1, i_2) \in MLI \times MLI; L < i_1, i_2, il > \Rightarrow il = \max\, pl(i_1)$

These links require that $\max\, pl(i_1) \geq \max\, pl(i_2)$ at their creation time. The i_1 sending operations are constrained with $pl(i_1) = \max\, pl(i_1)$. The i_2 can not change its integrity level higher than $\max\, pl(i_2)$ and $\max\, pl(i_2) \leq il$, so the i_2 receiving operations are be performed without checking for the integrity level violation.

Temporal Integrity Policy
The temporal integrity policy requires that isolates have temporal predictable communications with each other. A bounded wait free queue is used to achieve this. The read and write operations can be performed on the same queue object at the same time without blocking. Timeouts are provided on all send and receive operations for the queue full and queue empty conditions.

Messages Combination Policy of Data Integrity for Communications
A message combination integrity policy is defined to allow different integrity level isolates to communicate without losing their integrity level. For example, when two L0 messages from two different independent isolates are combined by a MLI to form one message, the output message has the integrity level L1 and then it can be passed to integrity level L1 isolates. The policy of combining the independent redundant integrity messages is defined in the table 1. The integrity level of a message can be enhanced only when it is combined with one or more equal, or higher integrity level messages that come from independent redundant isolates.

Table 1. Messages combination integrity policy.

Messages from isolate 1	Messages from other independent isolates	The combined messages
L3	L3	L3
L3	L2	L3
L3	L1	L3
L3	L0	L3
L2	L2	L3
L2	L1	L2
L2	L0	L2
L1	L1	L2
L1	L0	L1
L0	L0	L1

3 Implementation and Run-Time Architecture

To enable Ravenscar-Java to be used in high integrity real-time embedded systems, an analyzable and verifiable implementation strategy is proposed as illustrated in Figure 2.

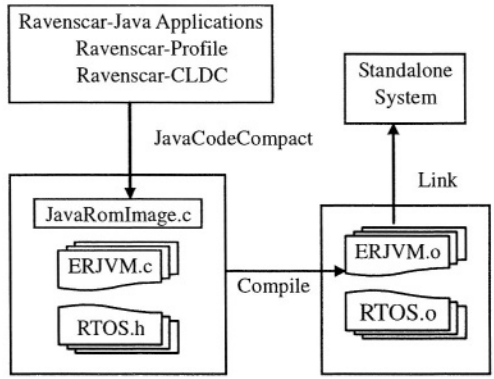


Fig. 2. An implementation of Ravenscar-Java applications.

The class files of the Java applications with the Ravenscar-Profile and the Java support libraries are ROMized by the Java Code Compact tool [17] , which pre-processes all the Java class files into a compact image form. This means that they have been pre-loaded, pre-linked and pre-verified. Java Byte Code optimization, safety and timing analysis are also performed to improve the performance, safety and predictability of the Java byte code execution. Any shared Java classes are also processed into multiple application sharable versions. This image is then compiled and linked with the Extended Ravenscar-Java Virtual Machine (ERJVM). The ERJVM uses the real-time guarantee services of the real-time operating system (RTOS) to provide a predictable and analysable execution environment. The ERJVM can also be statically linked with the RTOS to produce a standalone system that can be burned into the device ROM or downloaded by devices across network for execution.

This implementation strategy has the following advantages:

- Small memory requirement - all the classes have been pre-loaded, pre-linked and pre-verified. No extra space is needed for dynamic class loading and verifying. We also only compact the classes which are needed.
- Simple, reliable and efficient execution model - all the dynamic class loading and verifying features can be removed from the Java Byte Code, resulting in a more efficient deterministic worse case execution time. The code of the ERJVM is also reduced by removing the facilities which support dynamic class loading and verifying.
- The functional portability and the temporal portability of Java applications are enhanced. The Java code compact tools can be extended to perform platform-independent timing analysis to produce portable WCET information [19] and safety analysis. When an application targets for a new RJVM, the WCET information is instantiated with the new RJVM timing model [20]. The resources (CPU, Memory) can be divided according to their requirements when different components (isolates) are integrated to form the single system.
- The system could target platforms that do not provide a file system. For example real-time operating systems which only support the Minimal Real-time System Profile (PSE51) [21].

3.1 Java Code Compaction

The goal of the Java Code Compact (JCC) tool [17] is to offline pre-process Java classes into the internal virtual machine data structures representing Java classes, which can be statically linked into the virtual machine. It creates a platform-independent Java class ROM image from a collection of Java classes. The JCC can take class, Jar and zip files as its input which contains the Connected Limited Device Configuration (CLDC) libraries and applications, and it parses them into their processing runtime structure of the classes in a ROMized form. Sharing of information between the classes and resolving all symbolic references considerably reduce the memory footprint of the Java classes.

One motivation for isolates is to improve the scalability by safely sharing the runtime data structures as much as possible. For the shared runtime classes, the sharable contents include the constant pool and the methods of the classes. The non-sharable contents, which should be identical to each isolate, include the class static fields and the class monitors. The separation of static fields for each isolate is essential to achieve spatial isolation. The separated class monitors allow schedulable objects of different isolates to synchronize accessing classes on their own without interfering with other isolates.

The proposed runtime class structure for class sharing is illustrated in Figure 3. The runtime classes data structure shown in grey are built by JCC offline by analyzing the Ravenscar-Java applications instead of during runtime [18]. The internal shared

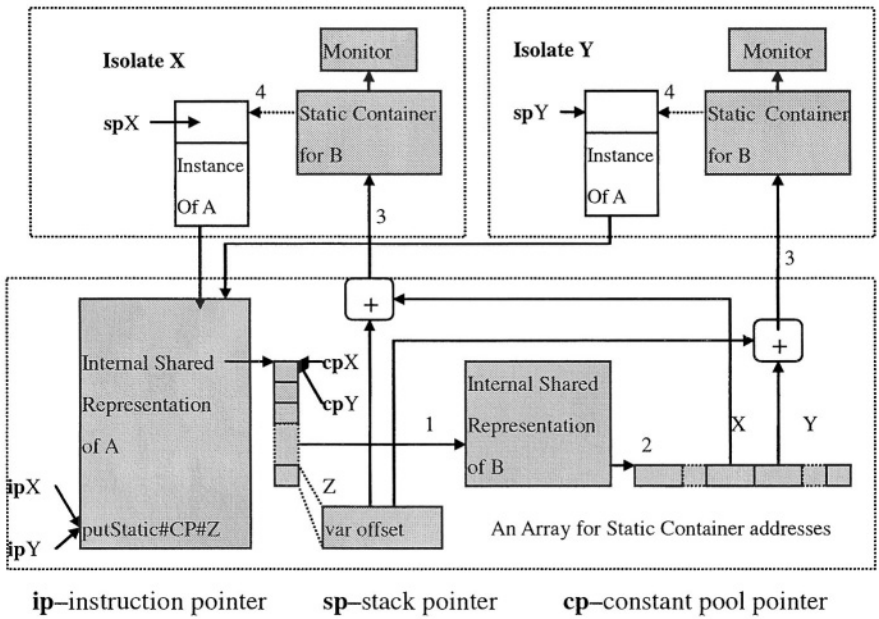


Fig. 3. A shared class runtime structure

runtime representations of classes contain the constant pool of the classes and the methods of the classes. The shared runtime representations of classes also have an array with the size of the number of isolates sharing the classes. The array contains the addresses of the static containers for each isolates. The static container contains the static variables, a pointer to a monitor and some information that indicates the state of the class such as whether the class has been initialized. The direct address of a static variable is resolved by combining the variable offset from the constant pool entry with the address of the static variable container according to the currently accessing isolate.

Figure 3 also illustrates an example where two isolates X and Y both execute the same instruction (putStatic) in class A, which put a newly created instance of A of their own into a static variable of class B of their own. The runtime class structure allows 4 steps to finish the instruction:

1. get the address of the internal representation of B from the constant pool entry which is resolved by JCC.
2. get the addresses of the static containers of the calling isolates.
3. get the offset of the static variable from the constant pool entry and combine the addresses of the containers to get the direct address of the static variables.
4. put the addresses of the isolates own created instances of A into their own direct addresses of static variables of B.

All the shared classes are pre-processed to the runtime class representations by the JCC. The Ravenscar-Java isolates requires that all the classes have been loaded, linked and verified before the mission phase. The off-line processing supports pre-class loading, linking and verifying. The class entries of constant pools are resolved and directly refer to the internal class representations. The creation of static variable containers and their monitor for each isolate is also done off-line by JCC. Ravenscar-Java also requires that all the classes have been initialized before the mission phase; the static initializers of all isolates initialize all static variables of the static containers to its proper initial values during the initialization phase. This prevents recursive class initializations during the mission.

3.2 Run-Time Architecture of ERJVM

In this section, a predictable execution engine is introduced and issues of memory management, scheduling and communication are considered.

The Execution Engine

Basically, there are four main techniques for the execution of Java Byte Codes, the Java processor, the Just In Time (JIT), Ahead of Time compiler (AOT) and the interpreter. The interpreter is a reasonable choice if we balance the portability, predictability and memory requirement for high integrity real-time embedded systems. The interpreter execution engine is extended to support multiple applications without violating temporal and spatial isolation. For handling of synchronized static

methods, the current isolate finds the monitor that is referenced by the isolate static variable container and synchronizes on it. The virtual machine instructions which access the static variables of classes are modified to access the static variables of the current isolate.

Spatial Isolation Support

Each isolate has its own immortal memory and linear-time scoped memory (LTM). Immortal memory is used for permanent objects and LTMs provide the dynamic memory allocation and reclamation mechanism for the isolate. No garbage collection is allowed for any isolates. Within each isolate, objects created in its LTM must not be referenced by objects created in its immortal memory in order to prevent dangling references. The objects created by separate isolates must not be referenced by each other in order to isolate faults.

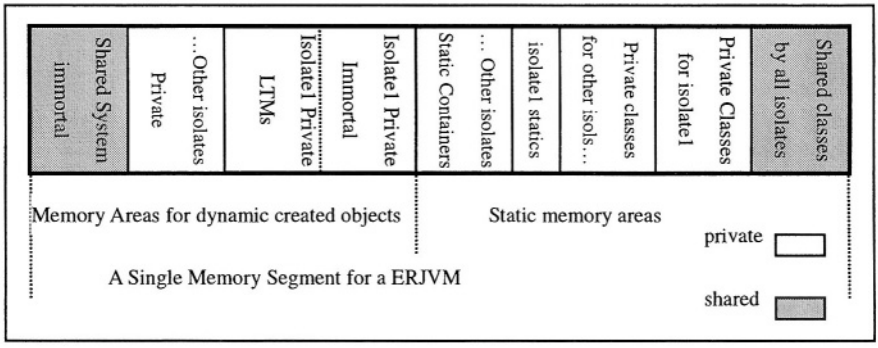


Fig. 4. The layout of memory areas in a ERJVM

The proposed ERJVM executes all the isolates in a single memory segment to increase scalability, facilitate the communication between isolates and reduce context switch overhead. Protection is provided by Java's strong typing and security mechanisms, and by allocating objects for different applications in separated RTSJ memory areas. The layout of memory areas in the ERJVM is illustrated in Figure 4. The shared system immortal area is used for some system lifetime objects which do not belong to any isolates such as the communication channel between isolates. The shared system immortal area is only used for dynamic objects creation during the initialization phase. No dynamic objects will be created in the system immortal during the mission phase. To isolate faults between isolates, each isolate has its own immortal memory and LTMs for dynamic created objects and all the shared runtime data (for instance, the shared class representation) are not allocated memory from any private memory area of each isolate. The shared runtime classes, the separated static containers of the shared classes for different isolates and the private classes of each isolates are also statically created and linked into the ERJVM.

Temporal Isolation Support
Two level scheduling is defined for Ravenscar-Java isolates. To achieve two level scheduling, we map each isolate to two real-time POSIX threads[1]. The first thread has the CPU budget and the priority of the isolate. It performs like a RJVM for a single Java application. The second thread is a maximum priority background POSIX thread which periodically recharges the budget of the first thread. We use the scheduling facilities of the underlying RTOS to provide first level scheduling which is preemptive fixed priority-based scheduling with CPU budget for isolates. Taking the advantages of fast context switch and portability of a green thread model, each isolate has its own runtime data structures for scheduling. The scheduling within isolates is transparent to the underlying RTOS. Each isolate has a timer queue and runnable queues for each priority. When an isolate is scheduled, it wakes up the schedulable objects in the timer queue and put them into the runnable queues. The first schedulable object of the available highest priority runnable queue is scheduled to execute.

Isolate Communications Support
For each isolate, the disjoint object graphs must be maintained to achieve spatial isolation. The communication channels between isolates are created in the system's immortal memory area, which does not belong to the memory areas of any isolates. Each link has an object queue to hold the references of the communication objects, a lock for reader waiting when the queue is empty and a lock for writer waiting when the queue is full, and an event and event handler are also attached to each link to facilitate the predictable synchronized communications. These objects are all created in the system's immortal memory. The real-time thread of the sender isolate creates reusable communication objects in its immortal memory and put their references in the queue of the communication channel. The real-time thread of the receiver isolate deep copies the communication objects in its own memory space either in its immortal memory or its LTM through the references traced from the communication channel. The deep copy mechanism of communication objects is essential to achieve spatial isolation of isolates and prevent isolates operating the objects of each other.

4 Related Work

JSR121 [8] first proposed the notion of an isolate and provided programming-level support for managing and communicating with separate Java programs. However, JSR121 does not address resource isolation issues such as CPU, memory usage etc. A subset of JSR121 is defined for J2ME, however, it still has a complex computational models and the rendezvous communication model between isolates does not address

[1] If the Ravenscar-Java environment shares the processor with other activities via an underlying full POSIX-compliant operating system, the Ravenscar-Java environment is a single process within the POSIX environment assuming a PSE54 environment [21], and the approach outlined in Fig. 2 must be modified and The shared class runtime structure in Fig. 3 and Fig. 4 can be built during the class loading.

the issue of temporal isolation and an integrity policy. RTSJ [3] provide facilities to bound the CPU and memory usage of threads, however, all these facilities are restricted to threads within a single Java application.

There has been an attempt by Higuera-Toledano [5] to consider multiple RTSJ programs executing within a single real-time JVM. She proposes a memory management solution to allow the concurrent execution of RTSJ applications and introduces a CommonHeap and CommonImmortal memory for objects sharing between applications. This memory management model will add extra complexities and overhead to the reference-checking problem of RTSJ. To prevent longer life objects referring to shorter life objects, the reference checking protocol are not well defined between objects in the commonSpaces and protectedMemory. The directly sharing objects in CommonImmortal also prohibit loosely coupled systems.

Dillenberger et al [14] introduces an approach where execution of a new Java application involves creation and initialization of a new instance of the Java virtual machine; but they share a memory region to stored data that can be re-used by all the instances of JVM. Loading, linking and verifying classes happens only once. This provides total application isolation and reduced startup time of application. However, most embedded systems do not have adequate memory to start multiple JVMs. Sometimes, there is even no underlying operating system to launch multiple JVMs.

KaffeOS [11] provides a solution to target the problem of running untrusted applications and prevent denial-of-service attacks. It supports the operating system abstraction of processes to control their resource consumption. If the resource demands are too high, the Java applications will be terminated. To account for the memory consumption and GC time, each application has its own heap which is separately garbage collected. Shared heaps need to be created for directly sharing objects between processes. A kernel heap is used for new created objects of trusted code. From a real-time perspective, this hierarchy memory management is highly inefficient and unpredictable. The directly objects sharing and kernel heap sharing break the spatial and temporal isolation of application.

The J-Kernel[12] and Balfanz and Gong[13] achieve multi-application support based on class loader protection without making changes to the JVM. The spatial and temporal behaviour of applications are tightly coupled. This approach replicates application code and may result undesirable inter-application interactions.

5 Current Status and Future Work

A prototype implementation is being carrying out based on a Ravenscar version of the KVM[4] and the Shark RTOS [16]. Currently, we are modifying the JCC to compile the shared classes into multi-application version in a ROMized form. The integration of the ROMized runtime classes, the modified KVM and the Shark RTOS will allow us to evaluate our approach.

Future work will include: scheduling analysis for the Ravenscar-Java Isolates.

References

[1] ARINC653: Avionics Application Software Standard Interface(Draft 15). Airline Electronic Engineering Committee (AEEC), June 17th, 1996

[2] A. Burns, A.J. Wellings, "Processing Group Parameters in the Real-time Specification for Java", pp.360-370, JRTES, On The Move to Meaningful Internet Systems 2003: OTM 2003 Workshops, LNCS 2889, Springer-Verlag 2003

[3] G. Bollela, B.Brosgol, P.Dibble, S.Furr, J.Gosling, D. Hardin, and M.Turnbull, "The Real-time Specification for Java", Addison Wesley,2000

[4] Hao Cai, Andy Wellings, "Towards a High Integrity Real-time Java Virtual Machine", pp.319-334, JRTES, On The Move to Meaningful Internet Systems 2003: OTM 2003 Workshops, LNCS 2889, Springer-Verlag 2003

[5] M. Teresa Higuera-Toledano, "Memory Management Design to the Concurrent Execution of RTSJ Applications", pp.479-489 JRTES, On The Move to Meaningful Internet Systems 2003: OTM 2003 Workshops, LNCS 2889, Springer-Verlag 2003

[6] J. Kwon, A.J. Wellings, and S. King, "Ravenscar-Java: A High Integrity Profile for Real-Time Java", Proceeding of the Joint ACM Java Grande – ISCOPE 2002 Conference, 2002

[7] P. Puschner and A.J. Wellings, "A Profile for High Integrity Real-time Java Programs", IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2001

[8] Peter Soper et al, "JavaTM Application Isolation API Specification (JSR-121) Public Review Draft", Sun Microsystems, http://jcp.org/en/jsr/detail?id=121 accessed Oct, 2003

[9] Eric Totel, Ljerka Beus-Dukic, Jean-Paul Blanquart, Yves Deswarte, Vincent Nicomette, David Powell, Andy Wellings, "Multilevel Integrity Mechanisms", pp.99-119, A Generic Fault-Tolerant Architecture for Real-time Dependable Systems, Kluwer Academic Publishers, 2001.

[10] Giuseppe Lipari, Enrico Bini, "Resource Partitioning among Real-time Applications", Euromicro Conference on Real-time Systems 2003.

[11] Godmar Back, Wilson C. Hsieh, Jay Lepreau, Processes in KaffeOS: "Isolation, Resource Management, and Sharing in Java", Proceedings of the 4th Symposium on Operating Systems Design & Implementation, 2000

[12] Chris Hawblitzel, Chi-Chao Chang, Grzegorz Czajkowski, Deyu Hu and Thorsten von Eicken "Implementing Multiple Protection Domains in Java". 1998. Proceedings of the USENIX 1998 Annual Technical Conference, pages 259-270. New Orleans, LA.

[13] D. Balfanz and L. Gong. "Experience with Secure Multi-Processing in Java". In Proc. Of the Eighteenth International Conf. on Distributed Computing Systems, May 1998.

[14] D. Dillenberger, R. Bordawekar, C. W. Clark, D. Durand, D. Emmes, O. Gohda, S. Howard, M. F. Oliver, F. Samuel and R. W. St. John, "Building a Java virtual machine for server applications: The JVM on OS/390", IBM Systems Journal volume 39, number 1, 2000.

[15] Hao Cai, Andy Wellings, "A Real-time Isolate Specification for Ravenscar-Java", IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2004

[16] Paolo Gai and Luca Abeni and Massimiliano Giorgi and Giorgio Buttazzo, "A New Kernel Approach for Modular Real-Time systems Development", Proceedings of the 13th IEEE Euromicro Conference on Real-Time Systems 2001, June

[17] Sun Microsystem, The K virtual machine (KVM), http://java.sun.com/products/cldc/wp/index.html , 2004

[18] Grzegorz Czajkowski, Laurent Daynes, "Multitasking without Compromise: a Virtual Machine Evolution", ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications, 2001

[19] G. Bernat, A. Burns, and A.J. Wellings, "Portable Worst-Case Execution Time Analysis Using Java Byte Code", Proceedings of the 12th EuroMicro Conference on Real-Time Systems, Stockholm, June 2000

[20] E. Yu-Shing Hu, A. J. Wellings and G. Bernat, "Deriving Java Virtual Machine Timing Models for Portable Worst-Case Execution Time Analysis", pp.411-424, Java for Real-time Embedded Systems (JRTES), On The Move to Meaningful Internet Systems 2003: OTM 2003 workshops, LNCS 2889, Springer-Verlag 2003

[21] POSIX.13 (1998). IEEE std. 1003.13-1998. Information Technology –standardized application environment Profile – POSIX Realtime Application Support (AEP), The Institute of Electrical and Electronics Engineers.

Optimizing Real-Time Embedded Systems Development Using a RTSJ-Based API

Marco A. Wehrmeister[1], Leandro B. Becker[2], and Carlos Eduardo Pereira[3]

[1] Computer Science Institute
Federal University of Rio Grande do Sul, Brazil
mawehrmeister@inf.ufrgs.br
[2] Automation and Systems Department
Federal University of Santa Catarina
lbecker@das.ufsc.br
[3] Electrical Engineering Department
Federal University of Rio Grande do Sul, Brazil
cpereira@eletro.ufrgs.br

Abstract. Over the last years, Java gained popularity as a suitable programming language for embedded real-time systems. This popularity influenced the definition of the Real-Time Specification for Java (RTSJ), which constitutes a high-level programming interface for creating real-time applications using Java. The current work presents an API based on the RTSJ that optimizes real-time embedded systems development. Using the provided API, programmers can make use of high-level mechanisms to represent concurrency and timing constraints in their Java applications. The developed application can be further synthesized to a Java processor. The paper illustrates the use of the proposed API by means of a case study that implements a crane control system. This case study highlights the benefits and advantages on using the proposed API.

1 Introduction

Over the last years, Java gained popularity as a suitable programming language for embedded and real-time systems development. The definition of the Real-Time Specification for Java (RTSJ) [1] is the most prominent example of such popularization in the real-time domain. The RTSJ defines an Application Programming Interface (API) for the Java language that allows the creation, verification, analysis, execution, and management of real-time threads, whose correction also depends on the satisfaction of timing requirements. Unfortunately the popularization and real-use of this API is still low given the lack of Java Virtual Machines (JVM) that conforms to such specification.

On the other hand, the situation is different in the embedded systems domain, once there are several proposals that optimize the JVM to embedded targets. Such optimizations aim to reduce the required footprint for a JVM as well as to improve the application's performance. The Sashimi environment (see 7) is an example of JVM optimi-

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 292–302, 2004.
© Springer-Verlag Berlin Heidelberg 2004

zation for embedded systems. It provides a powerful and easy-to-use development environment for embedded systems that has been successfully applied to different case studies. An open issue in the Sashimi environment is that it lacks a programming model for representing concurrent and real-time processes.

The goal of the current work is to fill the gap observed in the Sashimi environment by providing an API that supports the specification of concurrent tasks and additionally the specification of timing constraints. The proposed API optimizes real-time embedded systems development. All classes and interfaces available are derived from the RTSJ. Some modifications are performed in the RTSJ to optimize the API to the Sashimi environment. Using the provided API together with Sashimi environment programmers can design concurrent real-time applications and synthesize them into the FemtoJava processor, which is also described in 7.

The remaining of the paper is divided as follows: section 2 presents an overview of the Sashimi environment and the FemtoJava processor; section 3 details the proposed API by describing its class hierarchy; than section 4 presents a case study that elucidates the use of the proposed API, remarking the achieved benefits; finally section 5 highlights some conclusions and future work.

2 Sashimi Environment and FemtoJava Processor

According to the Sashimi environment definitions, designers develop their application directly in Java. In order to fulfill the environment constraints, some programming restrictions must be followed. For example, only integer numbers are allowed, and programmers should only use APIs provided by the Sashimi environment rather than the standard Java-API. Additionally, designers should use only static methods and attributes, since currently there is no support for object allocation. The Java source code is compiled using a standard Java compiler. The generated classes can be tested using libraries that emulate the Sashimi API in the development host.

In the next phase of the development cycle, the application is deployed into the FemtoJava processor, which is a stack-based microcontroller that natively executes Java bytecodes. It was designed specifically for the embedded system market. Therefore, the Java bytecodes of the application are analyzed, and a customized control unit for the FemtoJava processor is generated, supporting only the opcodes used by that application. The size of its control unit is directly proportional to the number of different opcodes utilized by the application software.

Further works extended the Sashimi API to allow concurrent programming, adding an operating system layer which allows dynamic task scheduling. Different scheduling algorithms are supported by environment. An evaluation of the impact of these algorithms in terms of footprint, power consumption and real-time performance are described in [4]. A drawback from such environment is the lack of high-level real-time constructs, enforcing designers to write programs “polluted” with low-level system calls to interact with the scheduler. Additionally, there is mechanism to express clearly the tasks timing constraints. These issues are addressed by the proposed API described in the next section.

3 The Proposed API

As already mentioned, the main goal of the present work is to provide support for high-level real-time programming constructs for a configurable embedded hardware/software architecture based on the FemtoJava microcontroller. An API based in the Real-Time Specification for Java (RTSJ) has been proposed. It allows the use of schedulable objects, which are instances of classes that implement the *Schedulable* interface, like the *RealtimeThread.* It also specifies a set of classes to store parameters that represent a particular resource demand from one or more schedulable objects. For example, the *ReleaseParameters* class (super class from *AperiodicParameters* and *PeriodicParameters*) includes several useful parameters for the specification of real-time requirements. Moreover, it allows the expression of the following elements: time values (absolute and relative time), timers, periodic and aperiodic tasks, and scheduling policies. The term 'task' derives from the scheduling literature, representing a schedulable element within the system context (and in this work used with the same semantic as a schedulable object). The class diagram of the proposed API is shown in Fig. 1. Follows a brief description from the available classes:

- **RealtimeThread:** extends the default class java.lang.Thread and represents a real-time task in the embedded system. The task can be periodic or aperiodic, depending on the given release parameter object. If it receives a *PeriodicParameters* object then the task is periodic, otherwise if the object is an instance of *AperiodicParameters* or *SporadicParameters* class then the task is aperiodic.
- **ReleaseParameters:** base class for all release parameters of a real-time task. It has attributes like cost (required CPU processing time), task deadline, and others. Its subclasses are *PeriodicParameters* and *AperiodicParameters,* which represent release parameters for, respectively, periodic and aperiodic tasks. *PeriodicParameters* has attributes like the start and end time for the task, and the task execution period. The *SporadicParameters* class is a subclass from *AperiodicParameters* class and, as the name suggests, represents a sporadic task that may occur at any time after a minimum interval between two occurrences.
- **SchedulingParametes:** base class from all scheduling parameters that are used by the *Scheduler* object. *PriorityParameters* is a class that represents the task priority and that can be used by scheduling mechanisms like the *PriorityScheduler.*
- **Scheduler:** abstract class that represents the scheduler itself. In the API, the sub-classes *PriorityScheduler, RateMonotonicScheduler,* and *EDFScheduler* represent, respectively, fixed priority, rate monotonic and earliest deadline first scheduling algorithms.
- **HighResolutionTime:** base class from all classes that represent a time value. It has three 32 bits integer attributes that represents days, milliseconds and nanoseconds. The subclass *AbsoluteTime* represent an absolute instant of time which is based in the same date/time reference as specified in java.util.Date class (01/01/1970 00:00:00) [8]. The subclass *RelativeTime*

represents a time relative to other instant of time that is given as parameter. The days, milliseconds and nanoseconds attributes represent, respectively, the quantity of days, milliseconds and nanoseconds relative to a given time instant.

- **Clock:** represents a global clock reference. This class returns an *AbsoluteTime* object that represents the current date and time of the system.
- **Timer:** abstract class that represents a system timer. The derived class *OneShotTimer* represents a single occurrence timer, and the derived class *PeriodicTimer* represents a periodic one.

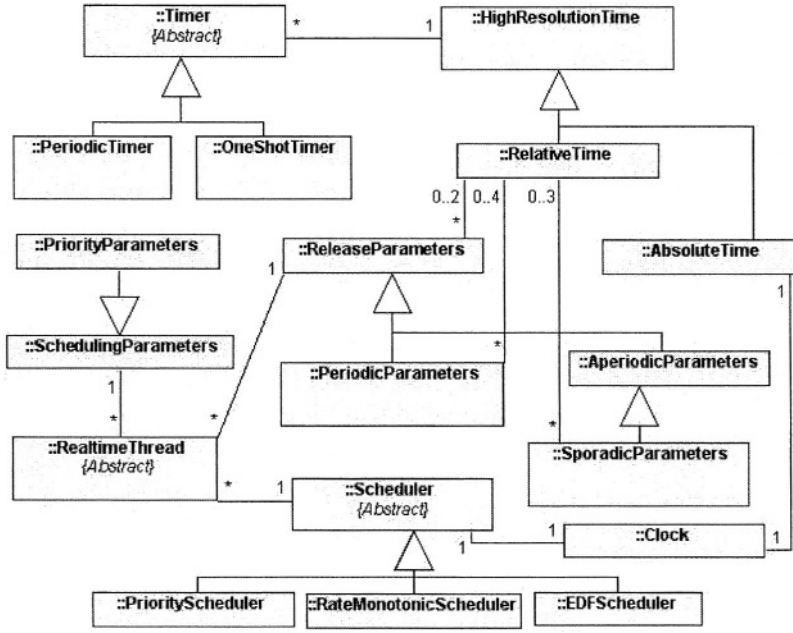


Fig. 1. Class diagram of the proposed API

The implementations from some of the proposed API classes have slightly differences in comparison to the RTSJ. This is due to constraints in the FemtoJava architecture and also for clarity matters. An example of such differences appears in the *RealtimeThread* class. In the proposed implementation it has two abstract methods that have to be implemented in the derived subclasses: *mainTask()* and *exceptionTask()*. They represent, respectively, the task body – equivalent to the *run()* method from a normal Java thread – and the exception handling code applied for deadlines misses. The latter substitute the use of an *AsyncEventHandler* object, which should be passed to the *ReleaseParameters* object, as specified in the RTSJ. If the task deadline is missed, the task execution flow deviates to the *exceptionTask()* code. After the exception handling code execution, the execution flow may deviate to the *run()* method or even terminated, depending on the real-time task characteristic. If the task is periodic, than the *run()* method should be restarted. This difference was proposed to provide

support to scheduling algorithms that use the concept of task-pairs, like the Time-Aware Fault-Tolerant (TAFT) scheduler [3], and also to enhance the entire task source code readability. provide more legibility to source code of the entire task code

Other class that has a different implementation is the *Timer* class. It has a abstract method named *runTimer()* that must to be implemented in the subclass and represents the code executed as the timer expires. Note that this method appears both in *OneShotTimer* and *PeriodicTimer* classes.

To support the utilization of the proposed API, some development tools must be modified and extended. For example, the Sashimi tool must support the synthesis of objects, as well as the analysis of the Java class files must be adapted. Other example, in the FemtoJava microprocessor four new opcodes must be supported to provide support to objects. Incoming sections details some required modifications to support the proposed API in the related development tools.

3.1 Proposed Extensions to Sashimi Environment

As mentioned in section 2, the original version of the Sashimi environment provided no support for object creation. Therefore, some adaptations are needed in order to provide full support for the proposed API in the FemtoJava platform. Firstly it was necessary to extend the Sashimi to support the synthesis of objects. According to the performed modifications, applications objects are allocated statically at synthesis time. In other words, all objects in the system are defined a priori, allowing the determination of the total memory necessary to hold them into the RAM. Although such practice incurs into higher memory usage, it is a suitable practice in real-time development, as it avoids the use of the garbage collector, on which introduced overhead can not be tolerated in real-time applications.

Another required adaptation in that tool is changing the bytecode analysis of the Java class files, in order to identify the objects that must be statically allocated. Thereby the reference for the allocated object (RAM memory address of the first attribute in the object) is be inserted into the bytecode in substitution of the *new* opcode. All bytecode occurrences related to the Java command *new* are replaced. It is important to emphasize that the object will never be released from memory, once there is no garbage collector running.

To allow accessing the object attributes, the *getfield* and *putfield* opcodes also need to be supported in the synthesis of the embedded system. As there is no constant pool in the embedded system generated by the Sashimi tool, the *getfield* and *putfield* opcodes are required. The reference to constant pool was replaced by the memory offset of the field related to the initial address of the object. In summary, to access an object field its memory address should be calculated based on the object initial address and the respective field offset.

Other important extension implemented is the frame allocation for all objects modeled as real-time tasks (derived from *RealtimeThread* class). The code for the task frame allocation in the stack is inserted automatically by the Sashimi tool in the synthesis of the embedded system. The maximum number of task allowed in the embedded system generated by Sashimi tool is direct related with the amount of available RAM memory.

3.2 HW Extensions to FemtoJava Microprocessor

The FemtoJava microprocessor also needed to be enhanced in order to support the proposed API. In the generated new version, it supports four new opcodes: *getfield, putfield, invokevirtual* and *invokspecial.* The first two opcodes are related to object fields' access. They are used, respectively, for getting and setting values. The other two opcodes are related to method invocation. The *invokevirtual* is used to invoke a public or protected method, and the *invokespecial* is used to invoke constructors and private methods.

Another required change in the FemtoJava microprocessor is the addition of a real-time clock, used to provide the notion of time in the embedded system. The real-time clock is composed of three 32 bit integers. The first integer represents the days past from 01/01/1970 (according to the Java class java.util.Date); the second integer represents the milliseconds passed from the 00:00:00 of the day; the last integer represents the nanoseconds passed from the millisecond. This clock should be used both by the API elements and also by the scheduling layer.

4 Case Study

This section illustrates the use of the proposed API in an embedded system project. The system to be modeled consists of a crane control system, proposed as a benchmark for system level modeling [2]. This system incorporates hard real-time constraints. The design solution to be presented along the section follows the UML model of the crane system that is presented in [4]. Fig. 2 shows the UML class diagram from such model.

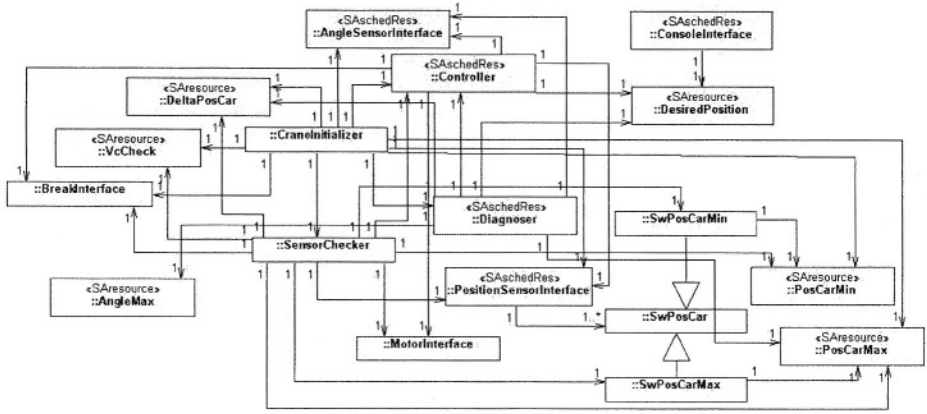


Fig. 2. Class Diagram from the Crane System

An important aspect from this diagram is that it is decorated with stereotypes derived from the UML profile for performance, schedulability, and time, or simply UML-RT [6]. An example from such stereotypes is the «*SASchedulable*», which de-

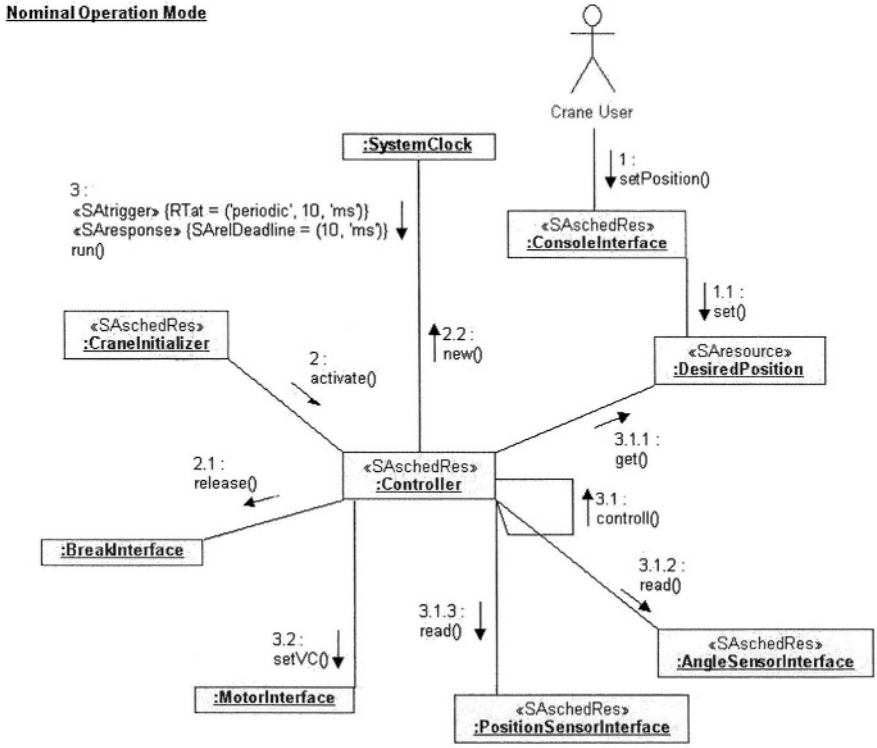


Fig. 3. Object Collaboration Diagram for the Crane System

notes a concurrent task in the system (see *Controller* class). Before illustrating the use of the proposed API, let's also analyze the object collaboration diagram presented in Fig. 3. The *Controller* class is also decorated with the stereotypes «*SATrigger*» and «*SAResponse*», which are responsible, respectively, for indicating that this task is triggered periodically every 10 ms, with a deadline of 10 ms. This is a typical timing constraint present in a real-time system.

Although the system is composed of several classes, as shown in Table 1, due to space limitations the paper focuses on describing only the *CraneInitializer* and *Controller* classes. Nevertheless, authors consider that they are representative enough to depict the use of the API elements. The mentioned description is presented in the next section.

4.1 Using the Proposed API

The main class for the crane system is named *CraneInitializer*. This class is responsible for objects allocation, initialization, and starting (applied for the real-time tasks). Its source code is shown in Table 1. As it can be observed, only static objects are allocated. This constitutes a design restriction from the current environment that was already discussed in section 3.1. Another important aspect from the code of Table 1

relates to the *initSystem( )* method, which represents the starting point from the application execution flow. Therefore, it provides objects initialization and real-time tasks startup. The real-time tasks startup can be identified by the call to the *start( )* method. The last consideration relates to calling the *sleep( )* method. This means that this initialization class will not be used anymore, and that it should be blocked. Such practice should be performed only in the system initialization method.

Table 1. Crane System's Main Class

```
public class CraneInitializer {
   // Allocation of Application objects:
   public static Controller nominalCtrl = new Controller();
   public static ConsoleInterface consoleInterface = new Con-
soleInterface();
   public static DesiredPosition desiredPosition = new Desired-
Position();
   public static BreakInterface breakInterface = new BreakInter-
face();
   ... //aditional objects are instantiated

   public static void initSystem() {
      // Objects initialization:
      Crane.breakInterface.setOn();
      Crane.posCarMin.set(1); // minimum position in meters
      ... //initializes remaining objects
      // Real-time tasks startup:
      Crane.angleSenorInterface.start();
      Crane.nominalCtrl.start();
      ... //startup remaining tasks
      while (true) {FemtoJava.sleep();}
};
```

The following discussion relates to the *Controller* class, on which the associated stereotype denotes a concurrent real-time task in the system. As previously mentioned, this task is triggered periodically every 10 ms, with a deadline of 10 ms (see the collaboration diagram presented in Fig. 3). To implement such features using the provided API, the Controller class needs to inherit from *RealtimeThread,* as shown in Table 2. Moreover, it must define release parameters to implement the modeled features. Therefore, it is used the class *PeriodicParameters* from the API, which instance is passed as parameter for the super-class constructor (see Table 2). A *RelativeTime* object is used to represent the 10 milliseconds from the task period and deadline.

Table 3 depicts the remaining parts from the *Controller* class. It presents two important methods: *mainTask( )* and *exceptionTask( ).* The former represents the task body, that is, the code executed when the task is activated by calling the *start( )* method (see Table 1). Since this task is periodic, there must be a loop which denotes the periodic execution. The loop execution frequency is controlled by calling the *waitForNextPeriod( )* method. This method uses the tasks release parameters to inter act with the scheduler and control the correct execution for the method. The *exceptionTask( )* method represents the exception handling code that is triggered in case of

Table 2. Controller Class

```
import saito.sashimi.realtime.*;

public class Controller extends RealtimeThread {
 private static RelativeTime _10_ms =new RelativeTime(0, 10, 0);
   private static PeriodicParameters schedParams =
     new PeriodicParameters(null, start_time,null, //
            end_time,_10_ms, period) null, // cost
                                  _10_ms);// deadline
   public Controller() {
      super("Controller", null, schedParams);
      // do other initializations
   }
   ... //continues
};
```

Table 3. Controller Class

```
Public class Controller extends RealtimeThread {
   ... //continuation
   public void mainTask() {
      Crane.breakInterface.release();
      while(isRunning == true){     // periodic loop
         this.controll();
         Crane.monitorInterface.setVC(m_vc);
         this.waitForNextPeriod();}
   }

   private int controll() { ... }

   public void exceptionTask() {
      // activated when deadline is missed
   }
};
```

deadline miss, that is, if the *mainTask()* method does not finish up to the established deadline.

The last discussion relates to the ConsoleInterface class, which is responsible for control the interaction between the crane user and the crane system. This class was chosen because it exemplifies the timer usage. The scenario exposed in this sample relates to an operation parameter change. The user chooses the parameter to change and, afterwards, has up to 15 seconds to save the new value. If he does not save the new value until the time limit, then the crane user interface will be restarted. In the Table 4 one can observe the *ParameterTimeOut* class that is responsible to signal the timeout. Note that the *ParameterTimeOut* extends the *OneShotTimer,* in other words, the timer will be executed once per timer activation.

Table 4. ParameterTimeOut class

```
public class ParameterTimeOut extends OneShotTimer {
   private ConsoleInterface m_owner;

   public ParameterTimeOut(ConsoleInterface owner, HighResolu-
tionTime timeout) { super(timeout); m_owner = owner;}

   protected void runTimer() { owner.abortUserInput();}
};
```

The ConsoleInterface class can be analyzed in Table 5. Note that in the *getParameterFromInterface()* method the timer will be started and the routine will remain in loop until the user saves the new value or until the operation is aborted because the timeout occurrence.

Table 5. ConsoleInterface class

```
public class ConsoleInterface extends RealtimeThread {

   private RelativeTime _15_s = new RelativeTime(0, 15000, 0);
// 15 seconds
   private ParameterTimeOut paramTimeOut = new ParameterTime-
Out(this, _15_s);
   private m_UserInputOK = false;
   private m_Abort = false;
   ... // additional code

   public void getParameterFromInterface() {
      ... // Print user interface and wait the input
      m_UserInputOK = false;
      m_Abort = false;
      paramTimeOut.start();
      while ((!m_UserInputOK) || (!m_Abort))
         FemtoJava.sleep();

      ... // make apropriate finalizations
   }
   public void abortUserInput() { m_Abort = true; }
};
```

4.2 Generating the Embedded System

The next step in the project cycle is to synthesize the embedded system. The tool used to make the synthesis is the Sashimi environment. It that takes as input the Java class files generated by the Java compiler to generate the hardware from the FemtoJava processor (in form of VHDL files) and the software of the embedded system.

It is important to highlight that the hardware generated by Sashimi is optimized. More specifically, the generated FemtoJava processor supports only the Java opcodes used by the embedded system software. Thereby it allows economizing in area, which is a design constraint from actual embedded systems.

Another point to be observed is that once the application objects are allocated at synthesis time, there is no need for using a garbage collector. Although this leads to

higher memory consumption, it provides the required determinism from real-time embedded applications.

5 Conclusions and Future Work

The current work presented an API based on the RTSJ that optimizes real-time embedded systems development. This API is target to the FemtoJava processor, which is a stack-based processor specially design to execute Java bytecodes. Using the provided API, programmers can make use of high-level mechanisms to represent concurrency and timing constraints in their Java applications. Only minor extensions were proposed to the proposed API in comparison to the RTSJ. These changes relates basically to the definition of a timeout exception handler for the concurrent real-time tasks operations, which is activated as soon as an operation violates its deadline. The provided method provides a clear code, which is considered easier to be understood.

Extensions to the adopted modeling tool in order to provide code generation from the UML diagrams to Java using the API elements is planned as future work. This will make possible to generate the embedded application directly from the high-level UML-RT model.

Acknowledgments. This work has been partly supported by the Brazilian research agency CNPq - SEEP research project. Special thanks are given to our colleagues, Flavio Wagner, Luigi Carro, and Julio Mattos for their valuable discussions.

References

1. Bollella, Greg; Gosling, James; Brosgol, Benjamin (2001). "The Real-Time Specification for Java", http://www.rtj.org/rtsj-V1.0.pdf
2. E. Moser and W. Nebel. Case Study: System Model of Crane and Embedded Control. In: Proceedings of DATE'1999 – Munich, Germany, March 1999
3. E. Nett, M. Gergeleit, and M. Mock "Enhancing OO Middleware to become Time-Awere", Special Issue on Real-Time Middleware in Real-Time Systems, 20(2): 211-228, March, 2001, Kluwer Academic Publisher. ISSN-0922-6443.
4. L. Brisolara, L.B. Becker, L. Carro, F. Wagner, and C.E. Pereira. "Evaluating High-level Models for Real-Time Embedded Systems Design". To be published in: IFIP Working Conference on Distributed and Parallel Embedded Systems. Toulouse, France. 2004.
5. L.B. Becker, M.A. Wehrmeister, L. Carro, F. Wagner, and C.E. Pereira. "Assessing the Impact of Traditional Real-Time Scheduling Algorithms on Top of Embedded Applications" To be published in: 29th Workshop on Real-Time Programming. Istanbul, Turkey, 2004.
6. Object Management Group (2003). "UML Profile for Schedulability, Performance and Time Specification", http://www.omg.org/cgi-bin/doc?ptc/02-03-03
7. S.A.Ito, L.Carro, R.P.Jacobi. "Making Java Work for Microcontroller Applications". IEEE Design & Test of Computers, vol. 18, n. 5, Sept/Oct. 2001, pp. 100-110
8. Sun Microsystems, "Java 2 Platform API Specification", http://java.sun.com/j2se/1.4.2/docs/api/

Lightweight Firm Real-Time Extensions for Low Memory Profile Java Systems

Helge Böhme and Ulrich Golze

Technische Universität Braunschweig, Abteilung Entwurf integrierter Schaltungen,
Mühlenpfordtstraße 23, 38106 Braunschweig, Germany
{boehme,golze}@eis.cs.tu-bs.de

Abstract. This paper shows a novel approach to real-time extensions for Java running on low memory profile systems. Dynamic time-driven firm real-time scheduling is rendered possible by merely using regular Java threads. A three level scheduling scheme using earliest deadline first (EDF), fair-share and round-robin policies is integrated into a Java virtual machine kernel. The application programming interface (API) of this extension is elementary. Just one class representing deadlines must be added to the system library. Time critical code regions are encapsulated using standard synchronization techniques on deadline objects. They are applicable to oneshot, periodic and sporadic tasks. The underlying concepts of this technique are characterized in detail and applications are presented that already make use of these real-time capabilities.

1 Introduction

The vision of the future household with communicating devices and an integrated control of the whole system motivates our researches. The key point of the smart home are low priced embedded systems. Every electrical component will need an integrated controller to communicate over a home network. Node software can be quite complex because the system has to support complex control scenarios. We aim to use object oriented software to handle the physical objects of the house.

1.1 The JControl Project

The JControl project focuses on embedded control especially in the future household. We have developed a Java virtual machine running on 8-bit microcontrollers for home network node realization. To reduce overall system cost, these nodes must be small and cheap, performance is not an issue. Applications range from embedded regulators to human user interfaces using a graphical display and touch screen. All implementations fit in a 16-bit address range. Typical controllers have 60 KBytes on-chip ROM and less than 2 KBytes of RAM, some are equipped with external flash memory.[1]

[1] Our current reference implementation resides on ST-Microelectronics ST7 microcontroller running at 8 MHz core clock. VM kernel and most parts of the class library are natively implemented in assembler [1].

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 303–314, 2004.
© Springer-Verlag Berlin Heidelberg 2004

The virtual machine is a clean room implementation of the specification [2] with reduced primitve data type set similar to the CLDC 1.0 configuration of the J2ME [3] (no support of `float`, `double` and `long`). The JavaVM supports multithreading and garbage collection using a concurrent conservative mark-sweep-compact scheme. The VM is using a microkernel architecture with bytecode interpreter, memory manager and thread scheduler. Other components of the Java runtime (e. g. class preparation and garbage collection) are implemented on top of the kernel in a thread.

To handle our application domain under this restriced memory situation we implemented our own class library. The JControl API consists of a reduced set of core Java classes needed by the Java language (some classes of the `java` package) and our own classes for communication and control (in the `jcontrol` package). A typical implementation makes use of 37 classes, interfaces and exceptions.

To improve user interface feedback and in particular implement feasable and accurate regulators we intended to have real-time extensions that fit into our memory profile and class library.

1.2 Related Work

We did not found many projects providing Java virtual machines in the 64 K memory range. Real-time capabilities of these systems are not distinct nor existent. The simpleRTJ [4] supports multithreading but no priorities nor preemptions (possibly simple round-robin scheduling). A simple scheme for asynchronous events is implemented to handle native interrupts. The muvium microVM [5] uses ahead of time compiler techniques to be executed on a PIC microcontroller and doesn't support multithreading at all. The TinyVM [6] for use with Lego Mindstorms RXC module supports multithreading using a preemtive round robin scheduler. No timer is used for preemtion but a bytecode counter. There are no further real-time extensions.

Real-Time Extensions for Java. We considered to use a subset of an established real-time extension for Java. The Real-Time Core Extension (RT-Core, [7]) installs a second class hierarchy for real-time tasks (the *Core*). Core classes use a different verificator, scheduler and memory model than non-real-time classes (the *Baseline*). Using RT-Core would be a good choice for a new designed larger system but not for simple real-time attachments.

The Real-Time Specification for Java (RTSJ, [8]) is a more careful approach mainly using an additional API with virtual machine interface. A thread and event inheritance tree is used for specifying different kinds of real-time behavior. The system is completed by memory scopes and installable and configurable schedulers. The RTSJ is far too large to be used with our memory profile. Most interesting would be a subset of the RTSJ or a similar approach found in [9]. But also these approaches are too large or restrict real-time capabilities.

1.3 Our Approach

This paper shows a different approach for providing small Java systems with real-time capabilities. First we present in section 2 our real-time kernel to give an overview of the principles and functionalities that stand behind our real-time extension. We show how our EDF and fair-share scheduler works and explain some refinements of real-time scheduling. In section 3 we present our minimized real-time API using a time-based approach rather than an event-based. In section 4 we show that this concept is utilizable using some real-life examples. Furthermore we present some measured results how scheduling time depends on various parameters.

2 The Real-Time Kernel

Our virtual machine is not designed to run on top of an operating system. In fact all system management tasks are performed by the VM itself. A main part of the VM is the thread scheduler and dispatcher. Multithreading is implemented without native interrupt usage. After calling the bytecode interpreter to execute code of a thread, the bytecode interpreter is responsible for a reliable return to perform a context switch. The advantage of this behavior is simplicity. The bytecode interpreter will never leave a thread or an object in an undefined state. There is no need for locks on variables, all variables behave as if they were declared `volatile`.

The decision to return from interpreting a thread is taken by some events or directly. A direct return is performed if a thread needs a resource only availabe in another thread (e. g. if a class must be prepared) or by invocation of `Thread.yield()`. Event triggered returns are performed by frequently inspecting a bit field between the execution of bytecodes. Setting a bit works like a native interrupt at bytecode level. *Bytecode interrupts* may be triggered by native interrupts of the controllers interface hardware. A special bytecode interrupt is triggered by a programmable timer implementing a *millisecond counter*. This timer is used by the thread scheduler to apply time slices to thread execution (see below). The millisecond as time base was chosen to be sure that all non-interruptible activities (bytecodes and portions of native code, e. g. memory movements of the garbage collector) will fit into one period (see measurements in section 4). So thread scheduling jitter will always be less than one time increment.

2.1 Thread Dispatching

To simplify thread management the use of multiple run queues (e.g. for different priorities) was abandoned. All `Threads` reside on the Java heap and are linked to a single list. The list is circular and an internal VM variable points to the currently executing or last executed thread. This list is processed every time the bytecode interpreter returns. The thread scheme presented in [10] was implemented but refined by sub-states (see figure 1). The sub-states of *blocked* are mainly for internal VM block cause distinction. If an event has occured, all

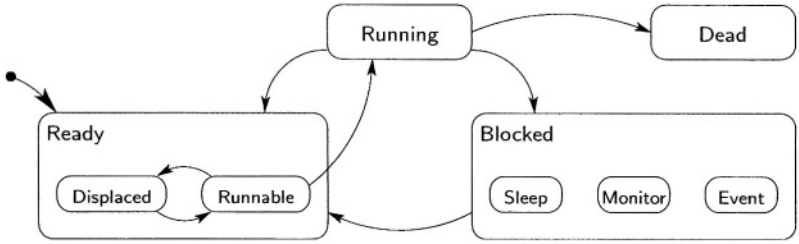


Fig. 1. Thread states

threads with matching state are put to *ready* state and they will check again their blocking condition on execution (and eventually fall back to *blocked* if the condition hasn't changed). Hence, the thread dispatcher doesn't need to know any blocking causes. Threads in the state *blocked by sleep* are handled by the thread dispatcher itself. Thread dispatching follows these steps:

- the scheduler scans the thread list and chooses the thread to be executed next (this is explained in the following section),
- if all threads are in a *blocked* state the system is put into sleep mode and rescheduling is performed after wakeup. If some of these threads are in *blocked by sleep,* the millisecond timer is programmed to wakeup the system just in time when the first of these threads should be scheduled next,
- if one *ready* thread is chosen by the scheduler the bytecode interpreter is called. But first the millisecond timer is programmed to interrupt bytecode execution to reschedule any awaking *blocked by sleep* thread just in time. If no thread has to be awoken by the thread dispatcher in near future, the interruption is done not later than a chosen *time slice*[2].

The thread list scan is always started at the next thread pointed by the last executed thread, so the default behavior (for non-blocked threads of the same priority) is preemptive round-robin scheduling with time slice.

2.2 The Scheduler

Our Java system is designed for interactive and real-time applications. According to [10] mapping the standard Java priorities using one of these two policies were considered:

Priority Scheduling. Only threads with the highest priority are allowed to run. This policy is used in most Java- and real-time systems. The advantage is a high scheduling determinism if all threads have their own dedicated priority and if no unexpected system coherences affect scheduling.

[2] The time slice (CPU quantum) is chosen for balancing scheduling costs and fluent concurrency [10], in our reference implementation it's 64ms.

Fair-Share Scheduling. A thread's priority defines a rule for its *execution probability.* The advantage is fairness of CPU assignment to all *ready* threads even if some threads are not cooperative to other threads.

Fair-share scheduling was chosen for mainly two reasons. First the system is designed to accept third party software components. Thus prediction how thread-friendly a component will be (by intention or by mistake) is not possible. The second reason for a fair-share scheduler is the designated VM architecture. Some maintenance tasks like garbage collection are running in a thread. This thread must never be blocked by an application thread as this would compromise system reliability. Moreover, this thread (especially the garbage collector) should be unobtrusive to the application so this thread must run at low priority (read more on garbage collection below). The lack of scheduling determinism is compensated by our real-time extension.

2.3 Real-Time Scheduling

Like others our approach should support *periodic* and *sporadic* real-time tasks. Many Java real-time implementations support different tasks using different classes extending `Thread` or some real-time thread class. Every future task is prepared and then represented with an object similar to `Thread` on the heap waiting for its event to fire. To reduce overall object usage in the real-time implementation, the use of already existing threads for realtime tasks was determined. The idea is to add some attributes to the already existing `Thread` object to interact with the scheduler. This interaction is realized by a single class explained in the next main section.

Above, fair-share scheduling for non real-time threads was chosen. So for real-time threads priority based scheduling is not reintroduced but a time based scheduling paradigm is used. All a programmer of a real-time application knows is the designated *time constraint* of some periodic or sporadic task. The system should run out-of-the-box not requiring off-line feasability analysis of static scheduling policies. *Earliest Deadline First* (EDF) scheduling fits best these constraints. It's an efficient and easy to implement dynamic best-effort scheduling approach [11] enabling firm real-time behavior as shown in the following section.

2.4 Scheduler Implementation

To implement EDF scheduling a `deadline` value is associated with any `Thread`. A deadline is created at the point of time a thread enters a time-critical region relative to the current value of the millisecond counter. If no deadline is associated with a thread, it's handled as a thread with an infinite time constraint. So non-real-time threads become real-time threads with always lowest priority and the scheduler doesn't need to distiguish two kinds of threads.

To implement fair-share scheduling a `priorityCounter` is associated with any `Thread`. Every time the scheduler is called by the VM, the counter of all threads is incremented by its current Java `priority` (execution probability). So

Java priorities accumulate over time and threads with lower execution probability will also have a high counter value if not called for a long time.

Scheduling is done by two pass processing of the list of threads:

Pass 1. increment all priority counters, find the deadline that is in the nearest future and store the maximum counter value of the associated threads (additionally the nearest point of time to awake any *blocked by sleep* thread will be noticed here),

Pass 2. if any thread is *ready* find the first thread in the list matching the shortest deadline and maximum priority counter value.

Using this simple scheme, scheduling is done in three layers:

1. according to the EDF policy all *ready* threads with the earliest deadline are *runnable* and are priorized over all other threads which are *displaced,*
2. all *runnable* threads are passed to the fair-share scheduling policy,
3. *runnable* threads of the same execution probability are scheduled using the round-robin policy.

So the Java priority is a second order demand after deadlines. But scheduling using this model does not avoid priority inversion for its own.[3]

2.5 Priority Inversion Avoidance

The priority inheritance policy was implemented and used for deadlines in the first instance. Priority and deadline inheritance is performed on-the-fly by the scheduler. If a thread is entering the *blocked by monitor* state the thread holding the lock on the object is noted in the blocked thread. The scheduler also examines threads in the state *blocked*:

- If a deadline is associated with the blocked thread the scheduler is switched to the blocking thread and uses the minimal deadline of both threads for scheduling, so if any thread is blocking another thread with an earlier deadline it gets its privileges.
- The Java priority of blocked threads is also accumulated to blocking threads, so blocking threads get exactly the CPU time blocked threads release to the system unnoticable by unaffected threads.

In both cases this flow is recursive. As a side-effect the scheduler may detect dependency cycles and abort execution throwing a `DeadlockError`.

In addition a mechanism to improve responsiveness on events was implemented. If a thread changes from *blocked* to *ready* the priority counter is pushed to a high value considering the Java priority and it gets a probability boost. This mechanism doesn't affect deadlines.

[3] For example priority inversion occurs if a thread with an earlier deadline is *blocked* by a non-real-time thread with no associated deadline and there is another deadline later. The non-real-time thread is *displaced* until the later deadline is dismissed and probably the earliest deadline has already passed.

2.6 And What About Garbage Collection?

In many Java based real-time systems the garbage collector draws the attention because it compromises real-time behavior. Main problems with the garbage collector arise with priority scheduled threads. Because the garbage collector must be able to run everytime without blocking, it must have a high priority. Then the garbage collector decides when it runs and how often. But in most cases the garbage collector runs without knowledge of real-time scheduling requirements. So many real-time implementations use an incremental garbage collector (e. g. called after a `new`). But this slows down object allocation rate.

Using fair-share scheduling the garbage collector thread can run concurrently at low execution probability. So not the garbage collector decides when it runs but the real-time scheduler as for any other thread. The garbage collector is also involved in the priority and deadline inheritance scheme. If a thread is *blocked* because it runs out of memory the garbage collector gets its privileges until it completes. Other real-time and non-real-time threads not requiring memory allocations are running continuously. But also using this policy the execution time of real-time tasks may rise unpredictably on memory runout if the `new` operator is used inside a time-critical region. This must be kept in mind of the application programmer (use of prepare and mission phase, see next section).

Currently our garbage collector always runs with the same (lowest) probability. In some applications with huge memory allocation rate (e.g. `String` manipulations) affected threads may run frequently out of memory and block unavoidably until the garbage collector completes memory compaction. To reduce thread block rate in the future the garbage collector execution probability should be at least proportional to the memory allocation rate [12].

3 A Minimized Real-Time API

All real-time scheduling capabilities are only reclaimable if they can easily be used by the application programmer using an appropriate API[4]. As suggested above already existing threads are used and their scheduling attributes are changed at runtime. A naive approch is a pair of `static` methods in a `Thread` extending class to control the `deadline` attribute of the current thread. This concept works in principle but has a number of flaws:

1. it's up to the application programmer *always* to pair the instructions for entering and exiting time-critical regions,
2. it's up to the application programmer to check all invoked code inside the time-critical region for the recursive use of these instructions. If this is the case, the real-time behavior will be not as expected for both, the application code and the invoked code.

[4] The full API specification can be found in [13].

3.1 The Deadline Object

To implement an elegant solution to the stated issues, the `Deadline` object is introduced. It has the facility to access the private attributes of the current `Thread` using native code. To obtain a paired construction for entering and exiting deadlines real-time code is bound to synchronized code blocks.

Inside the virtual machine synchronization is performed by tagged methods or using the two bytecodes `monitorenter` and `monitorexit`. These bytecodes have the same problem as paired methods but in contrast the Java compiler takes care that these bytecodes are always paired. Furthermore, the `synchronize` construct has some security facilities. It is not possible to use the same object for synchronization at the same time by multiple threads. Recursive synchronizations on the same `Deadline` object have no effect because no further monitor is aquired. Recursive time constraints are possible using other `Deadline` instances. Using `Deadline` objects has a security effect itself, it's up to the programmer to publish the object to other classes or use it privately.

Changes inside the virtual machine are minimized to two code blocks representing entering and exiting a monitor (which are also used for synchronized methods). These routines were extended by a type check. If the synchronizing procedure was passed and the synchronized object is an instance of `Deadline` then equivalent code for entering and exiting an EDF region is called. On entering the `deadline` value is calculated from specified time constraint and current millisecond counter value. It's attached to the current thread object, the previous value is stored in the `Deadline` object. At exit the `deadline` is set to its previous value and the thread sleeps the remaining time until the current deadline expires.

The user interface usage of this scheme is outlined in figure 2. All code executed or invoked inside the synchronized block is bound to the specified deadline. One drawback can also be found in this scheme, since the `synchronized` command doesn't provide any possibility to throw an exception on a deadline miss. Instead this is performed by methods defined in `Deadline`. These methods behave similar to `Object#wait()` as an `IllegalMonitorStateException` is thrown in the case if they are not invoked from inside the dedicated synchro-

```
Deadline d=new Deadline(200);           // store time constraint
-> non-real-time code (prepare phase)
try {
    synchronized(d) {                   // set deadline after 200ms
      -> real-time code, max. 200ms (mission phase)
        d.check();                      // checks deadline, may throw dme
    }                                   // sleep until deadline expires
  -> non-real-time code
} catch(DeadlineMissException dme) {
  -> non-real-time code (emergency procedure)
}
```

Fig. 2. Oneshot task using `Deadline`

```
Deadline d=new Deadline(200);
-> non-real-time code (prepare phase)
synchronized(d) {                          // set deadline after 200ms
    for(;;) {
        try {
          -> real-time code, max. 200ms (mission phase)
            d.append(200);                 // appends new deadline, may throw dme
        } catch(DeadlineMissException dme) {
          -> real-time code (emergency procedure)
        }
    }                                      // continue, even after miss
}                                          // sleep until last deadline expires
```

Fig. 3. Periodic task using `Deadline`

nized block. This ensures that the deadline is in use. The `check()` command just compares the current millisecond counter value with the deadline and throws a `DeadlineMissException` if required. This enables a simple run-time check of deadlines if this is the last command before leaving the synchronized block. The application programmer can decide where the exception is catched, this can be either outside the time critical section (as shown in the example) or even inside, then the emergency code sequence is still executed under real-time privileges. The programmer may even decide to implement soft real-time tasks and simply omit the invocation of `check()`.

The example shows only an oneshot real-time task. Deadline objects can also be used for periodic (time-driven) and sporadic (event-driven) tasks just using special design patterns:

Periodic tasks. are designed by concatenating further time constraints to an existing deadline using the command `Deadline#append(..)` inside some loop statement (see figure 3). The value of the concatenated time constraints may differ from the one associated with the deadline object for modulated periodicity. `append(..)` works similar to exiting and re-entering a synchronized block, the thread sleeps until the previous deadline expires. Appended deadlines are not relative to the current time found in the millisecond counter but relative to the previous deadline. So there will be no period drifting. As deadlines are adjoined, one missed deadline is no problem for the entire sequence, further periods are able to catch up.

Sporadic tasks. are designed straight forward using the already existing `wait()/""notify()` scheme (see figure 4). Invoking `wait()` on some synchronized object, the objects monitor is released and the thread is put into the *waiting for event* state. In the case of a `Deadline` object the deadline is also released. Later if the `Deadline` receives its notification signal, it reacquires the monitor and immediately enters a new deadline using its default time constraint.

Of course time critical regions can also be specified by extending `Deadline` and using synchronized methods.

```
try {
    Deadline d=new Deadline(200);
    synchronized(d) {                    // set deadline after 200ms
        for(;;) {
            d.wait();                    // releases monitor and deadline
          -> real-time code, max. 200ms (mission phase)
        }
    }                                    // sleep until deadline expires
} catch(DeadlineMissException dme) {
  -> non-real-time code (emergency procedure)
}
```

Fig. 4. Sporadic task using `Deadline`

4 Proof of Concept

Applications. We have used our real-time scheduling scheme in our 8 MHz reference system not expecting too much scheduling throughput. So applications requiring only low rate real-time capabilities were implemented. For example one of our GUI components represents an analog clock. The second hand advance is implemented using a periodic task using a `Deadline` object with time constraints of 1000ms. This clock runs correct compared to a reference clock outside the system. Another GUI component using `Deadline` is a text scroller.

A more audible application is a iMelody player component.[5] Notes are played using the integrated PWM (pulse width modulation) controller of our reference system microcontroller. A periodic task with a `Deadline` object is used for note and pause durations in sequence. So `append(..)` is very useful in this case because it is able to fire many deadlined tasks in short time without memory throughput.

Finally, we are using `Deadline` in our intended system architecture: home-automation. Many devices are communicating using specified protocols.[6] Some protocols define communication time constraints, e. g. handshake exchange time-outs. So these time critical regions are assisted by our software.

Measurements. A critical part of a real-time system is the reaction time and its variation (jitter). To measure these key values our reference implementation kernel was exploited by additional native code. This code slightly affects the measurements by 48 μs per pass so this value is removed from the results. A Java test program was created doing several tests for 3 seconds each. Tests of three groups were done:

[5] iMelody is a text file standard of the Infrared Data Association (IrDA) for specifying melodies for one voice with note sequences [14], it's used by many mobile phones.

[6] We are using the FT1.2 protocol over RS232 serial communication for data exchange with power line modems. Currently we're implementing a high level protocol for use with the CAN bus.

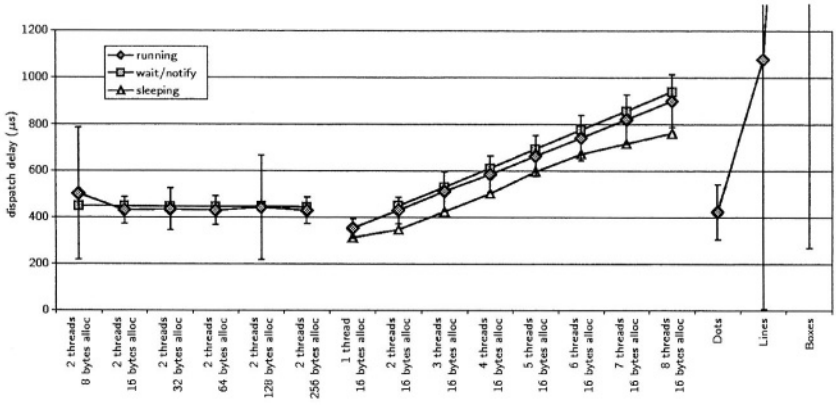


Fig. 5. Event processing delays

1. Memory usage (or garbage collector stress) using a fixed number of threads allocating arrays of different sizes in an endless loop. The garbage collector uses atomic memory block moves, larger blocks (arrays of 256 bytes) take 720 μs and are expected to affect event processing delay.
2. Thread scheduling costs using the same routine but instead changing the number of threads. Threads are processed in a linked list by the scheduler so event processing delays should correlate linearly.
3. Some basic graphical routines using random coordinates. The graphical routines are implemented using atomic native routines.

Figure 5 presents the results. Some tests were replayed using `wait()/notifyAll()` pairs or `sleep()`. Note that this graph doesn't show overall system performance but system reaction time. The results are further influenced by two other threads, the `main()` thread starting the tests and the system thread performing garbage collection. Some results are not as expected. The garbage collector utilization with larger memory blocks doesn't affect the dispatch delay in a predictable way (see deviation marks). Here the block moves are uncorrelated to the events and may be finished just in time before a scheduling initiating event. In contrast the number of threads scales as predicted because thread processing is performed within the scheduler and scheduling is correlated to the events. Most measured delays are within a millisecond so using this value as system timer was a good choice. Programming real-time applications special care must be taken if a graphical display is used then the delay can exceed 5 ms.

5 Conclusion and Future Work

In this paper we presented our real-time implementation for low memory Java systems. In the range of Java systems below 64 K, explicit real-time capabilities are almost unique. The implementation depends on the Java virtual machine because main parts of the system (scheduler, deadline-interface using synchronized

blocks) are integrated into the kernel. We introduced our three level dynamic best-effort policy using EDF, fair-share and round-robin scheduling with time slice. The scheduler is implemented as simple as possible using a single linked thread list. The scheduler uses each threads priority counter and deadline field and enables priority and deadline inheritance. The garbage collector is integrated into this concept.

The API of our real-time extension is very compact and consists of just one class (`Deadline`) and one exception. Our idea to bind real-time code regions to synchronized blocks enables compiler and VM assistance for consistency and security. Using special design patterns and methods of `Deadline` and `Object`, periodic and sporadic real-time tasks can be created and fired.

In the future the virtual machine will be redesigned to enable other target architectures than the ST7 microcontroller. The thread dispatcher and scheduler will be separated and the scheduling algorithm will be more configurable. It may be fully or partially implemented in Java and then integrated into the runtime system.

References

1. Böhme, H., Klingauf, W., Telkamp, G.: JControl – Rapid Prototyping und Design Reuse mit Java. In: 11. E.I.S.-Workshop, Erlangen. Volume GMM-Fachbericht 40., VDE-Verlag 2003 79–84 ISBN 3-8007-2760-9.
2. Bracha, G., Gosling, J., Joy, B., Steele, G.: The Java™ Language Specification. Second edn. Addison-Wesley 2000 ISBN 0-201-31008-2 `http://java.sun.com/docs/books/jls`.
3. Sun Microsystems: Java™ 2 Platform, Micro Edition (J2ME). `http://java.sun.com/j2me/`
4. RTJ Computing: simpleRTJ. `http://www.rtjcom.com`
5. Muvium: Muvium (PIC Java VM). `http://www.muvium.com`
6. Solorzano, J.: TinyVM (RCX Java), `http://tinyvm.sourceforge.net`
7. Real-Time Java™ Working Group: International J Consortium Specification, Real-Time Core Extensions. Technical report 2000 `http://www.j-consortium.org/rtjwg/`.
8. Bollella, G., et al.: The Real-Time Specification for Java. Technical report, The Real-Time for Java Expert Group 2002 `http://www.rtj .org`.
9. Schoeberl, M.: Restrictions of Java for Embedded Real-Time Systems. `http://www.jopdesign.com`
10. Tanenbaum, A.S.: Modern Operating Systems. Second edn. Prentice Hall 2001 ISBN 0-13-031358-0.
11. Mathai, J.: Real-time Systems Specification, Verification and Analysis. Prentice Hall 1996 ISBN 0-13-455297-0, `http://www.tcs.com/techbytes/htdocs/book_mj.htm`.
12. Nilsen, K.: Adding real-time capabilities to Java. Communications of the ACM **41** 1998 49–56 `http://www.acm.org/pubs/citations/journals/cacm/1998-41-6/p49-nilsen/`.
13. Böhme, H., Klingauf, W., Telkamp, G.: Jcontrol platform api specification, jcvm8 edition, `http://www.jcontrol.org/html/javadoc/`
14. The Infrared Data Association: imelody v1.2 specification, `http://www.irda.org`

Towards an Understanding of the Expressive Power of the RTSJ Scoped Memory Model

Andrew Borg and Andy Wellings

University of York, UK,
{aborg,andy}@cs.york.ac.uk,
http://www.cs.york.ac.uk/rts

Abstract. The Real-Time Specification for Java (RTSJ) introduces a memory model that includes immortal and scoped memory areas that are not subject to garbage collection latencies. Unfortunately, it is often argued that the RTSJ's memory model is unwieldy and even insufficiently expressive for describing the memory dynamics of simple and commonly used structures and patterns. In this paper we propose a simple approach to expressing known lifetime object information using scoped memory. By developing algorithms that give the best ordering for scoped memory based on this lifetime information we evaluate the expressive power of the RTSJ within this context. We use this evaluation to propose a minimal extension to the RTSJ and explain how a previous extension we have developed complements our programming model.

1 Introduction

The Real-Time Specification for Java (RTSJ) [1] introduces an alternative memory management model based on the concept of memory regions [2]. Besides the traditional heap, immortal and scoped memory areas are defined which are not garbage collected. Scoped memory regions provide the only mechanism by which applications that dynamically create arbitrary amounts of objects can free these objects when no longer required without suffering any latency from the garbage collector. Scoped memory as presented in the RTSJ has often been criticised for making the programming of trivial structures and patterns overly complex [3,4]. The majority of criticism levelled at the RTSJ memory model comes in the form of subjective arguments based on experience of using the model [5]. For example, it is known that dynamic collections can not be implemented using traditional algorithms as they would be developed in garbage collected or explicitly memory managed environments. The available solutions for implementations in the RTSJ include recycling items of the collection or making use of more complex patterns for using scoped memory [3]. The failure of the RTSJ is one of the complexity required to express known object lifetime knowledge. Capturing this concept in a non-subjective manner is not trivial. In this paper we attempt to do just this; we provide a framework which allows us to evaluate how the RTSJ allows the developer to express this knowledge. We introduce a simple programming approach called the *Region Partitioning Model* that we use to describe known

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 315–332, 2004.
© Springer-Verlag Berlin Heidelberg 2004

lifetime semantics of a program and evaluate the RTSJ within this context. We develop algorithms that show how object lifetime knowledge is best expressed in the RTSJ when using this model. We describe how a minor extension to the current RTSJ and a previous extension [6] can help to better express this lifetime knowledge. Nevertheless, we can show that for some programs where lifetime knowledge is expressed using the Region Partitioning Model, no scope assignment is possible without the RTSJ requiring infinite memory.

The rest of this paper is set out as follows: Section 2 provides the motivation for this paper where we argue that a memory model can be viewed as a way of expressing lifetime knowledge. The RTSJ memory model does exactly this and what we require is a way of capturing how expressive the model is. Section 3 introduces the Region Partitioning Model, our proposed approach to expressing know lifetime information that targets the RTSJ scoped memory model. Section 4 investigates how much lifetime information it is possible to express in the Region Partitioning Model and Section 5 proposes extensions to the RTSJ that would make it easier for an application to target. Finally, Section 6 briefly describes some of our current work related to this area and finally concludes.

2 Memory Models and Lifetime Expressibility

The dynamic use of memory in real-time applications has typically been frowned upon by real-time developers. This often leads to a practice in which memory allocation and deallocation is only carried out during non-critical parts of the application such as at initialization or in a pre-mission phase. The argument made for the necessity of using this approach is the high space or time pessimism of Dynamic Memory Allocators (DMAs) due to fragmentation [7]. DMA algorithms make use of techniques that provide a tradeoff between the degree of complexity of the algorithm and resultant overheads and the level of fragmentation. A large amount of research has been invested in DMA technologies and the resulting algorithms perform within reasonable bounds of pessimism and with acceptable levels of fragmentation. However, this tends to be true only when memory is managed explicitly by the application using *malloc()*- and free*()*-style functions. The introduction of automatic memory management (garbage collection), particularly when using object-oriented languages, has proven to be much harder to solve. The state-of-the-art garbage collectors [4] experience much higher space and time overheads than those of DMAs used in explicitly memory managed environments, albeit that there are obvious similarities in the strategies used to allocate objects of different sizes and those used to serve requests for varying-size memory blocks. There are two causes of this:

- There is an overhead involved in scanning for live objects that is not present when memory in managed explicitly. Real-time garbage collectors are incremental and also need to employ read and/or write barriers in order to maintain application integrity.

- The development process in automatic memory managed environments typically makes use of a larger number of objects which are often short-lived. This is perhaps a result of the convenience for the developer of not having to worry about memory issues.

Despite these two causes guaranteeing that the cost of real-time garbage collection will always be greater than that of explicit memory management, the popularity of Java has reignited interest in this area. Prior to the RTSJ's introduction, the only options available to real-time Java developers were to either preallocate objects before the execution of critical code and then switch off the garbage collector during execution of that code or to accept the overheads of a real-time collector. The approach taken by the RTSJ is based on the premise that the state-of-the-art in real-time garbage collection is still a way off from being suitable for real-world applications. The use of scoped memory pushes memory management back into the domain of the application developer but with a key difference to traditional explicit memory management characterized by *malloc()*- and *free()*-style functions: memory management is *coarse-grained* and *structured.*

2.1 The RTSJ Approach

The coarse-grain approach to memory management recognises the fact that, very often, groups of objects in an application exhibit *temporal aggregation.* This phenomenon occurs when a number of objects exhibit similar lifetimes during the execution of the program as a result of the semantics of the program. A grouping of such objects we term an *aggregate.* The main advantage of this approach is that the number of allocation and deallocation requests is reduced and therefore the use of a DMA is acceptable. The main disadvantage is the space overhead that arises either from retaining objects for longer than required because the lifetime of the objects in an aggregate is not exactly the same or because of the necessity of assuming the worst case path through the code.

The tradeoff between different memory management schemes can be seen as two opposing forces: input from the developer of known memory usage over time versus runtime space and time overheads. As shown in Figure 1, the result is a spectrum of mechanisms that range from garbage collection to explicit memory allocation and deallocation through to pre-allocated memory. In the latter case, expressing exactly where memory is going to be needed in the future and where it can be freed could make it possible for a DMA to have 0 fragmentation, and therefore no space or time overheads. The diagram shows the RTSJ performing as badly as coarse-grained memory management in the spatial domain but as well as coarse-grained memory management in the time domain. This shows the RTSJ to be more acceptable of higher memory requirements than time overheads. In a real-time environment, this compromise would be the expected choice.

The existence of aggregates is often inherent in a program. For example, generational garbage collection takes advantage of the fact that many objects exhibit a grouping pattern in their lifetimes that depends on their age. These

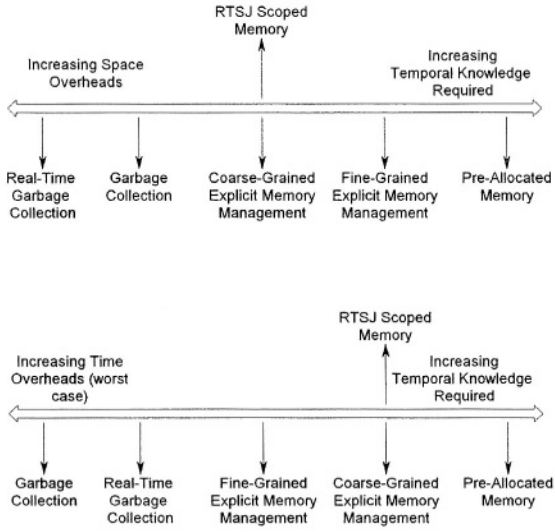


Fig. 1. A Spectrum of Memory Management Mechanisms

collectors can provide better performance by using this fact to selectively scan objects that would be most likely to become garbage. The futility of using synthetic applications to gauge fragmentation as demonstrated in [8] is proof of this. In a real-time environment, such approaches are less useful as it is still the worst case that needs to be considered and no assumption of the generational nature or spatial locality of the application can be made. The only alternatives to take advantage of temporal aggregation are therefore to either use compile-time analysis to extract information about the lifetime of objects [9] or to have the application developer identify aggregates explicitly. The RTSJ opts for the latter approach but specification of aggregates is done in a unique manner.

The most common approach to specify an aggregate is to make use of memory pools. Objects in an aggregate are placed in a pool which is explicitly allocated and freed. In the RTSJ, objects in an aggregate are placed inside a scoped memory region that is allocated and freed in a stack-like fashion that depends on the progress of the executing thread. This ordering enforces a *structure* that is maintained by means of the rules for scope trees and the reference rules for objects. In this way, explicit control of memory regions is no longer required. This approach is crucial for the RTSJ as it retains the safety of Java. However, two issues need to be considered:

- **What are the overheads of the underlying DMA when using this approach?:**
- **How easy is it to target the memory model?**

In this paper we focus on the item above (though we are currently undertaking research into the first issue). The ability of applications to target the RTSJ mem-

ory model is crucial to the success of the RTSJ. The success of automatic memory management algorithms is gauged by their average and worst case overheads over a large set of applications. Similarly, the success of a structured memory management approach is gauged by the model's ability to be targeted by a large set of applications. If a programming pattern does not fit into this structure then the result is an increase in programming complexity for the developer. Whereas the overhead in automatic memory management algorithms comes in the form of space and time overheads, in a structured memory model the penalty is a more complex development process as the application needs to be designed around the memory model. The goal of this paper is to provide a framework in which to evaluate the success of the RTSJ memory model in this context and thereby motivate a number of extensions to the model that make it easier for applications to target.

3 A Programming Paradigm for the RTSJ: The Region Partitioning Model

The paradigm we propose for applications to target the RTSJ is necessarily minimal. It is, however, a good representation of what we believe to be the intuitive approach RTSJ development should follow. The most intuitive way to express lifetime knowledge in the RTSJ is to divide the execution trace of a thread into a set of *regions* based on the aggregation of object lifetimes within that trace. We only consider single-threaded applications or multi-threaded application in which threads do not need to share objects. We call this approach to describing aggregates the *Region Partitioning Model.* A single-thread in an application is defined as a sequence of regions that each describe an aggregate. The region partitioning model explicitly restricts the use of alternative memory management techniques such as object recycling or using scoped memory to implement memory pools.

3.1 A Formalisation of the Region Partitioning Model

A single-threaded program P is made up of a number of a sequence of regions $P = r_1, r_2, \ldots, r_n$ which are executed by a machine in the given order. For some subsequence of P, $r_i, \ldots, r_j$ we use the short notation $r_{\overrightarrow{i,j}}$. Each of the regions creates a number of objects for which memory is required. The size of the memory required to allocate all objects in region r_i is denoted as $|r_i|$. All memory required to execute each region must be allocated at some point *before* the start of execution of that region. For a region r_i, the latest point in the program at which this memory can be allocated is just before r_i begins to be executed. This point is called the Latest Allocation Point (LAP) of r_i and is denoted as $LAP(r_i) = r_i^-$ where r_i^- is the point in the program just before the first instruction of r_i (and after the last instruction of r_{i-1} if $i > 1$). At some point during the execution of the program, the memory of a region r_i may be reclaimed if no objects created in r_i are required after that point. This point is called the Earliest Deallocation Point (EDP) of r_i and is denoted as

$EDP(r_i) = r_j^+$ where $j >= i$ and r_j^+ is the point in the program just after the last instruction of r_j (and before the first instruction of r_{j+1} if $j < n$). Note that the location between regions r_i and r_{i+1} refers to both the points r_i^+ and r_{i+1}^-. These locations between regions are the only places in which memory can be allocated and freed. No memory can be allocated or freed inside a region. We shall use the notation r_i^- and r_i^+ respectively when talking about the allocation and deallocation points of a region. We also define $<$ and $>$ to provide a total order on the region sequence in P and to include these inter-region locations. Therefore, for all regions r_i in a program $(i > 1)$, $r_{i-1} < r_{i-1}^+ = r_i^- < r_i$.

Given a Program P, and for each region r_i in P the EDP of r_i, and by assuming that the memory of r_i is allocated at r_i^- (i.e. $LAP(r_i) = r_i^-$), we can derive the minimum memory requirements at each point in the program and therefore the minimum memory requirements of the entire program. We denote the minimum memory requirements when executing r_i to be $MinMemReq(r_i)$ and can be found by summing the memory required by all regions between the start of the program and r_i and which can only be deallocated at some point after r_i has finished executing:

$$MinMemReq(r_i) = \sum\{|r_x| \; ; r_x <= r_i \wedge \; EDP(r_x) >= r_i^+\} \tag{1}$$

The mimimum memory requirements of P is therefore $Max(MinMemReq(r_i))$ for all r_i in P. We call this value the *MINMEM* value of *P:*

$$MINMEM(P) = Max(MinMemReq(r_i)) \;\; ; \forall r_i \epsilon P \tag{2}$$

MINMEM(P) is derived by assuming that the memory required to execute some r_i is allocated just before r_i begins executing and is deallocated exactly at $EDP(r_i)$. For some memory model M, this may not be the case if M cannot identify exactly the EDPs and LAPs of each region. Therefore, memory for some region r_i may need to be allocated at the start of an earlier region than r_i and deallocated at some point after the EDP of r_i. We call these points the Real Allocation Point (RAP) and Real Deallocation Points (RDP) of r_i respectively and denote these as $RAP^M(r_i)$ and $RDP^M(r_i)$. Modifying where memory is allocated and deallocated changes the memory requirements at each point in the program and therefore the minimum memory requirements of the whole program. Crucially, the invariant $RAP^M(r_i) <= LAP(r_i)$ and $RDP^M(r_i) >= EDP(r_i)$ must always hold. For some memory model M, we derive the real memory requirements at each r_i in P and then the real memory requirements of *P:*

$$\begin{aligned} RealMemReq^M(r_i) = \sum\{|r_x| \; ; & \\ & (r_x <= r_i \wedge RDP^M(r_x) >= r_i^+) \\ & \vee \\ & (r_x > r_i \wedge RAP^M(r_x) <= r_i^-)\} \end{aligned} \tag{3}$$

$$REALMEM^M(P) = Max(RealMemReq^M(r_i)) \;\; ; \forall r_i \epsilon P \tag{4}$$

Clearly, Equation (1) is a special case of Equation (3). The condition for adding $|r_x|$ to the memory requirements at r_i in Equation (3) resolves to the same condition as that in Equation (1) when for every r_x where $r_x <= r_i$, $RDP^M(r_x) = EDP(r_x)$ and $EDP(r_x)$ is still greater or equal to r_i^+. Also, for all r_x where $r_x > r_i$, $RAP^M(r_x) > r_i^-$. Note that the second part of this conjunction when applied to all regions r_i in P is simply the condition that validates the assumption that $RAP^M(r_i) = LAP(r_i) = r_i^-$.

It is trivial to prove that given a set of allocation and deallocation points for regions in a program *P,* delaying a deallocation point can increase the memory requirements of P but not decrease them. Similarly, allocating memory earlier than the provided allocation point can only increase the memory requirements of *P.* It therefore follows that Equation (2) does indeed give the minimum memory requirements of a program and this occurs when a memory model M can guarantee that for each region r_i in *P,* $LAP(r_i) = RAP^M(r_i) = r_i^-$ and $EDP(r_i) = RDP^M(r_i)$. Ensuring that this is the case is a sufficient but not necessary test to guarantee that using M will only require the minimum amount of memory necessary. The test is not necessary as some modification of allocation and deallocation points may increase the memory requirements at certain points in *P,* but not to an extent which is greater than *MINMEM(P).* An optimal test is simply to derive $REALMEM^M(P)$ and then compare the value to *MINMEM(P).* A fully expressive memory model is one that can express arbitrary aggregate lifetime during the execution of any program. Therefore, a memory model M is said to be *fully expressive* if for all *P,* $REALMEM^M(P) = MINMEM(P)$.

4 The RTSJ Memory Model for Finite Programs

In order to gauge the expressibility of the RTSJ we need to derive the semantics of the two methods that are used to describe memory allocation and deallocation: *enter()* and *executeInArea().* As our program definition has no stack representation to describe entering and leaving an invocation, these methods are symbolised by four annotations: **ENTER, EXIT, EIA-ENTER** and **EIA-EXIT** to describe respectively entering and exiting a memory area, starting execution in a memory area on an *executeInArea()* method and finally exiting when *executeInArea()* returns. The RTSJ defines the lifetime semantics of regions between the annotation pairs and sets rules that ensure the proper nesting of annotations. Once we have specified these semantics we seek to prove whether or not the RTSJ is fully expressive.

4.1 The RTSJ's Semantics

Given a sequence of regions $r_{\overrightarrow{i,j}}$, an annotation pair of **ENTER** and **EXIT** can placed in $r_{\overrightarrow{i,j}}$ so that if $S = r_{\overrightarrow{i,j}}$, S can be split into three so that $S = ABC$ and $A = r_{\overrightarrow{i,x}}$, $B = r_{\overrightarrow{x+1,y}}$ and $C = r_{\overrightarrow{y+1,j}}$ with an **ENTER** annotation

placed at r^{-}_{x+1} and an **EXIT** annotation at r^{+}_{y} and for which *A* and/or *C* can be empty. Alternatively, a new annotation can be added to a region sequence that already contains previous annotation pairs. In the former case, all regions in *B* have a RDP[1] at earliest at r^{+}_{y} and a RAP at latest at r^{-}_{x+1}. The regions that lie outside the **ENTER** and **EXIT** boundaries retain their original allocation and deallocation points. All regions are initialised with a RAP at the start of the program and a RDP at the end. In order to guarantee proper nesting in a partitioning, the RTSJ allows the addition of a new annotation to the temporal aggregation specification only if the new annotation pair encloses an equal number of previously existing **ENTER** and **EXIT** pairs. The semantics for such an annotation are as follows: All regions enclosed in existing annotation pairs retain their existing allocation and deallocation point bounds. The remaining regions form a set with each region having a RAP at latest at the location of the new **ENTER** annotation and a RDP at earliest at the location of the new **EXIT** annotation.

executeInArea() differs somewhat from that of *enter()* in that a specific memory area within the current scope stack must be identified as the location which is to become the memory context for executing the code passed in the `Runnable` parameter for this method. We use the notation **EIA-ENTER**^{N} where *N* is some integer ($N >= 0$) to show that the regions between the annotation pair are to be executed *N* positions up the memory stack. The example in Figure 2 illustrates how the index is used to identify the memory context for executing region r_4. The straight arrows represent **ENTER** and **EXIT** annotations with arrows of the same length representing a pair. The memory context to execute r_4 for different values of *N* are identified.

The semantics of *executeInArea()* are defined by the semantic meaning of what we described above as "executing in a memory context". Given an annotation pair **EIA-ENTER**^{N}/**EIA-EXIT** enclosing regions r_i to r_j, let *X* be the position of the **ENTER** annotation at index *N* and *Y* be the position of the pairing **EXIT.** For all regions r_x in $r_{\overrightarrow{i,j}}$, $RAP(r_x) = X$ and $RDP(r_x) = Y$. Note that unlike *enter(),* the semantics of *executeInArea()* set the RAP and RDP for a region exactly. This is because further annotation can not change these points.

4.2 Algorithms for Scope Ordering Using *enter()* and *executeInArea()*

The annotation process effectively aims to restrict the allocation and deallocation points of the regions between the **ENTER** and **EXIT** points. In order to evaluate the expressibility of this mechanism, we assume a program $P = r_{\overrightarrow{1,n}}$ for which every region has some arbitrary EDP. We shall prove that the RTSJ is indeed not fully expressive, that is there exists a program for which no annotation exists that will give $REALMEM(P) = MINMEM(P)$. The proof

[1] We shall henceforth suppress the RTSJ superscript. Therefore RDP implies RDP^{RTSJ}, REALMEM(P) implies $REALMEM^{RTSJ}(P)$, etc.

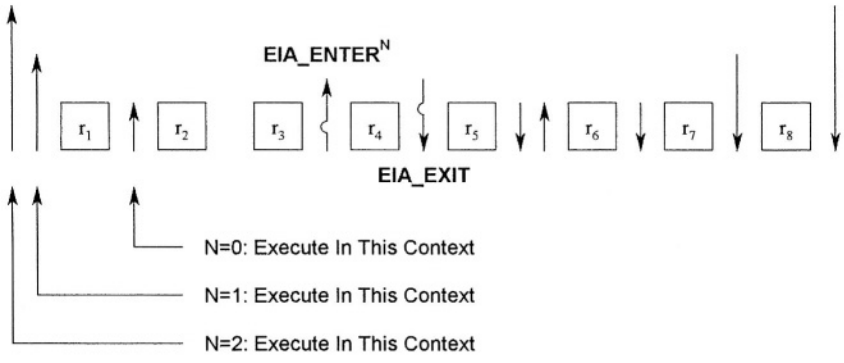


Fig. 2. Indexes for Identifying Memory Contexts in *executeInArea()*

is by construction: we provide an algorithm that gives the minimum possible real memory requirements of P that can be expressed using *enter()* and *executeInArea()* and showing that there exists a program for which this value is greater that *MINMEM(P)*.

The algorithm for minimising memory requirements when using *enter()* alone is trivial and is given in Algorithm 1. By placing an **ENTER** annotation just before r_i we can move the RAP point to r_i^-, the latest and optimum position. The RTSJ requires that the pairing **EXIT** be placed at some position r_j^+ where $r_j^+ > r_i$ such that all regions between r_i and r_j^+ have an EDP at r_j^+ at the earliest. We would always prefer to put this annotation at the first legal position that corresponds correctly with these semantics as this is the earliest point at which the enclosed regions can be deallocated. If $EDP(r_i)$ satisfies this constraint then the **EXIT** annotation is placed in this position. In this case, $RDP(r_i) = EDP(r_i)$ and r_i will not cause *REALMEM(P)* to be greater than *MINMEM(P)*. The algorithm proceeds until an annotation pair has been made for all regions. However, if some region r_j between r_i and $EDP(r_i)$ has an EDP at some point after $EDP(r_i)$ then it is no longer possible to place the **EXIT** annotation at $EDP(r_i)$. Instead it will have to be placed at the first point r_y^+ such that all regions between r_i and r_y^+ have an EDP at earliest at r_y^+. In this case, for all regions r_p where $EDP(r_i) < r_p < r_y^+$, $RealMemReq(r_p)$ is increased by $|r_i|$. If the point where *MINMEM(P)* occurs anywhere in the range of r_p then $REALMEM(P) > MINMEM(P)$.

We next develop an algorithm that uses both *enter()* and *executeInArea()* to find the best ordering for regions. Let the program $P = r_{\overrightarrow{1,n}}$ be a program with n regions all of which have some arbitrary EDP. Consider region r_1 with $EDP(r_1) = r_j^+$. An **ENTER** annotation is placed at r_1^- for which a pairing **EXIT** annotation must be found. Ideally this would be at r_j^+ so that r_1 could be deallocated at the earliest possible point. Let r_g be some region where $r_1 < r_g <= r_j$ and $EDP(r_g) = r_m^+$ and $r_m^+ > r_j^+$. Without *executeInArea()*,

Data : P: `Program` $= r_{\overrightarrow{1,n}}$: `RegionSequence`
Result : AnnP: `AnnotatedProgram` =AnnSeq: `AnnotatedRegionSequence`
foreach $r_i \epsilon P$ **do**
  Place **ENTER** in AnnP at r_i^- after any existing **EXIT** annotations
  r_j^+: `InterRegionLocation`=first position where
  $r_j^+ > r_i \wedge \forall r_x \epsilon r_{\overrightarrow{i,j}}, EDP(r_x) <= r_j^+$
  `/*Having found where to place the EXIT, we need to ensure correct nesting */`
  if $\exists$ *a number of* ***EXIT*** *annotations* r_j^+ **then**
    Add **EXIT** in AnnP at r_j^+ such that number of **EXIT**s before the new **EXIT**=number of **ENTER**s after r_i
  else
    Add **EXIT** in AnnP at r_j^+
  end
end
return AnnSeq=AnnP

Algorithm 1: AnnotateWithEnter(P): Algorithm for Minimising *REALMEM(P)* using *enter()*

Algorithm 1 would place the **EXIT** annotation at r_m^+. This will delay the deallocation of r_1 and increase $RealMemReq(r_x)$ where r_x represents all regions between r_1^+ and r_m^+ by $|r_1|$. The memory for r_g, however, can still be allocated and deallocated at the ideal positions and therefore r_g would not contribute to increasing $REALMEM^{RTSJ}(P)$. With *executeInArea()* we now have an alternative: The pairing **EXIT** for the **ENTER** at r_1^- can still be placed at r_j^+ whilst r_g is wrapped in an **EIA-ENTER**^{N}**/EIA-EXIT** annotation pair as represented by the curved arrows. We now need to find the ideal position in which to execute r_g by placing a new **ENTER** and **EXIT** annotation pair which will be indexed by **EIA-ENTER.** The ideal position is that which places the RAP as late as possible and before r_m and the RDP as early as possible after $EDP(r_m)$. Given that an **ENTER/EXIT** pair have been placed at r_1^- and r_j^+ respectively, a new legal annotation pair can be placed at best at r_1^- and r_m^+. The **EIA-ENTER** is then set to index this location. In this way, $RDP(r_g) = r_m^+$ which is ideal but $RAP(r_g) = r_1^-$. As shown in Figure 3, this will increase $RealMemReq(r_x)$ where r_x represents all regions between r_1^- and r_g^- by $|r_g|$. The memory for r_1, however, is allocated and deallocated at the ideal positions and therefore r_1 will not contribute to increasing *REALMEM(P).* The algorithm proceeds with processing of the regions in the resultant partitions. This example shows that there are two possible options, one of which must be selected. The algorithm for finding the minimal use of memory follows: simply choose that option which in increasing the respective memory requirements as described incurs the smallest increase (if any) to *REALMEM(P).*

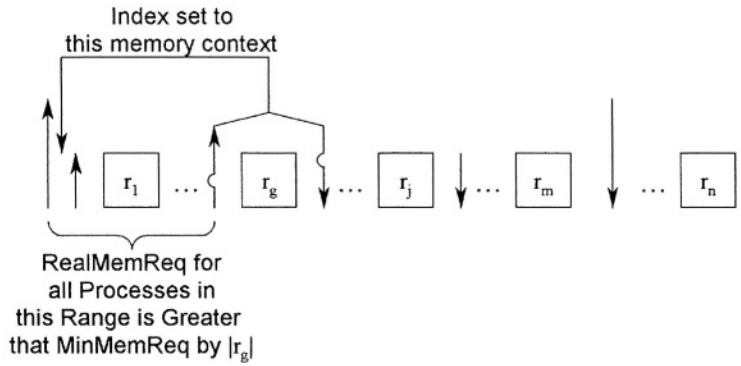


Fig. 3. An Annotation Solution using *executeInArea()*

The algorithm as described so far makes two assumptions: Firstly, there is only one region between r_1 and r_j^+ which has an EDP beyond r_j^+. Secondly, all regions between r_j^+ and r_m^+ have an EDP by r_m^+ at the latest, implying that an **EXIT** placed at r_m^+ is legal. We need to generalise the algorithm to remove these assumptions. The full construction of this algorithm is described in detail in [10] and for brevity we exclude here. Briefly, the algorithm first generates an *S set* for each region. For a given region, the EDP of regions in this set are the locations for which an **EXIT** annotation must be tested when processing this region. As each region is processed, all possible locations for a valid **EXIT** annotation are obtained using this set and each tested in turn. For each test, all regions that fall between the **ENTER** annotation of the region under investigation and the tested **EXIT** annotation must be wrapped in an *executeInArea()* and another **ENTER/EXIT** annotation pair added again for each of these regions. In the end, all possible annotations that could lead to the minumum memory requirements are derived and tested using Equation (4). The complete Algorithm is given in the Appendix in Algorithm 2. A complete listing with all sub-functions is given in [10].

4.3 How Expressive Is the RTSJ?

Having derived an algorithm that will give the annotation with the minimum possible memory requirements, we show that Algorithm 2 still fails to give an annotation for some programs whereby *REALMEM*(P) = *MINMEM*(P), even with the extra flexibility of *executeInArea()*. We choose that program in which every region r_i has an EDP at r_{i+1}^+ except for the last region r_n which has an EDP at r_n^+. We can show that in this case, *executeInArea()* provides some benefits over using *enter()* alone but is still not sufficient to reduce memory requirements to that of explicit coarse-grain memory management.

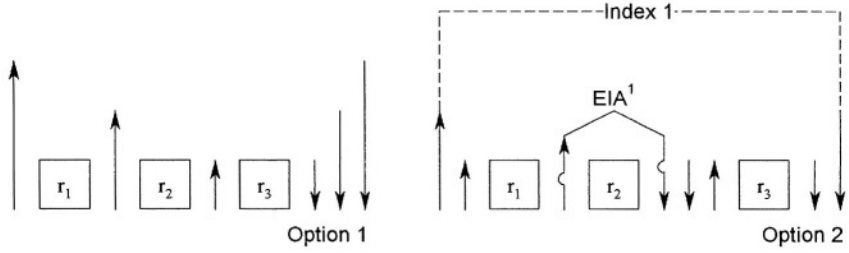


Fig. 4. 2 Possible Annotations from 3 Region Program

Let $P = r_{\overrightarrow{1,n}}$ and for all regions r_i where $r_1 <= r_i < r_n$ and $n >= 2$, $EDP(r_i) = r_{i+1}^+$ and $EDP(r_n) = r_n^+$. Trivially, if $n = 2$ then $REALMEM = MINMEM(P)$. If $n = 3$ then Algorithm 2 would generate two possible options. As shown in Figure 4, it is clearly the second option in the diagram that is most suitable as $REALMEM(P) = max(|r_1| + |r_2|, |r_2| + |r_3|)$. Therefore, for $n = 3$ it is also true that $REALMEM(P) = MINMEM(P)$. Consider the case when $n = 4$. There are 5 possible solutions that the algorithm would generate. The memory requirements for the five options are: $max(|r_1| + |r_2| + |r_3| + |r_4|)$, $max(|r_1|+|r_2|+|r_3|, |r_1|+|r_3|+|r_4|)$, $max(|r_1|+|r_2|+|r_3|, |r_3|+|r_4|)$, $max(|r_1|+|r_2|, |r_2|+|r_3|+|r_4|)$, and $max(|r_1|+|r_2|+|r_3|, |r_3|+|r_4|)$. In every case the result will always be greater than is possible with explicit collection as each solution has at least three consecutive regions with allocated memory at the same time. For values of n greater that 3, this program proves the inexpressibility of the RTSJ within the Region Partitioning Model.

Clearly, *executeInArea()* does provide some benefit over using *enter()* alone and it is important to understand where these benefits come from. With *enter(),* it is only possible to express lifetime knowledge exactly if the lifetimes of objects form aggregates with lifetimes ordered in stack order with respect to the execution of the thread. *executeInArea,* however makes it possible to arrange the order of regions to take advantage of this greater freedom. We have used Algorithm 2 to understand the best possible annotation for common patterns of region lifetimes. For example, for the program described above, we can show that if all regions have the same memory requirements $|r_i|$, then only $\lfloor log_2 n + 1 \rfloor$ memory regions of size $|r_i|$ are required.

5 Extensions for a More Expressive RTSJ

We have show that even with *executeInArea(),* it is still not possible to achieve $REALMEM(P) = MINMEM(P)$. This means that a program with regions with arbitrary EDPs will require more memory than would have been necessary with explicit coarse-grain allocation and deallocation. We propose a simple solution that will enable an application to achieve these minimum memory re-

quirements but which still enforces the scope structure as currently present in the RTSJ. In order to avoid introducing new methods, we propose a strengthening of the semantics of the constructors for scoped memory areas.

We have seen that when *executeInArea() is* used, the RDP of a region can be made to be closer to its EDP than if *enter()* alone is used. The cost of this is that the region's RAP was made to be earlier than the LAP of that region. If it is possible to provide a solution in which *executeInArea()* could be used to provide the proper nesting required to maintain the deallocation order of regions while at the same time delaying allocation until the start of the region, then the model could be equally expressive as explicit region allocation and deallocation. The nesting of regions could be arranged to concur with the order of deallocation using a combination of *enter()* and *executeInArea()* but the actual allocation of memory for regions would be deferred. This requires consideration of the semantics of *enter()* and *executeInArea()* when used together: Given an annotation pair **EIA-ENTER**^{N}**/EIA-EXIT** enclosing regions r_i to r_j, let X be the position of the **ENTER** annotation at index N and Y be the position of the pairing **EXIT.** For all regions r_x in $r_{\overrightarrow{i,j}}$, it remains true that $RDP(r_x) = Y$ but $RAP(r_x)$ becomes r_i^-.

In order to provide these semantics it is only necessary to distinguish between the two uses of *enter()* and *executeInArea().* Rather than introducing new methods that capture these semantics, we propose distinguishing between the two cases by the choice of constructor used when creating the scoped memory area instances. The RTSJ already allows the constructors for `VTMemory` and `LTMemory` to have an initial and maximum size specified when creating an instance of these classes. However, there is no requirement of when an implementation would physically allocate the necessary space. Therefore, by allowing the initial size to be 0 and then allocating the memory required for the region once the first object is created in the region on calling *executeInArea()* it is possible to obtain the new semantics described above. Nonetheless, as each invocation of *executeInArea()* and *enter()* requires stack space as well as an object representing memory areas, the RTSJ still incurs additional overhead. For the example used to prove the inexpressibility of the RTSJ in Section 4.3, even though the memory representing a memory region need not be allocated, a total of $\lfloor log_2 n + 1 \rfloor$ scoped memory objects are still required. Therefore for infinite n in a program in an infinite loop, the RTSJ will still run out of memory. This shows that, at least when using the Region Parititioning Model, the RTSJ does in fact fail to allow enough lifetime knowledge to be expressed so that the application does not run out of memory. This may be used to argue the necessity of an alternative or complementary memory approach such as memory pools.

In order to achieve the best nesting using these two new methods, we develop an algorithm that makes use of these new semantics. We assume any new memory area created using these semantics will have been created using a constructor that specifies with an initial memory size of 0. The algorithm is as follows: at each stage of Algorithm 2, the best annotation will always be that which chooses to use *executeInArea()* whenever possible rather than delaying an **EXIT** annotation.

The algorithm which gives this annotation whilst minimising the number of memory area instance required at one time can be found in [10]. We note in passing that what this extension provides is an escape from the strict nesting order of scopes required by the RTSJ. By removing this restriction it is possible to mark explicitly the beginning and end of an aggregate and therefore full expressibility is achieved. Nevertheless, even though the order of nesting of scopes is not constrained any longer, the scoping of the *enter()* and *executeInArea()* methods is maintained and ensures that every aggregate is marked explicitly with a RAP and RDP.

In [6] we introduced an extension to the RTSJ called Scoped Reference Objects. A Scoped Reference Object encapsulates a reference between two objects so that the RTSJ scoped reference rules are safely bypassed. We have developed Scoped Reference Objects to be used within the context of the Region Partitioning Model. The RTSJ reference rules may inhibit the developer from taking advantage of object lifetime knowledge if the direction of a reference does not coincide with the relative lifetimes of these objects. That is, if two objects O_1 and O_2 exist in two different aggregates and O_1 has a field that holds a reference to O_2, then either O_2 must be created in the same region as O_1 (using the *executeInArea()* or *newInstance()* methods) or the entire aggregate that O_2 forms a part of must be placed in a region with an EDP after the EDP of that region used to define O_1's aggregate. The result of either option is an added overhead, in the first case with O_2 potentially living longer than required and in the second case the entire region holding O_2 living longer than required.

Within the context of our memory model description, this problem can be described as a failure of the RTSJ to allow developers to express lifetime knowledge that is available. So far we have defined the EDP of a region by the knowledge of the relative lifetime of that region in the program's execution. However, the scoped reference rules redefine the EDP to also include the direction of references between objects. Scoped Reference Object remove this second condition so that it is the true lifetime knowledge of an aggregate that defines the EDP for that region's aggregate. The description in [6] shows how by decoupling the lifetime of an object from the references to that object from other objects, it becomes possible to implement common idioms such as collections without high memory overheads. In a collection, it is common that the items of a collection have a lifetime that is more closely related to a particular part of the program rather than the other items in the collection or the collection object itself. However, because of the reference rules, all the items in the collection and the collection object itself would have to gravitate into one region. Moreover, the items would tend to pull all the other objects in their respective aggregates into the same region. The result is that all available lifetime knowledge is lost and the resulting high overheads make the use of such idioms unfeasible. With Scoped Reference Objects however, this information can still be expressed and, by doing so explicitly, the safety of Java is also maintained.

6 Future Work and Conclusion

This paper briefly introduces our initial work into a detailed exploration of the RTSJ scoped memory that we are currently undertaking. The introduction of the region partitioning model in particular has allowed us to gain a much clearer understanding of how different patterns of memory usage are best expressed in the RTSJ. It also allows us to make stronger arguments for the extensions that we propose to the model. There are three key areas of research that we are focussing on at the moment. The first is an extension of the region partitioning model to describe multithreaded applications in which objects share scoped memory and which have threads that loop forever. The second is a study of how the concept of a region can be made a first class entity within the programming model rather than an abstract concept. Each thread in the application would then include a description of its region sequences and the decision of which region an object is created in becomes a cross-cutting concern. This approach creates a separation between the logic in an object's code and the use of scoped, tying the latter to the thread instead. Finally, we are extending this work to consider distributed systems and use the region partitioning model and RTSJ extensions in our research in this area [11].

In this paper we proposed a simple approach to describing how the RTSJ allows developers to express known lifetime aggregate and object lifetime knowledge. We have chosen as simple an approach as possible based on the argument that the RTSJ adopts a coarse-grained structured approach to dynamic memory management. We evaluated how lifetime knowledge would be expressed in the RTSJ using this programming model. An algorithm was developed that uses scoped memory to give the minimum possible memory requirements that can be expressed and this was shown to fail to be able to express known coarse-grained lifetime unless the lifetime order of these coarse-grain entities followed a strict ordering. We introduced a minimal extension to the RTSJ memory model that together with a previously developed extension [6] makes it possible to express known lifetime information without resolving to alternative memory management techniques.

Acknowledgements. The authors would like to acknowledge the contributions of Ron Cytron, Chris Gill and Angelo Corsaro from the University of Washington in St. Louis for some of the ideas expressed in this paper.

References

1. G. Bollella, J. Gosling, B. Brogsol, P. Dibble, S. Furr, D. Hardin, and M. Turnbull, "The Real-Time Specification for Java," 2000. [Online]. Available: www.rtj.org
2. M. Tofte and J. Talpin, "Region-Based Memory Management," *Information and Computation,* vol. 132, no. 2, pp. 109–176, 1997.

3. E. Benowitz and A. Niessner, "A Patterns Catalog for RTSJ Software Designs," in *On the Move to Meaningful Internet Systems 2003: OTM 2003 Workshops,* Z. T. e. a. Robert Meersman, Ed., vol. LNCS 2889. Springer, November 2003, pp. 497–507.
4. D. Bacon and P. Cheng, "The Metronome: An Simpler Approach to Garbage Collection in Real-Time Systems," in *On the Move to Meaningful Internet Systems 2003: OTM 2003 Workshops,* R. Meersman and Z. Tari, Eds., vol. LNCS 2889. Springer, November 2003, pp. 466–478.
5. E. Benowitz and A. Niessner, "Experiences in Adopting Real-Time Java for Flight-Like Software," in *On the Move to Meaningful Internet Systems 2003: OTM 2003 Workshops, Z.* T. e. a. Robert Meersman, Ed., vol. LNCS 2889. Springer, November 2003, pp. 490–496.
6. A. Borg and A. J. Wellings, "Reference Objects for RTSJ Memory Areas," in *On the Move to Meaningfull Internet Systems 2003: Workshop on Java Technologies for Real-Time and Embedded Systems,* vol. LNCS 2889. Springer, 2003, pp. 397–340.
7. J. Lindblad, "Reducing Memory Fragmentation," *Embedded Systems Engineering,* pp. 26–28, April 2004.
8. M. S. Johnstone and P. R. Wilson, "The Memory Fragmentation Problem: Solved?" *ACM SIGPLAN Notices,* vol. 34, no. 3, pp. 26–36, 1999.
9. B. Blanchet, "Escape Analysis for Java(TM). Theory and Practice," *ACM Transactions on Programming Languages and Systems,* vol. 25, no. 6, pp. 713–775, Nov. 2003.
10. A. Borg, "On The Development of Dynamic Real-Time Applications in the RTSJ - A Model for Expressing Dynamic Memory Requirements," University of York, UK, Tech. Rep. YCS-2004-379, June 2004.
11. A. Borg and A. Wellings, "A Real-Time RMI Framework for the RTSJ," in *Proceedings of the 15th Euromicro Conference on Real-time Systems,* July 2003.

Appendix: Algorithms for *executeInArea()* and Delayed Allocation

Data : P: `Program` $= r_{\overrightarrow{1,n}}$: `RegionSequence`
Result : AnnP: `AnnotatedRegionSequence`
SSets=GenerateSSets(P)
AnnP=GetBestSequenceAnnotation($r_{\overrightarrow{1,n}}$)
return AnnP

Algorithm 2: GetBestAnnotation(P: `Program`): Returns Best Annotation that minimises $MemReq_{LSD}^{RTSJ}(P)$

Data : Seq=$r_{\overrightarrow{i,j}}$: `RegionSequence`
Result : AnnSeq: `AnnotatedRegionSequence`
if *EDP(SSets[i][|SSets|])*=r_j^+ **then**
  /*$r_{\overrightarrow{i,j}}$ `is a chain` */
  AnnSeq=GetBestChainedSequenceAnnotation($r_{\overrightarrow{i,j}}$)
else
  /*$r_{\overrightarrow{i,j}}$ `is not a chain` */
  x: `Region`=RegionFromEDP(EDP(SSets[i][|SSets|]))
  chain: `RegionSequence`=$r_{\overrightarrow{i,x}}$
  y: `Region`=r_{x+1}
  AnnSeq=JoinChains(GetBestChainedSequenceAnnotation(chain),
  GetBestSequenceAnnotation($r_{\overrightarrow{y,j}}$))
end
return AnnSeq

Algorithm 3: GetBestSequenceAnnotation(Seq: `RegionSequence`): Returns Best Annotation for Region Sequence Seq

Data : Seq=$r_{\overrightarrow{i,j}}$: `RegionSequence`
Result : AnnSeq: `AnnotatedRegionSequence`
PossibleAnnotations: `Vector of AnnotatedRegionSequence`
for *count: int=1 to |SSets[i]|* **do**
  c: `AnnotatedRegionSequence`=Clone(Seq)
  Add **ENTER** in c at r_i^-
  Add **EXIT** in c at EDP(SSets[i][count])
  LastRegion: `Region`=RegionFromEDP(EDP(SSets[i][count]))
  EIASet: `Vector of`
  `Region`=$\{r_x; r_i < r_x <= LastRegion \wedge EDP(r_x) > EDP(SSets[i][count])\}$
  NextRegionSequence: `RegionSequence`=$r_{\overrightarrow{i+1,LastRegion}} - EIASet$
  if *(NextRegionSequence)* $\neq \emptyset$ **then**
    F: `AnnotatedRegionSequence`=
    GetBestSequenceAnnotation(NextRegionSequence)
    ReplaceAnnotation(c,NextRegionSequence,F)
  end
  if *(EIASet)* $\neq \emptyset$ **then**
    PossibleAnnotations += DoEIA(r_i,EDP(SSets[i][count]),c,1,EIASet)
  end
  AnnSeq=SelectBestAnnotation(PossibleAnnotations)
  return AnnSeq
end

Algorithm 4: GetBestChainedSequenceAnnotation(Seq: `RegionSequence`): Returns Best Annotation for Chained Region Sequence Seq

Data : start=r_i: `Region`
Data : last=r_j^+: `InterRegionLocation`
Data : c: `AnnotatedRegionSequence`
Data : nesting: `int`
Data : EIASet: `Vector of Region`
Result : AnnSeqVector: `Vector of AnnotatedRegionSequence`
P: `Vector of Region`=SortByEDP(EIASet)
r_p:Region=P[1];
TestForr_p: `Vector of Region`=SSets[p];
AnnSeqVector: `Vector of AnnotatedRegionSequence`
for *count:* `int`*=1 to* $|TestFor r_p|$ **do**
  currentEDP: `InterRegionLocation`= EDP(TestForr_p[count])
  newc: `AnnotatedRegionSequence`=Clone(c)
  ToWrap: `Vector of`
  `Region`=$\{r_x; r_x \epsilon EIASet \wedge EDP(r_x) <= currentEDP\}$
  Add **ENTER** in newc at r_i^- (outmost position if other **ENTER**s exist)
  Add **EXIT** in newc at currentEDP (outmost position if other **EXIT**s exist)
  ToWrap: `Vector of`
  `Region`=$\{r_x; r_x \epsilon EIASet \wedge EDP(r_x) <= currentEDP\}$
  $\forall r_y; r_y \epsilon ToWrap$, wrap in **EIA-ENTER/EIA-EXIT** with index *nesting*
  all r_y
  NewProblematicRegions: `Vector of`
  `Region`=$\{r_x; r_j^+ < r_x < currentEDP \wedge EDP(r_x) > currentEDP\}$
  r_{pcur}: `Region`=RegionFromEDP(currentEDP)
  RemainingAnnotation: `RegionSequence`=$r_{\overrightarrow{j+1, pcur}}^-$
  NewProblematicRegions)
  if *(RemainingAnnotation)* $\neq \emptyset$ **then**
    F: `AnnotatedRegionSequence`=
    GetBestSequenceAnnotation(RemainingAnnotation)
    ReplaceAnnotation(newc,RemainingAnnotation,F)
  end
  RemainingProblematicRegions: `Vector of Region`= EIASet - ToWrap + NewProblematicRegions
  if *(RemainingProblematicRegions)* $\neq \emptyset$ **then**
    AnnSeqVector +=DoEIA(start, currentEDP, newc, nesting++, RemainingProblematicRegions)
  else
    AnnSeqVector +={newc}
  end
end
return AnnSeqVector

Algorithm 5: DoEIA(start: `Region`, last: `InterRegionLocation`, c: `AnnotatedRegionSequence`, nesting: `int`, EIASet: `Vector of Region`): Return Possible Annotations when applying *executeInArea()* on c

Memory Management Based on Method Invocation in RTSJ

Jagun Kwon and Andy Wellings

Real-Time Systems Research Group
Department of Computer Science
University of York
York, England
{jagun, andy}@cs.york.ac.uk

Abstract. In this paper, we present a memory management model for the Ravenscar-Java profile. Because of the complexity and run-time overheads in verifying the proper use of the RTSJ's scoped memory, it is unfavourable in the area of high integrity systems where any unpredictability must be cast out. Our approach maps one anonymous memory area to a user-specifiable method by means of our Java 1.5 annotation types. This straightforward model eliminates single parent rule checks and simplifies other run-time checks that are the main cause of unpredictability and overheads. In fact, it also makes the programmer's job easier since he/she does not have to worry about creating and maintaining memory areas. All the annotated methods will be automatically converted by a transformer into an RTSJ/Ravenscar-Java compliant version. The semantics of the RTSJ remains the same, meaning that any program in our model when transformed is also a legal RTSJ program. Our key contribution is the definition of a predictable memory model and guidelines that will reduce/eliminate run-time overheads. A bonus to this is a less complicated programming model.

1 Introduction

Programmers do not want to be distracted by avoidable system-level activities, such as memory allocation/de-allocation. The RTSJ has added the notion of scoped memory that is intended for a more predictable execution of real-time Java programs. This can become an unnecessary burden to the programmer as well as to the underlying virtual machine. Every time an object in a memory area is assigned to a different memory area, that assignment has to be checked at run-time to prevent any dangling references. Programs are also more difficult to analyse with the additional features since, for example, memory areas can be nested and thus illegal nestings may result.

Moreover, programs can become inefficient in terms of their use of memory. Because now memory management is up to the programmer, he/she may wrongly decide to keep some unused memory areas active for a prolonged time, in which other components may be starving for more memory space. Portability and maintainability of RTSJ programs can also become problematic. On the whole, the new memory management model can introduce more programming errors and reduce the run-time performance of the whole system. However, we still want our programs to be predictable, i.e., we would like to know when memory will be allocated/de-allocated, and the overheads of such operations.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 333–345, 2004.
© Springer-Verlag Berlin Heidelberg 2004

As a countermeasure to this problem, we propose here a programming utility that makes use of the original RTSJ's scoped memory to iron out some of the aforementioned concerns of Java in real-time systems. Novel to this utility is a memory model that matches one anonymous memory area to one user-decidable registered method. Programmers need no knowledge about what and how memory areas are used. The utility will automatically create a memory area for a registered method and execute that method as well as all other *non-registered* subsequent method calls that the method invokes in the memory area. Only when the registered method returns, the memory area will be reclaimed and freed (this acts as if each registered method has its own stack for all objects it creates). For this reason, the single-parent rule of the RTSJ no longer needs to be checked.

Programmers have the freedom to fine-tune memory usage of the program by executing certain methods in a new memory area, others in an existing one (i.e., caller's memory). For example, a small and infrequently invoked method can choose not to have its own memory area, but to depend on its caller's memory, so that the overhead of creating and finalizing a memory area is kept to a minimum at the expense of some additional memory space in the caller's memory area.

Assignment checks can also be reduced or completely eliminated if the programmer follows our guidelines presented in this paper. With an extensive analysis, however, such guidelines can be relieved. An analysis tool may be developed such that it checks escapement of references through method calls, local objects, and returned objects. References to outside objects[1] can be passed to the current method as parameters to that method, or as internal objects defined within the enclosing object that the method belongs to. Class objects or any other shared objects can also be assigned with a local reference. All these cases can be checked by syntactically analysing the logic of each method. Yet, this can be greatly simplified if our guidelines are in position. As outlined above, our key contribution is the definition of a predictable memory model and guidelines that will reduce/eliminate run-time overheads. A bonus to this is a less complicated programming model.

This paper is structured as follows: Section 2 will briefly review the Ravenscar-Java profile and issues related to the RTSJ. Next, we will introduce our proposed approach in detail in Section 3. This section will cover the key features, our annotations and an example. Issues on the implementation and design of our tool will be discussed in Section 4, before we briefly review some related works. Conclusions will be drawn at the end.

2 Brief Review of the RTSJ and Ravenscar-Java Profile

2.1 Ravenscar-Java Profile

In recent years, there has been a major international activity, initiated by Sun, to address the limitations of Java for real-time and embedded systems. The *Real-Time Specification for Java* (RTSJ) [2] attempts to minimise any modification to the

[1] By outside objects, we mean objects that are not created within the current method and memory area pair.

original Java semantics and yet to define many additional classes that must be implemented in a supporting virtual machine. The goal is to provide a predictable and expressive real-time environment. This, however, ironically leads to a language and run-time system that are complex to implement and have high overheads at run-time (Sources of run-time overhead include interactions between the garbage collector and real-time threads, assignment rule/single-parent rule checks for objects in different memory areas, and asynchronous operations). Software produced in this framework is also difficult to analyse with all the complex features, such as the asynchronous transfer of control (ATC), dynamic class loading, and scoped memory areas.

Following the philosophy of the Ravenscar profile for Ada [4], we have proposed a high integrity profile for real-time Java (called Ravenscar-Java [8]) along the lines of the set of software guidelines produced by the U.S. Nuclear Regulatory Commission (NRC) [6]. This restricted programming model (or a subset of Java and RTSJ) offers a more reliable and predictable programming environment by preventing or restricting the use of language features with high overheads and complex semantics. Hence, programs become more analysable in terms of timing and safety and, ultimately, become more dependable. The profile is intended for use within single processor systems.

The computational model of the profile defines two execution phases, i.e. *initialisation* and *mission phase.* In the initialisation phase of an application, all necessary threads and memory objects are created by a special thread *Initializer,* whereas in the mission phase the application is executed and multithreading is allowed based on the imposed scheduling policy. There are several new classes that will enable safer construction of Java programs (for example, *Initializer, PeriodicThread,* and *SporadicEventHandler),* and the use of some existing classes in Java and RTSJ is restricted or simplified due to their problematic features in static analysis. For instance, the use of any class loader is not permitted in the mission phase, and the size of a scoped memory area, once set, cannot be altered. Objects that do not need reclaiming should be allocated in the immortal memory (thus in the initialization phase). For further restrictions, see [8].

2.2 RTSJ's Memory Model

There is one major concern with the use of the RTSJ's memory model in high integrity applications, that is, the runtime overheads of checking the single parent and assignment rules are often unpredictable and thus undesirable. A number of approaches have been proposed to reduce the overheads of dynamic checks, but we claim in this paper that such expensive operations can be completely eliminated by a few well-defined programming guidelines and a program transformer or virtual machine support.

2.3 Relationship to the Profile

The work presented in this paper should be regarded as an optional extension to the Ravenscar-Java Profile. In particular, it relaxes one of the rules, i.e., "no nested scoped memory areas" constraint.

3 Scoped Memory Method Invocation

3.1 Motivation

Because in Java every object is allocated in the heap and reclaimed by the garbage collector, one does not have to care about allocating and freeing memory space for new objects. However, its unpredictability and possible interferences with real-time threads means that it is not a convincing option for hard real-time applications.

The RTSJ has added the notion of scoped memory to solve this problem. Due to the vast amount of additional features, however, a very complex programming model can be created and, especially, memory areas can be utilized in an erratic way. That means the virtual machine must check for misuses of memory areas at run-time, i.e., single-parent rule and assignment rule checks. The overheads incurred by such operations are high and unpredictable, and extremely undesirable in resource-sensitive applications. This may also hinder portability, maintainability, and analysability of RTSJ programs.

3.2 The Idea in a Nutshell

In many imperative and procedural languages, programs are organized into blocks of code that are called *functions, procedures,* or *methods* in the case of Java. Each method has a functional purpose, such that it takes an input, processes it, and returns a result. A chain of method calls is constructed at runtime that can be represented as a stack, as shown below. In most cases, parameters that are passed to a method are only read and used in the production of the result. Each method has its own stack to store parameters, immediate results, and any local/temporary variables. Any other objects are allocated in the heap in Java by default.

This model can be extended to take on board scoped memory in a way that the program becomes simpler to analyse and run-time checks are minimized. A method can be associated with an anonymous memory area created by the run-time, and that memory area is entered whenever that method is invoked. All objects will be created within the current memory area, and the area will be reclaimed when the method returns, in the same way as the method stack is allocated and freed.

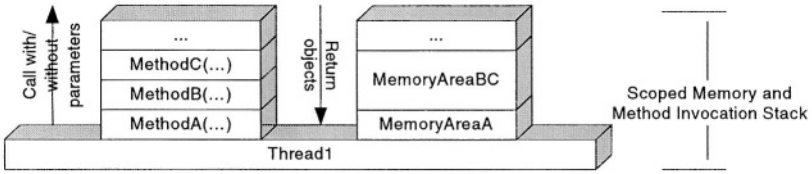


Fig. 3.1. Scoped Memory and Method Invocation Stack

Programmers can have the freedom to fine-tune memory usage of the program by executing certain methods in a new memory area, others in an existing memory area (i.e., caller's memory area). For example, a small and infrequently invoked method can choose not to have its own memory area, but depend on its caller's memory, so that the overheads of creating and finalizing a memory area is kept to a minimum at

the expense of some additional memory space in the caller's memory area (see Figure 3.1 above).

In order to associate a method to a memory area, we use annotation types that take advantage of the Java 1.5's annotation facility [10] (See Figure 3.2 below for an illustration). Memory areas are created anonymously, so that programmers cannot gain a direct access to them. This prevents misuses of memory areas; especially, the single parent rule check is no longer required at run-time by the virtual machine. Illegal assignments can also be checked using a static Escape analysis technique [1]. We will explore these matters further in section 3.4 and 3.5.

```
…
    @ScopedMemoryMethod(   size = 10000,
        isAssignmentCheckRequired = false,
        allocateCalleesReturnedObject = true,
        allocateCalleesExceptionObject = true
        reuseMA = false)
    public anObject complexCalculation(int param) {
        @ReturnedObject(type = anObject)
        anObject rtn = new anObject(…);
        …
        for (int j = 0; j>param; j++) {
            rtn.result = method1(j));
        }
        return rtn;
    }
```

Fig. 3.2. Annotating a method to associate with an anonymous memory area

Objects that are to be returned to the caller can be allocated in the caller's memory area, if such objects are identified and annotated in advance[2] (see `@ReturnedObject` annotation in the above example). That way, when the current method returns, the returned object need not be copied back to the caller's memory area, saving valuable computation time.

3.3 Key Features

In this model, threads have no knowledge about memory areas (except the immortal memory). Each method that has an associated memory area acts as if every object created in that method is stored in the method's local stack, and they are freed after the method returns. Below are the key features of our model.

- One to one matching between a scoped memory area (MA in short hereafter) and a user-specifiable method. There is no thread-oriented memory area stack.
- An MA object can be reused after the method returns or simply be reclaimed. Programmers can specify this requirement in the annotation. When the same

[2] Programmers have the knowledge on which object is to be returned to the caller method. Such objects can be annotated, so that they will be allocated within the caller's memory area. This saves copying objects between memory areas. But, of course, the size of a returnable object must be bound.

method is invoked next time, it will be given the same anonymous MA if its reuse flag is set to true; otherwise, the MA object itself will be reclaimed.

- A method can have its own MA to use, or choose to be dependent on its outer/caller method's MA. E.g., an infrequently called method can choose not to have its own memory area.
- MAs are all independent of each other and invisible at the source code level. No MAs will be entered by more than one thread/method at the same time. In fact, threads do not know which MAs they enter (i.e., they are invisible and anonymous!). For this reason, the single-parent rule in RTSJ is no longer required.
- Programs can either be transformed into a RTSJ compliant version that utilizes scoped memory, or a virtual machine can be developed, so that it takes care of creation, allocation, and reclamation of MAs at run-time.
- Referencing objects in the same method (this MA) does not cause any concern.
- Any references to objects created in the caller methods (thus outside this MA) is allowed. Such references can be passed as parameters to the method or as local objects of the enclosing object that the method belongs to. However, references to objects in a *callee* method are not allowed. This must be analysed.
- Illegal references can be checked by a method oriented analysis tool that incorporates the escape analysis.
- Objects in the immortal memory can always be referenced, whereas those in the heap cannot. This is to prevent any interactions between the garbage collector and real-time threads.

3.4 Single Parent Rule Checks

In the RTSJ, it must be checked that a thread does not enter a scoped memory area that is already active. In other words, if a thread has entered memory area A, B, and C in sequence, it cannot enter A again and allocate some objects that can refer to objects created in B and C, because the lifetime of A is in fact longer than B and C, and dangling references can be created.

In our model, threads simply cannot obtain references to memory areas, and all memory areas are anonymous. Even if a thread wants to enter the same memory area twice, there is no way it can do that! All the memory areas will be created internally by the virtual machine when a registered method is invoked. This will completely eliminate runtime overheads for checking the single parent rule.

3.5 Assignment Violation Checks

Assignment checks fail if and only if the following condition is satisfied in our model.

> *"A reference to an object within the current MA-registered method is assigned to another object that resides in a MA-registered* ***caller****."*

Assignment Violation Condition

More specifically, there are the following two cases.

Case 1. Direct Assignment of a method-local object to outside objects. It is fine if the enclosing object and the method share the same memory area; there is no need for any check. But if the method has entered a new memory area, then this is an illegal operation and a dangling reference can result when the method returns. The following example illustrates this point.

```
public class aClass {
    private anObject outsider;
    @ScopedMemoryMethod( … )
    public anObject aMethod(int param) {
        anObject insider = new anObject();
        outsider = insider;     // *** Assignment violation ***
        …
    }

    public anObject methodCaller(int param) {
        outsider = aMethod(param);
      …
    }
}
```

Fig. 3.3. Direct Assignment of a method-local object to an outside object

As shown above in *aMethod(),* when *insider* is assigned to *outsider,* an assignment violation occurs. This can be avoided if *outsider* is declared as *final,* so that the reference cannot be modified in methods. Alternatively, the method can create a local copy of the object, use it, and return it as a *ReturnedObject* mentioned earlier. If annotated by the *ReturnedObject* type, that object will be created in the parent MA. This means that the returned object shares the same memory scope as the object-local object. However, this is only possible when the parent method (the one that invokes *aMethod* in the first place, i.e., *methodCaller*()) and the object *outsider* are in the same scope.

Case 2. Indirect Assignment/Escapement as a parameter to a method call. If this *callee* method does not have its own memory area, then assignments are allowed unless that method invokes another one with a memory area associated with it, and passes the object to that method. The passed object and its members must only be read or '*used*' by all the callees, but not any member objects to be 're-*defined*' or assigned a new local reference by invoking a member method – this is an assignment violation. All subsequent method calls must be checked until the object in question is not passed any more to other methods and objects. Therefore, if an object can change any of its member objects' references by invoking a method, that method must share the same memory area as that of the enclosing object. Otherwise, an illegal reference can be created.

In short, it is best to declare any escapable objects as *final.* However, this seems too restrictive in some cases. Therefore, a static escape analysis can be performed to identify only those that are indeed assigned with references to method-local objects in different memory areas. The Static Single Assignment (SSA) form [5] is useful here since a program in that form can reveal whether a variable or object is *defined* (or assigned to a new value). So, first we need a graph of method invocations by conducting an escape analysis, and the SSA form representation of code for each of

```
public class aClass {
    private anObject outsider;

    @ScopedMemoryMethod( … )
    public void aMethod(anObject param) {
       anotherMethod(param);
        …
    }

    @ScopedMemoryMethod( … )
    public void anotherMethod(anObject param) {
      anotherObject x = new anotherObject();
      …
        param.xSet(x);        // *** Assignment violation ***
    }
}
```

Fig. 3.4. Indirect Assignment/Escapement as a parameter to a method call

the methods. Going through each method, a tool can examine whether the parameter's assignment counter, as well as the parameter's member objects' counters, ever becomes greater than 0. If the counter goes up to 1, it means that the parameter or some member object is modified and the assignment will raise an exception at runtime. Therefore, a method-level static escape analysis can identify illegal assignments statically.

3.6 Dealing with Returned Objects

There are situations where an object must be returned to a distant caller in a series of method invocations. When the methods do not share a memory area, such a returned object has to be relayed back to the original caller's memory area and its reference to become available for the original caller method. When the chain of methods is long, and there are more than two memory areas involved, overheads can be high due to copying the returned object from area to area. In order to resolve this, we propose an additional parameter that can specify where to store a returned object from the beginning of a method call (this parameter is added in the **`@ScopedMemoryMethod`** annotation; see the next section for details).

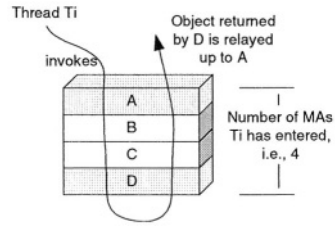


Fig. 3.5. Returned objects can be stored in one of parents' memory areas

As an illustration, assume that a thread has invoked a number of methods, say A, B, C, and D, which all have their own memory areas (see Figure 3.5 above). The

returned object from D can be relayed back to A, and in which case that object can be stored in A's memory area from the very beginning of D. This means that the object will not be copied back to C, B, and then A at run-time, reducing the overheads of copying objects. All returned objects must be annotated with `@ReturnedObject` to take advantage of this feature. Assignment checks are still required since a returned object can acquire a reference to an outside object, e.g., an object in D, C or B. Since the returned object is identified by the programmer, it can be checked by our tool that such objects are not allocated with any method local references.

This solution takes care of conditional invocations of methods. When there is an 'if' statement and two different sets of methods to call with different requirements on returned objects, we have a way of specifying that a returned object from one conditional branch will be stored in which method's memory area. Consider Figure 3.6 for a basic mode of operations.

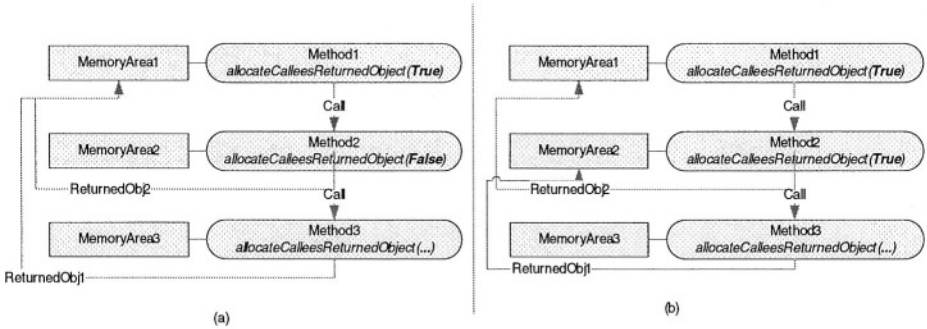


Fig. 3.6. Different ways of dealing with returned objects

The new attribute *allocateCalleesReturnedObject* is by default set to true (b), meaning that any returned object from a callee will be stored in the direct caller method's memory area. If it is set to false (a), any returned object will not be stored here, but in a memory area up in the chain where this attribute is set to true.

An interesting problem occurs, however, if both branches invoke a number of different methods, and eventually one common method that returns an object, as illustrated in Figure 3.7 below. When one branch requires the returned object to be stored in somewhere other than the memory area that the original conditional branch instruction belongs to, an appropriate *allocateCalleesReturnedObject* attribute can be set to *true* or *false,* depending on where we want the object to be stored. In case 1 in Figure 3.7, the returned object will be created in MA1 from the beginning, whereas in case 2, it will be Method5's memory area that the object will be created and stored.

The size of every returned object must be bound in order for our model to work. It becomes impossible to analyse and specify the size of a memory area if an arbitrarily sized object can be returned. Returned objects must be annotated, so that the transformer or an analyser can check the type and the caller memory area's size. It also has to be checked that a returned object does not take any method local references with it. This can cause an assignment check violation as explained in the previous section.

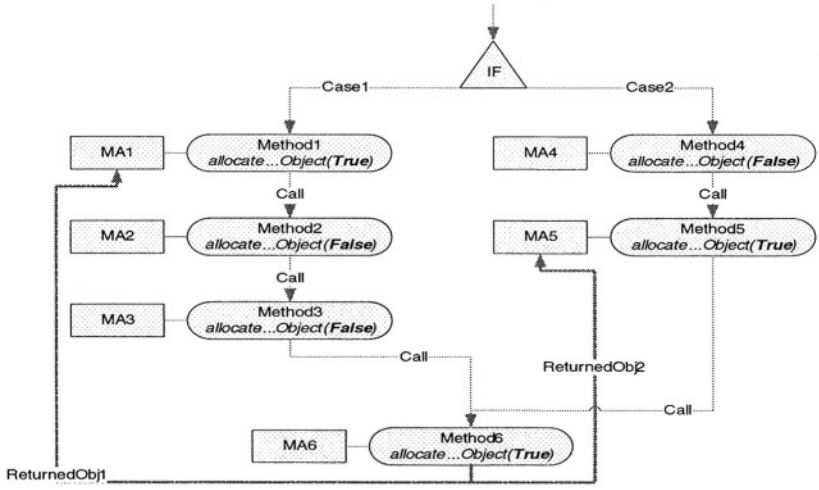


Fig. 3.7. Dealing with returned objects: conditional branches with a common method call

In some cases, although an object is not syntactically returned to a caller (with a return statement), it can eventually have the same effect. If a MA-associated method needs to allocate an object in one of the callers' MAs, then that object should also be annotated with **@ReturnedObject** annotation. This can prevent illegal assignments.

3.7 Dealing with User Exceptions

Exceptions are represented as objects in Java. A new exception object is created at runtime when a particular exception is raised. Similar to returned objects in the previous section, exception objects should be treated in the same way, so that we can save copying exception objects between memory areas when exceptions can be propagated or relayed. We have added a new attribute in **`@ScopedMemoryMethod`** annotation type, *allocateCalleesExceptionObject,* to cater for such situations. Exception objects should also be annotated with **`@PropagatedException`** annotation to take advantage of this feature, which is defined in the next section.

An analyser or the transformer will have to check if an annotated exception is indeed handled in the method whose *allocateCalleesExceptionObject* is set to true.

3.8 Annotation Types and Programming Procedures

We make use of the annotation facility of Java 1.5. Our annotation types are declared in the following way and illustrated below with an example.

ScopedMemoryMethod
This annotation type is used to declare that the subject method will be invoked with an associated memory area. The size of a memory area must be specified; all other properties have default values.

```
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.METHOD})
public @interface ScopedMemoryMethod {
     long size();                    // Required size of memory area
     boolean isAssignmentCheckRequired()          default true;
     boolean allocateCalleesReturnedObject()      default true;
     boolean allocateCalleesExceptionObject()     default false;
     Boolean reuseMA() default false;
            // Force to keep this MA object and reuse
}
```

We can tell the runtime that whether dynamic assignment checks are required or not. If we can be certain that such checks are not necessary by means of static analysis, then *isAssignmentCheckRequired()* field should be set to *false.*

As explained previously, it is also possible to avoid copying returned objects between scoped memory method calls. *allocateCalleesReturnedObject()* field is supplied to specify where to store a returned object in a chain of method calls with memory areas. If set to true, the returned object of any calls made by this method will be stored in the current method's memory area. This becomes useful when there are asymmetric method invocations with different requirements on where to keep returned objects.

ReturnedObject

This annotation is used to specify that a particular object is to be returned to the caller.

```
@Retention(RetentionPolicy.RUNTIME)
@Target({TYPE, FIELD, LOCAL_VARIABLE})
public @interface ReturnedObject
{    Class type();  }              // Type of the returned object
```

PropagatedException

User-defined exception objects can also be pre-allocated in one of the parent memory areas.

```
@Retention(RetentionPolicy.RUNTIME)
@Target({TYPE, FIELD, LOCAL_VARIABLE})
public @interface PropagatedException
{    Class type();  }              // Type of the exception object
```

Example

We present a trivial program illustrating the use of our annotations described above. In the program below, the *run*( ) method calls *complexCalculation*( ), which then invokes *method1*( ). Two memory areas are involved, and the returned object, *rtn,* will be created in *run*( )'s memory area.

```
public class Thread1 extends PeriodicThread {
   @ScopedMemoryMethod(size()=1000,isAssignmentCheckRequired()=false)
    public void run() {
        Complex x = new Complex(…);
        …
        try { Complex i = complexCalculation(x);
        } catch (Exception e) {   …   }
        DoSomething(i);
    }
```

```
    @ScopedMemoryMethod(size=10000, isAssignmentCheckRequired=false,
        allocateCalleesReturnedObject = false)
    public Complex complexCalculation(Complex param){
        @ReturnedObject(type() = Complex)
        Complex rtn = new Complex(…);
        …
        for (int j = 0; j>5; j++) {
            rtn.add(method1(j));
        }
    }

    // This method is dependent on complexCalculation's MA
    public Complex method1(int i) {…}
}
```

4 Implementation

There are two possible implementations of the memory model we presented in this paper. One implementation would be a transformer that takes a class file, read all the annotations, analyse them, and convert the code into a RTSJ compatible version. The other way is to build a high integrity virtual machine that is aware of our annotations, and provides all the appropriate operations. In the former case, we can still keep using a RTSJ compliant virtual machine since our model does not require any changes in the semantics. This transformation will involve wrapping up annotated methods with an *enter()* method of a memory area. That way, when the method returns, the memory area will be reclaimed automatically. Annotated returned objects can be created in one of the callers' memory areas using the RTSJ's methods of *MemoryArea.* At the present time, we are implementing a transformer that can also analyse and verify class files.

5 Related Work

There are a number of different approaches to removing or optimizing runtime checks for RTSJ, for example [3, 9]. It is impossible to list all of them here. Most of them analyse RTSJ programs to locate non-escaping objects, and allocate such objects in a method's stack or a memory region (see [1] for a comprehensive survey). Programmers can also specify stackable objects explicitly (see [7]).

6 Conclusions and Future Work

We have presented a memory management model for the Ravenscar-Java profile. Our approach maps one anonymous memory area to a user-specifiable method by means of our Java 1.5 annotation types. This straightforward model eliminates single parent rule checks and simplifies other run-time checks that are the main cause of unpredictability and overheads. In fact, it also makes the programmer's job easier since he/she does not have to worry about creating and maintaining memory areas.

All the annotated methods will be automatically converted by a memory allocator/transformer into a RTSJ/Ravenscar-Java compliant version. The semantics of the RTSJ remains the same, meaning that any program in our model when compiled is also a legal RTSJ program. Benefits of using our model are obvious, i.e., a more straightforward programming model, reduced overheads caused by run-time checks, and use of existing/proven RTSJ virtual machines to our advantage. We believe that the idea presented in this paper will help facilitate the use of Java in the area of high integrity systems in the future.

However, in order for our approach to work best, we need an analysis technique for determining the worst-case memory consumption for each method. This can become complex since objects can be created dynamically in Java. We hope that the restrictions in the Ravenscar-Java Profile will make this analysis easier. We will implement our transformer and evaluate the results in the future.

References

[1] B. Blanchet, Escape Analysis for Java: Theory and Practice, ACM Transactions on Programming Languages and Systems (TOPLAS), Vol. 25, Issue 6, November 2003.
[2] G. Bollella, et al, The Real-Time Specification for Java, http://www.rtj.org.
[3] C. Boyapati, et al, Ownership types for safe region-based memory management in real-time Java, Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation, San Diego, CA, USA, 2003.
[4] A. Burns, B. Dobbing, and G. Romanski, The Ravenscar Tasking Profile for High Integrity Real-Time Programs, In L. Asplund, editor, Proceedings of Ada-Europe 98, LNCS, Vol. 1411, pages 263-275, Springer-Verlag 1998.
[5] R. Cytron et al, Efficiently Computing Static Single Assignment Form and the Control Dependence Graph, ACM Transactions on Programming Languages and Systems (TOPLAS), Vol. 13, Issue 4, October 1991.
[6] H. Hetcht, et al, Review Guidelines for Software Languages for Use in Nuclear Power Plant Systems, NUREG/CR-6463, U.S. Nuclear Regulatory Commission, 1997.
[7] J Consortium, International J Consortium Specification: Real-Time Core Extensions, Revision 1.0.14, available at http://www.j-consortium.org.
[8] J. Kwon, A. Wellings, and S. King, Ravenscar-Java: A High Integrity Profile for Real-Time Java, to appear in the Concurrency and Computation: Practice and Experience Journal, Special Issue: ACM Java Grande-ISCOPE 2002 Conference, Wiley.
[9] F. Pizlo, et al, Real-Time Java Scoped Memory: Design Patterns and Semantics, Proceedings of the 7th IEEE Intl., Symposium on Object-Oriented Real-Time Distribute Computing (ISORC'04), Vienna, Austria, May 2004.
[10] Sun Microsystems, JSR 175: A Metadata Facility for the Java Programming Language, available at http://www.jcp.org/en/jsr/detail?id=175.

Encapsulation of IEC 61499 Function Blocks Using Real-Time Java According to the RTSJ

Marcus Tangermann[1], Christian Schwab[1], Arndt Lüder[1], Luca Ferrarini[2], and Carlo Veber[2]

[1] University of Magdeburg, Center Distributed Systems CVS@IAF, Universitätsplatz 2, 39106 Magdeburg, Germany
{marcus.tangermann,christian.schwab,arndt.lueder}@mb.uni-magdeburg.de
http://www.uni-magdeburg.de/iaf/cvs

[2] Dipartimento di Elettronica e Informazione, Politecnico di Milano, P.za Leonardo da Vinci 32, 20133 Milano, Italy
{ferrarin,veber}@elet.polimi.it
http://www.elet.polimi.it/

Abstract. Existing trends in information sciences currently also entering the manufacturing control field. Especially object oriented programming, distributed systems, and platform independent programming are interesting approaches to improve control system design at the factory floor. Within this context the international standard IEC 61499 has been adopted. It enables a modular and hierarchical control application design on a functional level based on control function blocks. But it does not address implementation issues. This paper will present an implementation and application approach to map the elements of an IEC 61499 Function Block Network to Java classes according to the Real-Time Specification for Java and to implement on this basement most flexible control applications. This approach will enable the usage of Java also on the field device level where beside of reliability, Real-Time is one of the major requirements and will improve the application of the IEC 61499 standard. This work is part of the research project TORERO, which is funded by the European Commission under the IST program.

1 Introduction

Resulting from the current market situation with increasing market power of customers and increasing globalisation, the producing industry is forced to enlarge the flexibility of production systems and simultaneously increase to system design and maintenance costs. This situation has lead to a strong trend towards distributed systems and in automation towards Distributed Control Systems (DCSs) based on intelligent field devices and distributed intelligence [1].
In addition, an upcoming trend in automation industry is the increasing amount of Information Technology (IT) used within control systems for several purposes as remote management or engineering and maintenance support for example [2].
From the software point of view, a major milestone for the design of DCS applications on a higher level is the standard IEC 61499 “Function Blocks for Industrial

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 346–358, 2004.
© Springer-Verlag Berlin Heidelberg 2004

Process Measurement and Control Systems" [3]. It proposes the standardisation of control functions at a higher level than pure programming languages and provides a vendor independent approach to program DCSs, whereas the current standards IEC 61131-3 "Programming Languages for Programmable Logic Controllers" [4] and IEC 61804 "Function Blocks for Distributed Control Systems." [5] are integrated and extended respectively.

The main advantages of the application of IEC 61499 conform function blocks are

- the possibility of a structured, modular, and hierarchical control system design
- the model based control system design with a standardised modelling framework,
- the possibility of an event based control flow.

But the IEC 61499 standard does not define implementation issues. Since the IEC 61499 defines a complex interaction system among the functions blocks based on events and data connected to the events, the execution of a IEC 61499 based function block control architecture is equivalent to the execution of a small operating system. In addition a set of different implementation approaches are possible depending on the mapping of function blocks to operation system entities and behavioural conditions [6]. Several proposals and tools exist for the application of IEC 61499 function block based modelling approaches. Most known are the FBDK [7] and the CORFU architecture [8]. But a wide variety of problem solutions such as:

- to design distributed control applications independent from the underlying resources (device hardware, communication protocol, etc.),
- to manage the automatic allocation of the control application to the specific field devices depending on the resources,
- to implement communication related code automatically depending on the allocation result, and deploy the allocated control application code portions to the field devices,
- to integrate the Internet into the DCS and support the whole life cycle management of the system,

are still under development.

Regarding to this, the research project TORERO (Total life cycle web-integrated control) [9], which is funded within the IST (Information Society Technologies) initiative of the European Commission, focuses on the whole life cycle of DCS. In particular, it aims at specifying an architectural platform for a DCS including a new type of Internet enabled field device (TORERO Device), an appropriate Ethernet based infrastructure, and an Integrated Development Environment (TORERO IDE).

The TORERO architecture and methodology focus on the control application itself rather than on individual devices, and thus allow for the design of distributed control applications independent from the underlying resources (e.g. communication protocol, device hardware, operating system) and the subsequent allocation of this control application to the TORERO Devices.

Based on the TORERO architecture a control engineer will be able to implement a control application device independent based on per-given and self-designed function blocks by simply connecting them and automatically distributing them among devices including an automatic integration of necessary interaction mechanisms.

To enable the mentioned device independence, modularity, and automatic distribution a hardware independent and object-oriented implementation languages is required for the control implementation. Hence, Java has gained importance within the implementation approaches for IEC 61499 function block based control systems. It will be used

for the function block framework as well as for the control algorithms, both running on field devices. At this level it is obvious to consider Real-Time constraints, thus the Real-Time Specification for Java (RTSJ) [10] will be applied in the TORERO approach.
This paper will present a method – which is called Encapsulation – containing a mapping of the elements of an IEC 61499 Function Block network to Java classes according to the RTSJ. To get an overview about the TORERO concept and to provide a basement for the further understanding, within the second chapter the TORERO engineering approach and within the third chapter the IEC 61499 standard and their implementation approaches will be described. Based on this, Chapter 4 contains the Definition of the Java Real-Time Encapsulation of IEC 61499 Function Blocks and their connections. In the last chapter, achievements of this work will be summarised.

2 TORERO Methodology

The TORERO scenario proposes a new methodology for a DCS which main part is called engineering process and which differs from the traditional one [11]. Main steps of the Engineering process are the following.
In the **first step,** the control application of the DCS will be designed. Therefore a set of pre-defined and/or self-developed IEC 61499 Function Blocks will be connected independent of the underlying resources.

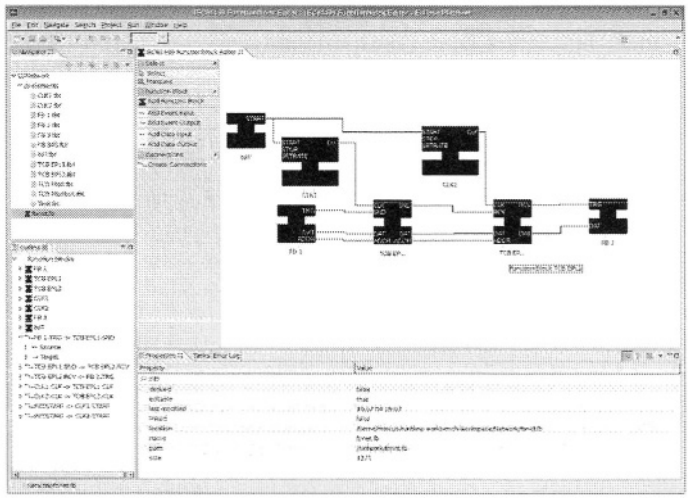


Fig. 1. Function Block Editor

Then, the Functions Blocks, its internal details and algorithms, and the connections between Function Blocks will be implemented automatically according to Real-Time Java classes (Encapsulation of Function Blocks to RT-Java). This process will be supported by the Eclipse [18] based TORERO IDE.
In parallel, all TORERO Devices within the network announce their presence and their connections by means of UPnP, the Universal Plug-and-Play mechanisms [19]. The

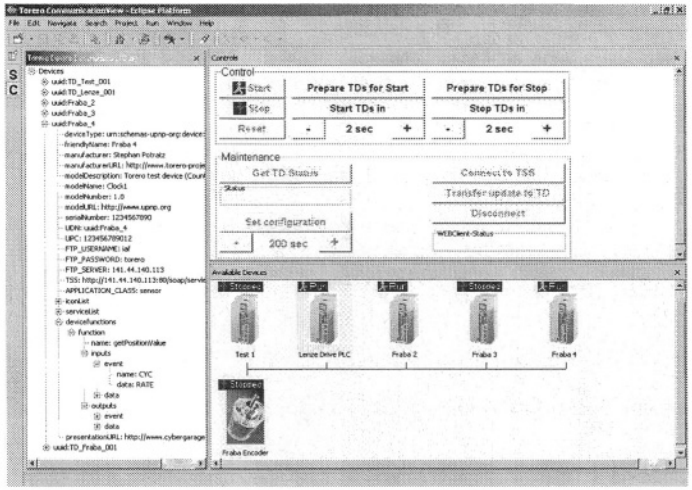


Fig. 2. Communication View of the TORERO IDE

announced network of TORERO Devices will be displayed in the TORERO IDE's "Communication View".

In the **next step,** the control application will be allocated automatically to the TORERO Devices depending on resource parameters of the devices. The output is an allocation table which will be processed further in the next step [12].

In the **following step,** and considering the allocation results, communication related code will be generated automatically by using AspectJ [13].

In the following **two steps,** first the complete code – the control application and the communication related code – will be downloaded to the TORERO Devices using ftp mechanisms and second, the control application's life cycle, ranging from start / stop functionalities to automatic update mechanisms via the Internet will be managed.

It is easy understandable that the implementation of Function Blocks in Real-Time Java is of most importance for the efficiency, correctness, and quality of the complete approach. Hence, the Encapsulation approach has got huge interest in the complete TORERO approach. But this Encapsulation is strongly dependent on the behavioural model of the IEC 61499 Function Blocks on the one hand and on the intended application field of the control system and its special behaviour on the other hand. Both will be sketched quickly in the following.

3 Implementation of an Execution Model of an IEC 61499 Application

After a short introduction of a Function Block according to the IEC 61499 standard, in the second subsection possible Java based implementation approaches for the execution of such Function Blocks will be given.

3.1 Basic Function Block

The IEC 61499 standard, applied within TORERO, is based on a fundamental module, the Function Block (FB), which represents a functional unit of software, associated to a hardware resource of a control system. A FB instance is characterised by: its type name and instance name, sets of event inputs/outputs and data inputs/outputs, internal data, an Execution Control Chart (ECC) consisting of states, transitions and actions, which invokes the execution of control algorithms in response to event inputs, and a set of algorithms, associated with the ECC states [14].
An incoming event will be read by the ECC and can enforce the invoking of an algorithm in dependence of the active state of the ECC and the possible state changes enforced by the incoming event. When the execution of an algorithm is invoked, the needed input and internal data values are read and new values for output and internal data may be calculated. Furthermore, upon completion of execution of an algorithm, the execution control part generates zero or more event outputs as appropriate. This structure is given in the Fig. 3.

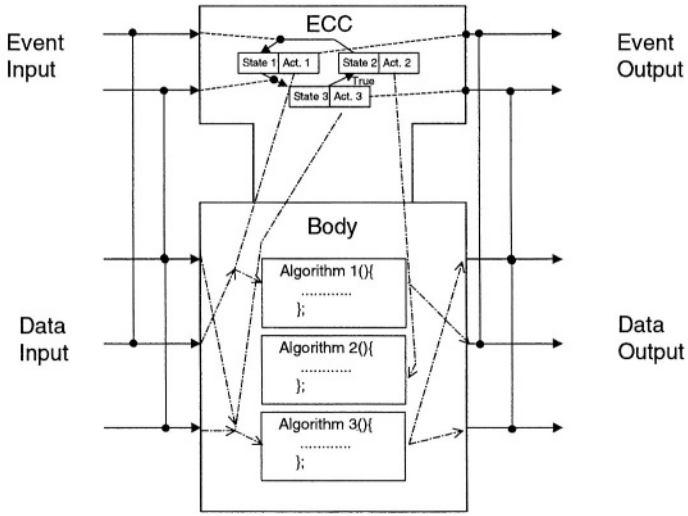


Fig. 3. IEC 61499 Function Block

By properly connecting more then one FB, a distributed application can be defined. The event flow between ECCs of FBs determines the scheduling and execution of the FB algorithms and thereby the behaviour of the complete control application.
An application can be distributed among several devices. A device uses the causal relationship specified by the application FBs to determine the appropriate responses to events, which may include communication and process events, utilising the different resources associated to the devices.
A resource is considered within the IEC 61499 standard as a logical subdivision within the structure of a device, which has independent control of its operations. The functions of a resource are to accept, process and return data and/or events to the process and/or communication interfaces, as specified by the applications utilising the resource, i.e. by the FBs associated to the resource.

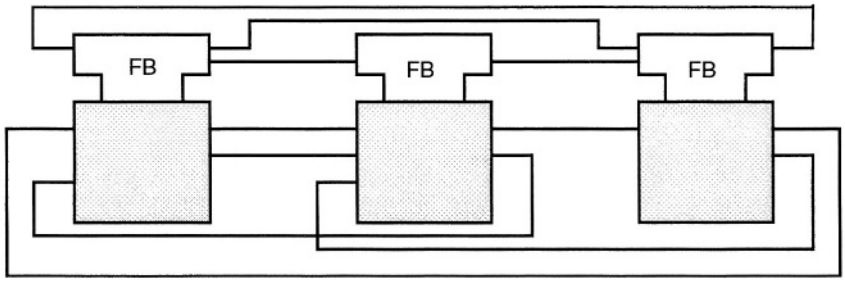


Fig. 4. IEC 61499 Function Block Network

3.2 Implementation Approaches

Within the IEC 61499 standard an exhaustive definition of the event based execution model of the individual function blocks is given [14]. But it does not specify the execution of the interactions of the function blocks as well as some details of the internal orders of Function Block execution. Hence, the standard opens up different approaches for its implementation.

The possible implementation approaches, based on the object model deriving from the one given in the standard, can be grouped according to the criteria Function Block Scan Order (FBSO) and Multitasking Implementation (MI) as main criteria.

With the Function Block Scan Order the order of tests on executable events arriving at a Function Block is specified. These order is essential for the event propagation within a Function Block network.

For FBSO it is possible to distinguish between two opposite cases:

- In the first case the scan order is fixed a priori, similarly to what happens in a PLC (Programmable Logic Controller)-like implementation.
- In the second case, the scan order is not fixed but depends only on the event occurrence.

With the Multitasking Implementation the association of Function Blocks to threads and thereby ability of a quasi parallel execution of function blocks is described.

For MI it is possible to distinguish between two main groups:

- The case where MI is not used.
- The Case where MI is used. The thread execution can be either without control or controlled on the basis of a time slice or of the execution of a FB.

The various possible combination are synthetically represented in Table 1 and explained in the following.

Table 1. Possible Implementation Approaches

MI / FBSO	Not used	Used, not controlled	Used, controlled, time slice	Used, controlled, FB slice
Not fixed	A0	A1	A2	A3
Fixed	A4	x	A5	A6

A0) The FB is implemented as a "simple" (in contrast with the solution that use threads) object. The generation of an event is implemented through a direct call to the run method of the connected FB.

A1) The FB is implemented as a thread-object. All threads run in parallel and the generation of an event is implemented simply setting a synchronised variable of the event connection.

A2) The FB is implemented as a thread-object. In the initialisation, all threads are forced to sleep except the one(s) corresponding to the "start" FB. The ECC and the algorithms are executed in a time slice mode.

A3) The FB is implemented as a thread-object. In the initialisation, all threads are forced to sleep except the one(s) corresponding to the "start" FB. The generation of an event is implemented as a notification to the threads "sleeping on" the event connection. The thread returns in a sleeping state when the ECC goes in the idle state.

A4) The FB is implemented as a "simple" object. The execution is defined by a PLC-like algorithm that, following a predefined list, directly calls the run method of the FB. The run method executes the ECC until it return in the idle state.

A5) The FB is implemented as a thread-object. In the initialisation, all threads are forced to sleep. The execution is defined by a PLC-like algorithm that wakes up the threads following the list of the active ones. The threads are activated for a pre-fixed DT time.

A6) The FB is implemented as a thread-object. In the initialisation, all threads are forced to sleep. The execution is defined by a PLC-like algorithm that wakes up the threads following the list of the active ones. The run method executes the ECC until it returns in the idle state.

The symbol "x" in Table 1 shows that such a combination is not applicable.

The different implementation approaches have advantages and disadvantages regarding their Real-Time behaviour. Main outcome of the test shows the following (For a detailed description of the tests and the results, please see [6]):

- A0 is the best solution when the FB network is composed by a series of FBs connected in cascade mode, but presents unexpected behaviour when more then one event output must be generated at the same time and in presence of event connection loops,
- A3 is the one that most follows the expected behaviour and the idea of the standard of FBs that run in parallel; its execution times are comparable with A0,
- A4 allows to predict the temporal performances of the application (which is impossible in A0, A3) but they strongly depend on the order of the sequence of FBs;
- A5 presents a more (virtual) parallel execution then A4 but its temporal performances depend on the order of the sequence of the FBs and the DT value.

Based on this results, version A3, the Multitasking Implementation of the FBs with not fixed Function Block Scan Order will be implemented in Java regarding the RTSJ since it seems to have the most suited behaviour of execution of Function Blocks in accordance to the not written spirit of the IEC 61499. This will be part of the next chapter.

4 Real-Time Encapsulation of an IEC 61499 Function Block Network

The fundamentals of the Real-Time Encapsulation are the direct conclusions from the decision to implement version A3 and the necessary event propagation within the Function Block network. These requirements have to be matched to RTSJ implementation possibilities.

The required multitasking abilities as well as the not fixed scan order will enforce the application of one thread per Function Block. Since, within the Function Blocks the control algorithms will be executed which may access the controlled plant the threads of the Function Blocks need to be Real-Time treads and need to operate on a memory which is outside of the scope of the garbage collection. Hence, a class `FunctionBlock` is required which is derived from the class `NoHeapRealTimeThread` of the RTSJ.

The event and data propagation within Function Block networks is given by connections among Function Block outputs and inputs. These connections need to be implemented in a way not directly affecting the execution of the Function Blocks but enabling a safe event propagation. Hence a connection has to be a Real-Time thread and therefore the class `Connection` will also be derived from the class `NoHeapRealTimeThread` of the RTSJ.

An event itself is encapsulated within an `EventContainer` instance. This instance contains the name of the target event input as well as the appropriate amount of `DataContainer` instances representing the data belonging to the event.

One of the main problems of implementing the IEC 61499 using the RTSJ is the variable amount of simultaneous events which leads to a variable amount of simultaneous existing `EventContainer/DataContainer` instances. To circumvent the problem of creating new instances in a non heap memory, the encapsulation follows an approach suggested by [16], which proposes the use of mutable objects in immortal memory as containers. Hence, the class `ContainerFactory` provides reusable instances of `EventContainer` and `DataContainer` located in the immortal memory. Therefore, the `ContainerFactory` maintains a single linked list with all container objects of the system. In case a new container (event or data) is requested by invoking the static methods `getDataContainer(int type)` or `getEventContainer` the `ContainerFactory` searches the list for a free container. If a container is available, this container instance is marked as in use and returned to the caller of the method. A container object keeps marked as in use until it is explicitly marked as free by the using instance of the container. In case there is no free container available at the moment, the `ContainerFactory` creates a new container instance in immortal memory, adds this container to the list and returns the container to the calling object.

The Execution Control Chart as described in section 3.1 is modelled within the encapsulation by an instance of `ExecutionControlChart` in collaboration with according instances of the class `State`. The ECC itself only contains the actual state, the knowledge of transitions triggered by events is delegated to the State instance. Therefore, a State instance manages all states that can be triggered by an event in a hash table with the event name string as the key. In case an event is triggered, the

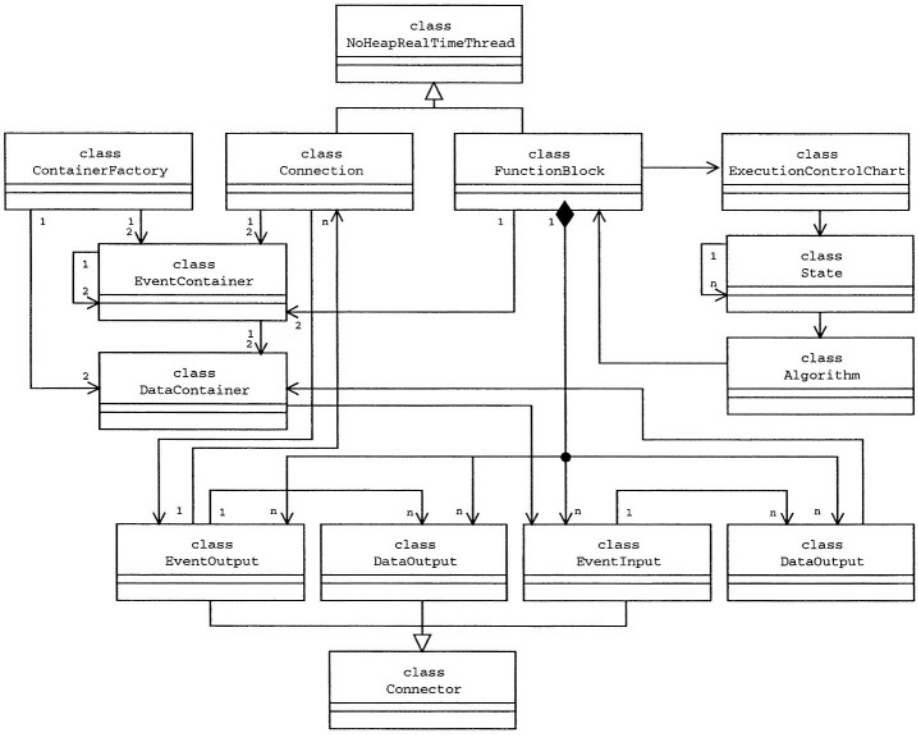


Fig. 5. Class Diagram of the RT-Encapsulation

ECC requests the transition from the state by invoking the `doTransition (String name)` function. Based on the next state that this event follows, the new state is returned to the ECC and is then marked as the new state of the ECC. Additionally the according algorithm of this transition contained in the Algorithm instance of the new state is executed. The described class system is given in Fig. 5.
The behaviour of the event propagation in the Function Block network and the behaviour of a Function Block for event execution are realised as follows.
The event propagation is initiated by a running instance of `Algorithm`. As a first step for triggering an event, an algorithm has to set the appropriate data of data outputs that are associated with the event output to trigger. Data of a `DataOutput` instance can be set by invoking the `setData` method it provides. On calling this method the `DataOutput` requests a `DataContainer` of the appropriate type by calling `ContainerFactory. getDataContainter (int type)`. Within this `DataContainer` the according data of the `DataOutput` will be stored.
As a next step, the `Algorithm` instance may trigger the event by calling `triggerEvent ()` of the `EventOutput` instance that the algorithm wants to trigger.
For each connection that is associated, the `EventOutput` will then request an `EventContainer` from the `ContainerFactory`. Adjacent, the `DataContainer` from each `DataOutput` is requested by calling the `getData` method of `DataOutput`, cloned and added to each `EventContainer`. Subsequently the

`EventContainer` instances are added to each Connection instance that is associated with this `EventOutput` by invoking the method `addEvent(EventContainer event)`.
This method call adds the `EventContainer` to the `Connection` instance list of events to deliver. In case the `Connection` instance thread sleeps since the event list was empty before this event reached the list, it is woken up.
The `run()` method of the connection detects a further `EventContainer` in its container list and delivers it to the target function block of the connection.
At the target the `run()` method of the `FunctionBlock` instance retrieves the event name of the `EventContainer` and triggers the event after the different `DataContainer` instances are set to the `DataInput` instances that are associated with the event. The event itself can now be processed by the target which marks all container as free after the processing is finished.
This event propagation behaviour is given in the Fig. 6 as sequence diagram.
Within this chart, the main characteristics are visible. The three main partners of the event propagation, the both used instances of `Algorithm` and the instance of `Connection` are decoupled with respect to the control flow, since they are located within different Real-Time threads. In addition, the necessary instance of `ContainerFactory` will provide the necessary control over the allocated data storage.

5 Useful Extensions of the RTSJ

The control of used storage is one benefits of the TORERO encapsulation approach. But it is also the main problem of the approach.
During the control design process the algorithms within the Function Blocks need to be implemented. Since the class `FunctionBlock` is derived from `NoHeapRealTimeThread` and the algorithms belonging to a `FunctionBlock` instance are, therefore, located in the memory area of the `FunctionBlock` instance and, hence, outside of the area of the garbage collection, the implementation of an algorithm need to be done very careful with respect to memory allocation and object instantiation.
It is impossible to allow an arbitrary PLC programmer to do this job since she/he is surely not familiar with the special details of the necessary approach.
There are two main solutions for that problem. The first one is based on the specification of an algorithm by a IEC 61131-3 compatible PLC programming language [4] and then transforming that definition automatically into Java code. The second one requires the extension of the RTSJ by a Real-Time garbage collection as it is used for example in the Jamaica VM [17].
Since the second solution seems to be the better choice with respect to an most efficient approach, the authors of that paper will support the extension of the RTSJ by a Real-Time garbage collection.

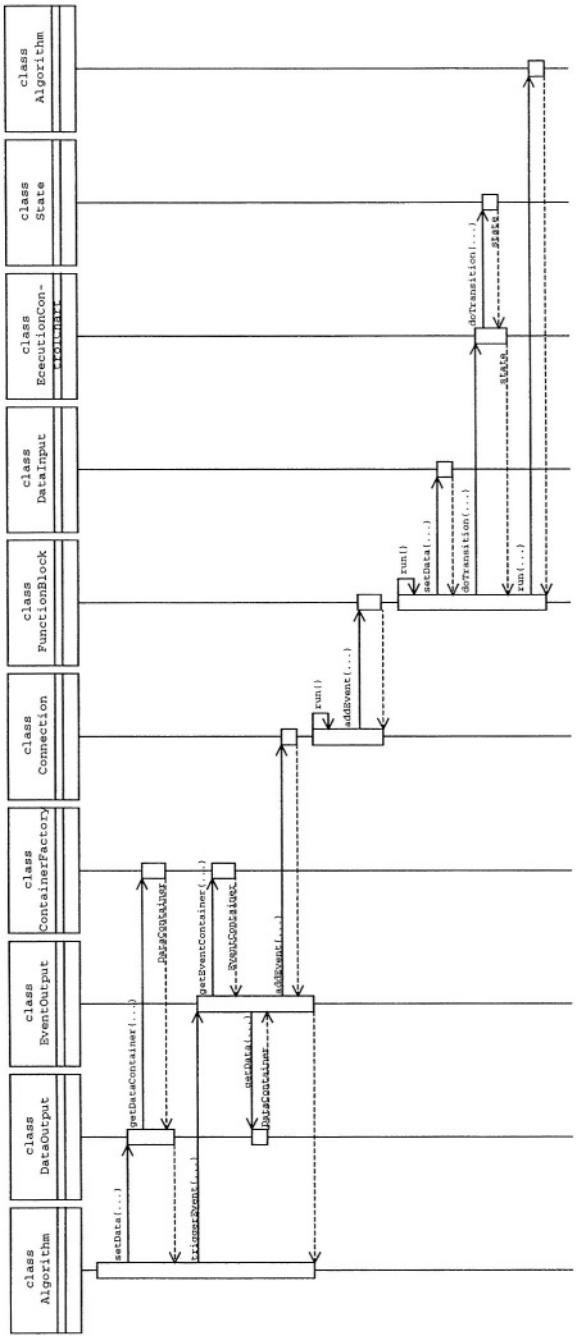


Fig 6. Sequence Diagram of the Event Flow

6 Summary and Outlook

Within this paper an application of the Real-Time Specification for Java within the field of field control has been described. Within the TORERO approach the benefits of Java will be useable within Real-Time control. It improves the reusability of control code and will decrease control system design efforts all based on the Java benefits.
Within this paper the special details of the use of the RTSJ within the TORERO approach have been described and some current drawbacks have been mentioned.
Within future work the application of Real-Time Java within field control will be extended. One step within this direction will be the automatic translation of IEC 61131-3 compatible control code to Real-Time Java.

References

1. Kühnle, H., Klostermeyer, A., Lorentz, K.: A Paradigm Shift to Distributed Systems in Plant Automation, In: Fares Sebaaly, M. (Ed.): The International NAISO Congress on Information Science Innovations ISI' 2001, Dubai, VAE, Proceedings, pp. 463-469.
2. Walczuch, R., Sans, K. Lemmink, J., Streukens, S.: Acceptance of Information Technology by Technology Ready Service Employees, International IMS Forum 2004 – Global Challenges in Manufacturing, Como, Italy, 2004, Proceedings, pp. 257-264.
3. Standard IEC 61499, Function Blocks for Industrial-Process Measurements and Control System, IEC TC65/WG6, 03-01-24, www.iec.ch.
4. Standard IEC 61131-3, Programming Languages for Programmable Logic Controllers, 2nd Edition, IEC TC65/WG6, 03-06-02, www.iec.ch.
5. IEC 61804-2 Ed. 1.0 E: Function blocks (FB) for process control - Part 2: Specification of FB concept and Electronic Device Description Language (EDDL), IEC SC65/WG7, 04-05-01, www.iec.ch.
6. Ferrarini, L. Veber, C.: Implementation approaches for the execution of IEC 61499 applications, 2nd International Conference on Industrial Informatics (INDIN'04), Berlin, Germany, 2004, Proceedings, pp.612-617.
7. Function Block Development Kit, online Editorial and Demonstration http://www.holobloc.com/doc/fbdk/FBEditor.html, 2004.
8. Thramboulidis, K.: Using UML in Control and Automation: A Model Driven Approach, 2nd International Conference on Industrial Informatics (INDIN'04), Berlin, Germany, 2004, Proceedings, pp.587-593.
9. The TORERO Project, Project home page: http://www.uni-magdeburg.de/iaf/cvs/torero, 2004.
10. Bollella, G., Brosgol, B., Dibble, P., Furr, S., Gosling, J., Hardin, D., Turnbull, M., Belliardi, R.: The Real-Time Specification for Java™ (First Public Release), Addison-Wesley, 2001.
11. Kalogeras, A. P., Prayati, A., Schwab, C., Tangermann, M., Ferrarini, L., Papadopoulos, G.: Integrated Web Enabled Control System Design Methodology, accepted to IEEE International Workshop on Factory Communication Systems – WFCS, Vienna, Austria, September 2004, Proceedings.
12. Prayati, A., Koulamas, C., Koubias, S., Papadopoulos, G.: A Methodology for the Development of Distributed Real-Time Control Applications with Focus on Task Allocation in Heterogeneous Systems", IEEE Transactions on Industrial Electronics, Special Section on Distributed Network-based Control Systems and Applications, 2004.

13. Tangermann, M., Schwab, C., Kalogeras, A., Lorentz, K., Prayati, A.: Aspect-Orientation of Control Application Code for Distributed Automation Systems: the TORERO Approach", First Workshop on Java Technologies for Real-Time and Embedded Systems – JTRES 2003, Catania, Italy, November 2003, in R. Meersman, Z. Tari et al. (Editors): On the Move to Meaningful Internet Systems 2003: OTM 2003 Workshops, Lecture Notes in Computer Sciences 2889, Springer, Berlin, 2003, pp. 335-345.
14. Christensen, J.: Basic Concepts of IEC 61499, www.holobloc.com, 2004.
15. Lewis, R.: Modelling control systems using IEC 61499, IEE control engineering series; n. 59, The Institution of Electrical Engineers, London, UK, 2001.
16. Dibble, P.: Real-Time Java Platform Programming, Sun Microsystems Press, Prentice Hall, 2002.
17. Siebert, F.: Bringing the Full Power of Java Technology to Embedded Real-Time Applications, 2nd Mechatronic Systems International Conference, Winterthur, Switzerland, October 2002, Proceedings, pp. 41-46.
18. www.eclipse.org
19. Cohen, Aggarwal, Goland: Universal Plug and Play Architecture, 2000, www.upnp.org.

No Heap Remote Objects: Leaving Out Garbage Collection at the Server Side

Pablo Basanta-Val, Marisol Garcia-Valls, and Iria Estévez-Ayres

Drequiem Lab
Distributed Real Time Systems and Multimedia Laboratory
Department of Telematics Engineering
Universidad Carlos III de Madrid
Avda. de la Universidad, 30
28911, Leganés (Madrid), Spain
{pbasanta, mvalls, iria}@it.uc3m.es

Abstract. This paper presents an approach to provide real-time support for Java RMI integrating it with the RTSJ memory model to leave out garbage collection. A new construct for remote objects is proposed: the *no heap remote object (NhRo).* The usage of a NhRo guarantees that memory required to perform a remote invocation (at the server side) does not use heap memory. The aim is to avoid garbage collection in the remote invocation process, improving predictability and memory isolation of Java-based distributed real-time applications. The Sun RMI implementation has been modified to integrate the NhRo model for static environments. The protype implementation and the performed experiments show the feasibility of the model.

1 Introduction

Distributed real-time systems require that the underlying communications middleware have real-time support. Java RMI is a very popular object-oriented middleware that includes the many interesting features of the Java technology, such as object orientation, strong typing, code mobility, virtual machines, security, etc. However, real-time applications require that the middleware includes real-time support for predictable management of threads, memory, and network communication. Although there is an on-going research effort to come up with the specification for a real-time RMI [11] merging RMI with RTSJ [1], currently Java RMI does not have any support for distributed real-time applications.

Such merging is not a simple task due to some handicaps the RTSJ presents for the RMI implementation internals. The work presented in this paper is mainly concerned with two of them: the RTSJ memory model and the Java garbage collection mechanism. Among the main obstacles of Java to real-time are its complex memory model and its garbage collection mechanism, that introduces unpredictable pauses in program execution. There are some proposals to solve the garbage collection problem, but none of them is perfect. On one hand, turning the garbage collector off introduces changes in the coding paradigm, besides complicating the usage of the available Java

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 359–370, 2004.
© Springer-Verlag Berlin Heidelberg 2004

implementations (libraries). On another hand, real-time garbage collection techniques [3] clean up the memory in a predictable way, but at the cost of introducing overheads in program execution. Eventually, region-based solutions (like *scoped memory* defined in the RTSJ [1] or allocation contexts defined in RTCE [2]) require that the programmer be aware of the life duration of objects to group them appropriately in the same memory region instaces; somehow, this complicates the programmer's job.

Among the common challenges to be solved in the integration of distributed real time middleware and RTSJ are (1) if scoped memory may substitute the garbage collector or not in distributed systems, and (2) what is the impact of that replacement in the programming paradigm and middleware infrastructures.

In this paper, we propose a combination of the remote object model defined in RMI with RTSJ scoped memory, that is called the *no heap remote object (NhRo)* model. Our model avoids the use of the garbage collector in favor of scoped memory. A remote object that inherits from a NhRo is guaranteed to use only memory that is not heap memory. This avoids using the garbage collector in the remote invocation process, improving predictability and memory isolation of distributed Java applications.

This paper is organized as follows. Section 2 presents the related work and gives the new ideas presented in this paper wih respect to the current state of the art. Section 3 outlines the main concerns of the integration of RMI and the no heap memory areas of RTSJ, stating some of the main problems to overcome in the integration. Section 4 presents the *no heap remote object* (NhRo) model; also, the model is ilustrated by giving a programming example. Section 5 gives the server-side architecture of our modified RMI to support the NhRo model; it shows how the model is mapped to static environments, and how it is implemented without violating the RTSJ memory model. Section 6 presents the experimental results obtained from the implementation of a prototype RMI that is compliant with the NhRo model; the implementation is based on the Sun RMI package. Finally, section 7 presents the conclusions and future work directions.

2 Related Work

Java technology has become very popular due to, among other reasons, the many conceptual features it introduces (virtual machines, code mobility, automatic garbage collection, security, etc.) that allow a wide target of applications. Among the many application fields of the Java technology, there are distributed systems (that require the existance of a communications middleware) and real-time systems (that require real-time virtual machines). Currently, there is an emerging research trend focused on the integration of middleware architectures like CORBA [15] and RMI [5] with Real Time Java [1][12] in order to build predictable distributed real-time systems [8] [9] [10] [11] using the Java language. The existance of Real-time Corba Specification (RT-CORBA)[15] seems to make RTSJ and RT-CORBA synthesis much easier than RTSJ and RMI, as it is argueed that it is only required to define the mapping between RT-CORBA and RTSJ as it was done in past for other languages

like C or Java. This is not exactly true because there are several RTSJ features, like scoped memory or no heap real time threads, that make integration with RT-CORBA not trivial. In the other hand, the RMI architecture has two advantages over RT-CORBA model that facilitate its integration with RTSJ: a plain architecture, without *ORB* or *POA* and the fact that there is not still defined an RT-RMI specification. This later characteristic leaves space for experimentation of different approaches to help make the path to a definite proposal.

Overcoming garbage collection problems is another area of research in real-time support for middleware. One of the approaches consists of leaving out garbage collection by using memory types that do not suffer garbage collection. In the Java world this implies avoiding the use of heap memory. There is not much work in designing pure no-heap memory based systems. No heap memory is thought more as a complementary feature than as a main programming paradigm [13]. In [19], it is proposed a real-time component model based in no-heap RTSJ memory called *scratchpads*. Although this work is focused on centralized (mono-virtual machine) systems and ours on distributed systems, both share some similar principles. Both make a difference between the state of a component (or NhRo) and the set of temporal objects that change this state, placing them in different memory areas (or memory contexts). Also, both fix the maximum life of objects created in LTMemory (when the `run` method exits or when the remote invocation ends). However, distribution introduces the necessity of having thread and *memory area* pools.

JSR-50 [10] is working on defining the Distributed Real Time Specification for Java (DRTSJ). This specification works around RMI and RTSJ to provide support for predictability and end-to-end timeliness of trans-node activities. In [11], it is defined a tagging interface, `NoHeapRealtime`. It ensures that the underlying transport of messages and remote object invocation does not use heap memory. However, in that work, the impact of this choice on the middleware is not examined. The memory model of our NhRos and memory area pools defines a way to support this semantics.

On the RT-CORBA implementation side, Zen [14] works on maintenance of servant (similar to RMI remote object) programming paradigm unchanged meanwhile it uses RTSJ scoped memory in POA and ORB to effectively reduce collector sweeps. This approach is less aggressive than ours; however, it requires the existence of a garbage collector at clients and servers that again keeps the garbage collection problem in the scene.

3 RMI and RTSJ No Heap Memory Integration

Integrating no heap memory into RMI may be directed at two kinds of scenarios: static and dynamic. A static scenario is one where once created, a remote objects is never destroyed and each remote object may receive or perform an unlimited number of remote invocations, depending on whether it is a remote server or a remote stub. In a dynamic scenario, there are no constraints neither on creation of remote object nor on destruction at any time of program execution.

To integrate RMI with the no heap memory model of RTSJ, one must deal with the problem of associating each object created in the middleware context with a particular type of memory area of RTSJ. The RMI context creates Java objects each time a remote object is either: (1) created, (2) receives a remote invocation, (3) invokes a remote method, or (4) is destroyed. It has to be decided in which memory area these objects will be created, that can be:

- *Immortal memory:* Only, a limited number of objects can be created in immortal memory area and typically requires *reusing* these objects in order to avoid memory leaks. However these objects may be referenced from any other object.
- *Scoped memory.* Potentially, an unlimited number of objects can be created in scoped memory areas. However, the life time of objects must be known and determined a priori to be able to keep the rules associated with this memory type.

Though the use of immortal memory simplifies implementation, it is not always the best choice. Thus, during the remote invocation process, temporal objects are created as a result of the de-serialization remote invocation parameters or sending the results of remote invocation to client. These objects cannot be directly created in immortal memory; if they were directly created in immortal memory, sooner or later all immortal memory would be consumed. Therefore, it has no sense to impose any constraints on the maximum number of remote invocations that threads may perform over a remote object instance.

One solution to this problem may be to change the Java Serialization mechanism [17] to be able to reuse objects in immortal memory instead of creating them from scratch at each invocation. However, this requires many changes to the serialization process. So, to keep the serialization mechanism unchanged the best option is to use scoped memory. This allows us to remove all temporal objects created during remote invocations when they are no longer in use.

4 No Heap Remote Objects

As a `UnicastRemoteObject` is a template for a remote object in a general purpose RMI environment, a *No Heap Remote Object* (NhRo) is a template for RTSJ-enabled RMI environments. A NhRo allows to be exported, and it can receive remote invocations. However, an object that inherits from the NhRo template is guaranteed that memory required to perform remote invocations (at the server side) is not allocated in heap; the invocation is satisfied in scoped memory. Therefore, the NhRo avoids garbage collection in remote invocations, improving predictability and memory isolation of Java-based distributed applications. The general definition of NhRos is targeted to dynamic and static environments.

The main difference between classical remote object concept and NhRo memory models is that the later memory model is not plain. In a NhRo, the memory is fragmented in two memory contexts, the one that is related with its creation process and the one that is used to store objects of remote invocations:

- *Creation context.* The remote object instance and the objects referenced by its attributes belong to this context; it stores the state of the remote object. Objects stored in this context are never destroyed before the NhRo instance is destroyed. This context therefore contains a set of scoped memory instances and immortal memory.
- *Invocation context.* In this context, temporal objects needed to perform remote invocations (e.g. serialization and de-serialization objects or remote method invocation parameters) are stored. This context therefore contains a set of scoped memory instances. Objects created in remote method invocations are created by default in a scoped memory instance of the invocation context. Scoped memory instances that belong to the invocation contexts are created and managed by the RMI internals. Objects created in the scoped memory of this context are destroyed after the remote invocation has ended.

No Heap Remote Object Assignment Rule

The life of objects created in the invocation context is shorter than the life of objects created in the creation context. Therefore, an object of the creation context cannot hold a reference to an object created in the invocation context. The opposite is possible: an object created in the creation context can be referenced by an object created in the invocation context. We call this rule the *no heap remote object assignment rule.*

The NhRo memory-contexts definition is non restrictive; each context may be a single memory area instance or more than one. This allows multiple particularizations of the model depending on the scenario where the NhRo is to be deployed. Thus, in static environments, it is reasonable to use only immortal memory, whereas in dynamic environments, we may use scoped memory or immortal. In both scenarios, the No heap remote object assignment rule keeps its meaning: the life of objects created in the invocation context is shorter that the life of objects created in the creation context.

4.1 An Example of No Heap Remote Object Memory Model: A Distributed Calculator

This section illustrates the programming paradigm of the NhRo by means of an example shown in figure 1, `CalculatorImpl.java`, contains the server side of a remote object that represents a calculador. Methods that may be remotely invoqued by clients are defined in Calculator.java interface file. Our calculator has memory; it remembers the result of the last operation in a `DataInt` object. This result is referenced by the `lastResult` Calculator attribute.

Clients may invoque three operations in the remote calculator: `add`, `lastResult` and `doNothing`. `Add` performs addition of two `DataInt` objects, returns a `DataInt` to the client containing the result of the sum, and it points `lastInteger` to the result. Clients may retrieve the value of the object pointed by `lastInteger` using the method `lastResult`. Finally, the `doNothing` method is defined for

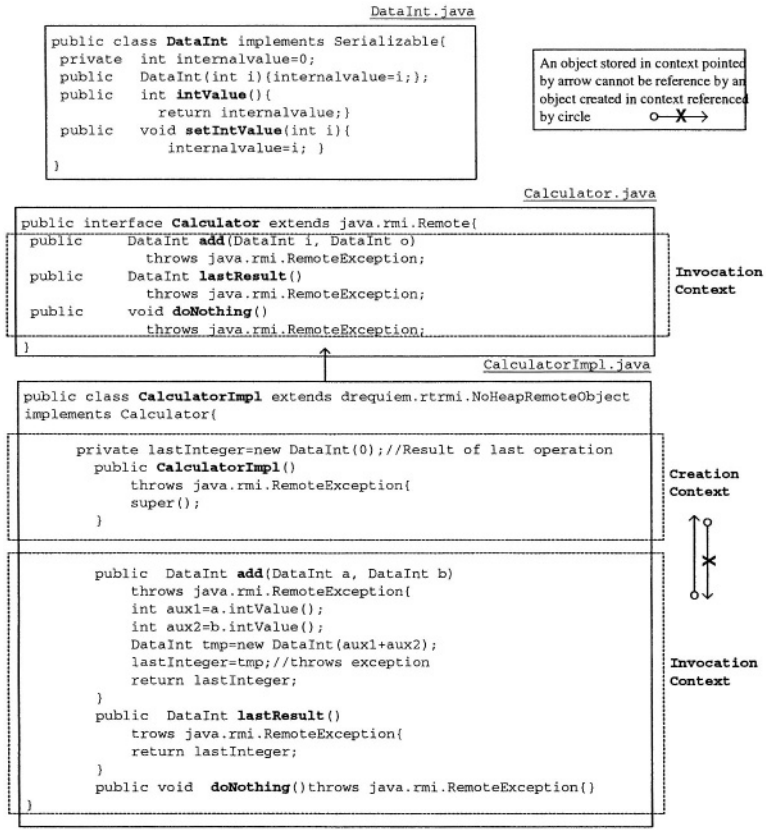


Fig. 1. Distributed Calculator implemented using a NhRo

benchmarking purposes, as it will be explained in the last section on experimental results.

In `CalculatorImpl` instance, objects created in the constructor and objects referenced by the attribute `lastInteger` belong to the creation context. Objects pointed by `a`, `b`, and `tmp` belong to the invocation context.

A violation of the *No Heap Remote Object Assignment Rule* can be observed in the `add` method, in the statement: `lastInteger = tmp;  //throws an Exception`

Here, a reference to an object created in the creation context, `lastInteger`, holds a reference to an object created in the invocation context (`tmp`).

To overcome this violation of the NhRo assignment rule, no reference assignment should take place. Instead, the value of the object pointed by `tmp` should be *copied* to the object pointed by `lastInteger`. So, the line above should be replaced by the following: `lastInteger.setIntValue(tmp.intValue());`

5 NhRo Model for Static Environments

In this section, the architecture of the integration of the NhRo model and RMI is presented for static environments. Static means that in this case it is not possible to destroy a NhRo once it has been created. In the design of our support for the NhRo model, the following constraints are observed:

- The NhRo's creation context contains only immortal memory, and it cannot contain any scoped memory instances.
- The NhRo's invocation context uses a single LTMemory instance.
- Threads that perform up-call are the RTSJ's *no heap real time threads.*
- The communication protocol between client and server is based on TCP/IP model defined in `UnicastRemoteObject` [5].
- All network connections between clients and server are pre-established and never closed. This is achieved using JRMP *stream-op* and *mux-op* subprotocols [5].

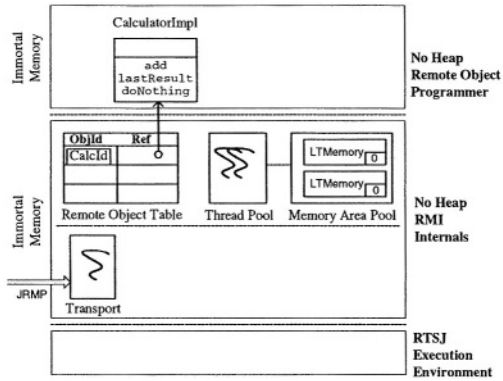


Fig. 2. Server side view of NhRo-RMI integration

Figure 2 shows our architecture at the server side for the integration of NhRo model into RMI for static environments. All objects are created in immortal memory and only those objects needed to perform remote invocations are created in scoped memory (LTMemory).

The Sun's RMI middleware solution has been slightly adapted to implement our NhRo-RMI. The key parts of our NhRo-RMI are the transport component, the remote object table, the thread pool, and the memory area pool.

- *Transport.* It allows communication between two Java virtual machines using Java's Socket API. Each transport has an associated RTSJ's no heap real time thread that monitors connections.
- *Remote Object Table.* It is the core of NhRo-RMI. It contains information of no heap remote object instances that can be remotely invoqued. Among others, this information contains remote object identifier (`ObjID`) and a pointer to remote object instance. Each time a new NhRo instance is created, its constructor attempts to register itself in this table.

- *Thread Pool.* This structure is based on RT-CORBA's thread-pool [15]. It limits the number of concurrent invocations at the server and improves server side remote invocation efficiency and predictability. Our thread pool requires that the programmer sets a single parameter, i.e., the number of no heap real time threads stored in the pool. Being the target a static environment, once created, its not possible to create threads dynamically like in RT-CORBA.
- *Memory Area Pool.* This structure is a completely new idea that comes out of the architecture. It complements the thread pool structure in the middleware for distributed real-time Java applications. The main idea behind the memory area pool is parallel to the thread pool idea. Each time a scoped memory instance is created, some memory resources in operative system are reserved. After using these resources to perform a remote invocation, why not reusing them instead of just releasing them?

As thread pools do, memory area pools also need to be dimensioned. The programmer must configure the number of instances of the pool, its size and relationship between thread pool instances, and memory areas instances. The default policy of the memory area pool requires that the programmer supplies a single parameter: the size of the memory area instances stored in the pool. All memory areas stored in the pool have the same size. The number of memory area instances stored in the memory pool equals the number of threads in thread pool. Hence, association between threads and memory areas is one-to-one; each thread uses a single LTMemory instance in a remote invocation. An RTSJ-compliant virtual machine is required to be underneath our NhRo-RMI model.

Functionality Supplied to No Heap Remote Objects

Our modified RMI that supports the NhRo model provides to NhRo objects the following functionality, just as the standar RMI:

- *Exportation.* The registry mechanism is maintained; a NhRo is registered in it in the same way as a normal remote object.
- *Remote Invocation.* A NhRo can be remotely invoqued in the same way as a normal RMI remote object.

Creation and Exportation of No Heap Remote Objects

Each time an instance of a NhRo is created, its constructor tries to register itself in the RMI internals in order to make it remotely accessible. RMI internals verify that the NhRo has been created in immortal memory. If not, it throws a remote exception notifying that the remote object instance must be created in immortal memory. The registration process implies the creation of a new ObjID instance; this ObjID is used by clients to refer to a particular remote object instance. After successful registration, the NhRo instance may be invoqued from other virtual machines.

Remote Invocation of No Heap Remote Objects

Each time the transport's thread detects that a client has started a remote invocation process; it selects a *handler thread* from the thread pool. This thread will complete the remote invocation process.

The handler thread gets a LTMemory instance from the memory area pool and enters it (using the `enter` method). This changes the default allocation context to the specified LTMemory. From this point on, all objects will be created in this memory area instance. After entering the specified memory area, the handler thread de-serializes the object identifier (ObjID), the method identifier, and the remote method parameters. All these objects are implicitly created in LTMemory. The thread uses the ObjID to query the *remote object table* and gets a reference to the NhRo instance. Using introspection, the API included in `java. lang .reflect,` the handler thread performs the remote invocation (e.g. `add` method of the remote calculator). Objects created in the remote object's method are created by default in LTMemory (e.g. object referenced by tmp). After performing the up-call, it sends the response to the client. Objects created at this stage are also created in LTMemory. Finally, after the remote invocation is performed, both handler thread and LTMemory instance return to their corresponding pools.

Realization of the NhRo Assignment Rule

In our solution, the NhRo assignment rule is implemented as a result of RTSJ assignment rules. All NhRo instances and their attributes are created in immortal memory meanwhile objects of the invocation context are in scoped memory. RTSJ assignment rule does not allow that a reference to an object created in LTMemory be stored in immortal memory. Therefore, our solution does not violate RTSJ rules. An instance to a remote object could always be referenced from the RMI internals because it is placed in immortal memory. LTMemory usage does not violate the single parent rule because LTMemory instances are never shared between two threads at same time.

Finally, we have hidden complexities of manipulation of scope stack structure to the NhRo programmer. Before invoking programmer code, threads change default allocation contexts using `enter`. The programmer only needs to know that there are two allocation contexts and that objects created in remote object methods disappear after the method has finished.

6 Empirical Results

We have developed a prototype of our architecture using TimeSys RTSJ reference implementation JVM [6] and J2ME RMI classes available in the personal package [7]. The hardware test platform is a 1.2 Ghz Pentium IV processor with 256 Mb running a Debian Woody Linux upon a Timesys Real Time Linux kernel.

Our implementation allows the creation and remote invocation of NhRo as defined in this paper. All objects needed to perform remote invocations are created in LTMemory instances avoiding use of heap memory. This eliminates garbage collection avoiding unpredictable pauses in program execution.

We have developed a trace utility: `drequiem.rtsj.tracer`. It allows reading the system clock with a precission of microseconds (using *gettimeofday* C function) and quering about heap, immortal, and current allocation context memory availability. The tracing mechanism does not consume memory in the sense that all objects needed to store information are created in immortal memory in the tracer constructor. Each trace takes 40us and introduces a jitter of 8us, numbers obtained due to the fact that we have used JNI [17] to access clock system.

Determining minimum size of LTMemory instances stored in the Memory Area Pool

The consumed memory in a remote object invocation depends on which particular method of a remote object is invoqued. As in our approach, in the memory area pool we have defined a single configuration parameter (the memory associated to each scoped memory), we have to dimension it for the worst case.

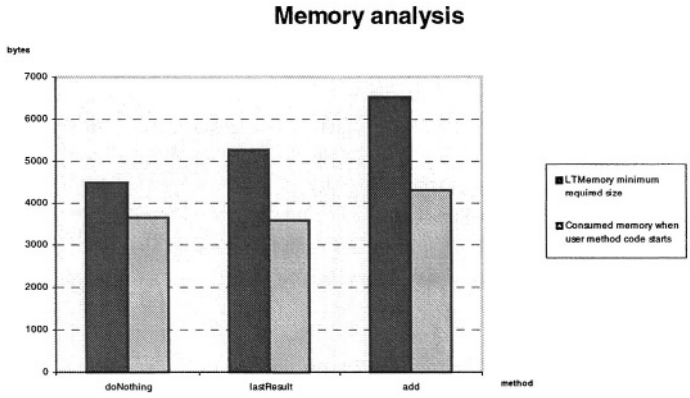


Fig. 3. Consumed memory in the remote invocation of calculator

Results of figure 3 show that the LTMemory instances created in the memory area pool always present a minimum size. This is because even when an empty `void` method like `doNothing` is invoqued, there are still objects that are created, such as the ObjID or those that are related to the reflection mechanism. The amount of memory consumed by these objects is 4338 bytes. On the other hand, as all instances of the memory pool have the same size, there is a waste of memory because not all remote invocations create the same number of objects, and we are using same LTMemory size for all. In our case this ratio is 1.4. We have observed that the memory consumed in remote invocations has an asymmetrical behaviour. It is consumed more memory before remote object code is invoqued than after such invocation. This is due to two effects: (1) de-serialization is a more memory consuming process than serialization, and (2) the invocation of remote objects that is performed using Java reflection creates many Java objects.

As the main conclusion, our prototype results show that memory consumption in processing the JRMP protocol adds a high overhead to program execution. Using our

prototype, it is possible to build applications with a best-case response time of 1.7 milliseconds where the remote object invocation process consumes at least 4.3 KB of memory.

7 Conclusions

In this paper, we have shown how it is possible to define a pure region based model for the Remote Object model of RMI using RTSJ. Our solution avoids pauses introduced by the garbage collector, but it introduces changes in the remote object programming paradigm and middleware infrastructures. From the perspective of programmers, we have introduced a new assignment rule, the *no heap remote object assignment rule* that separates objects that belong to the state of the remote object from those that belongs to the remote invocation process. In the middleware layer, we have defined a new entity, the *memory area pool* that is the responsible of destroying temporal objects that are created during the remote invocation process. Also, we have freed the programmer from the task of controlling the default allocation context; this action is now performed by RMI's middleware internals.
Finally, we have shown how to apply the NhRo definition to static environments. In this case, it is possible to implement it without violating RTSJ rules. In order to validate it, we have successfully developed a prototype using SUN J2ME RMI and the RTSJ reference implementation of TimeSys.

References

[1] RTJEG,"The Real-Time Specification for Java", Addison Wesley, 2000. Available on-line 4/6/2004 at http://www.rtj.org
[2] J. Consortium Inc. "Core Real-Time Extensions for the Java Platform". Technical Report. Sept 2, 2000. Available on-line 4/6/04 at http://www.j-consortium.org
[3] David F. Bacon, Perry Cheng, V.T. Rajan: Metronome: A Simplier Approach to Garbage Collection In Real Time Systems. Workshop on Java Technologies for Real-Time and Embedded Systems (Catania, Sicily, November 2003), Proceedings of the OTM Workshops, R. Meersman and Z. Tari, eds., Lecture Notes in Computer Science vol. 2889, pp. 466-478
[4] Ann Wollrath, Roger Riggs, Jim Waldo: A Distributed Object Model for Java Systems. USENIX Conference on Object-Oriented Technologies and Systems, Toronto Ontario, Canada 1996
[5] Java RMI Remote Method Invocation. Available on line 04/06/04 at ftp://ftp.java.sun.com/docs/j2se1 .4/rmispec-1.4.pdf
[6] Real Time Java Reference Implementation RTSJ-RI Available on line 04/06/04 at http://www.timesys.com
[7] JSR-62, Java 2 Micro-Edition Personal Profile v 1.0 reference implementation. Available 04/06/04 at http://www.sun.com
[8] Objective Interface Systems, Inc.: JCP RTSJ and Real-time CORBA Synthesis, realtime/2003-05-03. Available 05/05/04 at http://www.omg.org/docs/realtime/03-05-03.pdf
[9] J Consortium: J-Consortium RT Core and Real-time CORBA Synthesis, realtime/2003-05-04. Available 05/05/04 at http://www.omg.org/docs/realtime/03-05-04.pdf

[10] JSR-50, Distributed Real Time Specification for Java (DRTSJ). Available 05/05/04 at http://jcp.org/en/jsr/detail?id=050
[11] Wellings A. J., Clark R., Jensen D., Wells D.: A Framework for Integrating the Real-Time Specification for Java and Java's Remote Method Invocation. Proceedings of 5th IEEE International Symposium on Object-Oriented Real-Time Distributed Computting (ISORC) 2002 12-22.
[12] A. Corsaro, R. K Cytron: Efficient memory-reference checks for real-time java. In Proceedings of 2003 ACM SIGPLAN conference on Language, compiler and tool for embedded systems, pp 51-58. ACM Press, 2003.
[13] Peter C. Dibble, Real-Time Java Platform Programming. The Sun Microsystems Press. Java Series. ISBN 0-13-028261-8.
[14] Arvind S. Krishna, Douglas C. Schmidt and Raymond Klefstad. Enhacing Real-time Corba via Real Time Java features. In proceedings of 25th lEEEInternational Conference on Distributed Computing Systems (ICDS), May 23-26, 2004 Tokyo, Japan.
[15] OMG, Real time Corba,v2.0. formal/03-11-01. Available 04/04/04 at http://www.omg.org/cgi-bin/apps/do_doc?formal/03-11-01.pdf
[16] Java Object Serialization Specification. Available 04/04/04 at http://java.sun.com/j2se/1.4.2/docs/guide/serialization/spec/serialTOC.html
[17] Ling, S. The Java Native Interface Programmer's Guide and Specification, Addison Wesley Publishing Co. 320 pages. June 10, 1999
[18] Jagun Kwon, Wellings, A.; King, s; Predictable memory utilization in Ravenscar-Java profile. In proceedings of Object-Oriented Real-Time Distributed Computing 2003. 6th IEEE International Symposium on, 14-16 May 2003 pp. 24-17
[19] Bollela, G.; Canham, T; Carson V; et al. : Programming with Non-Heap Memory in the Real Time Specification for Java. 18th Annual ACM SIGPLAN Conference on Object Oriented Programming, Systems, Languages and Architectures (OOSPLA 2003), Anaheim, CA, October 2003.

A Time Predictable Instruction Cache for a Java Processor

Martin Schoeberl

JOP.design, Vienna, Austria
martin@jopdesign.com

Abstract. Cache memories are mandatory to bridge the growing gap between CPU speed and main memory access time. Standard cache organizations improve the average execution time but are difficult to predict for worst case execution time (WCET) analysis. This paper proposes a different cache architecture, intended to ease WCET analysis. The cache stores complete methods and cache misses occur only on method invocation and return. Cache block replacement depends on the call tree, instead of instruction addresses.

1 Introduction

Worst case execution time (WCET) analysis [1] of real–time programs is essential for any schedulability analysis. To provide a low WCET value, a good processor model is necessary. However, the architectural advancement in modern processor designs is dominated by the rule: '*Make the common case fast*'. This is the opposite of '*Reduce the worst case'* and complicates WCET analysis.

Cache memory for the instructions and data is a classic example of this paradigm. Avoiding or ignoring this feature in real–time systems, due to its unpredictable behavior, results in a very pessimistic WCET value. Plenty of effort has gone into research into integrating the instruction cache in the timing analysis of tasks [2,3] and the cache's influence on task preemption [4,5]. The influence of different cache architectures on WCET analysis is described in [6].

We will tackle this problem from the architectural side — an instruction cache organization in which simpler and more accurate WCET analysis is more important than average case performance. In this paper, we will propose a method cache with a novel replacement policy. The instruction set of the Java virtual machine contains only relative branches, and a method is therefore only left when a return instruction has been executed. It has been observed that methods are typically short [7] in Java applications. These properties are utilized by a cache architecture that stores complete methods. A complete method is loaded into the cache on both invocation and return. This cache fill strategy lumps all cache misses together and is very simple to analyze.

2 Cache Performance

In real–time systems we prefer time predictable architectures over those with a high average performance. However, performance is still important. In this section we will

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 371–382, 2004.
© Springer-Verlag Berlin Heidelberg 2004

give a short overview of the formulas from [8] that are used to calculate the cache's influence on execution time. We will extend the single measurement *miss rate* to a two value set, memory read and transaction rate, that is architecture independent and better reflect the two properties (bandwidth and latency) of the main memory. To evaluate cache performance, MEM_{clk} memory stall cycles are added to the CPU execution time (t_{exe}) equation:

$$t_{exe} = (CPU_{clk} + MEM_{clk}) \times t_{clk}$$
$$MEM_{clk} = Misses \times MP_{clk}$$

The miss penalty MP_{clk} is the cost per miss, measured in clock cycles. When the instruction count *IC* is given as the number of instructions executed, *CPI* the average clock cycles per instruction and the number of misses per instruction, we obtain the following result:

$$CPU_{clk} = IC \times CPI_{exe}$$
$$MEM_{clk} = IC \times \frac{Misses}{Instruction} \times MP_{clk}$$
$$t_{exe} = IC \times (CPI_{exe} + \frac{Misses}{Instruction} \times MP_{clk}) \times t_{clk}$$

As this paper is only concerned with the instruction cache, we will split the memory stall cycles into misses caused by the instruction fetch and misses caused by data access.

$$CPI = CPI_{exe} + CPI_{IM} + CPI_{DM}$$

CPI_{exe} is the average number of clock cycles per instruction, given an ideal memory system without any stalls. CPI_{IM} are the additional clock cycles caused by instruction cache misses and CPI_{DM} the data miss portion of the CPI. This split between instruction and data portions of the CPI better reflects the split of the cache between instruction and data cache found in actual processors.

The misses per instruction are often reported as misses per 1000 instructions. However, there are several drawbacks to using a single number:

Architecture dependent: The average number of memory accesses per instruction differs greatly between a RISC processor and the Java Virtual Machine (JVM). A typical RISC processor needs one memory word (4 bytes) per instruction word, and about 40% of the instructions [8] are *load* or *store* instructions. Using the example of a 32-bit RISC processor, this results in 5.6 bytes memory access per instruction. The average length of a JVM bytecode instruction is 1.7 bytes and about 18% of the instructions access the memory for data load and store.

Block size dependent: Misses per instruction depends subtly on the block size. On a single cache miss, a whole block of the cache is filled. Therefore, the probability that a future instruction request is a hit is higher with a larger block size. However, a larger block size results in a higher miss penalty as more memory is transferred.

Main memory is usually composed of DRAMs. Access time to this memory is measured in terms of latency (the time taken to access the first word of a larger block) and bandwidth

(the number of bytes read or written in a single request per time unit). These two values, along with the block size of a cache, are used to calculate the miss penalty:

$$MP_{clk} = Latency + \frac{Block\ size}{Bandwidth}$$

To better evaluate different cache organizations and different instruction sets (RISC versus JVM), we will introduce two performance measurements — memory bytes read per instruction byte and memory transactions per instruction byte:

$$MBIB = \frac{Memory\ bytes\ read}{Instruction\ bytes}$$
$$MTIB = \frac{Memory\ transactions}{Instruction\ bytes}$$

These two measures are closely related to memory bandwidth and latency. With these two values and the properties of the main memory, we can calculate the average memory cycles per instruction byte $MCIB$ and CPI_{IM}, i.e. the values we are concerned in this paper.

$$MCIB = (\frac{MBIB}{Bandwith} + MTIB \times Latency)$$
$$CPI_{IM} = MCIB \times Instruction\ length$$

The misses per instruction can be converted to MBIB and MTIB when the following parameters are known: the average instruction length of the architecture, the block size of the cache and the miss penalty in latency and bandwidth. We will examine this further in the following example:

We will use as our example a RISC architecture with a 4 bytes instruction length, an 8 KB instruction cache with 64-byte blocks and a miss rate of 8.16 per 1000 instructions [8]. The miss penalty is 100 clock cycles. The memory system is assumed to deliver one word (4 bytes) per cycle. Firstly, we need to calculate the latency of the memory system.

$$Latency = MP_{clk} - \frac{Blocksize}{Bandwidth} = 100 - \frac{64}{4} = 84 \text{ clock cycles}$$

With $Miss\ rate = \frac{Cache\ miss}{Cache\ access}$, we obtain MBIB.

$$MBIB = \frac{Memory\ bytes\ read}{Instruction\ bytes} = \frac{Cache\ miss \times Block\ size}{Cache\ access \times Instruction\ length}$$
$$= Miss\ rate \times \frac{Block\ size}{Instruction\ length} = 8.16 \times 10^{-3} \times \frac{65}{4} = 0.131$$

MTIB is calculated in a similar way:

$$MTIB = \frac{Memory\ transactions}{Instruction\ bytes} = \frac{Cache\ miss}{Cache\ access \times Instruction\ length}$$
$$= \frac{Miss\ rate}{Instruction\ length} = \frac{8.16 \times 10^{-3}}{4} = 2.04 \times 10^{-3}$$

For a quick check, we can calculate CPI_{IM}:

$$MCIB = \frac{MBIB}{Bandwith} + MTIB \times Latency = \frac{0.131}{4} + 2.04 \times 10^{-3} \times 84 = 0.204$$
$$CPI_{IM} = MCIB \times Instruction\ length = 0.204 \times 4 = 0.816$$

This is the same value as that which we get from using the miss rate with the miss penalty. However, MBIB and MTIB are architecture independent and better reflect the latency and bandwidth of the main memory.

$$CPI_{IM} = Miss\ rate \times Miss\ penalty = 8.16 \times 10^{-3} \times 100 = 0.816$$

3 Proposed Cache Solution

In this section, we will develop a solution with a predictable cache. Typical Java programs consist of short methods. There are no branches out of the method and all branches inside are relative. In the proposed architecture, the full code of a method is loaded into the cache before execution. The cache is filled on calls and returns. This means that all cache fills are lumped together with a known execution time. The full loaded method and relative addressing inside a method also result in a simpler cache. Tag memory and address translation are not necessary.

3.1 Single Method Cache

A single method cache, although less efficient, can be incorporated very easily into the WCET analysis. The time needed for the memory transfer need to be added to the invoke and return instruction. The main disadvantage of this single method cache is the high overhead when a complete method is loaded into the cache and only a small fraction of the code is executed. This issue is similar to that encountered with unused data in a cache line. However, in extreme cases, this overhead can be very high. The second problem can be seen in following example:

```
foo() {
    a();
    b();
}
```

The main drawback of the single method cache is the multiple cache fill of foo() on return from methods a() and b(). In a conventional cache design, if these three methods can be fitted in the cache memory at the same time and there is no placement conflict, each method is only loaded once. This issue can be overcome by caching more than one method. The simplest solution is a two block cache.

3.2 Two Block Cache

The two block cache can hold up to two methods in the cache. This results in having to decide which block is replaced on a cache miss. With only two blocks, Least-Recently

Table 1. Cache load and hit example with the two block cache

Instruction	Block 1	Block 2	Cache
foo()	foo	–	load
a()	foo	a	load
return	foo	a	hit
b()	foo	b	load
return	foo	b	hit

Used (LRU) is trivial to implement. The code sequence now results in the cache loads and hits as shown in Table 1. With the two block cache, we have to double the cache memory or use both blocks for a single large method. The WCET analysis is slightly more complex than with a single block. A short history of the invocation sequence has to be used to find the cache fills and hits. A memory (similar to the tag memory) with one word per block is used to store a reference to the cached method. However, this memory can be slower than the tag memory as it is only accessed on invocation or return, rather than on every cache access.

We can improve the hit rate by adding more blocks to the cache. If only one block per method is used, the cache size increases with the number of blocks. With more than two blocks, LRU replacement policy means that another word is needed for every block containing a use counter that is updated on every invoke and return. During replacement, this list is searched for the LRU block. Hit detection involves a search through the list of the method references of the blocks. If this search is done in microcode, it imposes a limit on the maximum number of blocks.

3.3 Variable Block Cache

Several cache blocks, all of the size as the largest method, are a waste of cache memory. Using smaller block sizes and allowing a method to spawn over several blocks, the blocks become very similar to cache lines. The main difference from a conventional cache is that the blocks for a method are all loaded at once and need to be consecutive.

Choosing the block size is now a major design tradeoff. Smaller block sizes allow better memory usage, but the search time for a hit also increases.

With varying block numbers per method, an LRU replacement becomes impractical. When the method found to be LRU is smaller than the loaded method, this new method invalidates two cached methods. For the replacement, we will use a pointer *next* that indicates the start of the blocks to be replaced on a cache miss. Two practical replace policies are:

Next block: At the very first beginning, *next* points to the first block. When a method of length l is loaded in the block n, *next* is updated to $(n + l)$ % *block count.*

Stack oriented: *next* is updated in the same way as before on a method load. It is also updated on a method return — independent of a resulting hit or miss — to point to the first block of the leaving method.

Table 2. Next block replacement policy

Instruction	a()	b()	ret	c()	ret	b()	ret	c()	ret	b()	ret	c()	ret
Block 1	A	→a	→a	C	A	a	a	a	a	B	b	→-	→-
Block 2	A	a	a	→-	A	a	a	a	a	→-	A	a	a
Block 3	→-	B	b	b	→b	→b	→b	C	c	c	A	a	a
Block 4	-	B	b	b	b	b	b	→-	→-	B	→b	C	c
Fill count	2	4		5	7			8		10	12	13	

Table 3. Stack oriented replacement policy

Instruction	a()	b()	ret	c()	ret	b()	ret	c()	ret	b()	ret	c()	ret
Block 1	A	→a	a	a	a	→a	a	a	a	→a	a	a	a
Block 2	A	a	a	a	a	a	a	a	a	a	a	a	a
Block 3	→-	B	→b	C	→c	B	→b	C	→c	B	→b	C	→c
Block 4	-	B	b	→-	-	B	b	→-	-	B	b	→-	-
Fill count	2	4		5		7		8		10		11	

We will show these different replacement policies in an example with three methods: a(), b() and c() of block sizes 2, 2 and 1. The cache consists of 4 blocks and is therefore too small to hold all the methods during the execution of the following code fragment:

```
a() {
    for (;;) {
        b();
        c();
    }
}
```

Tables 2 and 3 show the cache content during program execution for both replacement policies. The content of the cache blocks is shown after the execution of the instruction. An uppercase letter indicates that this block is newly loaded. A right arrow depicts the block to be replaced on a cache miss (the *next* pointer). The last row shows the number of blocks that are filled during the execution of the program.

In this example, the stack oriented approach needs slightly fewer fills, as only methods b() and c() are exchanged and method a() stays in the cache. However, if, for example, method b() is the size of one block, all methods can be held in the cache using the the *next block* policy, but b() and c() would be still exchanged using the *stack* policy. Therefore, the first approach is used in the proposed cache.

4 WCET Analysis

The proposed instruction cache is designed to simplify WCET analysis. Due to the fact that all cache misses are included in two instructions *(invoke* and *return)* only, the

instruction cache can be ignored on all other instructions. The time needed to load a complete method is calculated using the memory properties (latency and bandwidth) and the length of the method. On an invoke, the length of the invoked method is used, and on a return, the method length of the caller.

With a single method cache this calculation can be further simplified. For every invoke there is a corresponding return. That means that the time needed for the cache load on return can be included in the time for the invoke instruction. This is simpler because both methods, the caller and the callee, are known at the occurrence of the invoke instruction. The information about which method was the caller need not be stored for the return instruction to be analyzed.

With more than one method in the cache, a cache hit detection has to be performed as part of the WCET analysis. If there are only two blocks, this is trivial, as (i) a hit on invoke is only possible if the method is the same as the last invoked (e.g. a single method in a loop) and (ii) a hit on return is only possible when the method is a leave in the call tree. In the latter case, it is always a hit.

When the cache contains more blocks (i.e. more than two methods can be cached), a part of the call tree has to be taken into account for hit detection. The variable block cache further complicates the analysis, as the method length also determines the cache content. However, this analysis is still simpler than a cache modeling of a direct mapped instruction cache, as cache block replacement depends on the call tree instead of instruction addresses.

In traditional caches, data access and instruction cache fill requests can compete for the main memory bus. For example, a load or store at the end of the processor pipeline competes with an instruction fetch that results in a cache miss. One of the two instructions is stalled for additional cycles by the other instructions. With a data cache, this situation can be even worse. The worst case scenario for the memory stall time for an instruction fetch or a data load is two miss penalties when both cache reads are a miss. This unpredictable behavior leads to very pessimistic WCET bounds.

A *method cache,* with cache fills only on invoke and return, does not interfere with data access to the main memory. Data in the main memory is accessed with *getfield* and *putfield,* instructions that never overlap with *invoke* and *return.* This property removes another uncertainty found in traditional cache designs.

5 Caches Compared

In this section, we will compare the different cache architectures in a quantitative way. Although our primary concern is predictability, performance remains important. We will therefore first present the results from a conventional direct–mapped instruction cache. These measurements will then provide a baseline for the evaluation of the proposed architecture.

Cache performance varies with different application domains. As the proposed system is intended for real-time applications, the benchmark for these tests should reflect this fact. However, there are no standard benchmarks available for embedded real–time systems. A real–time application was therefore adapted to create this benchmark. The application is from one node of a distributed motor control system [9]. A simulation of

the environment (sensors and actors) and the communication system (commands from the master station) forms part of the benchmark for simulating the real–world workload.

The data for all measurements was captured using a simulation of a Java processor [10] and running the application for 500,000 clock cycles. During this time, the major loop of the application was executed several hundred times, effectively rendering any misses during the initialization code irrelevant to the measurements.

5.1 Direct–Mapped Cache

Table 4 gives the memory bytes and memory transactions per instruction byte for a standard direct–mapped cache. As we can see from the values for a cache size of 4 KB, the kernel of the application is small enough to fit completely into the 4 KB cache. The cache performs better (i.e. fewer bytes are transferred) with smaller block sizes. With smaller block sizes, the chance of unused data being read is reduced and the larger number of blocks reduces conflict misses. However, reducing the block size also increases memory transactions (MTIB), which directly relates to memory latency.

Which configuration performs best depends on the relationship between memory bandwidth and memory latency. Examples of average memory access times in cycles per instruction byte for different memory technologies are provided in Table 5. The third column shows the cache performance for a Static RAM (SRAM), which is very common in embedded systems. A latency of 1 clock cycle and an access time of 2 clock cycles per 32-bit word are assumed. For the synchronous DRAM (SDRAM) in the forth column, a latency of 5 cycles (3 cycle for the row address and 2 cycle CAS latency) is assumed. The memory delivers one word (4 bytes) per cycle. The Double Data Rate (DDR) SDRAM in the last column has an enhanced latency of 4.5 cycles and transfers data on both the rising and falling edge of the clock signal.

The data in bold give the best block size for different memory technologies. As expected, memories with a higher latency and bandwidth perform better with larger block sizes. For small block sizes, the latency clearly dominates the access time. Although the SRAM has half the bandwidth of the SDRAM and a quarter of the DDR, with a block size of 8 bytes it is faster than the DRAM memories. In most cases a block size of 16

Table 4. Direct–mapped cache

Cache size	Block size	MBIB	MTIB
1 KB	8	0.28	0.035
1 KB	16	0.38	0.024
1 KB	32	0.58	0.018
2 KB	8	0.17	0.022
2 KB	16	0.25	0.015
2 KB	32	0.41	0.013
4 KB	8	0.00	0.001
4 KB	16	0.01	0.000
4 KB	32	0.01	0.000

Table 5. Direct–mapped cache, average memory access time

Cache size	Block size	SRAM	SDRAM	DDR
1 KB	8	**0.18**	0.25	0.19
1 KB	16	0.22	**0.22**	0.16
1 KB	32	0.31	0.24	**0.15**
2 KB	8	**0.11**	0.15	0.12
2 KB	16	0.14	**0.14**	**0.10**
2 KB	32	0.22	0.17	0.11

bytes is the fastest solution and we will therefore use this configuration for comparison with the following cache solutions.

5.2 Fixed Block Cache

Cache performance for single method per block architectures is shown in Table 6. A single block that has to be filled on every invoke and return requires considerable overheads. More than twice the amount of data is read from the main memory than is consumed by the processor.

The solution with two blocks for two methods performs almost twice as well as the simple one method cache. This is due to the fact that, for all leaves in the call tree, the caller method can be found on return. If the block count is doubled again, the number of misses is reduced by a further 25%, but the cache size also doubles. For this measurement, an LRU replacement policy applies for the two and four block caches.

Table 6. Fixed block cache

Type	Cache size	MBIB	MTIB
Single method	1 KB	2.32	0.021
Two blocks	2 KB	1.21	0.013
Four blocks	4 KB	0.90	0.010

The same memory parameters as in the previous section are also used in Table 7. As MBIB and MTBI show the same trend as a function of the number of blocks, this is reflected in the access time in all three memory examples.

5.3 Variable Block Cache

Table 8 shows the cache performance of the proposed solution, i.e. of a method cache with several blocks per method, for different cache sizes and number of blocks. For this measurement, a *next block* replacement policy applies.

In this scenario, as the MBIB is very high at a cache size of 1 KB and almost independent of the block count, the cache capacity is seen to be clearly dominant. The

Table 7. Fixed block cache, average memory access time

Type	Cache size	SRAM	SDRAM	DDR
Single Method	1 KB	1.18	0.69	0.39
Two blocks	2 KB	0.62	0.37	0.21
Four blocks	4 KB	0.46	0.27	0.16

Table 8. Variable block cache

Cache size	Block count	MBIB	MTIB
1 KB	8	0.80	0.009
1 KB	16	0.71	0.008
1 KB	32	0.70	0.008
1 KB	64	0.70	0.008
2 KB	8	0.73	0.008
2 KB	16	0.37	0.004
2 KB	32	0.24	0.003
2 KB	64	0.12	0.001
٨B	8	0.73	0.008
4 KB	16	0.25	0.003
4 KB	32	0.01	0.000
4 KB	64	0.00	0.000

most interesting cache size with this benchmark is 2 KB. Here, we can see the influence of the number of blocks on both performance parameters. Both values benefit from more blocks. However, a higher block count requires more time or more hardware for hit detection. With a cache size of 4 KB and enough blocks, the kernel of the application completely fits into the variable block cache, as we have seen with a 4 KB traditional cache. From the gap between 16 and 32 blocks (within the 4 KB cache), we can say that the application consists of fewer than 32 different methods.

It can be seen that even the smallest configuration with a cache size of 1 KB and only 8 blocks outperforms fixed block caches with 2 or 4 KB in both parameters (MBIB and MTIB). In most configurations, MBIB is higher than for the direct–mapped cache. It is very interesting to note that, in all configurations (even the small 1 KB cache), MTIB is lower than in all 1 KB and 2 KB configurations of the direct–mapped cache. This is a result of the complete method transfers when a miss occurs and is clearly an advantage for main memory systems with high latency. As in the previous examples, Table 9 shows the average memory access time per instruction byte for three different main memories.

The variable block cache directly benefits from the low MTBI with the DRAM memories. When comparing the values between SDRAM and DDR, we can see that the bandwidth affects the memory access time in a way that is approximately linear. The high latency of these memories is completely hidden. The configuration with 16 or more blocks and dynamic RAMs outperforms the direct–mapped cache of the same size. As expected, a memory with low latency (the SRAM in this example) depends on the MBIB values. The variable block cache is slower than the direct–mapped cache in the 1 KB

Table 9. Variable block cache, average memory access time

Cache size	Block count	SRAM	SDRAM	DDR
1 KB	8	0.41	0.24	0.14
1 KB	16	0.36	0.22	0.12
1 KB	32	0.36	0.21	0.12
1 KB	64	0.36	0.21	0.12
2 KB	8	0.37	0.22	0.13
2 KB	16	0.19	0.11	0.06
2 KB	32	0.12	0.08	0.04
2 KB	64	0.06	0.04	0.02

Table 10. Caches compared

Cache type	MBIB	MTIB
Single method	2.32	0.021
Two blocks	1.21	0.013
Variable block (16)	0.37	0.004
Variable block (32)	0.24	0.003
Direct mapped	0.25	0.015

configuration because of the higher MBIB (0.7 compared to 0.3-0.6), and performs very similarly at a cache size of 2 KB. In Table 10, the different cache solutions with a size of 2 KB are summarized. All full method caches with two or more blocks have a lower MTIB than a conventional cache solution. This becomes more important with increasing latency in main memories. The MBIB value is only quite high for one or two methods in the cache. However, the most surprising result is that the variable block cache with 32 blocks outperforms a direct–mapped cache of the same size at both values.

We can see that predictability is indirectly related to performance — a trend we had expected. The most predictable solution with a single method cache performs very poorly compared to a conventional direct–mapped cache. If we accept a slightly more complex WCET analysis (taking a small part of the call tree into account), we can use the two block cache that is about two times better. With the variable block cache, it could be argued that the WCET analysis becomes too complex, but it is nevertheless simpler than that with the direct–mapped cache. However, every hit in the two block cache will also be a hit in a variable block cache (of the same size). A tradeoff might be to analyze the program by assuming a two block cache but using a version of the variable block cache.

6 Conclusion

In this paper, we have extended the single cache performance measurement *miss rate* to a two value set, memory read and transaction rate, in order to perform a more detailed evaluation of different cache architectures. From the properties of the Java language — usually small methods and relative branches — we derived the novel idea of a *method*

cache, i.e. a cache organization in which whole methods are loaded into the cache on method invocation and the return from a method. This cache organization is time predictable, as all cache misses are lumped together in these two instructions. Using only one block for a single method introduces considerable overheads in comparison with a conventional cache, but is very simple to analyze. We extended this cache to hold more methods, with one block per method and several smaller blocks per method.

Comparing these organizations quantitatively with a benchmark derived from a real-time application, we have seen that the variable block cache performs similarly to (and in one configuration even better than) a direct–mapped cache, in respect of the bytes that have to be filled on a cache miss. In all configurations and sizes of the variable block cache, the number of memory transactions, which relates to memory latency, is lower than in a traditional cache.

Filling the cache only on method invocation and return simplifies WCET analysis and removes another source of uncertainty, as there is no competition for main memory between instruction cache and data cache.

References

1. Puschner, P., Koza, C.: Calculating the maximum execution time of real-time programs. Real-Time Syst. **1** (1989) 159–176
2. Arnold, R., Mueller, F., Whalley, D., Harmon, M.: Bounding worst-case instruction cache performance. In: IEEE Real-Time Systems Symposium. (1994) 172–181
3. Healy, C., Whalley, D., Harmon, M.: Integrating the timing analysis of pipelining and instruction caching. In: IEEE Real-Time Systems Symposium. (1995) 288–297
4. Lee, C.G., Hahn, J., Seo, Y.M., Min, S.L., Ha, R., Hong, S., Park, C.Y., Lee, M., Kim, C.S.: Analysis of cache-related preemption delay in fixed-priority preemptive scheduling. IEEE Trans. Comput. **47** (1998) 700–713
5. Busquets-Mataix, J.V., Wellings, A., Serrano, J.J., Ors, R., Gil, P.: Adding instruction cache effect to schedulability analysis of preemptive real-time systems. In: IEEE Real-Time Technology and Applications Symposium (RTAS '96), Washington - Brussels - Tokyo, IEEE Computer Society Press (1996) 204–213
6. Heckmann, R., Langenbach, M., Thesing, S., Wilhelm, R.: The influence of processor architecture on the design and results of WCET tools. Proceedings of the IEEE **91** (2003)
7. Power, J., Waldron, J.: A method-level analysis of object-oriented techniques in java. Technical report, Department of Computer Science, NUI Maynooth, Ireland (2002)
8. Hennessy, J., Patterson, D.: Computer Architecture: A Quantitative Approach, 3rd ed. Morgan Kaufmann Publishers Inc., Palo Alto, CA 94303 (2002)
9. Schoeberl, M.: Using a Java optimized processor in a real world application. In: Proceedings of the First Workshop on Intelligent Solutions in Embedded Systems (WISES 2003), Austria, Vienna (2003) 165–176
10. Schoeberl, M.: JOP: A Java optimized processor. In: Workshop on Java Technologies for Real-Time and Embedded Systems. Volume LNCS 2889., Catania, Italy (2003) 346–359

Indoor Positioning for Location-Aware Applications on Java-Based Mobile Devices

Cristiano di Flora[1], Massimo Ficco[2], and Stefano Russo[1]

[1] Dipartimento di Informatica e Sistemistica
Università di Napoli "Federico II"
Via Claudio 21, 80125 - Napoli, Italy
{diflora, stefano.russo}@unina.it
[2] Consorzio Interuniversitario Nazionale per l'Informatica
Via Diocleziano 328, 80124 - Napoli, Italy
{massimo.ficco}@napoli.consorzio-cini.it

Abstract. The Java Community Process (JCP) has recently finalized a Java Specification Request (JSR) to cope with location-awareness (JSR-179) in Connected Limited Device Configuration (CLDC). Implementations of this specification may rely on several location methods, including satellite based methods like GPS, as well as short-range positioning methods based on Received Signal Strength (RSS). Though RSS is a good location fingerprint for indoor positioning, no standard technique to tailor RSS-based approaches to the JSR-179 API has been proposed yet. In this paper we propose such a technique, and we evaluate its effectiveness through a Bluetooth-based prototype. Specifically, we show how to extend the Java APIs for Bluetooth (JSR-82) in order to provide the Location API with RSS-based position-information. Moreover, we show how to adapt RSS-based approaches to Location-API's semantics. Our solution is based on the insertion of a specific component into the JSR-82 API. This is in charge of producing all the information needed to build Location objects as defined in the JSR-179. We evaluate the effectiveness of the approach by examining preliminary experimental results obtained from the first system prototype.

1 Introduction

The proliferation of handheld mobile devices and wireless networks is preparing the ground to the development of novel location-aware systems. During the last years a plenty of works presented systems and technologies for automatic location-sensing of either people or devices [1,2]. Each solution solves a different problem or supports different applications. Many mobile devices (e.g. smart-phones and PDAs) now support the Mobile Information Device Profile (MIDP) of the Java 2 Micro Edition (J2ME) platform. This platform provides a common yet flexible computing and communication environment that could be fitted for devices having different capabilities. Moreover, currently most of mobile devices have a wireless network device such as IEEE 802.11 and Bluetooth adapters. Since in future mobile systems several technologies will coexist, such systems will need a high-level Application Programming Interface for

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 383–393, 2004.
© Springer-Verlag Berlin Heidelberg 2004

technology-independent location sensing [3]. As for location-aware computing, no single location-sensing technology is likely to become dominant, since each technology can either satisfy different requirements or suit different devices. In order to deal with this heterogeneity, the Java Community Process (JCP) has recently finalized a Java Specification Request (JSR) to define a Location API (JSR-179) [4] for MIDP-compliant devices. This package provides applications with functionality for *i*) obtaining information about location and orientation of the mobile device, and *ii*) accessing a shared database of known locations (the so-called Landmarks). These specifications may be implemented by means of existing location methods, including satellite based methods like GPS, as well as short-range positioning methods.

Though the Received Signal Strength (RSS) has been used as a good location-fingerprint by several indoor positioning techniques [5,6], no standard approach to tailor such techniques to the JSR-179 API has been proposed yet. In this paper we propose an implementation of these specifications, and we evaluate its effectiveness by providing experimental results obtained from a bluetooth-based system. Specifically, we show how we extended the Java APIs for Bluetooth (JSR-82) [7] in order to provide the Location API with RSS-based position-information. Our solution is based on the insertion of a new component, called `RSSI_Provider`, into the JSR-82 API. This is in charge of producing information about signal strength, which could be needed to build `Location` objects. We bridged the two APIs by implementing a specific `LocationProvider` class (i.e. a location-providing module, generating `Locations`, defined in JSR-179), which exploits the `RSSI_Provider` to generate `Locations`. The proposed solution can suit a wide variety of mobile devices, since it requires no additional positioning device but a bluetooth adapter. Based on the designed extensions, we propose an implementation of the `LocationProvider`, which relies on a specific RSS-based indoor positioning technique [8].

The rest of the paper is organized as follows. In Section 2 we describe several approaches to developing Indoor-positioning systems for mobile devices. Section 3 casts some light on the Location API. In Section 4 we present our solution with some comments about the current prototype implementation. In Section 5 we provide preliminary experimental results. Section 6 concludes the paper and outlines the directions of our future research.

2 Related Work

Several techniques can be used to detect a mobile user's location [5]. The work in [2] proposed a taxonomy for location-sensing systems. This taxonomy is based on several parameters, such as the physical phenomena used for determining location, the form factor of the sensing apparatus, power requirements, infrastructure, and resolution in time and space. As for indoor positioning, in [8,9] location is calculated as a function of the proximity to a known set of points. Ekahau's Positioning Engine 2.0 [10] is a Java-based positioning server providing PC and PDA location coordinates and tracking features to client applica-

tions. Its tracking algorithm is based on Received Signal Strength Intensity from IEEE 802.11 Wireless Access Points. Several systems that use a triangulation technique have been proposed [1,6,11,12]; such systems provide location information by measuring either the distance between known points or the relative angle with respect to known referring poles. However, the presence of thick walls and channel-degradation phenomena (e.g., interference, and multipath fading) can make configuration of triangulation-based systems quite a complex task, and may require sophisticated sensor devices and/or more than three access points to leverage the accuracy of such techniques. This can make it hard to develop and maintain these positioning infrastructures. Conversely, our approach is easier than triangulation based approaches, for its infrastructure consists of no additional positioning devices but a cheap and easy-to-configure bluetooth adapter. Moreover, the mentioned solutions mostly depend on different programming languages than Java. They also rely on different technologies, such as Bluetooth, IEEE 802.11 RF, GPS, and ad-hoc solutions. Unfortunately, to the best our knowledge, they lack a common high-level software application programming interface for location sensing. Thus using these solutions may result in ever heterogeneous applications.

As no single indoor location-sensing technology is likely to become a de-facto standard, personal and mobile devices will need a high-level software application programming interface for technology-independent location sensing [3]. Several organizations have recently proposed location APIs to leverage the interoperability of outdoor positioning systems. The Open Mobile Alliance (OMA) Location Working Group (LOC), which continues the work originated in the former Location Interoperability Forum (LIF) [13], is developing specifications to ensure interoperability of Mobile Location Services on an end-to-end basis. This working group has defined a location services solution, which allows users and applications to obtain location information from the wireless networks independently of the air interface and positioning method. The Finnish Federation for Communications and Teleinformatics (FiCom) has built a national location API to enable location-information exchange between network operators, and between operators and service providers as well [14]. FICOM's API is compliant with the LIF location services specifications.

3 The Location API for J2ME CLDC Profile

The JSR-179 [4] defines a J2ME optional package to enable location-aware applications for MIDP devices. Specifically this package provides the following two main functionalities: i) obtaining information about location and orientation of the mobile device; and ii) accessing a landmark database stored on the device. Each functionality exploits specific objects as information containers; indeed, the `Location` class represents the standard set of basic device-location information, whereas the `Landmark` class represents known locations (i.e. the places that mobile users can go through). `Landmark` objects have a name and may be placed either into a single category or into several categories. Each cate-

gory is intended to group landmarks that are of similar type to the end user (e.g. restaurants, museums). Landmarks are stored into a persistent repository, called `LandmarkStore`, which provides JSR-179-compliant applications with a shared database for location description. The `LocationProvider` class represents a module that is able to determine the location of the terminal. This may be implemented by using existing location methods, including satellite based methods like GPS, and short-range positioning methods like Bluetooth Local Positioning. Actually, each device can have several location providers installed, each related to a different positioning technique (e.g., GPS and RSS-based triangulation). The API allows to specify selection criteria to choose the most suitable `LocationProvider`. Upon the selection of a specific `LocationProvider`, the application can retrieve `Location` objects by means of either periodic updates or asynchronous queries.

4 Enabling RSSI-Based Positioning for JSR-179 Compliant Applications

4.1 Location API for Indoor Positioning

JSR-179, as already mentioned, enables developers to write wireless location-based application and services for mobile devices by means of several location methods, including satellite based methods, as well as short-range positioning methods. Actually the accuracy of information provided through this API depends on the accuracy of the adopted positioning system. In this work we adopt an indoor-positioning technique which is suitable for detecting the symbolic position [2] of mobile users within a set of buildings. For the sake of clarity, in the rest of this sub-section we briefly describe this technique. The interested reader may refer to [8] for further details on the adopted approach. It is worth pointing out that we are interested in identifying the room that the device is moving in; thus the accuracy contained into `Location` objects is related to the coordinates-error probability (i.e. the probability of associating a device to a wrong room). Since an effective measurement of such an error probability goes beyond the scope of this work, we assume here the coordinates to have a constant accuracy. According to the adopted technique, each room of the building is represented by a symbolic location, called "zone"; we assume the presence of one Bluetooth position-sensor for each zone. The topology of the considered environment is represented by the *Sensor-Coordinates* table, which associates each Bluetooth sensor to a specific room, and the *Sensor-Neighbors* table, which logically links sensors each other between adjacent rooms.

The adopted technique uses the Received Signal Strength Indicator (RSSI) as a good fingerprint for the room the mobile device is in. It is worth noting that, although our representation of locations implies an additional layer of indirection to get the geometric information, it brings some benefits to our solution. First, storage and retrieval of symbolic location data might be more suitable to location applications than geometric location data. Second, a hierarchical (e.g.,

building-floor-room) symbolic data model can ease the integration of multiple wireless technologies within the same application. The third advantage is that the symbolic model might be used to predict user location, for it helps building secondary models of location information such as individual mobility patterns.

In the rest of this section we show how the adopted positioning technique can be mapped onto the JSR-179 API. Moreover, we show how we extended the JSR-82 API to bridge it to the JSR-179's `LocationProviders`, and provide some comments about the current implementation.

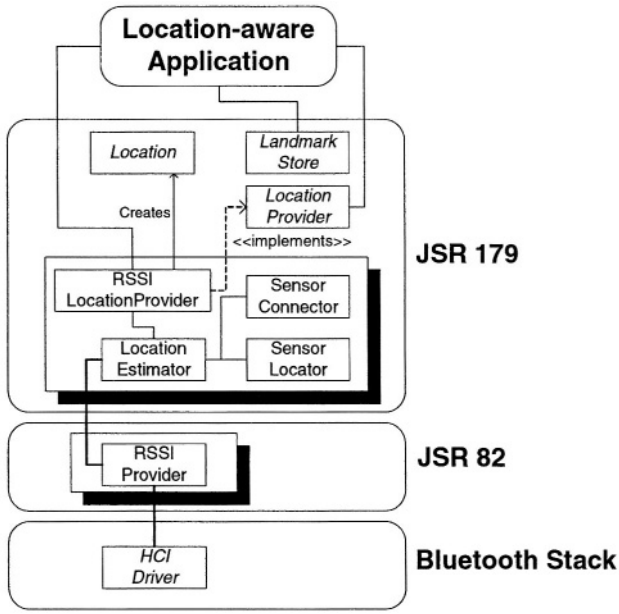


Fig. 1. High-level architecture of the proposed extensions to the APIs. Shadowed boxes contain the implemented components; the involved off-the-shelf components are put outside.

According to the zoning approach, each `Location` has a one-to-one relationship with a specific room of the building. Within the JSR-179 specifications, `Location` objects represent the up-to-date location in terms of timestamped coordinates, accuracy, speed, course, and information about the positioning method used for the location, plus an optional textual address. Thus, it is crucial that our technique be mapped onto JSR-179 semantics. Therefore, we associate each room to a specific set of latitude, longitude, and altitude parameters. In addition, as already mentioned, we also assume the accuracy to be constant. `Landmarks` and the already mentioned tables represent the topology of the considered environment. Figure 1 shows the overall architecture of the proposed extensions. We basically implemented a particular `LocationProvider`,

namely the `RSSILocationProvider`, in order to provide the overlying JSR-179 applications with our indoor positioning mechanism. As for `Landmarks`, we describe each known `Location` (i.e., each room) by means of a name that identifies the room to the end user (e.g., the Contemporary Art gallery within an exhibition). As for the topology tables, the `SensorLocator` component is in charge of managing the *Sensor-Coordinates* table; hence it associates each zone (i.e. each Landmark) to a specific Bluetooth sensor. The `SensorConnector` manages the *Sensor-Neighbors* table; hence it represents the interconnections between rooms.

4.2 Enabling the RSSI-Based Positioning Technique on a Bluetooth Network

In order to enable the proposed technique on a Bluetooth-based system, we added a new class to the Java API for Bluetooth (JSR-82) [7], namely the `RSSI_Provider`, which allows to measure the RSSI; this value refers to the communication between the device itself and another Bluetooth device (i.e., a sensor).

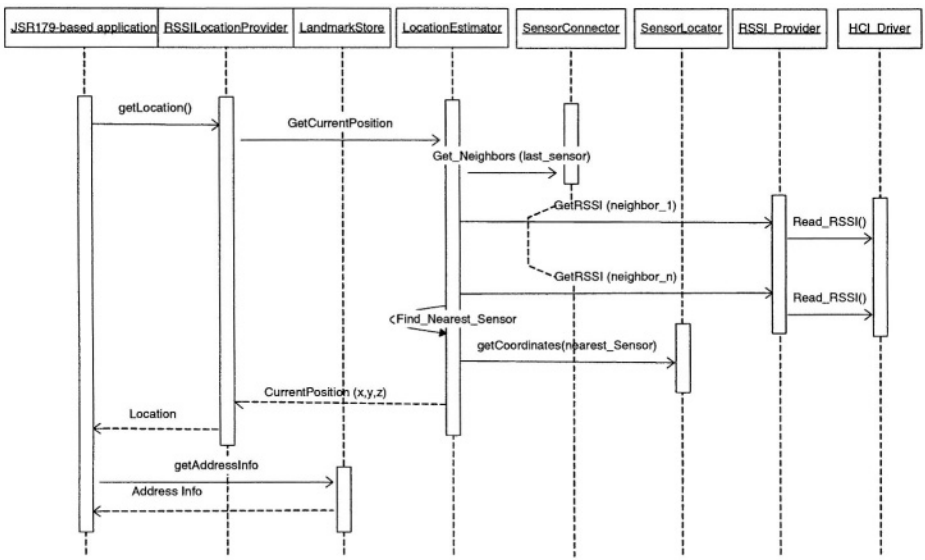


Fig. 2. Retrieval of a mobile user's location

Figure 2 shows the retrieval of user's location by means of the described extensions. It is worth noting that the `LocationEstimator` component exploits the `RSSI_Provider` to discover the zone where the mobile device is. It specifically analyzes the *Sensor-Neighbors* table by means of

the SensorConnector component, and reads the RSSI for each neighbored sensor. Upon the identification of the nearest sensor (i.e., the identification of the current room), the LocationEstimator can *i*) retrieve its spatial coordinates by means of the SensorLocator component, and *ii*) return them to the RSSILocationProvider.

Implementation details. We implemented the RSSI_Provider class as an extension to the JBlueZ library. JBlueZ is an implementation of the JSR-82 specifications, which relies on the official Linux Bluetooth protocol stack, namely Bluez [15]. Since BlueZ is implemented as library of native procedures written in C, the RSSI_Provider uses the Java Native Interface (JNI) to retrieve RSSI and quality information through the BlueZ's HCI. As for the Location API, we developed a JSR-179 implementation skeleton consisting of a LandmarkStore and a LocationProvider.

5 Experimental Results

In order to evaluate the behavior of our technique, we tested the solution on Compaq iPAQ 3970 PDAs running the Familiar 0.7.0 Linux distribution. As for sensors, we used ANYCOM Bluetooth dongles configured to accept connections from mobile devices. We performed several experiments in order to evaluate the proposed solution from different view-points.

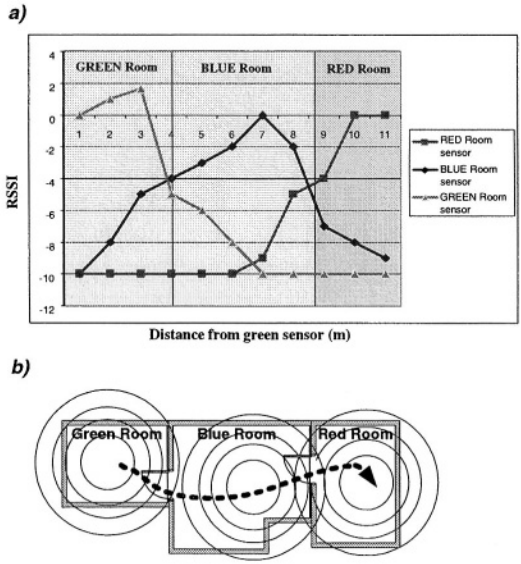


Fig. 3. a) Signal strength while moving between different rooms; b) Topology of the RSSI-measurement environment.

First of all, we measured the RSSI while moving through a simple sequence of three adjacent rooms, namely the Green Room, the Blue Room, and the Red room. This preliminary experiment aimed to verify that the RSSI could be used as a room-fingerprint. Figure 3a shows the measured values of the RSSI with respect to each room's sensor. Figure 3b shows the environment used to perform this experiment. Since our testing user starts moving from green-room sensor's location (as Figure 3b shows), we use the distance from the green sensor as a location variable for our results; the vertical dividing lines represent the separation between adjacent rooms (e.g., the wall between the green and the blue room is 4 meters far from the green room sensor). Preliminary results show that the highest RSSI value always represents the current room: for instance, when the user is 3 meters far from the green sensor (i.e., he is walking in green room) Green Room's RSSI is higher than the others, whereas when the user is 6 meters far from the green sensor (i.e., he is walking in blue room) Blue Room's RSSI is higher than the others.

We then evaluated the performance of the `RSSI_provider` component. Specifically we quantified the RSSI measurement time $t_{Rm} = t_{BC} + t_{RSSI}$, where t_{BC} is the time to connect to a Bluetooth sensor, and t_{RSSI} is the time to read the RSSI. Several factors influence t_{Rm}, such as channel-degradation phenomena (e.g., multi-path fading, interference, direct sunlight, physical obstacles), as well as the adopted Bluetooth hardware and software libraries. Since in this preliminary evaluation our measures focused on the performance of the implemented APIs, we did not take into account channel-degradation phenomena.

Each RSSI measurement refers to a certain sensor. Our preliminary experiments showed that t_{Rm} is not influenced by the distance d from the used sensor. Indeed, it is well-known that the higher the distance between bluetooth devices, the lower the performance of the bluetooth channel in terms of data-rate (due to packet-errors and error-correction procedures); since the amount of information exchanged by sensing devices using our protocol is extremely small, such a data-rate degradation does not significantly influence t_{Rm}. Specifically these experiments showed that $t_{BC} >> t_{RSSI}$, being $t_{BC} \approx 1175ms$ and $t_{RSSI} \approx 8ms$ independently of d.

Figure 4 depicts an example topology in which each room can have at most 3 adjacent rooms. According to the proposed approach, when the mobile user is walking through room Z_4, the `LocationEstimator` component continuously analyzes each adjacent rooms' sensor RSSI in order to decide whether the user has moved to a new room or not. Thus, when the user moves from Z_4 to Z_1, the Location Change Detection time t_{LCDT} depends on t_{RSSI}, whereas t_{BC} does not affect t_{LCDT} since the connection with the involved room sensors has been previously established. In this case (which is the worst case as each room can have at most 3 adjacent rooms), the performance of the `LocationProvider` component is suitable for the up-standing JSR-179 applications, as it is able to detect location-changes in less than 50ms; the `LocationEstimator` must compare the RSSI obtained from the current room's sensor with the value retrieved from the three adjacent rooms' sensors. Hence, in this case $t_{LCDT} \approx$

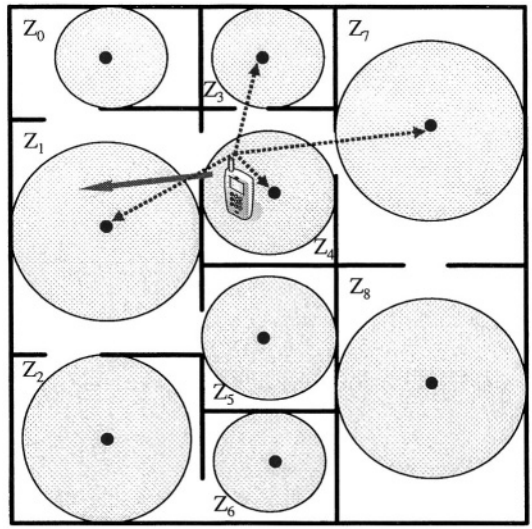


Fig. 4. Topology of the environment used to compare the Location Change Detection Time with the speed of a walking user.

$(t_{RSSI}(AdjRooms) + t_{RSSI}(CurrRoom)) = (3+1) \times t_{RSSI} \approx 40ms < 50ms)$, which is a reasonable time if compared with the speed of a walking user (i.e., $2m \cdot s^{-1}$).

6 Conclusions and Future Work

This work presented an approach to providing JSR-179 compliant applications with RSS-based positioning techniques. Specifically we proposed a technique to implement a `LocationProvider` component that estimates the zones where the mobile devices operate by measuring the Received Signal Strength. Our approach resulted in a versatile infrastructure model, which is independent of the used devices as well as of the wireless communication technologies. Since our solution is based on standardized specifications, it is suitable for a wide range of JSR-179 compliant mobile devices. It is worth noting that, even though the proposed technique relies on the possibility to measure the RSSI on the mobile device, our approach can be adapted to devices whose hardware do not allow the RSSI to be directly measured. Indeed, in this case a different `RSSI_Provider` must be implemented; such a component could exploit a serial connection with a remote sensor to retrieve the RSSI as measured on the room sensor's side rather than measuring the RSSI on the mobile device's side. The implemented extensions to the JSR-82 infrastructure can also be used to implement different positioning techniques; for instance, a different RSS-based technique can be

implemented by modifying only the `LocationEstimator`'s algorithm. Preliminary experiments showed that the performance of the implemented API does not depend on the distance between the mobile device and room sensors. Moreover, the performance of the implemented JSR-179 components is reasonable if compared with the velocity of walking users. Our future research will focus on the following major issues: *i*) evaluating the accuracy of the implemented mechanism with respect to different configurations; *ii*) evaluating the performance and the accuracy of the developed APIs on several kind of platforms and mobile devices (e.g., Symbian-based smartphones, and Windows PocketPC-based palmtops); *iii*) evaluating and improving the robustness of the provided positioning services; and *iv*) experimenting the proposed approach over several short-range wireless communication technologies (e.g., WiFi).

Acknowledgments. This work has been partially supported by the Italian Ministry for Education, University and Research (MIUR) in the framework of the FIRB Project "Middleware for advanced services over large-scale, wired-wireless distributed systems" (WEB-MINDS), and by Regione Campania in the framework of "Centro di Competenza Regionale ICT".

References

1. Bahl, P., Padmanabham, V.N.: RADAR: An In-Building RF-based User Location and tracking System. In: Proceedings of the IEEE Infocom 2000, Vol.2. Tel-Aviv, Israel (2000) 775–784
2. Hightower, J., Borriello, G.: Location Systems for Ubiquitous Computing. In: IEEE Computer, 34(8). IEEE CS Press (2001) 57–66
3. Patterson, C.A., Muntz, R.R., Pancake, C.M.: Challenges in Location-Aware Computing. In: IEEE Pervasive Computing, 2(2). IEEE CS Press (2003) 80–89
4. Java Community Process: Location API for J2ME Specification 1.0 Final Release (2003)
5. Hightower, J., Borriello, G.: Location Sensing Techniques. In: Technical report UW-CSE-01-07-01. University of Washington (2001)
6. Kotanen, A., Hannikainen, M., Lappakoski, H., Hamalainen, T.D.: Experiments on Local Positioning with Bluetooth. In: Proceedings of the International Conference on Information Technology, Computers and Communication (ITCC03). IEEE CS Press (2003) 297–303
7. Java Community Process: Java APIs for Bluetooth Specification 1.0 Final Release (2002)
8. Cornevilli, F., Cotroneo, D., Ficco, M., Russo, S., Vecchio, V.: Implementing Positioning Services Over an Ubiquitous Infrastructure. In: Proceedings of 2^{nd} IEEE Workshop on Software Technologies for Embedded and Ubiquitous Computing Systems (WSTFEUS '04). IEEE CS Press (2004)
9. Fukuju, Y., Minami, M., Morikawa, H., Aoyama, T.: DOLPHIN: An Autonomous Indoor Positioning System in Ubiquitous Computing Enviroment. In: Proceedings of the IEEE Workshop on Software for Future Embedded Systems (WSTFES'03). IEEE CS Press (2003) 53–56
10. Ekahau, Inc.: Ekahau Positioning Engine 2.0: Developer Guide (2002)

11. Harter, A., Hopper, A., Steggles, P., Ward, A., Webster, P.: The Anatomy of a Context-Aware Application. In: Proceedings of 5^{th} Annual International Conference on Mobile Computing and Networking (Mobicom 99). ACM Press, New York (1999) 59–68
12. Anastasi, G., Bandelloni, R., Conti, M., Delmastro, F., Gregori, E., Mainetto, G.: Experimenting an Indoor Bluetooth-Based Positioning Service. In: Proceedings of the 23^{rd} International Conference on Distributed Computing Systems Workshops (ICDCSW'03). IEEE CS Press (2003) 480–483
13. Location Inter-operability Forum: Mobile Location Protocol LIF TS 101 Specification, Version 3.0.0 (2002)
14. FiCom Location API Working Group: FiCom Location API 2.0.0 Interface specification (2002)
15. Bluez: Official Linux Bluetooth protocol stack: http://bluez.sourceforge.net/

Design Patterns for RTSJ Application Development

Angelo Corsaro[1] and Corrado Santoro[2]

[1] Washington University
Department of Computer Science and Engineering
1 Brookings Drive, BOX 1045, St. Louis, 63130 Missouri, USA
corsaro@cse.wustl.edu
[2] University of Catania
Dept. of Computer and Telecommunication Engineering
Viale A. Doria, 6, 95125 - Catania, ITALY
csanto@diit.unict.it

Abstract. The Real-Time Specification for Java provides a memory model, based on memory areas, which introduces some constraints on objects allocation and references. For this reason, many well-known design patterns cannot be used in a RTSJ environment since their structure violates such constraints. In this context, this paper proposes some new RTSJ design patterns and idioms, as well as a re-interpretation of existing patterns into a RTSJ perspective. Five patterns are presented. Three of them are an adaptation to RTSJ world of well-known design patterns—the *singleton,* the *abstract factory* and the *leader-follower.* The other two are new patterns—the *eager exception handler* and the *memory tunnel*—which allow to solve some peculiar problems deriving from the constraints of RTSJ memory model.

1 Introduction

Real-time Java is a fairly new technology. The increasing number of available JVMs [5,13,12,4,8,7], which comply with Real-Time Specification for Java (RTSJ) [3], are now making possible the design and implementation of Java-based real-time systems. However, despite the availability of such tools, real-time Java has still not gained, in the real-time community and industry, the expected success. Researchers agree that some reasons for this are due to the limitations that RTSJ introduces. Indeed, RTSJ memory model, which is based on scoped areas, even if it overcomes the unbounded priority inversion problem generated by garbage collection, introduces some constraints on object allocation and reference. This characteristic does not allow direct use of many programming idioms typical of both object-oriented technology and Java language: in many cases, their "as-is" use could either violate RTSJ memory model constraints or even introduce memory leaks.

The effort needed from researchers is thus to provide—new or re-thought—*idioms* and *patterns* that, being compliant with RTSJ, do not force the pro-

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 394–405, 2004.
© Springer-Verlag Berlin Heidelberg 2004

grammer of a Java real-time system to face the issues of explicit handling object allocation in scopes, checking for object references, etc. In this light, some papers have been published [2,1,9], which mainly present new patterns useful during engineering of RTSJ-based systems. But no current work[1] deals with the problem of making RTSJ-compliant the most widely known and used design patterns for object-oriented programming, *e.g.* that of provided by "Gang-of-Four" (GoF) book [6] and POSA2 [11]. This is a very serious problem since it impedes the use of well assessed design and implementation techniques, and invalidates—for RTSJ environments—many results obtained in the research field of object-oriented software engineering.

This paper deals with such an issue. It underlines those design techniques, often used in object-oriented systems, which are indeed "anti-patterns" since they violate the constraints of RTSJ. Then the paper provides a re-interpretation of these anti-patterns into a RTSJ perspective, thus making them suitable for the use in real-time Java software. The paper presents the adaptation to RTSJ of three well-known design patterns—*singleton* [6], *factory* [6] and *leader/follower* [11]—and proposes other two new patterns able to overcome some programming limitations imposed by RTSJ. The latter set includes the *eager exception handler,* which allows exception propagation to outer scopes in order to perform a correct handling, and the *memory tunnel,* which makes possible communication between threads running in different and not compatible memory areas. All the patterns presented are classified into three major categories: *creational, structural* and *behavioural,* which are dealt with in three separate Sections. Our conclusions are finally reported in Section 5.

2 Creational Patterns

2.1 Singleton

The Singleton Pattern, made popular by the famous GoF book [6], exposes some interesting issues in an RTSJ environment. A description of the pattern and the extension needed to make it applicable in an RTSJ context are reported below.

Intent. Ensure that a class has only one instance and provide a global point of access to it.

Example. In some applications, there are abstraction that have only one associated instance. For example, in a real-time system, it is very likely that there is a single scheduler managing the various tasks execution. In all the cases in which there is an abstraction that has to have a single instance, a good design practice is to (1) enforce, by design, that no more than one instance of the given class can be created, and (2) provide a global access point to it.

Problem. A design mechanism is needed in order to enforce that a given class has at most one instance and this singleton instance is accessible via global access point.

[1] At time this paper was written.

Solution. Define a class method that allows to get the unique instance of the class. Declare the constructor of the class protected and give the responsibility of creating the class to the class itself or a friend factory.

Implementation. The typical Singleton Java implementation is shown in Figure la. The solution works perfectly fine in a Java environment, but, respect to the RTSJ programming model, has a fatal flaw. The issue is that the Singleton instance, allocated by the **new** instruction executed in the `instance ()` method, will be placed in the current memory area of the calling thread. This is a big problem since static member of a class can only refer to objects allocated in the Heap or Immortal memory; therefore, whenever a thread, running within a scoped memory, will call the `instance ()` method, the runtime system will throw an invalid reference exception. Thus, it should be clear that the pure Java Singleton implementation is not robust in an RTSJ environment.

```
public class Singleton {

  protected static Singleton theInstance;


  public static synchronized
   Singleton instance() {
    if (theInstance == null)
      theInstance = new Singleton();
    return theInstance;
  }

  // Other methods

}
```

(a)

```
public class Singleton {
  protected static Singleton theInstance;

  static {
    ImmortalMemory im =
      ImmortalMemory.instance();
    theInstance =
      im.newInstance(Singleton.class);
  }

  public static Singleton instance() {
    return theInstance;
  }
  // Other methods
}
```

(b)

Fig. 1. A typical Singleton implementation in Java and with RTSJ

The RTSJ-compliant solution is reported in Figure 1b. In this case, we explicitly allocate the Singleton instance in immortal memory. The lesson that should be learned by the RTSJ's Singleton implementation is that static fields should be in general treated with great care in RTSJ and that the programming idiom that should be used when coping with such cases should be **Statics are Immortal.**

2.2 Memory-Area-Aware Factory

In the RTSJ memory model, each occurrence of the **new** operator allocates memory from the *current* memory area. Thus, unless some information is provided on the target memory area that should be used for allocating new instances of a given type, the current context will be used. One way to control the placing of certain types is to use a variation of the GoF Factory Pattern [6].

Intent. Provide an interface for creating families of objects without specifying their type nor their placement, *i.e.* their memory area.

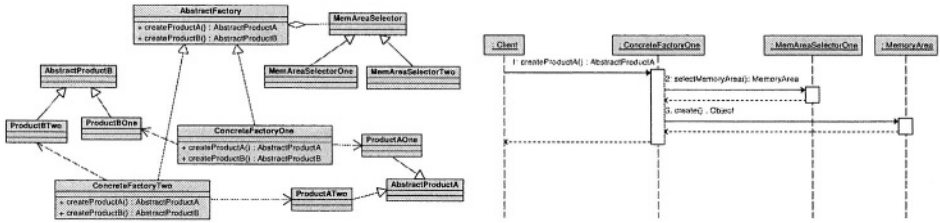


Fig. 2. The Memory-Area Aware Abstract Factory Pattern.

Example. Suppose you are writing a system in which both the real-time core and its GUI need to create some concrete types through a factory. However, while the GUI will be run by non real-time thread and will be using the heap, the application's real-time core will have to use no-heap real-time threads and thus rely only on immortal and scoped memories

Problem. You want to create instances of concrete types by relying on the Factory pattern [6], but at the same time you want the factory to allocate memory in different places based on the callee identity.

Solution. Use the Abstract Factory Pattern in synergy with the Strategy Pattern [6]; the latter will be used to encapsulate the allocation strategy. Each time a request arrives to the Abstract factory, it uses its strategy to decide where to allocate the newly created instance. The strategy can be programmed by the client.

Structure. The structure of the **Memory-Area-Aware Factory** pattern is obtained by the plain Abstract Factory structure and adding the memory area selection strategy. The resulting structure is shown in the left side of Figure 2, while the right side reports the typical sequence of messages exchanged (the example is shown for `ConcreteFactoryOne`).

Implementation. The implementation of this pattern is the combination of the implementations of the Abstract Factory and the Strategy Pattern, thus refer to [6] to see how to implement this.

3 Structural Patterns

3.1 Scoped Leader Follower

The leader-follower [11,10] is a is a well-known design pattern used to efficiently handle concurrent events coming from various sources. Because of the RTSJ memory model, this pattern cannot be directly applied as described in [11,10], but it requires some non-trivial modifications. In the reminder of this section we will outline a possible specialization of this pattern for the RTSJ platform.

Intent. Efficiently multiplex and dispatch event from several sources.

Example. Suppose you have to implement a real-time network server that has to concurrently serve incoming requests from several different connections. Incoming requests will have to be demultiplexed efficiently and the usage of the memory resource has to be minimized because of *e.g.* the software is running on a small embedded system.

Problem. The problem to solve is to handle time-critical events, using RTSJ software, taking into account the efficiency, correctness and safety requirements.

Solution. The **Scoped Leader-Follower** pattern uses a pool of threads allocated in a scoped memory area called the *pool scope,* and a *leader selector* thread, running in that memory area, which iteratively picks the leader thread from the pool and activates it. Each thread of the pool is associated with another scoped memory area, called the *handler scope.* This memory area is used to run the event handler. To hide to programmers the details on memory management and scope entering/exiting, the threads of the pool act as *trampolines,* thus preparing the execution environment for user-defined objects that implement the logic for event capture and handling. All the threads (those of the pool and the leader selector) are `NoHeapRealtimeThread`s running with a preassigned priority. Since before an event is caught its criticality is unknown, the handler is first activated with the highest priority and then, once the event is caught and thus known, the priority is adjusted according to the urgency of the event itself. Using this "conservative" approach ensures that capturing and processing an event with a high priority is not delayed by any other activity. The pattern also includes an *event selector* object that provides the interface to the system that generates or receives the event.

Structure. The structure of the Scoped Leader-Follower pattern is depicted in Figure 3.

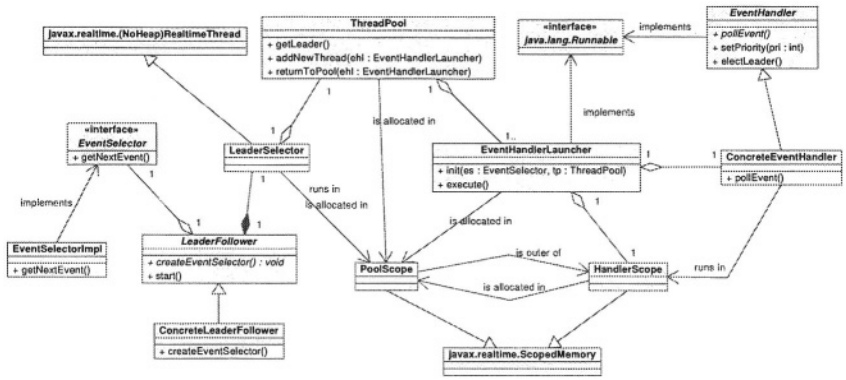


Fig. 3. Class Diagram of the Scoped Leader-Follower Pattern

The pattern is composed of the following classes:

- `LeaderFollower`. It is the responsible for starting the system, *i.e.* creating the thread pool and all the objects to be placed in the pool (threads, memory areas, etc.). It is also the main interface that a user has to control the system *e.g.* to change some parameters like priorities, etc. It is an abstract class because any realization must define the `createEventSelector` method, needed to set up the *event selector* object.
- `LeaderSelector`. It is a `NoHeapRealtimeThread` responsible for selecting the leader from the thread pool, activating it.
- `EventSelector`. It is the interface that *event selector* objects must implement to be used in the Scoped Leader-Follower.
- `EventHandlerLauncher`. Threads allocated in the pool are instances of this class. Each `EventHandlerLauncher` holds a reference to a *handler scope,* and another reference to an instance of a concrete `EventHandler` class. The task of the `EventHandlerLauncher`, each time it is activated, is to enter the (concrete) `EventHandler` in the scope, wait for `EventHandler`'s task completion, and finally return itself to the thread pool.
- `EventHandler`. It represents the abstract class for the implementation of event handlers. Each concrete instance of this class (`ConcreteEventHandler`) is activated by the corresponding `EventHandlerLauncher` and has the task of capturing an incoming event, suitably processing it. It has also the responsibility for triggering new leader election.
- `PoolScope`. It is the memory scope in which the thread pool and all its objects are allocated. It is also the scope in which the `LeaderSelector` runs.
- `HandlerScope`. It is the memory scope associated to a `EventHandlerLauncher` and in which the (concrete) `EventHandler` runs.

Implementation. Figure 4 reports the sequence diagrams of the Scoped Leader-Follower. The activities of the pattern begin when the method `start ()` of the `LeaderFollower` object is called. We suppose that, before this, an *initialization phase* has been performed, managed by `LeaderFollower` and `LeaderSelector` objects and aiming at creating the `PoolScope` and all the other objects allocated in this memory area (*i.e.* the `ThreadPool`, the `EventSelectorImpl`, all the `EventHandlerLaunchers`, together with their `HandlerScopes`, and `ConcreteEventHandlers`[2]). After this initialization phase, following invocation of method `start()` of the `LeaderFollower` object as Figure 4 reports, control is given to the `LeaderSelector` object that (A) selects the leader thread (`EventHandlerLauncher`) from the pool, (B) activates this thread by calling the `execute ()` method, and (C) waits for a signal, which will arrive from the `ConcreteEventHandler` object, triggering re-iteration and selection of the next leader. Each leader thread (`EventHandlerLauncher`), after being selected, (1) enters its `ConcreteEventHandler` object in the associated `HandlerScope`, and (2) when

[2] Such an initialization phase, which controls the creation of the objects of the pattern, aims at hiding memory management details to the programmer.

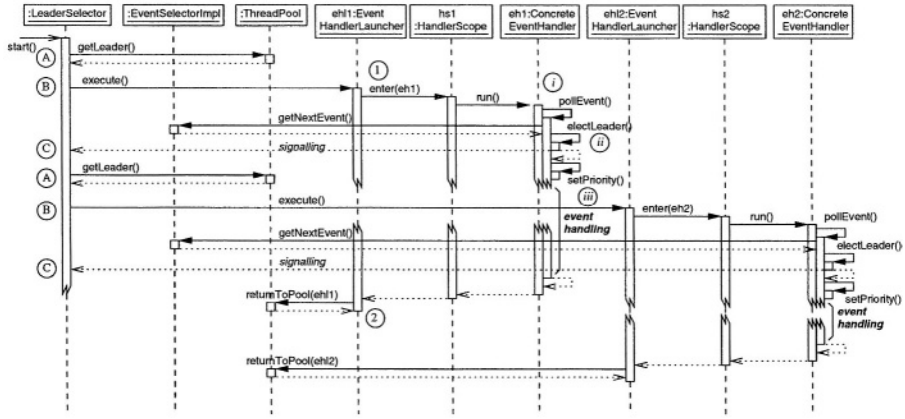


Fig. 4. Sequence Diagram of the Scoped Leader-Follower Pattern (Operation)

the handler terminates, return itself to the pool. Finally, each `ConcreteEventHandler`, once activated by the `EventHandlerLauncher`, first (i) calls `pollEvent()` (that in turn invokes method `getNextEvent( )` of the `EventSelectorImpl` object), then (ii), once it has acquired the events, triggers new leader election, and finally (iii) handles the event adjusting the thread priority accordingly.

3.2 Memory Tunnels

RTSJ does not provide any specific mechanism to transfer object graphs from one memory area to another. The only mechanism that could be used is the Java serialization, but this has some limitation since (1) it requires that all the objects to be serialized implement the `Serializable` interface, and (2) is not the most efficient mean of getting an object graph from one memory region to another. A possible solution is to use the *handoff* pattern, proposed in [9], but this pattern requires that the source and destination memory areas share the same parent area, a situation that could not occur so frequently.

Intent. Provide an efficient mechanism for threads to transfer object graphs across memory areas.

Example. Implementing a producer/consumer paradigm in RTSJ is usually very hard, since it conflicts with the programming model imposed by scoped memories. This issue has been faced by people who have tried to implement real-time Java ORB that rely on the RTSJ. The issue that often arise is that an object, or an object graph, from one memory area needs to get into another, but there is no neat way of accomplishing the goal.

Problem. Solve the problem introduced by the scoped memories which limits the possibility of data sharing between threads, as well as the implementation of producer consumer applications.

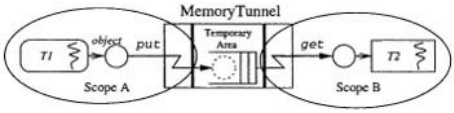


Fig. 5. A Memory Tunnel

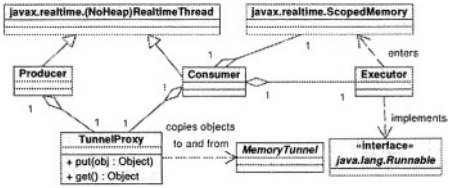


Fig. 6. Producer/Consumer problem with Memory Tunnels

Solution. The idea is that of "breaking the barriers" of scoped memories. To this aim, we propose a structure that can be used to perform object passing among different threads, forcing a violation of the memory constraint of RTSJ but without provoking consequences. Like in some diodes the "tunnel effect" allows electrons to break a barrier of potential, **memory tunnels** allow a message passing by-value among threads that live in different and not compatible memory areas. From an abstract point of view, a memory tunnel is a traditional queue exposing two operations—*put* and *get*—that can be used to write an object, at one side, and read it from another side (Figure 5). The underlying mechanism, like any other technique used to access data from different domains[3], is based on performing a *copy* of the object: when `put` is executed, the object is copied from the source memory into an *internal temporary area* that holds the queue in the tunnel; similarly, the effect of `get` is doing a copy the object from the temporary area into the memory of the destination domain. Since the real operation is *cloning* the object to be passed and the latter could contain references to other objects of the source (or a compatible) domain, a memory tunnels should provide two versions of `put`: a `deep_put`—which performs a *deep-copy* of the object—and a `shallow_put`—which instead does not copy the referred objects. In the latter case, when a `get` is performed, a suitable check verifies that references present in the copied object are still valid in the new domain.

Using this structure, data can be transferred without needing to share anything among the threads involved; for this reason, memory tunnels can be employed without any problem when traditional forms of data passing-by-reference fail due to RTSJ memory model constraints. In order to make this possible, the internal temporary area of a memory tunnel must be allocated in a memory space and using management mechanisms that are outside the RTSJ memory model, otherwise we easily fall again in the non-feasibility conditions caused by RTSJ constraints[4]. For this reason, a memory tunnel cannot be directly visible to Java programs and access is performed by means of a `TunnelProxy` object. The latter provides only methods to read and write objects, hiding tunnel's internals.

[3] Think, for example, to accessing user space by kernel code, operation usually done by copying data from user to kernel space.

[4] Our prototype implementation, made with jRate, is based on native methods that use the C++ heap and the `malloc()` and `free()` primitives.

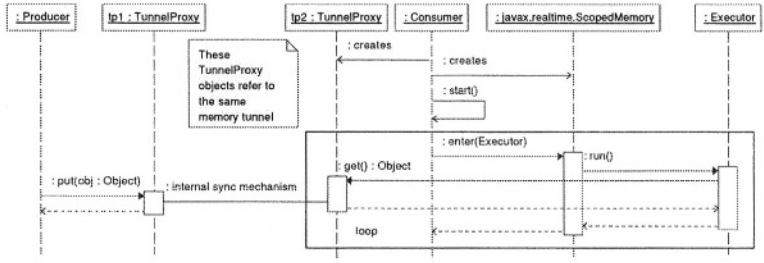


Fig. 7. Sequence Diagram of Dispatcher/Worker with Memory Tunnels

Structure. Figure 6 and Figure 7 show the structure of the solution in the context of a producer/consumer problem. In this solution, the memory tunnel is referred by two different `TunnelProxy` objects: one for producer side and the other one for consumer side. The behaviour of the producer is quite simple: once produced the message, it gets the `TunnelProxy` object and puts the message in it. On the other side, the task of consuming and handling messages is split between two different objects: the `Consumer` and the `Executor`. The former implements a loop that continuously passes the control to the latter by iteratively entering it in a scoped memory. The `Executor` has instead the real responsibility of reading messages from the tunnel and peforming the processing task. Since the `Executor` runs in a scoped area, the memory allocated by the picked message is automatically released when the task is finished and the `Executor` exits the scope.

Implementation. The prototype of the class `TunnelProxy` is given in Figure 8. As it can be noted, a memory tunnel is identified by a literal string that has to be specified when the proxy is created. The class has two constructors: one to open an existing memory tunnel and another to create a new memory tunnel, given the maximum number of items and the size of the temporary memory area to be allocated for the queue. Methods provided allow to put an object by using a deep or shallow copy, and to retrieve a stored object, checking also the validity of embedded references. The proxy provides also a `close()` method to signal that the opening/creating thread is no longer interested in using the tunnel. When a memory tunnel is no more used by any thread[5], it is destroyed and the memory allocated is released.

4 Behavioural Patterns

4.1 Handle Exceptions Locally

Due to the RTSJ restrictions on memory management and validity of references, designing code that uses structured exception handling is rather tricky. The **Eager Exceptions Handling** pattern provides a way of using structured exceptions in RTSJ applications so to avoid the traps and pitfalls that the platform might induce.

[5] *i.e.* when the last thread calls the `close()` method.

```
public class TunnelProxy {
  /* Creates a memory tunnel, given its name and its dimension.
     "size" gives the dimension of the temporary area and "items" gives
     the maximum number of objects that can be put in the queue. */
  public TunnelProxy(java.lang.String name,int itemSize,int items);

  /* Opens a memory tunnel, given its name. */
  public TunnelProxy(java.lang.String name);

  /* Writes the object in the tunnel using a shallow copy */
  public synchronized void shallowPut(java.lang.Object object);

  /* Writes the object in the tunnel using a deep copy. */
  public synchronized void deepPut(java.lang.Object object)

  /* Retrieves the object from the tunnel. It also checks whether the
     references contained are valid in the new domain. */
  public synchronized java.lang.Object get();

  /* Closes the memory tunnel and removes the internal memory
     if the reference count reaches zero. */
  public synchronized void close ();
}
```

Fig. 8. The `TunnelProxy` Class

Intent. Ensure that the use of exceptions do not cause invalid memory references, avoiding the coupling of scope structure to the handling of structured exceptions.

Example. In programming languages that supports structured exceptions, erroneous conditions are handled by raising a proper exception. The thrown exception encapsulates the kind of unexpected situation that has occurred. While using exceptions in Java is rather straightforward, the RTSJ introduces several complications. Due to the referencing rules imposed by the RTSJ memory model, exceptions should be handled either in the same memory area in which they were raised (allocated), or in a memory area from which is legal referencing the given exception object.

As an example, the scenario depicted in Figure 9a seems at a first sight rather innocent, but actually it could have several problems, depending on how the underlying RTSJ implementation copes with exceptions. If it does not take care of propagating exceptions upwards, from scope to scope[6], an invalid reference exception (or a segmentation fault) could occur when the exception is thrown in an inner scope to be handled in an outer scope (see Figure 9a).

Problem. The RTSJ does not mandate a specific way of coping with runtime exceptions that are thrown while executing within a memory area. Thus, exception handling in the RTSJ might have different behaviour based on the RTSJ implementation. This limits the portability of code that is not written carefully.

Solution. In order to write safe and portable RTSJ application, exceptions should be handled in the same scope in which they are raised. The notification of an exceptional condition has to be transfered to outer scopes, and then handled depending on the application. For instance this can be achieved by a status

[6] Please note that the most of currently available RTSJ implementations does not support exception propagation.

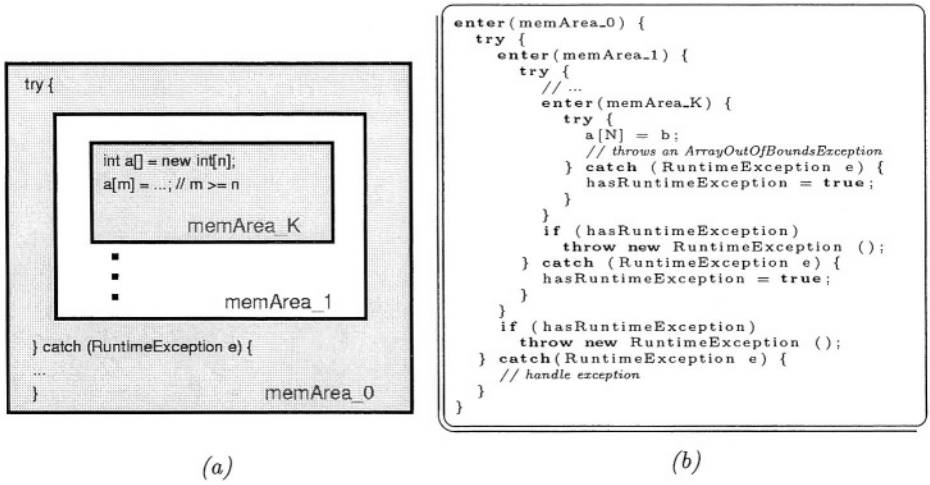


Fig. 9. Runtime Exceptions in the RTSJ: Problem (a) and Solution (b)

objects or variable can be used to set the information that might be needed by the logic executing in outer scopes.

Implementation. The implementation of the pattern is sketched in Figure 9b. Here, we use a pseudo-language where a new keyword, **enter,** represents that a given fragment of code is executed within the memory area passed an argument. This syntax (that in a real implementation is replaced by "`memArea.enter(new Runnable () { ...})`") makes it easier to understand the scoping of memory areas. As Figure 9b shows, the exception handler sets a global variable that is checked in the outer scope in order to re-throw the exception—if this is the case[7].

5 Conclusions

In this paper, we presented some object-oriented design patterns suited for the implementation of RTSJ-based Java real-time systems. The patterns reported are intended to provide the programmer with a set of ready-to-run solutions for some recurring problems that arise during the development of RTSJ-based systems. Following the results given by many years of research in the field of (object-oriented) software engineering, some patterns presented in this paper re-propose those made famous by other authors, suitably modified in order to make them compliant with RTSJ. The other patterns presented are instead new, and are provided as solutions to many implementation problems that are due

[7] In the example, we used a boolean variable indicating only if an exception has been generated. A more appropriate (and general purpose) implementation could use *e.g.* an integer variable assuming a value depending on the type of the exception caught and then use a factory pattern to re-create the exception in the outer scope.

to the constraints imposed by RTSJ memory model. The lesson learned is that, in general, software design with RTSJ—even if simple at a first sight—is in practice quite complex and has to be handled with a great care. For this reason, the effort needed from researchers is to provide programmers with a very rich pattern catalog, so as to make designing with RTSJ "simple and immediate", as in the tradition of engineering with the Java language.

References

1. E. Benowitz and A. Niessner. A Pattern Catalog for RTSJ Software Design. In *Proceedings of* 1^{st} *Workshop on Java Technologies for Real-time and Embedded Systems (JTRES 2003), OTM'03 Federated Conferences.* LNCS 2889, Springer, Nov. 2003.
2. G. Bollella, T. Canham, V. Carson, V. Champlin, D. Dvorak, B. Giovannoni, K. Meyer, A. Murray, and K. Reinholtz. Programming with Non-Heap Memory in RTSJ. In *Proceedings of the* 18^{th} *2003 ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA '03).* ACM Press, 2003.
3. G. Bollella, J. Gosling, B. Brosgol, P. Dibble, S. Furr, D. Hardin, and M. Turnbull. *The Real-Time Specification for Java.* Addison-Wesley, 2000.
4. A. Corsaro and C. Santoro. A C++ Native Interface for Interpreted JVMs. In e. a. Robert Meersman, Zahir Tari, editor, 1^{st} *Intl. Workshop on Java Technologies for Real-time and Embedded Systems (JTRES'03).* Lecture Notes in Computer Science 2889, Springer Verlag, 2003.
5. A. Corsaro and D. C. Schmidt. The Design and Performance of the jRate Real-Time Java Implementation. In R. Meersman and Z. Tari, editors, *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE,* pages 900–921, Berlin, 2002. Lecture Notes in Computer Science 2519, Springer Verlag.
6. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, Reading, MA, 1995.
7. M. Rinard et al. FLEX Compiler Infrastructure, `http://www.flex-compiler.lcs.mit.edu/Harpoon/`, 2002.
8. OVM/Consortium. OVM An Open RTSJ Compliant JVM. `http://www.ovmj.org/`, 2002.
9. F. Pizlo, J. M. Fox, D. Holmes, and J. Vitek. Real-time Java scoped memory design patterns and semantics. In 7^{th} *IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC 2004),* Vienna, 2004.
10. D. C. Schmidt, C. O'Ryan, I. Pyarali, M. Kircher, and F. Buschmann. Leader/Followers: A Design Pattern for Efficient Multi-threaded Event Demultiplexing and Dispatching. In *Proceedings of the 6th Pattern Languages of Programming Conference,* Monticello, IL, Aug. 2000.
11. D. C. Schmidt, M. Stal, H. Rohnert, and F. Buschmann. *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects, Volume 2.* Wiley & Sons, New York, 2000.
12. F. Siebert. *Hard Realtime Garbage Collection in Modern Object Oriented Programming Languages.* AICAS GmbH, 2003.
13. Timesys. JTime Virtual Machine, `www.timesys.com`, 2002.

Separating Soft Real-Time from Functionality Concerns in Complex Object-Oriented Applications

Antonella Di Stefano[1], Marco Fargetta[1],
Giuseppe Pappalardo[2], and Emiliano Tramontana[2]

[1] Dipartimento di Ingegneria Informatica e Telecomunicazioni
{adistefa,mfargetta}@diit.unict.it
[2] Dipartimento di Matematica e Informatica
{pappalardo,tramontana}@dmi.unict.it
Università di Catania, Viale A. Doria, 6 - 95125 Catania, Italy

Abstract. Typically, real-time applications are developed by introducing real time concerns into functionality concerns. This leads to difficulties in the reuse of applications, and to limited capabilities of the underlying runtime system to adapt itself for the sake of meeting the desired deadline.
In this paper we explore an approach that supports the automated development of real-time aware applications. This is made possible by a tool that performs a suitable analysis and transformation, aiming to automatically incorporate real-time concerns into an unaware application. The analysis serves to identify, within the application, subtasks and (possibly) subsubtasks, together with their minimum and maximum execution times and partial deadlines. The transformation has the twofold aim of monitoring critical timing aspects of the execution, and adapting the application on-the-fly to help meeting the deadlines.
Adaptation measures involve not only task scheduling, but also the substitution of application parts, as well as the activation of additional (possibly distributed) hardware and software resources.

1 Introduction

In typical real-time design approaches, applications are developed as time-aware ones right from the start, by incorporating into them the code intended to support real-time concerns. As a consequence, specific real-time aspects are "hard-wired" into the application and often spread across its code; such an application is not easy to reuse, should timing constraints or assumptions on the underlying environment need to be changed.

The approach advocated in this paper aims at designing real-time systems by separating real-time concerns from functional ones, in such a way that application developers need only focus on the latter.

Our targets are complex object-oriented applications featured with soft real-time deadlines, operating in environments where the loads may vary substantially

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 406–417, 2004.
© Springer-Verlag Berlin Heidelberg 2004

and unexpectedly, due e.g. to resources found busy; this would be the case, e.g., of multimedia systems. In such cases, the underlying scheduler, whether real-time oriented or not,[1] would be unaware of potential delays affecting the application while it is running, hence cannot compensate by giving it extra CPU cycles. Later, accumulating delays may cause a deadline to be missed, without the scheduler being able to react, due to late detection. Still, some delays might have been easy to prevent along the way, if properly detected, thus avoiding the missed deadline.

To overcome the above problem, the notion of *partial* deadline has been introduced in the literature [7]. In such approaches, developers have to decompose an application, or *task* into a number of *subtasks,* providing for each a partial deadline that contributes to meeting the final one.

This paper puts forward the idea of automatically introducing temporal constraints for any application activity that had been developed at an earlier time, ignoring time-related issues. For this purpose, we propose a solution for automatically estimating the execution time and (partial) deadlines of an application's subtasks.

At runtime, the management of temporal concerns is delegated to a software layer capable of observing the application, and checking whether the ongoing activities are executed within their calculated deadlines. Otherwise, appropriate actions are taken to overcome the problem, before it is too late. This allows a delayed execution to be detected, by monitoring partial deadlines, and adjusted as necessary, so as to allow final deadlines to be met.

Runtime monitoring and manipulation of the application is enabled by the mechanisms supplied by *computational reflection.* These ensure that real-time concerns are *seamlessly* integrated into an application, in the sense that its developer need only focus on functional concerns, whereas real-time issues are transparently handled by an additional reflective layer. In particular, the application developer need not manually change the previous design and implementation, spend any effort on achieving partial deadlines, or carry out any adjustment.

While we manage to ensure that most application deadlines are respected, we cannot guarantee one is occasionally missed. However, this is of course regarded as acceptable in a soft real-time framework.

An example application considered in our experiments is a file transfer utility designed without any time awareness, but later required to operate respecting appropriate deadlines. Our methodology allows appropriate adaptation measures to be transparently enacted, aimed at preventing the available bandwidth from being suddenly reduced by events occurring outside the application. Other scenarios that can benefit from our transparent adaptation methodology, in view of accommodating timing requirements unanticipated at design time, include image detection, data compression and (as discussed in [7]) trading-on-line applications.

[1] We essentially have in mind general purpose, non-real-time operating systems. However, our approach is still valid in the presence of real-time support, which, if anything, provides extra adaptation opportunities (cf. Section 5.4).

2 Outline of the Proposed Approach

We consider object-oriented applications that perform parallel execution of activities by means of concurrent threads. Since in real time terminology an application is also called a task, it is natural to identify subtasks with threads.

We deal with Java applications, for which we introduce a bytecode analyser and specialised algorithms for estimating execution time. Two features of the Java environment are crucial for letting the application be transparently monitored and adapted by the reflective metalevel:

- Java bytecode is relatively easy to instrument and many tools are available for this purpose (we employ the Javassist library, see Section 4.1). As a benefit, source code of applications is not required for our purposes.
- The classloader mechanism allows Java classes to be instrumented just before their execution.

The real-time concerns we cater for can be useful not only for applications for which real-time is a desired feature, but also for others whose overall "quality" can thus improve. Our aim is to support the development of soft real-time systems, since our approach cannot ensure the imposed deadlines are met *strictly;* this uncertainty is due to potential shortcomings in three areas: the possible insufficiency of the underlying platform's computing "power" at run time; the coarse knowledge that the proposed off-line analysis provides about the running application; and the possible unavailability of real-time support from the execution environment, at least as regards some components (e.g. scheduler, memory manager).

Assume a global deadline is to be imposed on a Java multi-threaded application developed regardless of time concerns. Our methodology for smoothly introducing into it the desired soft real-time support starts from the following off-line stages.

- Identifying the concurrent threads (i.e. subtasks) the application consists of. For this, an analysis of the application is performed, off-line, by a component called *Application Analyser.* This component reads the application code to find out its threads and their mutual relationships.
- Estimating the execution time of each identified thread. For this, two kinds of approaches have been proposed, based on observations of previous execution times, and a static analysis of executable code, respectively. We shall usefully combine both approaches for our purposes.
- Modifying the application bytecode by means of a component called *Transformer,* which selects some points of the application where *Sensors* are inserted. A Sensor is a component of the reflective layer able to capture control from an application, measure timings of interest during application execution and activate a component called *Controller,* which potentially triggers modifications on the application execution, as detailed in the following.

Once the application has been analysed and modified off-line as above, the following activities are performed on-line.

- Calculating a deadline for each thread. Based on the thread structure, determined by the Application Analyser, and knowing the specified global deadline, a component, called *Deadline Identifier,* computes the deadline of each thread before it starts.
- Analysing the timing data collected by Sensors. This is the responsibility of *Controller,* which compares collected data with deadlines. As an outcome, Controller determines whether runtime adjustments are needed to avoid missing subsequent deadlines.
- Adjusting application execution. When needed, Controller invokes a component called *Adapter,* which, depending on its settings, changes the application itself, typically to reduce the execution time of some of its portions.

The adjustment activities that can be performed depend on the underlying platform.

At the operating system/environment level, our approach is useful whether or not a real-time scheduler is at hand. If so, extensive scheduling control is likely to be available in order to ensure each thread's deadline is met. Otherwise, with a general purpose operating system, adjustment opportunities are more limited and consist essentially in varying thread priorities appropriately.

At the architectural level, the availability of a high performance parallel system gives us the flexibility to request additional CPUs when needed, so that, typically, the adjustment amounts to deciding how many CPUs to allocate for the application. In a different scenario, where (access to) computing power is constrained, adjustments aim at raising the scheduling priority of a critical activity, or substituting it with an alternative one faster albeit less precise.

In a distributed environment, adaptation can exploit the availability of multiple hosts by migrating application portions appropriately across the network.

3 Application Analyser

3.1 Threads and Their Dependencies

The *Application Analyser* is employed off-line to identify, within the application bytecode, the application threads and their mutual dependencies.

We distinguish two kinds of dependencies between threads. A thread has an *explicit dependency* from another thread, when the first is started by the second thread (executing a `start()`) or waits that the second thread finishes execution (executing a `join()`). A thread has an *implicit dependency* when it waits for an event that is started by another (unidentified) thread, e.g. a `synchronised` construct forces a thread to wait for other threads to exit such constructs, or a `wait()` invocation forces a thread to wait until another thread executes a `notify()` or a `notifyAll()`. The Application Analyser reveals all the explicit dependencies, whereas it only indicates some information about the implicit dependencies.

In order to obtain both explicit and implicit dependencies and the related temporal relationships between the threads of an application, we hold for a

thread the sequence of method invocations. This sequence is determined by a top-down bytecode analysis, starting from method `main()`. Threads are identified along the way, by noting whether an application class implements interface `Runnable` or extends class `Thread`.

Once threads have been discovered and characterised, we generate a timing relationship between them by estimating the execution time of each method and the starting time for each thread (see following subsection).

3.2 Estimating Execution Time

Our approach to estimating the execution time of methods combines those pioneered by Puschner and Koza [13] and Shaw [15]. The former introduces language constraints and techniques that allow the maximum execution time of a task to be calculated. The latter provides also means to give lower bounds on execution times. Thus, we are able to estimate the minimum and maximum execution times for Java bytecode, under the proviso that coding conforms to suitable rules, e.g. loops must be annotated with lower and upper bounds on the iterations they give rise to, and recursion should not be used.

For a thread the execution time is given by the sum of all the estimated execution time of the invoked methods and the other constructs it is made of.

Many runtime conditions can affect the accuracy of the estimation, such as the actual load on the executing host. Thus, we will integrate the estimation with runtime observations as explained in section 5.2.[2]

3.3 Timing Relationships

According to the estimated execution time of methods, the Application Analyser is able to infer when threads start executing, and which of them are executing concurrently at any given time.

The procedure employed for this purpose is the following. Let t denote a thread of the application, and *parent(t)* be the (only) thread that starts t. Then we can express the minimum and maximum starting time of t as:

$$tstart_{min}(t) = tstart_{min}(parent(t)) + \Delta_{min}(parent(t), t)$$
$$tstart_{max}(t) = tstart_{max}(parent(t)) + \Delta_{max}(parent(t), t)$$

where $\Delta_{min}(parent(t), t)$ and $\Delta_{max}(parent(t), t)$ are the minimum and maximum times elapsing from the start of $parent(t)$ to that of t. These are calculated by summing up the minimum and maximum execution times, respectively, of all the methods invoked by $parent(t)$ before that which starts t.

These formulas allow the starting time to be recursively determined for any thread of the application. The base case is given by the `main()` thread, which has both the minimum and maximum starting times equal to 0.

[2] In section 5.2 we show how to treat observed and estimated times homogeneously.

When a call to method `t.join()` is found, the duration of the invoking thread will be updated with the remaining execution time of the thread `t` on which `join()` has been invoked. The remaining execution time is obtained by summing up the starting time and duration of the thread, and subtracting the time when `join()` has been invoked.

4 Modifying the Application

Once the application structure has been determined and the execution times of threads have been estimated, as described in the previous section, the application is modified by means of component *Transformer.* Modifications are introduced by associating application classes with *metaobjects.* According to the *computational reflection* paradigm, metaobjects are capable of capturing control when operations are invoked on instances of their associated classes.

4.1 Computational Reflection

A reflective software system consists of a part that performs some computation, and another part that reasons about and influences the first part by *inspecting* it and *intercepting* its operations [9]. Usually, reflective systems are represented as two-level systems, where a *baselevel* implements an application and the *metalevel* monitors and controls the application. As reported in the literature, reflective systems have been used to enhance existing systems with support for additional concerns, e.g. synchronisation [17] and distribution [4].

One of the most widespread reflective models for object-oriented systems is the *metaobject model,* whereby instances of a metalevel class, called *metaobjects* intercept operations, e.g. *method invocations,* performed on objects that are instances of an *associated* baselevel class [9].

The association between a baselevel class and a metalevel class is performed in various ways, depending on the reflective environment employed. With the Javassist [2] reflective environment selected bytecode parts of baselevel Java application classes are injected with the appropriate jumps to the metalevel: thus, source code is unmodified and even unnecessary. Because of this advantage, and of its high quality, in this work we have chosen to use Javassist.

4.2 Transformer

Transformer associates selected application classes with metaobject `Sensor`, which is responsible to gather measures on the execution time of portions of the application, and transfer control to supporting classes that check and adjust the application at runtime.

Transformer chooses which application classes to associate, hence monitor, with `Sensor` on the basis of the (statically) estimated durations of threads that had been detected offline.

First, consider a thread having expected execution time substantially shorter than the remaining time between its end and the global application deadline. For such a thread, it suffices to monitor at runtime the starting and finishing time. Indeed, intervening in the application after the thread ends will not essentially have taken away many adjustment opportunities. Conversely, intervening before, which would require a finer monitoring, will not significantly add to them. E.g., if a thread is expected to last 10 ms and to finish 1s before the global deadline, intervening within the thread duration rather than just after it will hardly make any difference as to respecting the deadline.

In the above scenario, Transformer need only associate with metaobject `Sensor` all the application classes that potentially start a new thread, i.e. those that implement method `run()` or `main()`.

Conversely, for a thread that lasts significantly longer than the time between its end and the deadline, it seems reasonable to give `Sensors` a chance to take actions with a finer granularity, *along* its execution as well.

For this purpose, Transformer identifies methods that contribute substantially, with their own duration, to make up the duration of longer threads. Metaobject class `Sensor` is therefore associated with baselevel classes to which such methods belong. In this case, adjustment activities can be adopted whenever a monitored method terminates. As a consequence, deadlines need also be determined at the method level.

The overhead that metalevel operations introduce at runtime is kept low, since Transformer selects only some application classes for association with metaobjects. Moreover, when possible, jumps to the metalevel are inserted permanently into an application, thus avoiding the cost due to bytecode manipulation at runtime.

5 Software Architecture and Runtime Activities

5.1 Sensor

Metaobject `Sensor` is associated with application classes in order to capture events, i.e. thread creation, termination and intermediate stages, and invoke checking and adjusting activities as appropriate.

When a thread starts executing, `Sensor` notes its starting time and invokes metalevel class `DeadlineIdentifier` to calculate its deadline. When a thread finishes execution, `Sensor` measures the time again and determines the actual thread duration. The difference between the actual execution time and the statically estimated time for the thread is also determined and if it exceeds a threshold, method `check()` of metalevel class `Controller` is invoked to assess whether adjustment activities are necessary for the application.

By continually observing the application runtime behaviour, it is possible to detect the effects of components (both of the underlying infrastructure and metalevel ones) involved during execution. The overhead introduced by such components and potential scheduling anomalies can be treated by triggering adaptation activities (see Section 5.4).

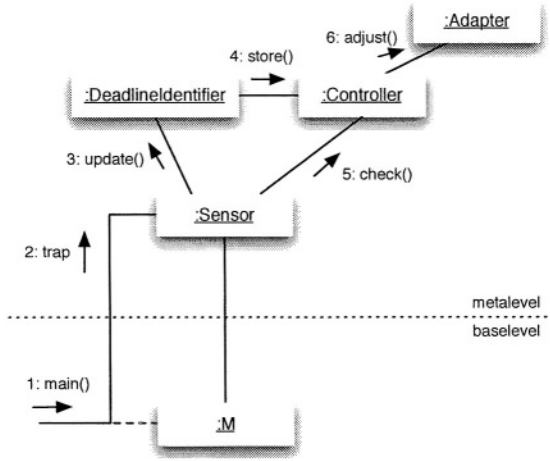


Fig. 1. Collaboration diagram between application classes and metalevel classes

The overall architecture and the interactions between the application and the observing and controlling metalevel classes are shown in figure 1.

5.2 Deadline Identifier

Metalevel class `DeadlineIdentifier` follows the approach proposed in [7] to derive subtask deadlines from a global task deadline. Conforming to the terminology used in [7], we identify the whole application and its threads with the main task and its subtasks, respectively. Mapping threads to subtasks agrees with the semantics of tasks in a real-time environment, because threads can be considered as work units scheduled by the underlying system. Moreover, all the threads constitute a complex functional unit that can be seen as the application [8].[3]

As discussed earlier, the proposed metalevel is capable of observing methods and threads. This allows deadlines to be calculated for subtasks (i.e. threads) and subsubtasks, which we identify with methods. Although the evaluation of subsubtask deadlines is costly, we only perform it when they contribute significantly to controlling the execution time as explained in section 4.2.

Deadlines are computed by means of the *Equal Flexibility Strategy (EQF)* formula from [7]. Basically, EQF sets the deadline of subtask k to its expected execution time eet_k plus a fraction of the total remaining slack proportional to eet_k itself.

For the sake of better estimating eet_k, we integrate the offline estimation $eet_{k,0}$ with the execution times $oet_{k,1}, oet_{k,2}, \ldots$ observed during the previous runs $1, 2, \ldots$ of subtask k.

[3] For Java applications the underlying system consists in the JVM, since the application cannot perform any operation outside the JVM.

Note that $eet_{k,0}$ is conceptually referred to a totally unloaded host, whereas, for observed execution times $oet_{k,1}$, $oet_{k,2}, \ldots$ to be comparable, they have to be normalised by the applicable host workload. We evaluate this with the *relaxation* concept [3], from which we derive a normalising factor that, multiplied by an observed time, yields the time that the subtask would require on an unloaded host. Thus, the normalised execution time for subtask k's ith run is

$$oet_{k,i} \cdot wl_{k,i}$$

where $wl_{k,i}$ is the normalisation factor that accounts for host workload during the run.

The expected execution time of subtask k when its (N+1)th run is about to start, denoted by $eet_{k,N+1}$, can now be estimated as follows:

$$eet_{k,N+1} = \frac{1}{wl_{k,N+1}} \cdot \left(\alpha \cdot \frac{1}{N} \sum_{i=1}^{N} oet_{k,i} \cdot wl_{k,i} + (1 - \alpha) \cdot eet_{k,0} \right) \qquad (1)$$

where $wl_{k,N+1}$ reflects the current workload, and α is a constant between 0 and 1. The higher α, the greater the influence, on the new estimate, of the previously measured versus the estimated execution time (in our experiments we had $\alpha = 0.6$).

5.3 Controller

The responsibility of class `Controller` is to determine whether the current state of execution for the application requires an adaptation. It is activated by `Sensor` when a thread takes more time than expected to finish. It holds the list of threads constituting the application, and the measured execution time of threads that have finished their execution, so it knows whether they have been executed within their deadline. At any time it is able to determine which threads are executing or going to be started in the following.

`Controller` decides to adapt the application on the basis of the evaluation of thread deadlines and minimum and maximum expected execution times. When one or more of the following conditions occur, an adaptation is triggered.

- The remaining time for the final deadline is less than the minimum expected remaining execution time.
- Some of the previously executing threads have also missed their deadlines.
- Some of the previously executing threads have executed for a time close to the maximum expected one.

The threads that have missed their deadlines are analysed so as to find out whether the same method has been executed by all of them. When this is the case, this method's name is passed as a parameter to class `Adapter` when invoking its method `adjust()`.

5.4 Adapter

Whenever an adaptation on the execution of the application is needed, class `Adapter` determines which adjustment is best to perform. This decision depends on both the state of the application and the underlying system. One or more of the following adjustments can be carried out.

- When threads of different applications are executing, we increase CPU availability for the application of interest by increasing the priority of those of its threads whose deadline is nearer.
- An alternative implementation is injected into the running application for one or more methods. The alternative implementation is typically characterised by a shorter expected execution time and produces similar results with a lower precision. This adaptation is essentially based on N-version programming. In this case, the invocation to the method selected by `Controller` will be re-directed to the alternative implementation (for the details on how to characterise and introduce a different version of a method refer to [5]).
- A portion of the application is executed on a remote unloaded host. For this purpose, selected application parts should have been replicated in advance on several hosts. The remote host to which computation is migrated is chosen according to its run time conditions, which should allow the deadline to be met, and the availability of the classes to be executed.
  Because of the way threads are described in Java, advance replication to remote hosts should be performed for classes implementing threads that might be migrated at runtime, and for classes these could (even indirectly) invoke. At runtime, before a thread can continue on a remote host, this should be enabled to update, as efficiently as possible, the state of the objects involved in executing the thread. We have presented an architecture supporting the runtime distribution of objects in [4].
  Finally, it should be noted that this adaptation strategy can only be used when the execution of a task does not depend on resources of the local host (e.g. local files). Such a dependence is revealed by the off-line analysis stage.

A wider range of effective adaptation strategies can be envisaged, if an execution environment is available that is more controllable than the standard JVM in terms of timing aspects, e.g. a RTSJ [14] compliant JVM.

6 Concluding Remarks

The reflective architecture we have proposed in this paper aims at suitably introducing real-time concerns into an unaware application. The reflective support allows us to compare the timing requirements and profiles of an application with its observed execution behaviour. This runtime knowledge is exploited to adjust dynamically the way a computation is performed, by determining which parts of the application should execute first or be supported by additional, possibly distributed hardware and software resources. As a result, we can avoid relying

exclusively on the a priori evaluation of worst case execution times, which helps reducing scheduling overhead and resource (over) provisioning. In this respect, the proposed reflective system affords increased flexibility, and, although it cannot ensure hard deadlines are always met, it can effectively aid the development of soft real-time applications.

A seminal work on reflection applied to real-time systems is [16]. Here, a reflective architecture is embedded into the operating system, in order to gain some knowledge of the running applications, and accordingly adjust task scheduling at runtime. Compared with our work, the information their reflective level uses is much wider, hence more effective as to distinguishing critical deadlines and adapting to meet them. However, the knowledge that the reflective level relies upon cannot be obtained in an automatic fashion, and developers are therefore asked to specify application requirements. Moreover, adaptation measures do not include the possibility of changing selected application parts.

In [10] a reflective system is proposed that performs on-line task scheduling to adapt to changes in the environment. Application developers provide deadlines for their tasks, and alternative tasks to execute when the specified deadline cannot be met. Changes that may be revealed at runtime concern: the system operating mode, and, for any task, its worst case (observed) execution time and the "utility" the system ascribes to it completing. Adaptation measures performed, when runtime conditions do not let an original task meet its deadline, may consist in raising its scheduling priority or executing an alternative task.

This approach presupposes that the developer specifies the deadline for a task, and requires the reflective system to decide whether to execute the whole task or an alternative one. In contrast, we exploit the automated decomposition of tasks into smaller activities, i.e. subtasks-threads and, possibly, subsubtasks-method calls. For all of these we can derive partial deadlines based on static analysis complemented by run-time observations. Thanks to this, both observations performed and adaptation measures taken enjoy a finer granularity. Single method calls can be re-directed, and deadline checking is performed throughout subtask execution, triggering early runtime adjustments when appropriate.

Nett et al. [11,12] propose code instrumentation to incorporate time awareness into object-oriented applications. Their runtime system consists of a monitoring part that collects measures on execution times, and a fault tolerant scheduler that compares the actual execution time of a task with its estimated time, stopping execution when these times differ. In this event, the scheduler handles the time fault by executing an exceptional task. This approach, unlike ours, is based on two assumptions: (i) a task, albeit interrupted, is still able to deliver an acceptable result, and (ii) beside ordinary tasks, an exceptional one is available to bring the application to a fail-safe state. Moreover, their runtime system is adaptive in the sense that the scheduler determines on-line which task to execute, whereas we adapt the application by executing alternative portions of it, possibly on a remote host.

Blair et al. [1] propose an adapting middleware that applications can inspect and reconfigure by means of metalevel interfaces. Their approach aims at,

e.g. choosing the most appropriate network protocol for the infrastructure and bandwidth available for multimedia streaming, rather than enabling originally time-unaware applications to take into account timing issues.

Also [6] focuses on transparent adaptation at the middleware rather than the application level.

References

1. G. Blair, G. Coulson, et al. The Design and Implementation of Open ORB 2. IEEE Distrib. Syst. Online 2, 6 (Sept. 2001).
2. S. Chiba. Load-time Structural Reflection in Java. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP'00),* volume 1850 of *Lecture Notes in Computer Science,* pages 313–336. Springer-Verlag, 2000.
3. A. Di Stefano, L. Lo Bello, and E. Tramontana. Factors Affecting the Design of Load Balancing Algorithms in Distributed Systems. *Journal of Systems and Software. Elsevier,* 48:105–117, October 1999.
4. A. Di Stefano, G. Pappalardo, and E. Tramontana. Introducing Distribution into Applications: a Reflective Approach for Transparency and Dynamic Fine-Grained Object Allocation. In *Proceedings of the Seventh IEEE Symposium on Computers and Communications (ISCC'02),* Taormina, Italy, 2002.
5. A. Di Stefano, G. Pappalardo, and E. Tramontana. An Infrastructure for Runtime Evolution of Software Systems. In *Proceedings of the Ninth IEEE Symposium on Computers and Communications (ISCC'04),* Alexandria, Egypt, 2004.
6. C. D. Gill, J. M. Gossett, J. P. Loyall, et al. Integrated Adaptive QoS Management in Middleware: A Case Study. In *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 04),* 2004.
7. B. Kao and H. Garcia-Molina. Deadline Assignment in a Distributed Soft Real-Time System. *IEEE Transaction on Parallel and Distributed Systems,* 8(12): 1268–1274, 1997.
8. J. W. S. Liu. *Real-Time Systems.* Prentice Hall PTR, 2000.
9. P. Maes. Concepts and Experiments in Computational Reflection. In *Proceedings of OOPSLA'87,* volume 22 (12) of *Sigplan Notices,* pages 147–155, 1987.
10. S. E. Mitchell, A. Burns, and A. J. Wellings. Developing a Real-time Metaobject Protocol. In *Proceedings of IEEE Third International Workshop on Object-Oriented Real-Time Dependable Systems,* February 1997.
11. E. Nett, M. Gergeleit, and M. Mock. An Adaptive Approach to Object-Oriented Real-Time Computing. In *Proceedings of IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC'98),* Kyoto, April 1998.
12. E. Nett, M. Gergeleit, and M. Mock. Enhancing O-O Middleware to Become Time-Aware. *Real-Time Systems,* 20(2):211–228, 2000.
13. P. Puschner and C. Koza. Calculating the Maximum Execution Time of Real-Time Programs. *Journ. Real-Time Systems,* 1(2):160–176, 1989.
14. Real-Time for Java Expert Group. RTSJ web site, 2004. http://www.rtsj.org/.
15. A. C. Shaw. Reasoning about time in higher-level language software. *IEEE Trans. on Software Eng.,* 15(7):875–889, Jan. 1989.
16. J. A. Stankovic and K. Ramamritham. A Reflective Architecture for Real-Time Operating Systems. In *Principles of Real-Time Systems.* Prentice-Hall, 1995.
17. E. Tramontana. Managing Evolution Using Cooperative Designs and a Reflective Architecture. In *Reflection and Software Engineering,* volume 1826 of *Lecture Notes in Computer Science,* pages 59–78. Springer-Verlag, June 2000.

Workshop on Modeling Inter-Organizational Systems (MIOS) PC Co-chairs' Message

The digital revolution has just begun. Globalization, increasing competitive pressure, mergers and acquisitions and, above all, new technological innovations force many enterprises to undergo drastic changes that cannot be coped with traditional change and restructuring programs.

Flexibility has become the new guiding principle for companies and requires flexible structures and business processes based on modern IT-infrastructures. At the core of this change processes are inter-organizational systems (IOS) which enable the integration of business processes between companies thus increasing flexibility of the business system and improving inter-company collaboration in value networks.

Those inter-organizational information systems are enabled by innovations in information and communication technologies, primarily the emergence of the Internet. However, many current practice initiatives in this field have not fully met expectations. One of the main reasons for this can be found in a lack of proven procedures to define cross-enterprise business processes and in the high level of complexity associated with the modeling of large value networks.

The objective of the MIOS workshop was to gather researchers working on different emerging topics concerning inter-organizational systems and cross-enterprise collaboration, covering issues ranging from modeling, conception and realization of inter-company application systems to standards and integration of application systems.

The MIOS 2004 workshop received a total of 24 submissions, out of which 12 papers were selected by the Program Committee. The papers selected cover a wide spectrum of topics, including cross-enterprise business processes, management of virtual enterprises, B2B integration processes, ontologies for organizations, collaboration models, component models, and more.

The MIOS 2004 workshop attracted people from 13 different countries, with a wide and uniform geographical distribution of the paper's authors. The MIOS workshop gave to the attendees the opportunity to enjoy a forum for discussing actual inter-organizational topics. The link to the workshop can be found at the following site: http://wi-se.wiwi.uni-augsburg.de/mios04.php

We would like to thank the contributing authors for making the MIOS 2004 workshop a success and do hope that the presented results and the interesting discussions during the workshop may contribute to your work. We count on you, as well as on many others, for the next year's MIOS edition!

August 2004

Antonia Albani
Klaus Turowski
(MIOS 2004 Program Committee Co-chairs)

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, p. 418, 2004.
© Springer-Verlag Berlin Heidelberg 2004

The e-Lollapalooza Global Workflow Modeler: A Registry-Based Workflow Modeling Methodology and System

Kwanghoon Kim, Hyukjae Ryoo, and Jaekang Won

Collaboration Technology Research Lab.
Department of Computer Science, Kyonggi University
Suwonsi Kyonggido 442-760, South Korea
{kwang,hjyoo,jkwon}@kyonggi.ac.kr
http://ctrl.kyonggi.ac.kr

Abstract. Recently, there are two newly emerging research issues in the workflow literature according to the proliferation of electronic commerce, electronic logistics, and electronic supply chain management markets - One is the inter-organizational workflow issue interconnecting workflow engines across interrelated organizations, and the other is the XML-based data and application integration issue interchanging workflow data and applications across the organizations. Based upon these emerging issues, we have been developing a registry-based global workflow management system, which is named 'e-Lollapalooza Global Workflow Management System,' especially targeting on the electronic logistics/commerce markets. The e-Lollapalooza consists of four components: e-Lollapalooza Builder, e-Lollapalooza Miner, e-Lollapalooza Actor and e-Lollapalooza Enactor. This paper deals with the e-Lollapalooza Builder, a registry-based global workflow modeling system, that defines cross-organizational workflows for e-logistics and e-commerce. The e-Lollapalooza Builder can be characterized by the concept of registry that is similar to ebXML's except registering inter-organizational workflows formatted in WPDL/XPDL. So, the registry component of the e-Lollapalooza Builder plays an important role in interconnecting workflow engines deployed in the associated organizations. We develop the builder by Java language and EJB framework approach so *as* to be deployed on various platforms without any further technical consideration. And its major components are the EJB-based DB component, graphical global-workflow modeler, organizational dependency analysis algorithm, organization, relevant data, invoked application, repository, and the registry component. The system is fully deployable and operable on EJB-based computing environment. In this paper, we describe detailed features of the e-Lollapalooza Builder in terms of conceptual and functional perspectives.

Keywords: Global Workflow, Workflow Modeling Methodology and System, e-Logistics.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 419–430, 2004.
© Springer-Verlag Berlin Heidelberg 2004

1 Introduction

There exist a lot of technologies which appear and disappear in our society. Among them, we can easily recognize that convergence of those technologies surviving recently has appeared to bring the world closer together, physically and conceptually. Furthermore, we can categorize those technologies into two groups according to the subjects of the world that they are aiming at - people or organization. That is, One is a group of technologies targeting on bringing *people* closer together, the other is a group of technologies making *organizations* much closer together. Groupware technology including traditional workflow, enterprise resource planning (ERP), and enterprise application integration (EAI) technology are typical examples of the former, while computer-aided logistics management (CALM) technology including e-logistics, e-commerce, e-market place including ebXML, e-supply-chain management, XML-based electronic data interchange (EDI) and inter-organizational workflow technology, which are recently proliferating and hyping in the information technology market, are of the latter. Conceptually speaking, the former technology is focusing on enhancing efficiency and productivity of *intra-organizational* group activities. And, the latter is to promote *inter-organizational* cooperative activities. Consequently, we would proclaim that the focus of information technological trend be shifted from supporting cooperative work of group of *people* to supporting cooperative work of group of *organizations.*

We guarantee indubitably that the workflow technology should play a very important role, in terms of the functional points of view, in both of the technology categories, supporting not only cooperative people works but also cooperative organization works, which can be dealt with the intra-organizational workflow and the inter-organizational workflow, respectively. Based upon these conceptual backgrounds, we have been developing a registry-based global workflow management system, which is named 'e-Lollapalooza Workflow Management System,' especially targeting on the electronic logistics for the postal service market. This three-year on-going research and development work, funded by the government (the Ministry of Information and Communication), is executed by the Collaboration Technology Research Laboratory of the Kyonggi University and the Electronics and Telecommunications Research Institute. The system has four components such as Builder, Miner, Actor, and Enactor. Currently, the e-Lollapalooza project has successfully completed the first year mission to develop the e-Lollapalooza builder, a registry-based global workflow modeling system, that defines cross-organizational workflows hooking up with a set of organizations co-related for e-logistics or e-commerce. The focus of the builder is on incorporating complex and non-sequential ebXML-based process models into the workflow models. In other words, we can say that ebXML is an open e-commerce/e-logistics framework for unknown multiple partners over the internet, while the e-Lollapalooza builder is a closed e-commerce/e-logistics framework only for pre-contracted multiple partners over the intranet/extranet. This paper describes the detailed architectural implementation and operational examples of the closed e-logistics framework, the e-Lollapalooza builder.

In the consecutive sections, we present the detailed design of the e-Lollapalooza builder including its overall architecture, global workflow modeling features and its registry mechanism, as well as the implementation and its operational examples of the system, respectively.

2 Overall Architecture of the e-Lollapalooza

In the inter-organizational workflow management, it is hard to deploy and manage a tightly coupled architecture of the build-time and the runtime components, because the workflows are interconnected for the aim of realizing the business process across organizational boundaries, each of which installs its own heterogeneous workflow enactment component. Conclusively in this paper, we propose and realize a way how to implement a tightly coupled architecture for the inter-organizational workflows management. We call the interoperation of workflows *global-workflow* and the total support technologies *global-workflow management mechanism.* We aim, in the ongoing project, that the global-workflow is expected as a supporting mechanism for business-to-business electronic logistics/commerce, and call it e-Lollapalooza. In this section, we briefly describe an overall architecture of the e-Lollapalooza workflow management system targeting on a type of the closed e-business framework, which is categorized into the 'heterogeneous workflow engine with registry', supporting global-workflows for smooth cooperation among organizations. The e-Lollapalooza consists of four components: e-Lollapalooza Builder, e-Lollapalooza Miner, e-Lollapalooza Ac-

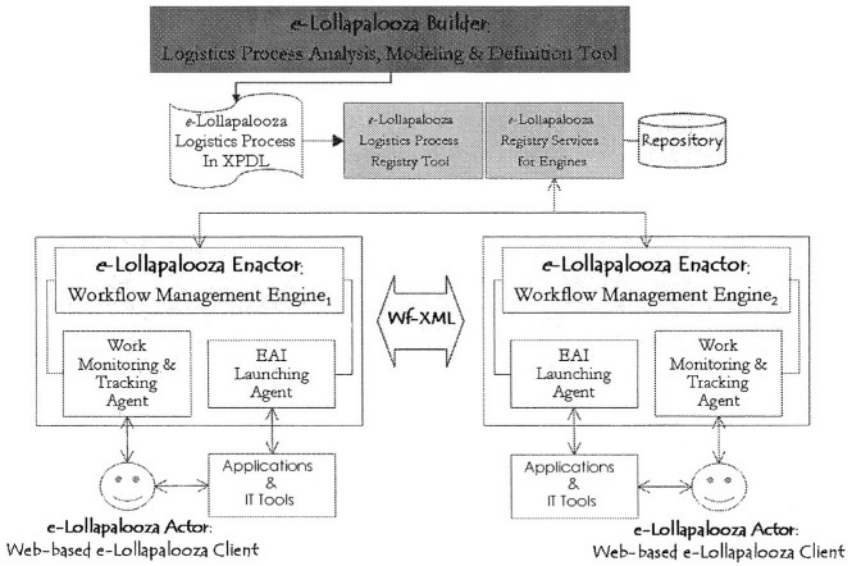


Fig. 1. Overall Architecture of the e-Lollapalooza Global WfMS

tor and e-Lollapalooza Enactor. We describe first about several system-level requirements that should be accomplished on the final stage of the e-Lollapalooza development, and then explain summarily each of the components depicted in Fig. 1.

3 The Global-Workflow Modeling Methodology

The e-Lollapalooza global-workflow modeler is focusing on the definition of cooperative business processes across industries and companies particularly targeting on e-Logistics and e-Commerce domains. And, in terms of the global-workflow modeling and definition of the cooperative business process, it is very important for the dilemma described in [5] to be resolved. That is, the more independence of organizations is guaranteed, the better each organization has the ability or the right to decide things on its own. However, in contrast to this, for the construction of a global-workflow, the more interaction among the cooperative business processes is precisely specified, the more the methodology for describing the global-workflow process in one place has advantages. For the sake of resolving the dilemma, we take a different approach from the [5]'s approach. We would name the [5]'s approach "black box approach," in which the internal process in each organization is treated as a black box. While on the other, our approach would be named "white box approach," because all of the internal processes associated with a global-workflow can be precisely described in one place where the e-Lollapalooza modeler is located.

3.1 The Global Information Control Net

The original Information Control Net [6] was developed by researchers (C.A. Ellis, G. Nutt, et el.) at Xerox PARC to describe and analyze information flow within offices. It has been used within actual as well as hypothetical automated offices to yield a comprehensive description of activities, to test the underlying office description for certain flaws and inconsistencies, to quantify certain aspects of office information flow, and to suggest possible office restructuring permutations. The ICN model defines an office as a set of related procedures. Each procedure consists of a set of activities connected by temporal orderings called procedure constrains. In order for an activity to be accomplished, it may need information from repositories, such as files, forms, and some data structures. An ICN captures these notations of procedures, activities, precedence, repositories, and organizations in graphical forms. In order to support global-workflow modeling with multiple organizations, we extend the basic ICN model so as to incorporate the notion of organization dependency model including global-actors, global-groups, global-organizations. Our extended global ICN definition fits the ICN schema defined in [6], and is thus a valid member of the ICN family of models. The following is the formal definition of the extended ICN for global-workflow processes:

Definition 1. *A global ICN is 6-tuple* $\Gamma = (\delta, \gamma, \varepsilon, \kappa, I, O)$ *over a set* A *of activities, a set* R *of repositories, and a set* C *of organizations, where*

- I *is a finite set of initial input repositories, assumed to be loaded with information by some external process before execution of the ICN;*
- O *is a finite set of final output repositories, perhaps containing information used by some external process after execution of the ICN;*
- $\delta = \delta_i \cup \delta_o$
  where, $\delta_o : A \longrightarrow \wp(A)$ *is a multi-valued mapping of an activity to its sets of (immediate) successors, and* $\delta_i : A \longrightarrow \wp(A)$ *is a multi-valued mapping of an activity to its sets of (immediate) predecessors; (For any given set* S, $\wp(S)$ *denotes the power set of* S.)
- $\gamma = \gamma_i \cup \gamma_o$
  where $\gamma_o : R \longrightarrow \wp(A)$ *is a multi-valued mapping (function) of an activity to its set of output repositories, and* $\gamma_o : R \longrightarrow \wp(A)$ *is a multi-valued mapping (function) of an activity to its set of input repositories;*
- $\varepsilon = \varepsilon_a \cup \varepsilon_p$
  where $\varepsilon_p : C \longrightarrow A$ *is a single-valued mapping of an activity to one of the roles, and* $\varepsilon_a : A \longrightarrow \wp(C)$ *is a multi-valued mapping of a role to its sets of associated activities;*
- $\kappa = \kappa_i \cup \kappa_o$
  where κ_i *: sets of control-transition conditions, T, on each arc,* $(\delta_i(\alpha), \alpha)$, $\alpha \in A$; *and* κ_o *: sets of control-transition conditions on each arc,* $(\alpha, \delta_o(\alpha))$; *where the set* T = {*default, or(conditions), and(conditions)*}.

3.2 The Global-Workflow Modeling Methodology

As pointed out the previous section, to avoid the dilemma, [5]'s black box approach is that at first the internal workflow in each organization can be treated as a black box, making only the linkage with others visible. Second, the point described as a black box can be added internal information in each organization. Therefore, they proposed a hierarchical process description methodology [5] for constructing interoperation among organizations. As we stated in the previous section, this methodology is reasonable for the open e-business framework. However, it needs additional education for modeling inter-organizational workflows as well as a set of translators. Especially, the number of translators is exactly same to the number of heterogeneous workflow engines that are associated with organizations, respectively. That's tough! Furthermore, the methodology's construction unit is not the activity but the process, because the internal process of a global-workflow is treated as a black box. On the other hand, after taking a lesson from the disadvantages of [5], we propose a global-workflow modeling methodology that should be perfectly applicable for the closed e-business framework.

(a) The Organization Dependency Analysis Framework

In order to realize the methodology for constructing the cooperative global-workflow model, we propose a framework that is a formal approach to analyze

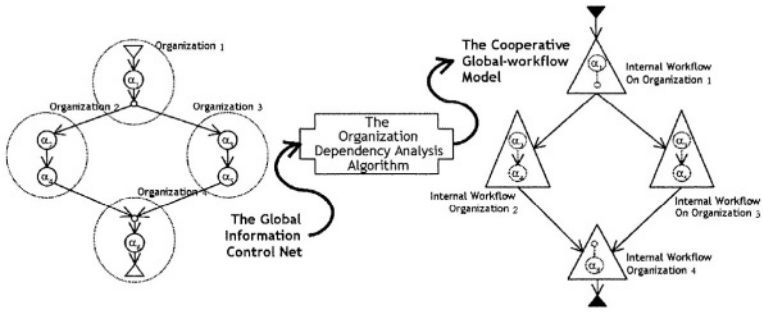


Fig. 2. The Organization Dependency Analysis Framework

dependencies among organizations existing on an extended global information control net model. Fig. 2 shows a fairly reasonable viewpoint regarding abstractions of the organization dependency analysis framework automatically transforming from the global information control net (a procedure-driven model) to the cooperative global-workflow model (a dependency-driven model), which is formally translated into a condition-driven model after all, through the organization dependency analysis algorithm. The detailed analysis works of the dependency driven model are described in the next section.

(b) The Dependency-Driven Model for the CooperativeGlobal-Workflow Model

An organization dependent net represents the order of transitions among organizations within a global-workflow. It is constructed out of the sets δ (control flow part) and ε (organization assignment part) in a global ICN model. (Note that the notations and their meanings are the same as that used in the global ICN's definition, if they are not redefined.) The followings formally and graphically define the organization dependent net, conceive its construction algorithm, apply the algorithm to the hiring global-workflow procedure [7] as an example, and formally specify the organization dependent net of the hiring global-workflow procedure.

Definition 2. *An organization dependent net is formally defined as* $\bigwedge = (\sigma, \psi, S, E)$, *over a set* $\mathbf{P}$ *of organizations and a set* $\mathbf{A}$ *of activities, where*

- $\sigma = \sigma_i \cup \sigma_o$
  where $\sigma_o : \mathbf{P} \longrightarrow \wp(\mathbf{P})$ *is a single-valued mapping of an organizationto its set of (immediate) successors including itself, and* $\sigma_i : \mathbf{P} \longrightarrow \wp(\mathbf{P})$ *is a single-valued mapping of an organization to its set of (immediate) predecessors.*
- $\psi = \psi_i \cup \psi_o$
  where ψ_i*: a set of activities,* $\alpha \in \mathbf{A}$*, on each arc,* $(\sigma_i(\eta), \eta)$*; and* ψ_o*: a set of activities,* $\alpha \in \mathbf{A}$*, on each arc,* $(\eta, \sigma_o(\eta))$*, where* $\eta \in \mathbf{P}$*;*
- $\mathbf{S}$ *is a finite set of the initial activities, assumed to be loaded by some global procedures;*

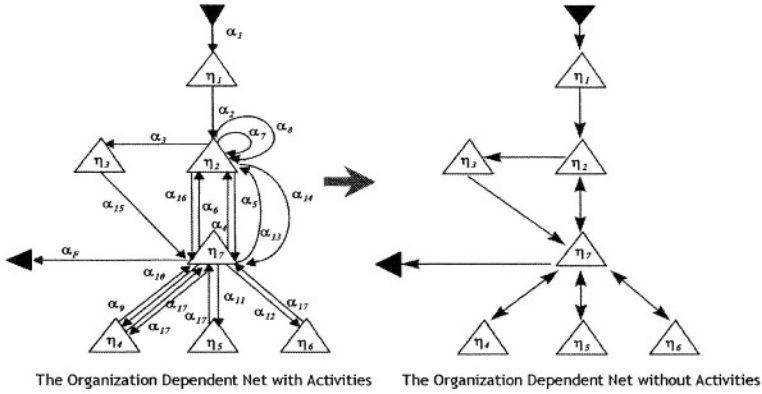


Fig. 3. The Organization Dependency Analysis Framework

– **E** *is a finite set of the final activities, perhaps containing activities of some global procedures.*

Organization Dependent Analysis Algorithm. An organization dependent net is constructed from a global information control net through the following algorithm. The construction algorithm of an organization dependent net for an global information control net can be computed in $\boldsymbol{O}(|\boldsymbol{A}|)$.

INPUT A global ICN;
OUTPUT An organization dependent net;
BEGIN
  FOR all $X \in \mathbf{A}$ *in the global ICN*
    Get an organization flow subnet.
    ADD $\varepsilon_p(X) TO \sigma_i(\varepsilon_p(\delta_o(X)))$;
    ADD $\varepsilon_p(\delta_o(X)) TO \sigma_o(\varepsilon_p(X))$;
    Get an activity on the arc between two organizations.
    ADD $XTO\psi_i(\varepsilon_p(\delta_o(X)))$;
    ADD $XTO\psi_o(\varepsilon_p(X))$;
  ROF;
END.

The Hiring Global-workflow Model [7]. Fig. 3 is to represent the organization dependent net automatically constructed by the organization dependency analysis algorithm. The left side of the figure is a graphical representation of the organization dependent net for the hiring global-workflow procedure. It shows the enactment flows of activities across organizations as well as the control transitions among organizations. The right side simplifies the organization dependent net to illustrate only the dependent relationships among organizations. Single arrow represents the directed dependent relationship, and a double arrow represents the mutual dependence relationship between two organizations.

(c) The Condition-Driven Model for the Cooperative Global-Workflow Model

After all, a procedure driven model can be physically represented and transformed into a set of organization-based transition conditions consisting of organization transition conditions, activity transition conditions, and data transition conditions. Each organization has its own organization transition conditions making up FROM and TO organization set. Once an active organization component takes over the control from one of members of the FROM organization set. It will then decide which activity shall be enacted by considering the activity transition conditions, PRE and POST activity set. At this point, the selected activity will check its data transition conditions out of the data transition conditions, INPUT and OUTPUT repository set.

As a result, the enactment part of a global-workflow management system can manage its enactment control of activities through data flow, activity flow, and organization flow information. The global-workflow enactment part can take the information from the repository managed by the registry component. The organization transition conditions come from the organization dependent net. The activity transition conditions come from the activities' precedence information. And the data transition conditions are generated by the relevant data flow information. That is, the condition-driven model of a cooperative global-workflow model is finally transformed into WPDL/XPDL and stored in the repository of the registry component.

4 Implementation Details of the e-Lollapalooza Builder

In this section, we describe the implementation of the e-Lollapalooza Builder, which is a part of the e-Lollapalooza global-workflow management system that the CTRL research group of the Kyonggi University has been developing especially for the e-Logistics and e-Commerce framework, and that is funded by the Korea Science Engineering Foundation (No. R05-2002-000-01431-0).

4.1 Overall Architecture of the e-Lollapalooza Builder

The functional architecture of the e-Lollapalooza Builder is presented in Fig. 4. It has two major components. One is the global-workflow modeler that is used to define global-workflow through the graphical user interface. The other is the organization-based registry that has a repository for XML-based global-workflow models and provides a connection to workflow engine components. And the interactions between the modeler and the registry, such as **Log-in**, **Request**, and **Resultset**, are performed by the registry agents residing in each component. The agent in the modeler side becomes a client registry, and then the one in the registry side is a server registry. We briefly describe about each component of the e-Lollapalooza Builder in this section.

The Global-workflow Modeler. The modeler provides a set of functions and capabilities to define, verify, and simulate ICN-based global-workflow processes.

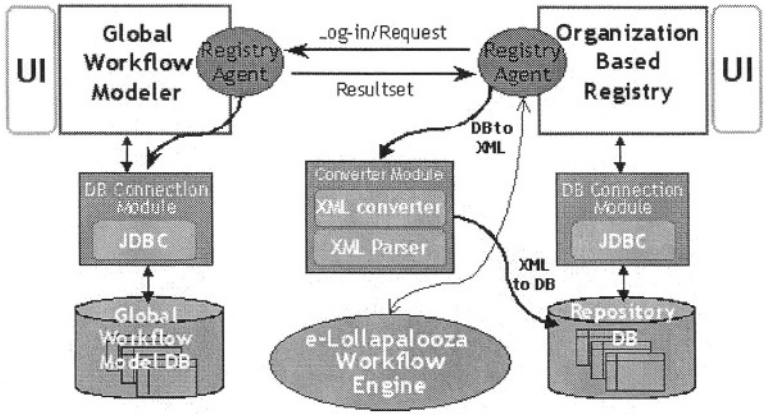


Fig. 4. Overall Architecture of the e-Lollapalooza Builder

User is able to draw a global-workflow through graphical icons that are exactly identical to the graphical notations of the information control net model. The major information dealt with the modeler consists of four dimensions of the global-workflow definition - global-workflow process models, global-organization information, relevant data structures, and the invoked application programs. After modeling a global-workflow process, those related information are stored on the global-workflow model database through the JDBC operations of the database connection module. Moreover, the modeler provides a feature for verifying the defined global-workflow processes through the graphical animation approach, which is called XML-based open global-workflow animation tool. At the same time, it has a functionality to automatically generate the global organization dependency net from the global-workflow models by the organization dependency analysis algorithm.

The Database Connection Module. The module (agent) provides a set of JDBC-based APIs that are used for the global-workflow modeler to access its database so as for the modeler to be unable to directly access to the database. That is, the module is located between the build-time clients and the database system. Also it is deployed on EJB sever as a component. Through the EJB technology, we are able to accomplish a stable database agent that can be characterized by the distributed object management, reliable recovery mechanism from system failures, reliable large-scale transaction management, and the security functions. The builder component may reside in a local or a remote to the enterprise bean, and never directly accesses the ECDbAgentEJB. The ECDbAgentHome acts as a factory to manage the life cycle of a ECDbAgentEJB. The ECDbAgent receives the method call from the build-time component, works with the container to check security and add transaction context before passing it to the ECDbAgentEJB. *The Organization-based Registry.* The registry manages the information of global-workflow models transformed in XML. It consists of two major modules, the XML converter module and the database connection module. The former

takes in charge of conversion of the organization-based global-workflow models, which is formed in the condition-driven model as stated in the previous section, into the XML format (WPDL/XPDL). The later has the database access functionality that is exactly identical with the database connection module in the model component. Moreover, it carries out connections with the global-workflow enactment engines that are operable on the EJB computing environment.

4.2 Operational Examples

In this section, we would present a couple of screen captures out of the e-Lollapalooza Builder. We would simply prove, through these selected screens, that the builder is completely workable on the real environment. When user first starts e-Lollapalloza builder, he/she can see the build-time management client window Then, the user can login through a valid build-time user ID. Through this client, the user is able to directly connect to all components of the e-Lollapalooza Builder - Modeler, Application Program Manager, Relevant Data Structure Manager (Repository), and Organization Manager.

Fig. 5 is a screen capture of the global-workflow modeler, which simply shows a global-workflow process definition containing four nested global-workflows. At the same time, we show a set of icons that represents, from the left-hand side, the start node, end node, activity, nested activity, cooperative activity, AND node, OR node, loop activity, relevant data, control-flow arrow, and the dataflow arrow, respectively. Especially, this modeling tool supports for a group of users to be able to simultaneously edit a global-workflow process. When user first starts modeling tool and load the sample global-workflow, he/she sees the modeling window as shown in Fig 5. There is a tree view on the right-hand side of the modeling tool window that shows all the objects that belong to global-workflow models. The left-hand side of the window is a work area that is used to display views of global-workflow elements. This can be the diagram view of a process or the properties that you can define for a selected object. At the bottom of the build-time window, there is a status bar. The status bar shows information such as the name of the database you are using and your user ID.

Organization Agent table representing members' information of the global organizations. It must be clicked on the organization index of the tree to see the detailed information. To define a global organization information, we implement the organization agent, which has three sub-panels (1. The tree panel, 2. summary panel, and 3. the detail form panel). The tabs at the top of the detail form view provide a quick way to switch between the different forms. The tabs indicate that user can display form for organization, department, role, member, grade, and search. The right-hand side of the window is a tree that has several organizations' names. It also displays another information, such as its associated department and role, staff names. If you click the wfms of department name, then on the summary panel the members of the wfms department is displayed, and on the detail form panel the wfms information automatically is given. At that time, if you click a member in the summary panel, then the detail form panel shows the member's detailed information. the registry client interface representing several

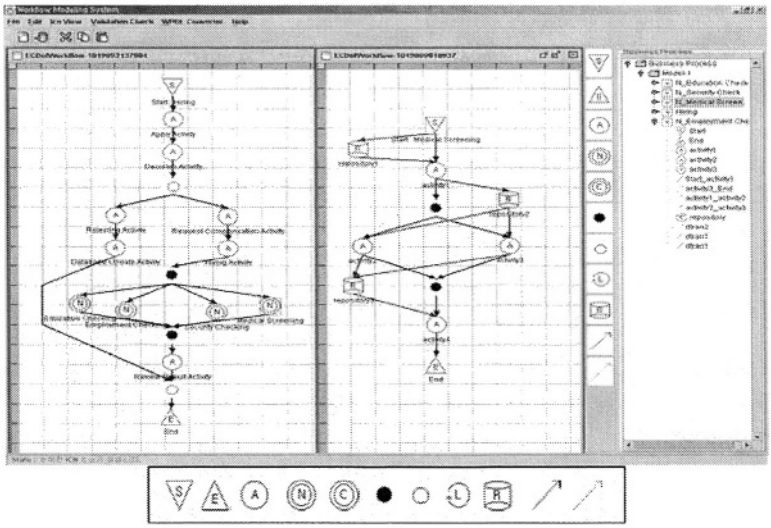


Fig. 5. Global-workflow Modeler and ICN Icon Palette

global-workflow models transformed in XML (WPDL/XPDL), which are stored in the repository database. The tabs on the detail form panel represent a series of registered global-workflow information. Especially, this screen shows the XML representation of the Hiring global-workflow model. Note that you can see the highlighted one on the tree of registry panel of the right-hand side of the screen.

5 Conclusion

So far, we have proposed a global-workflow modeling methodology and implemented a global-workflow modeling system, which is named the e-Lollapalooza Builder. We strongly believe that the modeling system fortifies the modeling methodology, and vice versa. Also we are sure that the modeling methodology and system fits very well into the closed e-business framework with heterogeneous workflow architecture with registry. Especially, the methodology might have a strong advantage in terms of the high degree of modeling ability for cooperative global-workflow processes. At the same time, it should be very applicable to the type of coordinative global-workflow model that is defined and characterized, by ourselves, with tightly coupled global-workflows in cooperative organizations.

Not only in South Korea but also in world-wide research arena, workflow and its related technological fields, including e-Business and e-Logistics, are catching great attentions from the society of information science and database management fields. So, there are a lot of challenges to develop and commercialize e-business solutions. This paper should be one of those active attempts for pioneering global-workflow modeling methodologies and systems toward supporting cross-organizational workflows in cooperation. As our further research works, we

are going to implement the complete set of the e-Lollapalooza global-workflow management system's components. Of course, the system will be based upon the modeling methodology and system described in this paper.

Acknowledgement. This work was supported by grant No. R05-2002-000-01431-0 from the Basic Research Program of the Korea Science & Engineering Foundation.

References

[1] Workflow Management Coalition Specification Document, "Workflow Coalition Interface 1: Process Definition Interchange Process Model", Document Number: WFMC TC-1016-P, August 5, 1998.
[2] Workflow Management Coalition Specification Document, "Workflow Management Application Programming Interface (Interface 2&3) Specification", Version 2.0e, Document Number: WFMC TC-1009, July 1998.
[3] UN/CEFACT and OASIS, "ebXML Registry Information Model v1.0", May 2001
[4] UN/CEFACT and OASIS, "ebXML Registry Services Specification v1.0", May 2001
[5] Haruo Hayami, Masashi Katsumata, Ken-ichi Okada, "Interworkflow: A Challenge for Business-to-Business Electronic Commerce", Workflow Handbook 2001, WfMC, October 2000
[6] Clarence A. Ellis, "Formal and Informal Models of Office Activity", Proceedings of the 1983 Would Computer Congress, Paris, France, April 1983
[7] Kwang-Hoon Kim, Clarence A. Ellis, "Performance Analytic Models and Analyses for Workflow Architectures", Information Systems Frontiers, Vol. 3, No. 3, pp. 339-355, 2001

Concepts of Flexibility for Efficient Integration of B2B Processes

Jelena Zdravkovic

Department of Computer and Systems Sciences
University of Gävle and Royal Institute of Technology
Kungsbäcksvägen 47, 80176 Gävle, Sweden
jzc@dsv.su.se

Abstract. Business-to-Business (B2B) e-commerce is emerging as trading parties are attempting to integrate electronically, to automate exchange of their services. To be able to collaborate, enterprise processes must expose compatible public behavior. It is a common need for a company to collaborate a business with many partners. A problem is that, even agreeing on information to be exchanged, the partners usually expose different requirements for protocol and logic of interactions. Therefore, it becomes necessary for companies to redesign process models to accommodate to a new partner. As the result, the company must engage considerable resources for designing and verifying new process specifications, as well as for maintaining them. In this paper, we propose a framework for flexible modeling of enterprise processes, to support a larger scale of B2B integrations. The proposal is based on a set of concepts of flexibility that enable partial process specifications, depicting thus a wider scope of business scenarios that a company is willing to conform in a B2B context. The complete process specification is made after B2B parties agree on a strict public protocol and enforced by runtime transformations. The proposed framework for flexible process modeling is aimed to speed up integration of partner processes as it increases ability for process matching without requiring changes in their design.

1 Introduction

The rapidly growing interest in e-business has emphasized the need for technologies able to support automated integration among enterprises. Except for the simplest forms of integrations, Business-To-Business (B2B) scenarios require for business logic to be executed during interactions. The business logic is expressed in form of processes. To be able to collaborate, enterprise processes must expose compatible business behavior.

It is a common need for companies to collaborate business with many parties. However, in many scenarios, involved parties cannot directly integrate even when they agree on information to be exchanged, as their processes are completely predefined and cannot adjust to a new protocol. As an example, it means that a company, modeling a strictly predefined order and selection of a group of activities, cannot directly

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 431–43, 2004.
© Springer-Verlag Berlin Heidelberg 2004

integrate with enterprises having different orders and selections, even the company is willing to conform to those orders. It also means that an enterprise sending and receiving information disposed in a set of messages and activities, cannot match with another enterprise that exchanges same information in a dissimilar aggregate of messages or/and activities. These examples illustrate that too prescriptive process specifications force enterprises to redefine them almost every time a new B2B agreement is made. As the result, companies must engage considerable resources for designing and verifying new process specifications, as well as for maintaining them. Moreover, there is no possibility to have a unique process view over a particular business context.

A solution to the problem is to model enterprise processes in a flexible manner to support a larger scope of B2B protocols that companies may conform to, without necessity for model redesign.

The need for flexibility in workflows is well recognized in the research community ([1], [2], [3] [4], [5]). Our work differs in two aspects. First, we discuss the concept of flexibility related to B2B integrations, i.e. the flexibilities in structures of modeled enterprise processes, needed for relaxation of matching requirements. Second, in our study, we do not concentrate only on flexibilities in composition of activities, but we evenly consider other aspects of process definitions. Our approach for modeling partial process specifications with addition of flexible elements is similar to the proposal given in [5]; the works differ, however, in structuring notion and scope of the proposed flexible elements.

There are several significant studies on relaxation for equivalence requirements between enterprise processes and public protocols ([6], [7]) through the concept of process inheritance. Our study coincides with them in the way we suggest more generic modeling of enterprise processes, for obtaining a larger scale of B2B matches.

The contribution of the paper lies in a comprehensive approach to flexible modeling of enterprise processes, to enable broad B2B integrations without requirements for process redesign. Process flexibility may be achieved in different ways. It may be modeled in the process specification, through the common control-flow constructs. For a company having many B2B partners, this could result in highly complex models with many alternative paths at all flexible points. Another way to achieve flexibility is to specify a separate process template for each particular B2B partner (i.e. contract). However, for a company with high number of template processes, it would become almost impossible to choose a right template for a new contract, as well as to reflect occurring business changes in all of template specifications. In our approach, a process is partially specified with both predefined and flexible elements, while the full specification is defined at contract time. Thus, a single, generic process model is obtained, encompassing whole range of B2B scenarios in a particular business context.

The rest of the paper is structured as follows. In Section 2, we define a framework that we utilize to classify different aspects of flexibilities in B2B processes. In Section 3 we describe our approach for flexible modeling of enterprise processes. In Section 4 we discuss implementation aspects of our approach. In Section 5, we conclude the paper with the discussion of the presented results.

2 Modeling Aspects of B2B Processes

In this section we introduce a conceptual framework, to classify different aspects that constitutes enterprise process specifications. The framework is based on modelling aspects of workflow processes, as proposed by Jablonski in [8] and Rausch-Scott in [9]. In our study, we concentrate on B2B integrations, i.e. on external (public) behaviour of enterprise processes. We, therefore, describe the modelling aspects identified by Jablonski and Rausch-Scott, as following:

- The *functional* aspect defines intention of an interaction, i.e. its goal. An interaction is a basic exchange of messages between partners, having a certain affect. The functionality of an interaction is determined by names and types (send/receive *SR,* receive/send *RS,* solicit send *S,* and solicit receive *R*) of activities of collaborating processes.
- The *behavioral* aspect depicts the flow protocol of message exchanges between B2B processes. This is determined by specifying the control flow of a process, i.e. when an activity is to be executed in relation to others.
- The *informational* aspect refers to the messages exchanged during B2B interactions. Messages are defined in partners' process activities, specifying documents to be included.
- The *transactional* aspect concerns consistent execution and recovery of a set of interactions. The design of the transactional aspect includes defining what to be considered as consistent states, depicting thus particular transactional boundaries (scopes) of collaborating processes. In case of errors in interactions caused by failures in process activities, the transactional mechanisms are used to perform adequate recovery procedures by compensation of already completed activities [10].

By the models of Jablonski and Rausch-Scott, the organizational aspect addresses the roles of businesses involved in a particular business scenario. Thus, this aspect is not a subject of change (i.e. flexibility), because in our study we consider integration of partners with already recognized business roles.

In the next section, we introduce our approach for modeling flexible B2B processes, by defining forms of flexibilities for each of described process aspects.

3 Flexible Modeling of B2B Processes

In this section we describe our approach for flexible modeling of enterprise processes. We first define flexible B2B processes. Then, we define notion and the structure of the flexible element, used to depict choices in process specifications. Based on the conceptual framework described in the previous section, we then define the classes of flexible elements.

3.1 Flexible B2B Processes

Flexibility is one of the most revenant issues in the context of process management. In this paper we consider flexibilities of the process structure. In addition, by following the classification given in [2], we concentrate on *flexibility by selection*. It means design of different choices in a process specification, such that a number of alternative execution paths may derive from.

Based on the described concept, we define flexibility in the B2B context as the ability for flexible modeling of the parts of a private process, to allow for alternatives in constitution of public protocols (i.e. public processes). The flexible model constitutes a partial process specification, comprising two parts:

- The predefined part, consisting of the common *process elements* - control-flow constructs, activities, messages (documents) and transactions.
- The flexible part, constituted of flexible process fragments containing sets of process elements and depicting alternatives in full specification of those sets.

The complete specification is made at the time of contracting, i.e. when the public process is created.

In the works related to flexible process modeling, flexibility by selection is associated with the number of *instance types*[1] that can be generated from a process model. In our study context, flexibility by selection indicates the number of public process specifications that an enterprise process may conform to.

3.2 Modeling Processes with Flexible Elements

Current process-based languages (such as BPEL4WS[11], YAWL[12], BPML[13]) support a certain degree of flexibility, with the common control-flow constructs such as sequence, iteration, AND/OR splits and joins ([3]). Creating more reach forms of flexibility using those constructs, leads to design of all alternative paths at all relevant points. This may drastically increase complexity of a process specification and, accordingly, decrease its readability. Another problem is that process-based languages cannot model some forms of flexibility, because they are related to implementations of activities, provided as services of background systems.

To enlarge extent of flexibility in process specifications, while keeping the model readability, we use *flexible elements*. A flexible element consists of one or more process elements and represents alternatives in the public part of the process specification that can be selected depending on the requirements at contract time. In the process model a flexible element is depicted in the form as shown in Figure 1. As it may be seen in the figure, the flexible element construct consists of three parts:

- *Type,* which labels the form of applied flexibility (each of the four process aspects defined in Section 2 supports various forms of flexibility, as it will be seen later).
- *Elements,* denoting one or more process elements belonging to a flexible element.

[1] An instance type is a set of process instances that follow the same execution path [5].

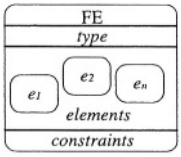


Fig. 1. Flexible element construct

– *Constraints,* which is a logical expression that describes what to include or exclude from a flexible element type. The expression consists of a number of *expression elements* connected with AND, OR, or NOT. An expression element may represent a process element (i.e. an activity, a document or a transaction) or a set of process elements (sequenced activities, for example) belonging to a flexible element.

The concept of flexible elements is thus used to model the fragments of a B2B process specification that are not completely defined, due to allowed alternatives in business collaboration protocol and logic.

For illustration of defined flexible elements we will consider business of a company (such as Sandvik[2], for example) that supplies its products to many worldwide partners. The company collaborates business by integrating its business process with a partner process and following agreed business protocol and logic. As it will be seen, requirements for protocol and logics of exchanged business documents may vary significantly. We will show how abilities for matching between partner processes considerably increase, when the processes are flexible modeled using the approach we suggest.

Behavioral Aspect

Following the definition in Section 2, the behavioural aspect depicts when an activity is to be executed in relation to others. This means that in B2B interactions, there could be differences in requirements for selection of activities to be executed, as well as in ordering of activities. To enable modelling of such differences, we identify two types of flexible elements – *selection* and *ordering* (with the subtype *sequencing*).

Selection. This flexible element depicts that activities from a process segment may be considered in B2B integration, or omitted. The *elements* part consists of one or more activities, which are optional. The *constraint* part is used to depict exceptions, if such exists. For instance, if activities a_1, a_2 and a_3 are optional in the way that any one is optional, while two others must be in the specification, then the constraint is defined as (a_1, a_2) OR (a_1, a_3) OR (a_2, a_3); if, however, the activities would be optional in the way that at least one should be executed, then the constraint would be NOT().

In the described product supply example, before sending an invoice, the company may send the price list (activity *a*), information on delivery terms (*b*), as well as a product offer (*c*). For the company, all three activities are optional, without restrictions. For partners, however, it could be different: some partners, for example, require to get all three documents; most of international partners ask for delivery terms; for some, a product offer suffices, while old partners often do not require any of the

[2] Sandvik is a global industrial materials engineering company with offices in 130 countries.

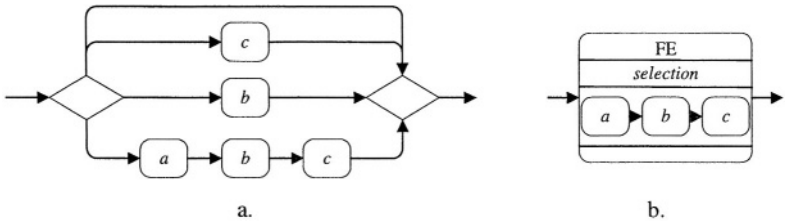


Fig. 2. Selective execution of a group of activities that exchange pre-sales documents a. BPMN model b. Model with the flexible element.

documents. Figure 2 depicts the differences in modeling the flexibility with BPMN (a promising notion for graphical modeling of business processes [14]) and the flexible element:

Ordering. This flexible element allows for execution of activities in arbitrary order, sequential and/or parallel. The element is used to model alternative orders for a segment of the public part of a B2B process. The *elements* part includes two or more activities. The *constraint* part depicts exceptions from the rule, if such exist. As an example, if activities a_1, a_2 and a_3 may be arbitrary ordered, except the activity a_3 may not go in parallel with a_2, and the activity a_2 cannot precede the activity a_1, then the constraint would be defined as $\mathrm{NOT}(a_3 \| a_2)$ AND $\mathrm{NOT}(a_2\text{-}..a_1)$), where the symbol $\|$ is used to denote the parallel order, the symbol "-" is used to denote the sequence order and the symbol "-.." is used to denote the preceding ordering.

In the described B2B scenario, depending on a partner, the company may allow for activities "receive payment" (activity *a*) and "send delivery information" (activity *b*) to execute in diverse orders. For instance, for an old partner, the company allows delivery before payment; for some other partners, those activities may execute concurrently, while for others payment must be done before products are delivered. In Figure 3, we depict the differences in modeling the "ordering" flexibility with BPMN and with the flexible element:

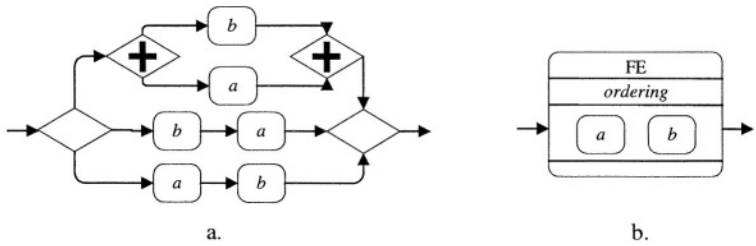


Fig. 3. Free ordering of payment and delivery information activities a. BPMN model b. Model with the flexible element.

Ordering/Sequencing. This element is the special case of the flexible element ordering. It is used to model allowed alternatives in sequential order of a set of activities. As an example, if activities a_1, a_2, a_3 and a_4 may be sequenced in arbitrary way, with

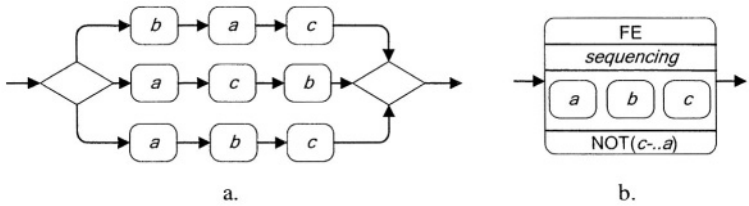


Fig. 4. Flexible sequencing of product testing activities a. BPMN model; b. Model with the flexible element.

the exception of the sequence a_4, a_3, a_2, a_1, then the constraint is defined as NOT(a_4-a_3-a_2-a_1); as another examples, if the activities may be arbitrary sequenced except that the activity a_4 cannot be executed before the activities a_1 and a_2, then, the constraint is specified as NOT(a_4-..a_1, a_2), In the study example, we can consider situation when the company must send information on results on three kinds of product testing (activities *a, b* and *c*) to its partners. As for different partners importance of performed tests may vary, the company may allow for flexibility in tests ordering, except the test *c* cannot be performed before the test *a.* In Figure 4, we illustrate the differences in modeling this flexibility, with BPMN and with the free sequencing element:

Functional Aspects

As described in Section 2, the functionality of B2B interactions is determined by names and types of activities of collaborating processes. In the concept of matching two processes, an activity in one process must be compatible with a corresponding activity in the other process. As an example, this means that if an enterprise sends a purchase order and receives the confirmation as one single activity, it cannot match with another enterprise that receives the order and sends the confirmation as two separate activities. In [15] the authors define a process compatibility concept that allows for differences in activity structures (i.e. types), as long as processes match semantically, i.e. in the communicated information and in the control flow of message exchanges. Based on that concept, to improve capability for functional matching between B2B processes, we define the following flexible element:

(De)Flattening. The flexible element is used to model flexibility in interaction patterns. The *elements* part includes either one activity (of type *SR* or *RS*) or two subsequent activities (of types *S* and *R*). The *constraint* part is not used. As an example, if an activity *a* is of type *SR,* it may be presented as two subsequent activities $a_1(S)$ and $a_2(R)$; as another example, two subsequent activities $a_1(R)$ and $a_2(S)$ may be presented as a single activity *a* of type *RS.* It does not mean, however, that these transformations may be applied for any activity in a process specification. Due to limitations of information systems it could be, for example, impossible to represent a *SR* activity as two separate activities (of *S* and *R* type), because a system is not capable of doing other operations before completing this activity. In the study example, the company sends delivery notification (*a*) and receive acknowledge (*b*) as two separate activities; however, it allows matching with partners that exchange these information in a single activity. Figure 5 depicts the corresponding flexible element:

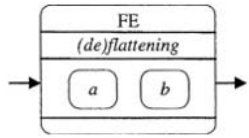


Fig. 5. Flexible element depicting composition of subsequent activities *a* (type *S*) and *b* (type *R*) into a single activity (type *SR*).

This flexibility cannot be presented by a BPMN model, because it is related to the implementations of the activities *a* and *b,* which are part of underlying applications.

Informational Aspects

The informational aspect refers to the documents included in messages exchanged during B2B interactions. A problem in B2B integrations is that business partners usually expose different requirements regarding exchanged documents and their structures. Thus, for efficient integrations, it would be beneficial for a company to be able to specify allowed alternatives in the exchanged information content and structure. Following this, we define two type of flexible elements – *omitting* and *(de)composition*.

Omitting. This flexible element allows for omitting input or output message (i.e. document) from a *RS* or *SR* activity. The *elements* part contains the document that is optional; the *constraints* part is not used. In the studied example, the company sends offer and receive acknowledge of the reception as a single activity *a.* However, the company is willing, even, to participate in collaborations with partners that do not send the confirmation of order reception. Figure 6 illustrates the corresponding flexible element:

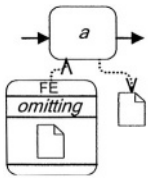


Fig. 6. Optional receive of the "confirmation of order" document.

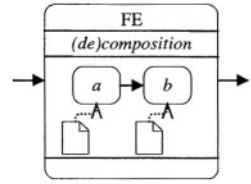


Fig. 7. Composition of documents "order rejection" and "counter offer" from activities *a* and *b* into a single document.

(De)Composition. The flexible element allows for restructuring of information provided by messages (documents) within activities. If a document is to be decomposed to a number of smaller documents, the *elements* part contains that document; the *constraints* part depicts allowed decompositions. As an example, if a document *d* may be decomposed to documents d_1 and d_2, then the constraint is (d_1, d_2). At contrast, if documents from several subsequent activities are to be sent (or received) integrally, then the *elements* part depicts those activities; the *constraints* part is not used. In the example case, if the company, after sending a product offer, receives the rejection and

a counter offer as two messages (i.e. as two receive activities *a* and *b*), it may comply with partners that send those documents within a single activity. Figure 7 depicts the corresponding flexible element.

Described flexibilities cannot be presented by alternative paths in BPMN models, because they are related to differences in implementation of selected activities.

Transactional Aspects

In B2B domain, support for transactional message exchange is important to enable consistent recovery of process states in exceptional circumstances. This means that partners have to expose similar scopes for corresponding transactions, as well as compatible recovery behavior. Companies aiming to enlarge number of B2B partners are often willing to conform to different transactional requirements, but underlying process models are too prescriptive to support that. To enhance flexibility in transactional integration of B2B processes we define, accordingly, two types of flexible elements – *scope* and *compensation.*

Scope. This flexible element allows for participation of different activities within a transaction. The element is intended to model alternative transaction scopes. The *elements* part is a single transaction; *constraint* part depicts activities that may be excluded from the transaction. For instance, if a transaction *T* encompasses activities $a_1, a_2, .., a_n$ whereas activities a_1 and a_2 may be excepted from it, then the constraint is defined as (a_1, a_2).

In the studied case, the company encompasses activities "receive order" (*a*), "send invoice" (*b*), "receive payment" (*c*) and "send delivery information" (*d*) within a transaction, to facilitate correct payment cancellation. However, the company may exclude the first activity (*a*) from the transaction if some collaboration requires so (i.e. when partner does not send order cancellation). In Figure 8, we illustrate the differences in modeling the "scope" flexibility with BPMN and with the flexible element:

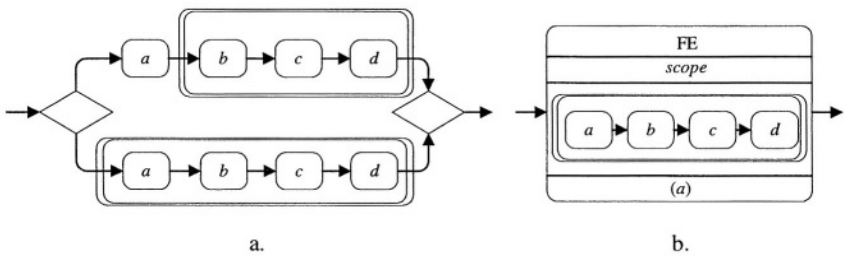


Fig. 8. Flexible scope of product ordering transaction a. BPMN model b. Model with the flexible element.

Compensation. The flexible element allows for optional compensation of an activity engaged in a B2B interaction. The *elements* part contains one or more activities (as an activity may be compensated by a set of activities); *constraint* part is not used.

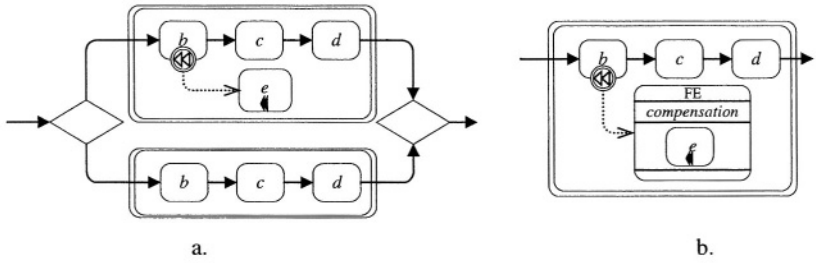


Fig. 9. Flexible compensation of "send invoice" activity a. BPMN model b. Model with the flexible element.

In the described example, if after payment (activity *c*) the transaction is aborted (by a partner initiative, for example), then the company returns payment, but does not require to receive an "official" invoice cancellation document. Figure 9 illustrates the differences in modeling the flexibility with BPMN and with the flexible element.

The above discussion reveals that, in our approach, the flexible element is a fundamental object for modeling alternatives in execution of B2B processes. By following the four main aspects of process modeling (Section 2), we identified a class of flexible forms. We modeled all forms with the proposed generic structure of the flexible element. The structure is fairly simple and compact; even more important, it is very simple for change (i.e. when new constraints are to be set, or new elements, etc.). In addition, the structure is open for extensions. First, new types of flexibilities may be introduced. Second, the *constraints* part logic could be further extended to describe new dimensions of rules; for instance, the elements depicting optional compensation and possible omit of documents could involve time restrictions. Finally, it is important to appoint that different forms of flexibilities may be nested within each other.

In the next section we discuss abilities for implementation of our approach, as well as integration of flexible modeled enterprise processes to complete specifications according to public protocol requirements.

4 Implementation Considerations

The flexible element construct is integrated in a process specification as a separate process fragment. In the previous section, we presented the flexible element object using a graphical notion. Even we argued for readability of process models, it is clear that it decreases as the number of flexible elements grows. This is directly related to the fact that, in general, process models have to depict various logics, such as composition, choices, transactions, events, errors, compensations, etc. One solution to this problem is to use layers of process view, as suggested in [16].

The graphical notion of the flexible element, as we defined it, could be easily translated to the form of a scripted tag with the corresponding structure (Figure 10 depicts a BPEL4WS segment enwrapped with the flexible element "sequencing").

```
<FE type="sequencing" constraints="">
  <receive partner="customer"
     portType="paymentPT"
     operation="recPayment"
     container="payment"
  </receive>
  <invoke partner="customer"
     portType="deliveryPT"
     operation="sendDelivery"
     inputContainer="delivery"
  </invoke>
</FE>
```

Fig. 10. A partial BPEL4WS code segment enwrapped with the flexible element.

```
<sequence>
  <receive partner="customer"
     portType="paymentPT"
     operation="recPayment"
     container="payment"
  </receive>
  <invoke partner="customer"
     portType="deliveryPT"
     operation="sendDelivery"
     inputContainer="delivery"
  </invoke>
</sequence>
```

Fig. 11. Code segment from Figure 10, converted to the full specification.

The main aspect is integration of flexible elements with process-based languages, to enable functionality of our approach. We realize the integration through a model we name *process integrator* (due to limited space, we cannot present the model details).

Process integrator is used to build the complete private process specification (integration phase) at contracting time according to a given public protocol. If processes match, the integrator, if necessary, performs some transformations during runtime (collaboration phase). Imported process definitions (for instance, a partial BPEL4WS specification of private process extended with flexible elements, and a public protocol) are first matched for equivalence [17]. Every time a flexible element is encountered, the full specification is built according to requirements in the public protocol. This means, for the class of flexibilities not altering implementation of activities, the flexible element tag is removed from the BPEL4WS specification and the included activities are fully structured by adding the adequate language semantics according to the public process requirement (Figure 11). The other class of flexible elements is, upon matching, simply removed from the process specification and included activities are indexed and recorded for necessary runtime transformations (as described in [15]).

5 Conclusion

In this paper we have addressed the problem of lack of broad process-based B2B integrations at the present time. Within the scope of the SERVIAM project [18], we have investigated reasons for absence of broader implementation of B2B integrations in Swedish companies. We concluded that the main reasons of the problem lay in inability of enterprise processes to support a large enough number of public protocols; thereafter, companies design new process models and implementation services for each business partner. We have, therefore, argued that an enterprise (private) process should be defined in a flexible and generic way, to depict the whole scale of B2B scenarios that a company is willing to conform to. Following this motivation, we proposed a set of concepts for partial modelling of the main structural aspects of enterprise processes, where the complete specification is obtained at contracting time, after matching with a public protocol.

This work has been conducted in several steps. We have first defined the four main aspects of process structures and then defined the related forms of flexibility. The concept of the flexible element in the way we defined it, represents a separate layer in the process specification. It frees the specification of design of alternative paths and, in addition, provides for expressing of reach forms of flexibilities. At contracting time, a class of flexible elements is replaced with the fully defined process segments; the other class (i.e. those related to implementation of activities) are also replaced, whereas the agreed execution structures are registered in the process integrator to enact necessary transformations at runtime.

The presented result can be applied in several ways. Firstly, the approach is not tight to a particular integration scenario (static or dynamic) as the basic idea is to enable a generic representation of a collection of model templates that altogether depict a single business. Secondly, the approach enables integration of a single process model with a variety of predefined or agreed B2B protocols.

In the future work within the project, it is our intention to develop a prototype of the process integrator and test the efficiency of its functions.

References

1. Ellis, C., Rozenberg, G.: Dynamic Change Within Workflow Systems. In: Conference on Organizational Computing Systems, Milpitas (1995)
2. Heinl, P., Horn, S., Jablonski, S., Neeb, J.: A Comprehensive Approach to Flexibility in Workflow Management Systems. In: Proceedings of International Joint Conference on Work Activities Coordination and Collaboration (WACC'99), San Francisco (1999)
3. Aalst, W. M. P. van der, Barros A.P., Hofstede A.H.M., Kiepuszewski, B.: Advanced Workflow Patterns. In: O. Proceedings of the Seventh IFCIS International Conference on Cooperative Information Systems (CoopIS 2000), Eilat, Israel. Lecture Notes in Computer Science, Vol. 1901. Springer-Verlag, Berlin (2000) 18-29
4. Klingemann, J.: Controlled Flexibility in Workflow Management. In: Proceedings of the 12th International Conference on Advanced Information Systems Engineering (CAiSE'00), Stockholm, Sweden. Lecture Notes in Computer Science, Vol. 1789. Springer-Verlag, Berlin (2000) 126–141
5. Sadiq, Sh., Sadiq, W., Orlowska, M.: Pockets of Flexibility in Workflow Specifications. In: Proceedings of the 20th International Conference on Conceptual Modelling (ER2001). Yokohama, Japan. November (2001)
6. Bussler, C.: Public Process Inheritance for Business-to-Business Integration. In: Proceedings of Workshop on Technologies for E-Services (TES 2002). Lecture Notes in Computer Science, Vol. 2444. Springer-Verlag, Berlin (2002) 19-28
7. Aalst, W. M. P. van der, Weske, M.: The P2P Approach to Interorganizational Workflows. In: Proceedings of the 13th International Conf. on Advanced Information Systems Engineering (CAiSE'01). Lecture Notes in CS, Vol. 2068. Springer-Verlag (2001) 140-156
8. Jablonski, S.: A Software Architecture for Workflow Management Systems. In: Proceedings of the Ninth International Workshop on Database and Expert Systems Applications (DEXA'98). Vienna, Austria, August 1998. IEEE Computer Society, (1998) 739-744.
9. Rausch-Scott, S.: TriGSflow – Workflow Management Based on Active Object-Oriented Database Systems and Transaction Mechanisms. PhD Thesis, University at Linz (1997)

10. Garcia-Molina, H.: Modeling Long-Running Activities as Nested Sagas. In: IEEE Data Engineering Bulletin, Vol. 14/1 (1991) 14–18
11. BEA, IBM, Microsoft, SAP and Siebel. Business Process Execution Language for Web Services (BPEL). http://www-106.ibm.com/developerworks/library/ws-bpel/, June (2004)
12. Aalst van der, W., Hofstede, A., Aldred, L. Yet Another Workflow Language (YAWL). http://www.citi.qut.edu.au/yawl/index.jsp, June (2004)
13. Arkin, A. Business Process Modelling Language Version 1.0, The Business Management Initiative, November (2002)
14. White, S. Business Process Modelling Notation Version 1.0, The Business Management Initiative, May (2004)
15. Zdravkovic J., Johannesson, P.: Cooperation of Processes through Message Level Agreement. In: Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAiSE'04), Riga, Latvia. Lecture Notes in Computer Science, Vol. 3084. Springer-Verlag, Berlin (2004) 564–579
16. Johannesson P., Perjons, E.: Design Principles for Process Modelling in Enterprise Application Integration. Information Systems Vol. 26 (2001) 165-184.
17. Glabbeek, R.J. van, Weijland, W.P.: Branching Time and Abstraction in Bisimulation Semantics (extended abstract). In: 11th Proceedings of the IFIP World Computer Congress. G.X. Ritter. Elsevier Science Publishers (1989) 613-618.
18. SERVIAM project, http://www.serviam.se, July 2004.

Using Patterns in the Design of Inter-organizational Systems – An Experience Report

John Novatnack[1] and Jana Koehler[2]

[1] Drexel University, Computer Science Department, Philadelphia, PA 19104, USA
jmn27@cs.drexel.edu
[2] IBM Zurich Research Laboratory, CH-8803 Rueschlikon, Switzerland
koe@zurich.ibm.com

Abstract. The modeling, design, and implementation of inter-organizational systems (IOS) is a challenging new problem. In contrast to previous systems, where components have clearly defined interfaces and serve a well-defined purpose, components of IOS exist in a distributed environment, where each component of the system may exist at a separate corporation and must conform to the interface of partner components. This creates a complicated problem for designers. In this paper, we investigate to which extent control flow, communication, and data manipulation patterns can help to ease the manual design process of IOS implemented in the Business Process Execution Language for Web services (BPEL4WS). By applying patterns of all three types in an iterative fashion we go from an abstract representation of the example process to a graphical solution with a one-to-one mapping to executable BPEL4WS code.

1 Introduction

The vision of Service-oriented architectures changes the way businesses are implemented and how they interact with each other. Business processes are no longer monolithic systems, but instead are assembled from existing service components. Companies interacting in a Business-to-Business (B2B) relationship as well as customers interacting with a company in a Business-to-Customer (B2C) relationship are no longer dealing with a single isolated corporation, they knowingly or unknowingly, interact with a composition of participating companies that work together to meet the customer's demands. The companies (and their business processes) participating in this composition form inter-organizational systems (IOS), consisting of independent and distributed services that come together to form a cohesive system.

The modeling, design, and implementation of these systems is a challenging problem. In contrast to previous IT implementations that had clearly denned interfaces and served a well-defined purpose, three trends of IOS can be observed: (1) Business requirements are much tighter linked to IT systems design, (2) IT systems are split into functional components (services) that have to serve very different purposes, and (3) the IT system boundaries break up - a system can no longer exist in isolation, but depends on various services which are offered by other partners.

These trends lead to a number of unresolved research problems [1]. One paramount problem is developing tools and techniques to design and implement these distributed

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 444–455, 2004.
© Springer-Verlag Berlin Heidelberg 2004

systems. Established or upcoming standards such as the *Web Service Description Language* (WSDL) [2], which defines the public interface of a Web service, and the *Business Process Execution Language for Web services* (BPEL4WS) [3], which describes the operational details of Web service orchestration, provide a unique technological basis for the design of IOS. Knowledge of these languages alone does not equip a designer with the ability to design and implement IOS. Designers must make a conceptual leap from being fully authoritative over a system design, to designing a single synergic component in a collective system. Currently, little development support is available that aids the designer in this task. MDA techniques can be applied to UML sequence diagrams or activity diagrams to automatically generated executable BPEL4WS [4,5]. Semantic Web approaches, e. g. [6], focus on the dynamic binding of Web services based on semantic annotations, which describe what a service does, under which circumstances it can be used, etc. The automatic composition of Web services is accomplished using intelligent planning techniques to achieve predefined goals. The approach requires to annotate individual BPEL4WS activities with pre- and postconditions in some logical language.

In this paper, we investigate to which extent *patterns* can aid in the *manual* design of IOS. More specifically, we investigate the role patterns play in the design of a Service-oriented Architecture based on WSDL and BPEL4WS and how patterns can be used to ease the development of the Web Service orchestration in BPEL4WS. While [4] focuses on *automatic,* one-to-one transformations from UML sequence diagrams to BPEL4WS code, but ignore details of data manipulation and communication amongst partners, we focus on *manual* transformations of UML sequence diagrams to BPEL4WS, where developers iteratively apply patterns to their design, all the while supplying the necessary data manipulation and communication details. As it has been observed in [6], "it is the BPEL4WS author's responsibility to manually construct a process model that follows the operational semantics of its service partners." We investigate how iteratively applying patterns to a manual Web service design helps an author to assume this responsibility, because the manual development of BPEL4WS or related process models will be the dominant software engineering approach in the industry in the foreseeable future.

The paper is organized as follows. In Section 2, we briefly review the origins and properties of patterns and the state of the art in using patterns in implementing business processes. In Section 3, we introduce a specific development scenario and capture it with UML models. We start with formulating use cases that are subsequently refined in a UML sequence diagram showing the interactions between the partners of IOS. In Section 4 we begin transforming the UML sequence diagram into executable BPEL4WS code by injecting control flow, communication and data manipulation patterns. In Section 5 we draw conclusions and sketch some directions future work can take.

2 The Role of Patterns

The Merriam-Webster Online Dictionary defines *Patterns* as "a form or model proposed for imitation". For the IT community, patterns have developed into "a software-engineering problem-solving discipline" which focuses on a culture to document and support sound design [7]. In general, there is wide agreement that defining what a pattern constitutes is hard. Instead, [7] declares a number of criteria shared by *good patterns.*

Following these criteria, a good pattern: (1) solves a problem, (2) is a proven concept, (3) proposes a nontrivial solution and (4) describes deep system structure and relationships.

The most popular patterns in software development originate from the Object-Oriented Community [8]. They were later extended to patterns that address general issues of software architecture [9] as well as concurrent and networked objects [10]. At the same time, patterns of control-flow were extracted from workflow systems [11].

2.1 Patterns for IOS

When designing IOS based on a choreography of Web services, three areas where patterns may be applicable can be identified:

1. *Control flow* between the various Web services,
2. *Communication* between a number of partner Web services, and
3. *Data manipulation* that is implemented in the choreography.

The workflow patterns defined by Aalst et al. [11] have been extracted from the control flow constructs provided by workflow systems. The patterns have been mainly used to classify and compare existing workflow products based on the control flow constructs that these products provide, and have recently been extended to compare proposed Web service standards [12]. There is, however, no work done up to this point that examines the impact and the applicability of the workflow patterns in the design and implementation of IOS. A set of communication patterns have been defined that may have relevance to the Web service domain [13]. The communication patterns are separated into two broad categories, patterns dealing with (1) synchronous and (2) asynchronous communication. A set of patterns for enterprise integration is given in [14]. In our study, we focus on the more simpler set of communication patterns of [13] as we are especially interested in differentiating between the synchronous and asynchronous communication as it occurs in BPEL4WS, and aiding in applying each. Recently, work has begun in the workflow domain to develop a set of data patterns [15].

3 A Case Study of Manual Incremental Pattern-Based Design

In our case study we bring three different trends together: Service-Oriented Architectures based on WSDL and BPEL4WS, Model-driven Architecture [16], and Patterns. Models play an increasing role in the design of software. In the following, we create models, expressed as UML use case and sequence diagrams, to capture the design of a simplified Web-based travel agency process. We use models to capture the major requirements and then inject patterns into our models in order to move towards executable BPEL4WS code. Thus in this approach, patterns are considered as guides that facilitate the refinement of models by an IT expert. We believe that some automatic tooling support may be available in the future to partially automate the pattern injection process (similar to applying OO Design Patterns in Rational XDE for example), but this issue is beyond the scope of this paper.

We study the applicability of patterns in the context of a specific example, namely a travel agency. The travel agency receives requests from customers over the phone. The

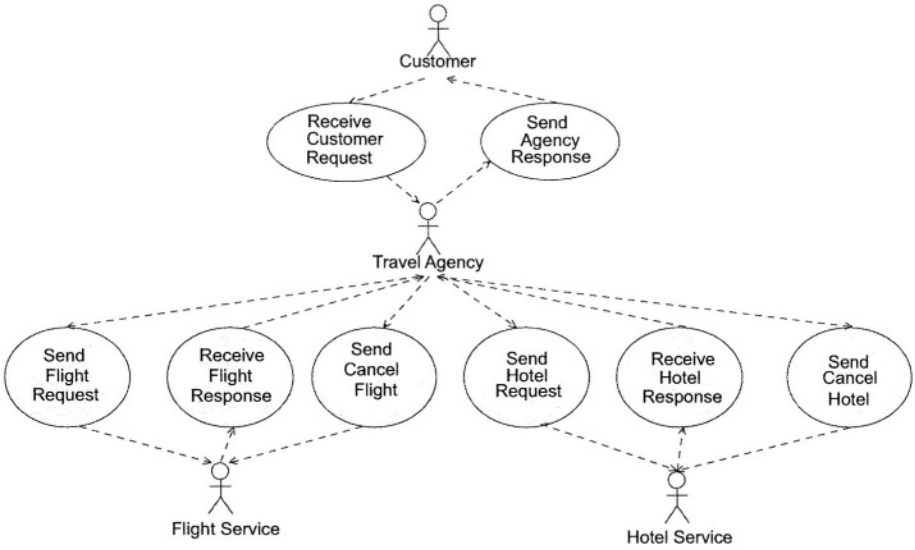


Fig. 1. Behavioral view of the travel agency process as a use case diagram.

requests are recorded in trip documents, from which several flight and hotel requests are derived by the agency. The flight and hotel requests are handled by independent Web services that exist outside and receive requests from the travel agency. The services report their success or failure in booking the requests back to the agency. If all requests were successfully handled, the agency puts together the trip documents for the customer and confirms the travel itinerary. If one of the services fails to make a requested reservation, the agency informs the other service to put all reservation requests on hold and contacts the customer to revise the trip itinerary. Our design goal is to arrive at a BPEL4WS specification that implements the interaction of the travel agency with its partners. In this paper we are not concerned with the internal details of the flight and hotel services, although the techniques described here can also be applied to their design.

As was stated in [17], Web services may be modeled from two views: *collaborative* and *behavioral.* In the collaborative view, IOS are modeled from the perspective of an external observer. The observer views the sequence of messages exchanged between Web service partners in the system. In the behavioral view, individual Web services are independently modeled from an internal perspective that treats partners like a black box. In this paper we model the travel agency from an internal, or behavioral, view. However, if the behavioral models of the customer, flight and hotel services were available, it would be possible to construct a collaborative view of the composite system. Figure 1 captures the main use cases of how the four partners, customer, travel agency, flight service, and hotel service, interact, modeled from the behavioral view of the travel agency.

Use case diagrams provide a high level view of the partner interaction. It is a natural next step to use a UML sequence diagram to further refine the required interaction between the partners, cf. Figure 2. Since our implementation will be based on BPEL4WS

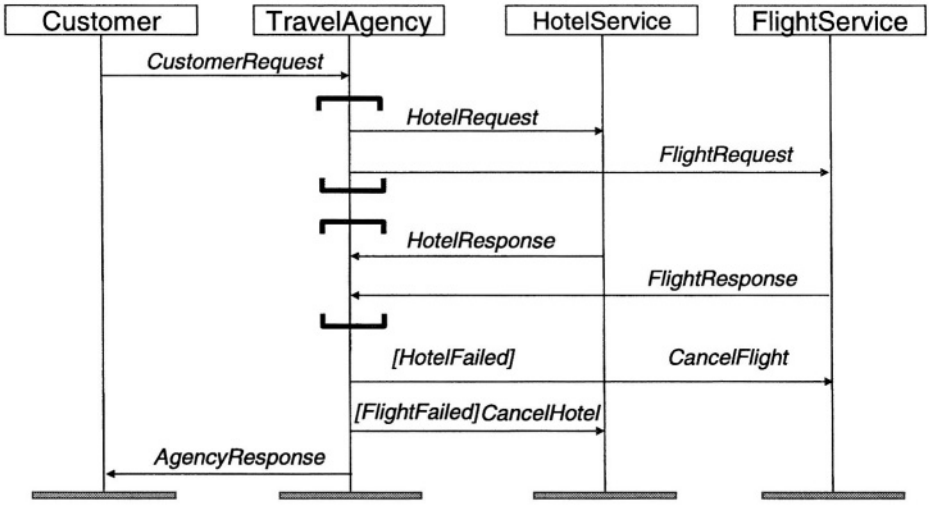


Fig. 2. Sequence diagram for the travel agency process.

and WSDL, the Web service messages which are exchanged in a specific order help in further understanding the design of the travel agency process. Figure 1 uses so called *co-regions* (depicted with bold brackets) from UML2 to specify that some messages can be sent in any order. Conditions in braces are added to the diagram to capture that the CancelFlight and CancelHotel messages will only be sent under certain conditions.

4 Applying Patterns to the Design

Further refinements are made to the sequence diagram by injecting control flow, communication and data manipulation patterns. The injection of these patterns allows the design to approach executable BPEL4WS code.

4.1 Injecting Control Flow Patterns

The set of workflow patterns that we consider are divided into six categories [11]:

1. Basic control
2. Advanced branching
3. Structural patterns
4. Patterns involving multiple instances
5. State based
6. Cancellation

Each category contains patterns that supply certain control flow constructs. In general this includes two types of patterns: patterns that initiate a sequential or concurrent split in the control flow and patterns that merge a number of sequential or concurrent

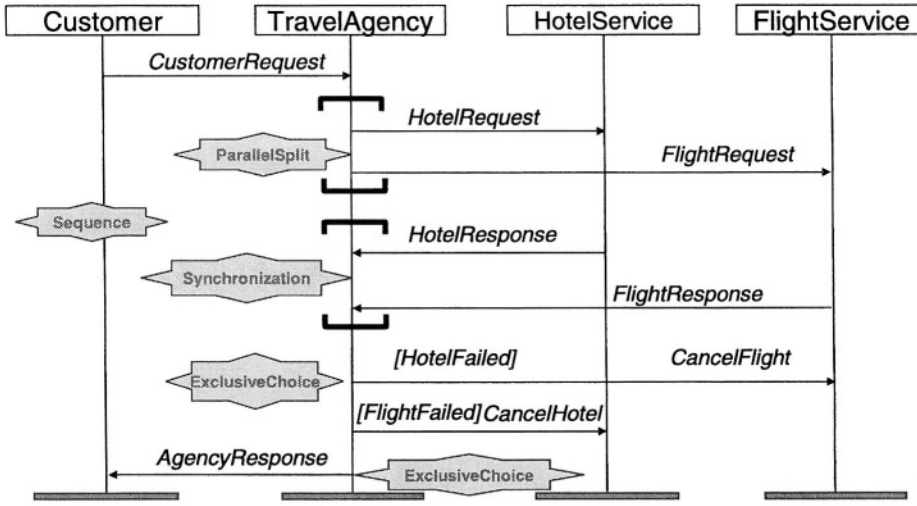


Fig. 3. Placing control flow patterns in the sequence diagram.

flows into a single chain of execution. Although captured in individual patterns in the above collection, it is often the case that the merge patterns are provided implicitly by BPEL4WS, depending on how the split is implemented.

Figure 3 shows places where the control flow patterns can be applied to the design of the travel agency. In designing and implementing the travel agency process, patterns from basically any of the categories can be applied. With extensive knowledge of the patterns and the system to be implemented, a designer can make a choice of the most applicable patterns. Animations from the workflow patterns web site [18] provide an intuitive basis for designers to choose which patterns to apply. However, there does not exist classification system as comprehensive as provided by the OO community, where patterns are listed along with *intent, motivation, applicability, consequences* and *known uses* [8].

We decide to apply several patterns from the set of basic control patterns, namely the *sequence, parallel split, synchronization,* and *exclusive choice.* The *sequence* pattern executes activities in order, where one activity begins after the previous activity has completed. This pattern is appropriate for the interaction between travel agency and customer. The *parallel split* pattern allows for the execution of a number of activities in parallel. It is applied to design the concurrent interaction of the travel agency with its partner services. The *synchronization* pattern is used to merge the parallel split. Following the synchronization pattern, the travel agency needs to make a choice depending on whether the partners report a success or failure. The *exclusive choice* pattern is applied to provide a conditional branching construct.

All four of these patterns have a direct mapping to BPEL4WS code, see also [12], where the sequence pattern is implemented using a `<sequence>` element, parallel split and the synchronization patterns are jointly implemented using a `<flow>` element and exclusive choice is implemented using a `<switch>` element. Alternative mappings for

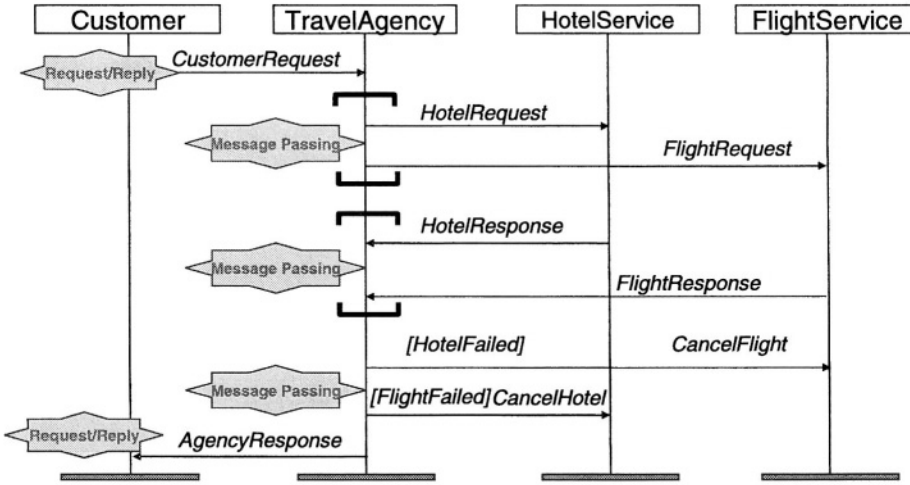


Fig. 4. Placing communication patterns in the sequence diagram.

all patterns can be provided to the designer by using `<flow>` elements with control links that supply transition and join conditions.

4.2 Injecting Communication Patterns

Web service communication can be distinguished as being either *synchronous* or *asynchronous.* In synchronous communication, the sender halts for a response after sending a message, whereas in asynchronous communication, a sender does not implicitly block for a response after sending a message to a partner. BPEL4WS naturally supports both forms through the constructs: `<invoke>`, `<receive>`, `<reply>` and `<pick>`. We consider a set of communication patterns which are organized into synchronous and asynchronous patterns [13]. The synchronous patterns include: (1) Request/Reply, (2) One-Way and (3) Synchronous Polling. The asynchronous patterns include: (1) Message Passing, (2) Publish/Subscribe and (3) Broadcast. Of the synchronous communication patterns we apply *Request/Reply* and of the asynchronous patterns we apply *Message Passing.* In the synchronous Request/Reply pattern a sender exchanges a message with a partner and blocks for a response. In the asynchronous Message Passing pattern a sender exchanges a message with a receiver without expecting a response.

Figure 4 shows the UML sequence diagram of the travel agency with the injected communication patterns. Determining which communication links will be synchronous and which will be asynchronous is up to the designer of the system. A customer which interacts with the travel agency expects that the travel agency will respond to the reservation request. The customer and travel agency engage in a synchronous exchange of message, where the customer halts and waits for the travel agency's response to the reservation request. The travel agency, on the other hand, may have the capability to interact with an arbitrary number of flight and hotel service partners. The responses given by

some of the partners may have long delays, or may never come. Asynchronous communication links between the travel agency and flight and hotel service partners allows for controlled communication, where one lost message will not block control flow within the process. Secondly, when the travel agency sends a CancelFlight or CancelHotel message, it does not expect a response, therefore a synchronous message exchange would cause the travel agency to block infinitely. Asynchronous message passing is *required* here, as there is no return message.

The Request/Reply and Message Passing patterns are both implemented using the BPEL4WS communication constructs: `<invoke>`, `<receive>` and `<reply>`, or `<receive>` and `<pick>`, see also [12]. The synchronous Request/Reply pattern may be implemented by the sender with an `<invoke>` element that specifies an input and output variable, and with a receiver that uses the `<receive>` and `<reply>` elements. The asynchronous Message Passing pattern is implemented by a sender that uses an `<invoke>` statement with only an input variable and a receiver that blocks with a `<receive>` element. If the initial sender requires a response to an asynchronous message, it is necessary for them to block with a `<receive>` element.

4.3 Injecting Data Manipulation Patterns

A good starting point for data manipulation patterns are the patterns originally designed by the OO community [8]. These patterns are very well explored and provide a basis for constructing the data manipulation patterns necessary for Web services. As a starting point, we define the *mediator,* inspired by the OO pattern of the same name. In the OO domain the mediator pattern is "an object that encapsulates how a set of objects interact" [8]. With respect to IOS, the mediator pattern encapsulates how a set of *partner services* interact. Web services exchange complex XML variables that may be instances of WSDL message types or defined by XML schemas. Collaborating Web services are independently designed and cannot be expected to have conforming interfaces. It is up to the Web services participating in the collaboration to manipulate data such that it conforms to the interface definition of each partner. The mediator pattern encapsulates the data manipulation that must be done in order for the data to be sent to partner services in a valid format.

Figure 5 shows an abstract view of the mediator pattern. The pattern is initiated with a complex message containing multiple data components going to independent partners. The mediator pattern splits the message into new messages, each conforming to the interface definition of a partner. Correlation data is added when it is necessary to distinguish between multiple instances of a single partner type. The mediator pattern receives responses from the partner services and aggregates the modified data into a format suitable for the initiator of the pattern.

A relevant portion of the mediator pattern's sample code in BPEL4WS is shown in Figure 6. The BPEL4WS code breaks the pattern into three sections: first areceive activity obtains the data from the triggering partner. Secondly, the data is split into two separate variables with an assign activity. Synchronous invoke statements send the messages and receive responses from the service partners. Finally, the output of the partner services is combined into a single message that is returned to the initially triggering partner. The sample code shows essential details of how messages are bound to variables and how

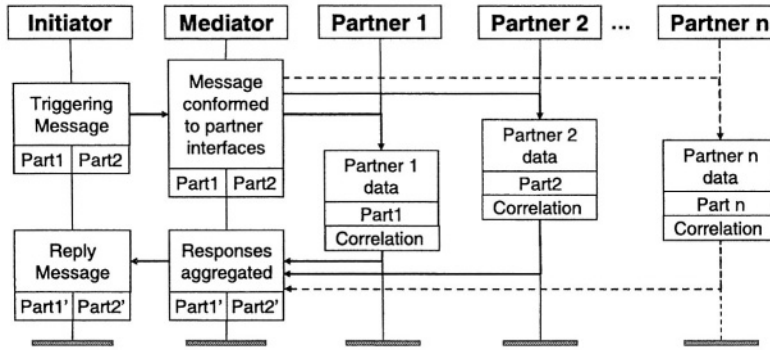


Fig. 5. Abstract view of mediator pattern.

```
<!-- Initial message reception and data breakup -->
<receive variable="triggerMessage" partnerLink="initiator" ... />
<assign>
  <copy>
    <from variable="triggerMessage" part="part1" />
    <to variable="partner1Message" part="part1" />
  </copy>
  <copy>
    <from variable="triggerMessage" part="part2" />
    <to variable="partner2Message" part="part2" />
  </copy>
</assign>

<!-- Communicating with independent partners -->
<invoke inputVariable="partner1Message"
        outputVariable="partner1Output"
        partnerLink="partner1" ... />
<invoke inputVariable="partner2Message"
        outputVariable="partner2Output"
        partnerLink="partner2" ... />

<!-- Aggregating the data and returning a response -->
<assign>
  <copy>
    <from variable="partner1Output" part="part1" />
    <to variable="triggerMessage" part="part1" />
  </copy>
  <copy>
    <from variable="partner2Output" part="part2" />
    <to variable="triggerMessage" part="part2" />
  </copy>
</assign>
<invoke inputVariable="triggerMessage" partnerLink="initiator" ... />
```

Fig. 6. BPEL4WS sample code of mediator pattern.

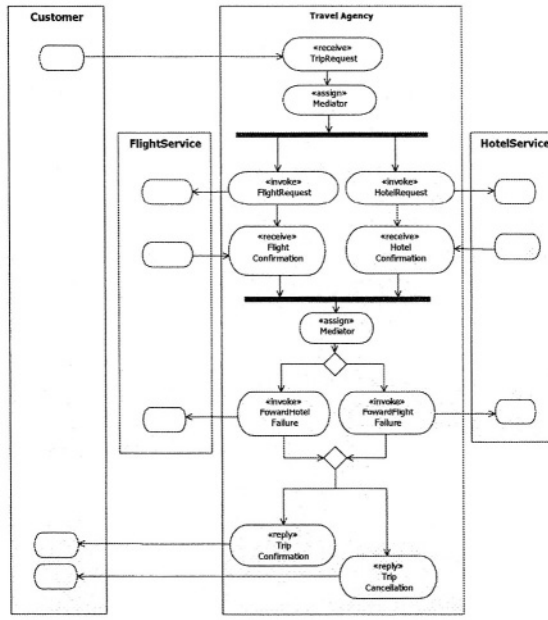


Fig. 7. The abstract BPEL4WS resulting from the pattern application.

variables are copied and assigned to each other. In addition, correlation sets may need to be added to the communication constructs to distinguish between multiple instances of partner services. Similar to the description of the OO patterns, sample code of sufficient detail needs to be provided.

4.4 Resulting BPEL4WS Flow

Figure 7 shows the graphical representation of the resulting BPEL4WS code of the travel agency obtained after successful application of the patterns, based on the UML profile for BPEL4WS [19]. The graphical representation can be seen as *executable,* as it has a one-to-one mapping to BPEL4WS code. The travel agency begins a sequential execution of activities after a `<receive>` element obtains the initial synchronous request from the customer. The travel agency asynchronously invokes the flight and hotel partner services in parallel with a `<flow>` element (depicted as a thick horizontal bar) and implements the mediator pattern to handle the data manipulation required for the interaction, with an `<assign>` element. The synchronization pattern is applied at the end of the parallel execution, and an exclusive choice, implemented with a `<switch>`, is done to determine the results of the reservation request. The travel agency returns a synchronous reply to the customer with a `<reply>` element.

Bringing the three patterns together in a consistent manner has allowed us to transform the initial UML use case diagram into an executable model. Throughout the application of the patterns it is necessary for the designer to supply the details of the variable

access and control flow decisions. Variables in a BPEL4WS process can be instances of a WSDL message, or defined by XML schema. Data manipulation patterns, such as mediator, must be supplied with the structure of these variables in order to access the correct components. In addition, a designer must explicitly define the Boolean propositions that will drive control flow decisions. Both of these tasks can be simplified through the use of design tools that provide an intuitive graphical interface.

Our approach of incremental design combining patterns that address different aspects of an IOS is similar to the approach described in [20]. As Janson et al., we start with the analysis of a business scenario and then refine it by investigating the collaboration and interaction between the partners in more detail. In contrast to Janson et al. the patterns we consider, provide control flow, communication, and data manipulation solutions, which are independent of a specific implementation, while their patterns focus on available ready-to-use software components, e. g. web client systems, adapters, workflow engines that are plugged together.

5 Conclusion and Outlook

In this paper we have combined techniques of Model Driven Architecture and reusable design techniques to transform an abstract view of a Web service to a graphical representation with a direct mapping to executable code. This was accomplished by applying *patterns* from three important aspects of Web service design: control flow, communication and data manipulation. Prior work has enabled a starting point for creating a set of control flow and communication patterns appropriate for Web service design. Data manipulation patterns still need to be extensively explored. The application of patterns to the design of a Web service is only beneficial when the patterns are applied in a way that is consistent in all three areas. For example, the application of a synchronous communication pattern will affect how the mediator pattern communicates with partner services.

Our work has shown the elements exist to apply Web service patterns to the design of IOS. The next step in the development of patterns for IOS must include a standardized method of classifying patterns, such that designers are aided in applying them. Similarly to the experience of the OO community, these classifications will not be possible until there is a larger amount of experience in the design of IOS. The identification of problems and the knowledge of creating solutions to these problems will allow more sophisticated Web service composition patterns to be created that address all three aspects of the system design in a consistent manner.

References

1. Hull, R., Benedikt, M., Christophides, V., Su, J.: E-services: A look behind the curtain. In: Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS). (2003) 1–14
2. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: The web services description language WSDL.
   http://www–4.ibm.com/software/solutions/webservices/resources.html (2001)

3. Andrews, T., et al.: Business process execution language for web services. www-106.ibm.com/developerworks/webservices/library/ws-bpel/(2002)
4. Bauer, B., Mueller, J.P.: MDA applied: From sequence diagrams to web service choreography. In Koch, N., Fraternali, P., Wirsing, M., eds.: Web Engineering: 4th International Conference (ICWE), Lecture Notes in Computer Science 3140 (2004) 132–136
5. Hauser, R., Koehler, J.: Compiling process graphs into executable code. In: Third International Conference on Generative Programming and Component Engineering. LNCS, Springer (2004) forthcoming.
6. Mandell, D.J., McIlraith, S.A.: Adapting bpel4ws for the semantic web: The bottom-up approach to web service interoperation. In: International Semantic Web Conference 2003. (2003) 227–241
7. Coplien, J.O.: A pattern definition. Hillside.net (2004)
8. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of reusable object-oriented software. Addison–Wesley (1986)
9. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-Oriented Software Architecture - A System of Patterns. John Wiley and Sons (1996)
10. Schmidt, D.C., Stal, M., Rohnert, H., Buschmann, F.: Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects. John Wiley and Sons (2000)
11. van der Aalst, W., ter Hofstede, A., Kiepuszewski, B.., Barros, A.: Workflow patterns. In: Distributed and Parallel Databases. (2003) 5–51
12. Wohed, P., van der Aalst, W., Dumas, M., ter Hofstede, A..: Analysis of web services composition languages: The case of BPEL4WS. In: 22nd International Conference on Conceptual Modeling (ER 2003). (2003) 200–215
13. Ruh, W., Maginnis,F., Brown, W.: Enterprise Application Integration: A Wiley Tech Brief. John Wiley and Sons, Inc (2001)
14. Hohpe, G., Woolf, B., Brown, K., DCruz, C.F., Fowler, M., Neville, S., Rettig, M.J., Simon, J.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley (2004)
15. Russell, N., ter Hofstede, A.H., Edmond, D., van der Aalst, W.M.: Workflow data patterns. White paper, Centre for Information Technology Innovation (2004) http://www.citi.qut.edu.au/babel/DP/DP.jsp.
16. The-Object-Management-Group: OMG model driven architecture. http://www.omg.org/mda (2003)
17. Frank, J.H., Gardner, T.A., Johnston, S.K., White, S.A., Iyengar, S.: Business processes definition metamodel concepts and overview. White paper, IBM Software Group (2004) http://www.omg.org/cgi-bin/apps/doc?bei/04-05-03.pdf.
18. Workflow Patterns. (2004) http://www.workflowpatterns.com/.
19. Amsden, J., Gardner, T., Griffin, C., Iyengar, S., Knapman, J.: UML profile for automated business processes with a mapping to BPEL 1.0. IBM Alphaworks http://dwdemos.alphaworks.ibm.com/wstk/common/wstkdoc/services/demos/uml2bpel/docs/UMLProfileForBusinessProcesses1.0.pdf(2003)
20. Janson, D.H., et al.: Patterns: Designing Process Integration Solutions. ibm.com/redbooks (2004)

Managing Ad Hoc Virtual Enterprises in Open SME Networks: The SCOOP Solution

Michael Ryba and Gerhard Wickler

University of Stuttgart, IPVS/ISE, Universitätsstraße 38, Stuttgart, D-70569, Germany
{Michael.Ryba, Gerhard.Wickler}@informatik.uni-stuttgart.de

Abstract. The aim of this paper is twofold. Firstly, we want to describe the results achieved in the SCOOP project. That is, we will describe an inter-organizational production planning system and broker that support human planners during order intake by supplying a number of alternative production plans using internal and external resources. Plans are rated according to rules specified by the user who only has to take the final decision. Four main software modules implement this functionality: a local PPS that assigns jobs to resources, an extension for communication with external PPSs allowing the seamless integration of external resources, and economic suitability analysis tool that rates production plans, and finally, a broker that opens up the networks to companies without PPSs. Secondly, we will describe and analyze two bottlenecks that still hinder the wide-spread use of the SCOOP solution. The analysis will reveal approaches how these bottlenecks can be addressed and we will describe our envisaged solution.

1 Introduction

Small and Medium-sized Enterprises (SMEs) form a dynamic and heterogeneous community, which is confronted by many challenges. These include increased competition resulting from the completion of the European internal market and the general trend for market globalisation, the constant threat of larger companies willing to outperform and/or buy them and the growing customer influence on production processes. More and more SMEs tackle these challenges by forming enterprise networks and strategic alliances.

The SCOOP project [1] [2] has addressed the problem of managing such enterprise networks and strategic alliances in dynamic SME networks using an innovative and advanced solution based on co-operative planning and control. The textile/garment and printing industries have been chosen as ideal representatives of highly dynamic business scenarios.

The first part of this paper describes the specific requirements identified in the SCOOP project, the solutions developed, and the results and benefits achieved. The second part describes two open issues identified in the SCOOP project and envisaged solutions, which will be subject to future projects.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 456–467, 2004.
© Springer-Verlag Berlin Heidelberg 2004

2 Virtual Enterprise Networks

These days most enterprises have to own very expensive resources in order to produce the quality goods and services that are required to be competitive in the market. Such investments are more difficult for SMEs than for large companies with higher capital. Thus, for SMEs to be able to stay competitive they have to make sure their resources are utilized to the full and avoid idle times in which fixed costs still have to be written off. These are exactly the issues we face in the textile and printing industries which consist largely of SMEs and micro companies producing quality goods using specialized skills competing in a global market place.

One approach to achieving higher resource utilization is the formation of virtual enterprises. The aim is to address demand fluctuations by using a distributed capacity network in which resources are utilized where they are currently available and jobs are matched to those resources that are optimally suited for them. An important issue here is the size of the network from which the virtual enterprise is formed: the more choice is available the more likely it is that an efficient virtual enterprise can be assembled. Furthermore, demand fluctuations can be better addressed in a larger network.

Production processes in virtual enterprise networks have to be optimised along the entire supply chain, which requires close co-operation between the participating companies. In addition to the (vertical) co-operation along the supply chain there is an increasing need for horizontal co-operation, especially in the European printing industry. In order to provide full service to their customers, companies accept orders although they do not have the capacity for processing these orders or they cannot produce at optimal cost due to a lack of specialised equipment. Therefore these orders are outsourced to co-operating companies, which either have free capacities or the equipment to produce at optimal cost.

Note that the formation of such networks is already standard practise in the printing industry. However, planning and controlling production processes in dynamic networks of co-operating companies is currently hindered by several problems:

- Especially small companies do not use IT systems for local production planning and control due to high investment costs and the inability of existing PPC systems to deal with highly dynamic changes of the production plan imposed by customer requests.
- Production planning across an entire network of co-operating companies is only possible if all companies use the same system for local planning and control and are willing to pass information about local planning and control to their partners.
- Vertical and horizontal co-operation requires negotiation, which is time-consuming and error-prone mostly due to a lack of agreed-upon cross-enterprise business processes.
- Finding suitable partners is not supported sufficiently by electronic marketplaces.

There have been attempts to use standard PPC systems (SAP, BAAN, etc.) in the past. However, generating production plans using these systems is far too time consuming to fulfil the high demands for short feedback loops. Therefore production planning in most cases still relies on magnetic whiteboards, i.e. all production planning including adjustments caused by changes of priority or external influences is performed manually.

3 The SCOOP Project

The problem of managing dynamic SME networks has been addressed in the SCOOP project, which has been funded under the fifth framework programme of the European Union (IST-2000-25200). The goal of SCOOP was to develop a methodology [3] for co-operative rough planning across company networks as well as a set of software tools [4] supporting this methodology. The textile/garment and the printing industry served as application scenarios as they are ideal representatives of highly dynamic business environments.

3.1 User Requirements

The requirements analysis performed with the end user companies involved in the SCOOP project made evident that the SCOOP software toolset had to provide four main functionalities:

- **Local rough planning:** First, the whiteboard-based planning had to be replaced by an IT-based approach. However, since detailed shop floor planning at an early stage of order processing does not make sense in highly dynamic end user scenarios, standard production planners were not usable here. A fast capacity-based rough planning, which only takes the capacity of organizational units and critical resources into account was required for individual companies.
- **Co-operative rough planning across a network of co-operating companies:** In order to make proactive and reactive subcontracting and/or outsourcing easier, the local rough planning systems had to be extended by mechanisms for exchanging planning data and the automatic negotiation of order shifting. In short, the local rough planners required an extension that enables automatic co-operation.
- **Decision support at order intake:** Support for make-or-buy decisions at order intake was required. This could be achieved by analysing the economic suitability of an order based on the technical specification of the requested product and using this analysis to compare different ways in which the product could be produced.
- **Broker services for establishing new partnerships:** In order to find new co-operation partners a broker service was required, which allows companies to register with a profile describing the services they offer. Thus potential co-operation partners can be searched by matching requested services with the stored profiles. This is particularly important for networks that have grown to a size where participating companies no longer know each other personally.

In addition to these functional requirements the following operational requirements had to be taken into consideration:

- **Platform independence:** Due to the diversity of hardware and operating system platforms used by the end users, the SCOOP software had to be portable.
- **Easy installation:** The software should be easy to install since many SMEs lack trained IT personnel.
- **Low operating costs:** In order to be affordable even for small companies the system should aim for low operating cost.

- **Security/confidentiality:** Since co-operative planning involves the exchange of confidential information the system should prevent unauthorized access.
- **Standards:** The system should conform to existing, de facto and emerging standards for software, hardware, communication, and data exchange.
- **User friendliness:** The system should be easy to use, even for persons having little experience with IT systems.
- **International use:** The system should support multi-lingual use.

3.2 Technical Approach

The core component of the SCOOP toolset is an extremely fast simulation-based rough-planning system [5], which allows online-planning at order intake. The rough planning is performed at the level of organisational units of a company and is based on a time frame model, i.e. the production capacity of each organisational unit is modelled by a row of chronological time frames each representing a specific "production capacity basket" as shown in figure 1.

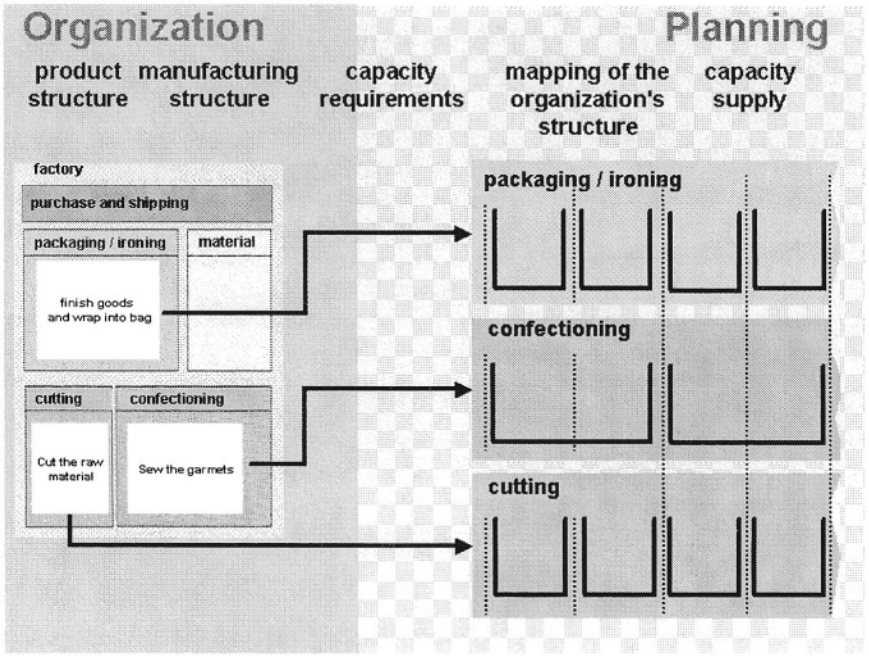


Fig. 1. The time frame model as basis for fast and reliable rough planning

For each time frame two capacity limits are defined:

- Normal capacity limit
- Maximum capacity (using overtime)

For incoming orders, bill explosion is only performed down to the end products of the organizational units and a scheduling running backward from the requested delivery date is done. This enables extremely fast rough planning cycles based on material-, capacity- and time-constraints. As soon as production capacity of one organisational unit (OU) has reached the normal capacity limit and is thus determined as a bottleneck, the user is informed. A negotiation between the human planner and production can clarify the situation immediately. The production unit should be able to determine, if capacity can be increased temporarily to fulfil the order. In addition, the resulting capacity load for the organisational units can be monitored in a capacity and load overview.

In order to make co-operative planning as easy and transparent for the end users as possible, most of the required functionality has been seamlessly integrated into the local rough planning system and is thus hidden from the end users. From the perspective of the human planner, partner companies and subcontractors are treated in the same way as local organizational units. The resulting scenario is illustrated in figure 2.

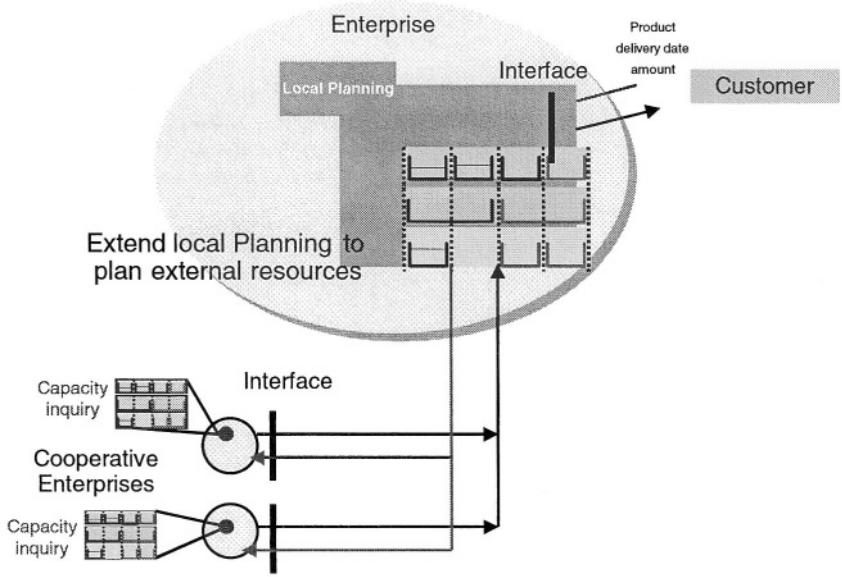


Fig. 2. Co-operative Planning based on local rough planning

However, the execution of the SCOOP rough planner requires capacity information from the remote planning systems of the partner companies. Therefore communication with these planning systems is required. Since a synchronous on-the-fly communication during the planning run would substantially slow down the planning process (or even stop it if a remote planner is down) an asynchronous three-phase approach has been chosen:

1. During a pre-processing phase electronic inquiries are generated and sent asynchronously and in parallel to the remote planning systems. During a defined period of time capacity offers from the remote planners are then collected.

2. Based upon the collected capacity offers and the local capacity situation possible planning scenarios are computed. The one suited best is then chosen by the human planner.
3. Orders are sent to the remote planning systems of the selected capacity offers. The other inquires are cancelled.

This communication protocol may appear rather rigid at first. However, since it was only intended for direct communication between the different planning systems, it does not constitute a problem in the context of co-operative planning. The messages that are being exchanged are XML and the according message formats are described by XML schemata. While this approach does not adhere to any predefined standard, the availability of the XML schemata makes it possible for other software to interoperate with the different planners across company networks.

For the human planner the co-operative planning involving external companies from a network often amounts to the availability of a wide spectrum of possibilities for producing given goods. The required decision support tool was thus provided in the form of an economic suitability analysis tool. To use this tool the human planner has to provide a set of rules with which a numeric value can be computed that rates the economic suitability of producing given goods on given production resources. Economic suitability is described by a number between 0 and 9, which states how desirable it is to receive a certain order. An economic suitability of 0 means not desirable at all and an economic suitability of 9 means absolutely desirable. Every company can define their own way to compute economic suitability as they see fit and the definition has to be done only once.

When an order comes in, the economic suitability can be computed automatically for every possible way of producing this order. The resulting economic suitability values are then displayed together with the corresponding production plans to the human planner who makes the final decision. This is to allow for criteria that cannot be expressed formally by the planners.

The following example from the printing end user scenario illustrates the meaning of the economic suitability value. Consider a printing company with a four-colour sheet printing machine. An order for printing 1.000 single colour business cards on such a machine is absolutely not desirable and therefore would be rated 0. Printing 10.000 copies of a four-page, four-colour brochure is optimally suited for the machine and therefore ranked 9. An order for printing 5.000 copies of a two colour flyer is not optimally suited but still desirable and could be rated 5.

The co-operative planning system described so far allows groups of co-operating companies to automatically shift orders within the network and thus to use each others free capacities. However, this automated way of co-operation is limited to companies, which already know each other.

In order to allow a more open kind of co-operation network, a broker service has been designed and implemented, which can be used to build an electronic market place. Companies can register at the broker providing a profile describing the services they offer. Other companies can sent requests for specific services to the broker, which matches the request with the stored service profiles and either provides a list of possible co-operation partners or directly forwards the request. There are two basic ways for accessing the broker service:

1. Companies, which are using the SCOOP planning system, can send requests to the broker using the same protocol used for the communication between planning servers. In the same way, requests forwarded by the broker can be handled.
2. Other companies can access the broker using a web browser. The functionality provided is essentially the same. However, the response times are slower, because there is no direct communication between planning systems.

The extended scenario including the broker is illustrated in figure 3.

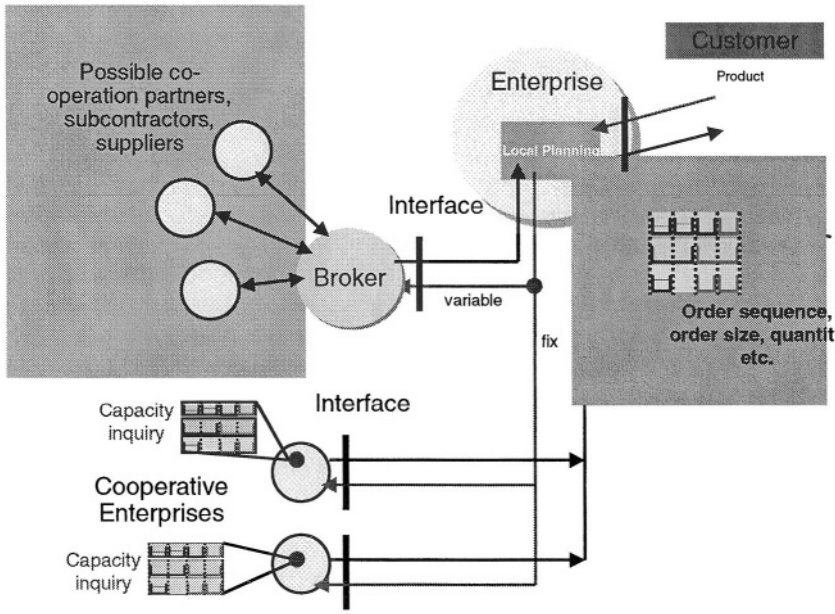


Fig. 3. Generalised SCOOP scenario

The most important question that needed to be addressed for the design of the broker was how to represent services. On the one hand, this representation had to be powerful enough to express even sophisticated service profiles, on the other hand it need to be sufficiently simple to allow for efficient matching algorithms and a user interface that the users can understand. The formalism used for the prototype that was available at the end of the project used a representation that provided three different views: products to be produced, machines available for production, and processes that can be realized. Each of these views is essentially propositional with a set of parameters to allow specificity.

One of the major concerns of the end users involved in the SCOOP project was trust between the co-operating companies. Therefore the broker keeps track of all co-operations established via the broker and asks the involved partners to assess each other after the completion of the business transaction. In this way trust profiles can be maintained. Furthermore, the broker only knows about services that are available in principle, but not about the actual capacity situation at any of the registered companies. This gives partners the privacy they required in the network.

3.3 Results Achieved

Within the SCOOP project we have provided a software framework that supports collaboration in the form of virtual enterprises in a network of SMEs. Individual companies can use the sophisticated production planning system (PPS) to assess resource availability for incoming jobs. The PPS then keeps track of the resources that have been assigned to accepted jobs on the basis of capacity baskets. Thus, the system has a good overview of production capacities available. Fast re-planning is possible at any time, making the system ideal for SMEs that need to be flexible due to the frequent change in customer demands and priorities. The technological basis for this flexibility lies in the capacity basket-based model and the rough planning algorithm that is used.

With a PPS at the individual companies that knows the current capacity situation, the door is open for the automated and fast formation of virtual enterprises for jobs at hand. For this purpose the SCOOP software allows external resources to be integrated into the local PPS such that these capacities can be taken into account when an assessment for a new job is made. When a new job is accepted the local PPS communicates with the PPS of the company that has the required resources and books the necessary capacity. Thus, an ad-hoc virtual enterprise is formed on demand for a job at hand.

4 Opening Up the Network

Two bottlenecks were identified at this point. Firstly, the production planning still takes place on a whiteboard in most companies. This prevents any automated participation in an ad-hoc virtual enterprise. While the web-interface provided by the broker constitutes a partial solution to this problem, the full set of benefits can only be realized by companies that use a PPS the can communicate with the broker directly. Secondly, service descriptions and production requests need to be standardized for the broker to be able to guarantee that they will be interpreted and reasoned about correctly. Otherwise there may mismatches, where a machine is not suitable for an assigned job, or the broker might not find a resource that is suitable for a given job. Thus, standardization is required for large open networks of companies to collaborate.

In the remainder of this paper we will analyse the reasons for these bottlenecks and identify the underlying issues that need to be addressed before the advantages that are already available to the closed network can be extended to the open network.

4.1 Self-Consulting Systems: The Idea

One of the most important prerequisites for an open collaborative business framework to be successful is that it requires a critical mass of participants. Most of these participants will be SMEs as they can expect most benefits from the virtual enterprise concept and are, by their nature, most in number. A requirement for the efficient participation in a collaborative business framework is the availability of complex software systems, e.g. a PPS, at the individual companies. This implies that SMEs must be equipped with complex software systems to form the critical mass required for open collaboration. However, such systems can be very difficult to install, configure, use,

and maintain. This problem is usually tackled by buying in external consultancy which can cost a multiple of the purchase price of the software itself, thus making it unaffordable for SMEs.

This is most certainly the experience we have had in the SCOOP project. While the software solution was very popular with the end users within the project and others that have seen the system, the cost of consulting always was a critical issue. This problem can be addressed by extending the PPSs used by the individual companies with a self-consulting system that uses an explicit self-model to perform the expensive tasks usually performed by the consultants [6]. It requires a self-model-based autonomous self-consulting and tutoring framework for decision support in collaborative software systems. As explained above, such a module is absolutely vital to achieve the critical mass in SMEs that is necessary to make networked collaborative business a success.

The framework to be developed will essentially consist of a generic self-consulting and tutoring module which augments an existing collaborative target system with self-consulting and tutoring functionality, and a complementary methodology for this augmentation process. The self-consulting functionality corresponds to the task the module accomplishes during the installation and configuration phase that is normally performed by human consultants. The tutoring functionality on the other hand corresponds to the support provided by an intelligent help system instructing the end user during daily operation of the target system. Both, self-consulting and tutoring will be focussed on providing support for decisions that need to be taken to operate the complex collaborative system efficiently and correctly, exactly when this support is needed.

The self-consulting and tutoring module will be autonomous, meaning that it does not require human intervention to perform its task. Furthermore, it will be self-model-based, meaning that it has an explicit and abstract model of the collaborative target system available which is used as a basis for the most important self-consulting and tutoring tasks in a collaborative context, namely explanation generation and target system simulation. Finally, the focus will be on collaborative software systems by which we mean systems that facilitate collaborative work. Such systems could, for example, consist of a collaboration manager, e.g. a broker, that coordinates a number of resource management systems, e.g. PPSs, existing in a group of enterprises, which, through the collaboration, form a virtual enterprise.

4.2 Semantic Capabilities: The Idea

As explained above, the broker currently uses the interface originally developed for communication between planners. While this works well for a limited set of companies, there are two problems when it comes to opening up the network.

1. XML only defines a generic syntax for the communication language and, even with the XML schemata definitions that constrain what constitutes an XML document that is not only well-formed but also valid, it remains a purely syntactical specification. The meaning of the different tags and what appears between them is only implicitly defined by what the communications do upon receipt of an XML message.

2. Even if the meaning of the different tags was defined in some way, what appears between the tags, the real content so to speak, cannot be defined that easily. The problem here is that XML cannot constrain this part of a message and thus, users are free to use any formalism or natural language they feel appropriate. This makes the task of matching a production request to a service description unsolvable in the general case.

Fortunately, there are currently a number of technologies available for the Semantic Web [8] that can be used to at least partially address both of the above problems. RDF is an XML-based standard for the description of generic resources. RDF comes with a model-based semantics that defines the meaning of RDF statements in the sense that it defines when it is valid to draw which conclusions from a given set of RDF statements. If both, production requests and service descriptions were described in RDF, it would be possible to formally define when the two match, and thus a mismatch should be impossible unless one or the other has been described incorrectly. The semantic web also provides a formalism to define a vocabulary that can be used to describe resources. This RDF-based formalism is OWL, a language for defining vocabularies in the form of ontologies. Currently in progress is the development of an extension of OWL specifically for the description of services. This language is OWL-S [9].

With RDF, OWL, and OWL-S the Semantic Web provides the means to define a vocabulary for the description of services that can be offered to other companies through a broker. However, it does not provide the vocabulary itself. For our broker to cope with the demands of a large open network three types of terms need to be defined: generic service description and eBusiness terms for the description of business processes in general, specific services that are relevant to different industries, e.g. printing, and finally, specific vocabulary for the description of products that are produced or consumed in different industries.

The use of Semantic Web technologies for the description of services also has an additional advantage: since OWL-S has been influenced by the AI Planning community, AI planning software may be used for service orchestration. That is, instead looking for one company that has advertised a service with which a given production request can be satisfied, the broker can combine the services offered by several companies in order to satisfy the request. This allows for much more complex collaborations in the network. An example of such a system based on a predecessor of OWL-S, DAML-S, is described in [10].

5 Conclusions

The fact that European Industry is based on an SME structure is one major reason for its strength and flexibility. Smaller companies can adapt more efficiently to changes in the economic environment. Also they will identify and occupy market niches quicker than bigger entities.

However, in capital-intensive segments like printing a certain minimum size is required for survival. A minimal investment into production equipment amounts to almost one million Euro and several millions on average. Such investments can only be financed, if acquisition skills and planning techniques can provide a constant work-

load. But this unfortunately is the main shortcoming of the "usual" SME-entrepreneur. Being too much occupied with skilful production tasks and running and maintaining his equipment, he usually does not care enough about economics and efficiency.

The aim of this paper was twofold: In the first part we have described the SCOOP project and the results achieved this far. In the second part we have elaborated on two remaining problems and presented an approach for a solution.

SMEs are facing many challenges today and one approach to addressing a number of them consists of the formation of strategic alliances in enterprise networks. The aim of the SCOOP project was to address the problem of managing co-operation in dynamic SME networks using an innovative and advanced solution based on co-operative planning and control. The SCOOP solution essentially consists of three layers. First, enterprises participating in a production network were equipped with a PPS based on a rough planner with capacity baskets that fits their needs. The second step was achieved by connecting the local PPSs such that each company could negotiate with the PPS of another company in the network and shift orders as required. Finally, a broker was added to the network in an attempt to open up the network and provide more flexibility.

The SCOOP software was evaluated in a number of enterprises from the textile and printing industry, both being ideal application scenarios for SCOOP as they consist largely of SMEs. A number of benefits were achieved for the participating companies, most importantly a reduction in production costs, a better utilization of production resources, and a reduction of risk.

However, the experience gained in the project also has shown that there are still issues that need to be addressed. Firstly, the introduction of the broker did not open up the network in the sense that it does not provide the same functionality and thus benefits to the open network that are available to the closed network. The roots of this problem lie in the underlying representation used by the broker, which lacks a semantic foundation. Furthermore, trust is a critical issue in inter-business relations and the SCOOP broker lacks sophisticated trust-building features. Only a system that has these features could truly open up the network and we have outlined a technical approach that could achieve this.

A second problem identified in SCOOP is the critical mass problem. A network only becomes efficient with a critical mass of participants. The most important obstacle for the critical mass, however, is the fact that PPSs are expensive, too expensive for small companies. An analysis of this situation has shown that the largest share of this cost comes in the form of consulting that has to be bought in, and again, we have described an approach we call self-consulting systems that addresses this issue.

References

1. M. Ryba, M. Matthiesen, B. Löffler, R. Aits, Á. Duarte de Oliveira: Sophisticated Co-operation in Dynamic SME Networks based on Co-operative Planning and Control, in: B. Stanford-Smith, E. Chiozza (eds.): E-work and E-commerce, Vol. 2, IOS Press, 2001
2. J. Fradinho Oliveira, Á. Duarte de Oliveira: Dynamic Management of Virtual Enterprises in the Textile/Garment Industry, in: B. Stanford-Smith, E. Chiozza (eds.): E-work and E-commerce, Vol. 2, IOS Press, 2001

3. R. Aits, B. Löffler, M. Ryba, M. Matthiesen: A Methodology for Co-operative Rough Planning in Dynamic SME Networks, in: B. Stanford-Smith, E. Chiozza, M. Edin (eds.): Challenges and Achievements in E-business and E-work, Vol. 1, IOS Press, 2002
4. M. Ryba, M. Matthiesen, R. Aits, B. Löffler: A Software Architecture for Co-operative Planning in Dynamic SME Networks, in: B. Stanford-Smith, E. Chiozza, M. Edin (eds.): Challenges and Achievements in E-business and E-work, Vol. 2, IOS Press, 2002
5. W. Sihn, P. M. Löllmann: Order Management by Means of the 3-Liter PPC Concept, In: Pacific Conference on Manufacturing, Vol. 2, pages 258-264, 2000.
6. M. Ryba and G. Wickler: Self-Consulting IT Systsms – An Appraoch to Increase Acceptance of IT Systems in SMEs. In: P. Cuningham, M. Cunningham and P. Fatelnig (eds.), Building the Knowledge Economy: Issues, Applications, Case Studies, pages 1374-1379, IOS Press, 2003.
7. M. Ryba and G. Wickler: Competitive SME Production Networks using Intelligent Semantic-Based Brokering. In: P. Cuningham, M. Cunningham and P. Fatelnig (eds.), Building the Knowledge Economy: Issues, Applications, Case Studies, pages 1404-1410, IOS Press, 2003.
8. http://www.w3c.org/2001/sw/
9. http://www.daml.org/services/owl-s/1.0/
10. D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau: Automating DAML-S Web Services Composition Using SHOP2. In:D. Fensel, K. Sycara, and J. Mylopoulos: The Semantic Web – Proc. 2nd ISWC, pages 195-210, Springer Verlag, 2003.

Engineering of Dependable Complex Business Processes Using UML and Coordinated Atomic Actions

Nicolas Guelfi, Guillaume Le Cousin, and Benoît Ries

Software Engineering Competence Center
Faculty of Science, Technology and Communication
University of Luxembourg, L-1359 Luxembourg-Kirchberg, Luxembourg
{nicolas.guelfi,guillaume.lecousin,benoit.ries}@uni.lu

Abstract. For many companies, it is widely recognized that languages and methods for modeling and analyzing distributed business processes are becoming more and more important. For improving efficiency, the modeling language should provide reusability, easy understanding by business analysts, and should ease the validation and verification tasks. In this paper, we present an approach for developing dependable complex business processes using UML that satisfies these requirements. The proposed UML notation is designed to be directly integrated with COALA, a syntactically and semantically well-defined fault-tolerant advanced transaction model based on Coordinated Atomic Actions. Structuring concepts like nested business processes and fault-tolerance through exception handling are first class concepts brought by our approach that are crucial for modeling cross-enterprise business processes. The modeling phase is followed by a validation phase by business analysts through animation of the business process model in a workflow environment. Due to the precise notation used, automatic verification of crucial properties is accessible through integration with an automatic verifier.

Keywords: Cross-enterprise business processes modeling, UML, advanced transaction model, fault-tolerance, validation, verification, methodology, tools.

1 Introduction

Business process modeling became a significant point for companies due to the growing need of competitiveness, responsiveness from the changes of market, but also for a better communication and comprehension of the businesses in an organization. Recently, many organizations cooperate forming partnerships to deliver solutions in a competitive way. They focus on their core business and outsource secondary activities to other organizations. Thus, companies require the ability to model business processes that are related to processes of their partners. As human dependence on technical infrastructure and systems grows, the threats of major failures grow accordingly. This fact calls for a concentrated effort to improve the quality of Information and Communication technology (ICT)-based systems along the following axes: Security, Dependability and Trust. Concerning dependability, there are many application areas where system failure may lead to loss of financial resources, and even loss of human lives (e.g., control of aircrafts and nuclear plants).

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 468–482, 2004.
© Springer-Verlag Berlin Heidelberg 2004

In such cases, scientifically rigorous evidence is needed in advance to back up promises about a product's future service. The problem is complicated by the need for practical ICT systems to evolve in response to changes in their requirements, technology and environments, without compromising their dependability. This is even stronger in applications where system boundaries are not fixed and are subject to constant urgent change (e.g. in e-business). Scientific and technological advances are required to (a) demonstrate that commercial and industrial-scale software can be developed to be truly dependable, and with less development risk than today; and (b) dependable systems can be evolved dependably including, for a class of applications, just-in-time creation of required services.

A business process model describes the activities that are carried out to reach a certain goal, and the execution order of these activities. It may involve many actors and perform many activities, which may be as simple as sending or receiving messages, or as complex as coordinating other processes or activities. An activity may also fail, which implies the integration of a rigorous fault-tolerant mechanism to deal with abnormal situations.

In order to represent such systems, the modeling language should describe actors with their role in the process, the activities execution flow (control flow) which may involve concurrency, and interactions between actors and between activities through messages exchange (data flow). The language should allow reusability of processes, and be easily understandable by non-computer specialists as business analysts. It must also have a well-defined semantics facilitating the validation and verification tasks, and thus ensuring to the involved organizations that the model is correct and fulfills the required properties.

We selected the Unified Modeling Language (UML) as the basis for our modeling language. Amongst all the UML [1] diagrams available, we consider that the UML activity diagrams syntax is well adapted for the behavior specification because it allows representing processes with actors, roles, activities execution flow and data flow in a graphic and comprehensive way [2], [3]. Activity diagrams syntax has been shown to be useful for business process modeling in [4]. UML Class diagrams represent the structure of data handled by the business processes. Moreover, UML provides extension facilities (stereotypes, tagged values and constraints) that allow building UML profiles for specific application domains, and it allows interoperability through the standardized open-source XML Metadata Interchange (XMI) format. Extensible Markup Language (XML) is a simple, very flexible text format, originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. Thus, in this paper, we define the syntax of our modeling language, which is a customization of UML activity and class diagrams.

The semantics is given using an automatic transformation of our UML models into a COALA model. The correspondence of formal semantics between UML and COALA is not an issue in this paper. The formal semantics is provided by the COALA model, thus business analysts must know that only the COALA models has well-defined meaning, and thus only the COALA models, but not the UML models, should be used, and modified if necessary, for the final system validation. The UML models and the transformation from UML models to COALA models ease the formal specification of transactions to business analysts by modeling in a visual and more intelligible way for non-experts. COALA is a textual modeling language based on the Coordinated Atomic Action (CAA) concept [5], [6]. CAA provides a conceptual

framework for dealing with cooperative or competitive concurrent processes. This is done by integrating three concepts:

- Conversations: provide cooperative activities, and is used to implement coordinated and disciplined error recovery;
- Nested transactions: maintain the consistency of shared resources in presence of failures and competitive concurrency, and allow a hierarchical transactions structure;
- Exception handling for error recovery.

Using the CAA concepts for modeling business processes allow cooperation between business partners, nesting transactions for a better reusability, fault-tolerance by using a transaction system [7], [8] for accessing shared resources, and exception handling for reacting as well as possible to an abnormal situation. The COALA language [9] provides a well-defined semantics of CAA and our approach aims at providing to COALA a UML-based syntax adapted to business process modeling, and suitable for verification and validation.

As for now, the modeling phase is followed by a validation phase handled by business analysts. During this phase, the models are transformed into XML Processing Description Language (XPDL) standardized format that is computed by a XPDL-compliant workflow engine. Business experts simulate the system behavior with the help of the workflow engine; i.e. they cooperatively validate the business model by playing roles of the business processes, and participate in the execution of the processes by receiving messages and sending answers. Since the validation cannot prove the absence of errors, but only detect some of them, we suggest to improve the current way of validation, by introducing an additional and complementary approach based on formal proofs and on the formal semantics of COALA given in terms of COOPN/2 (Concurrent Object-Oriented Petri Nets) expressions, allowing the verification of properties for all possible scenarios. We focus our work on the specification process, however an alternative approach for validation could be to use an execution platform that handles middleware implementations of CAAs like the DRIP platform [10]. This alternative approach differs by focusing on the generation of an executable prototype of the whole system.

This paper is structured as follows. Section 2 describes the coordinated atomic actions and COALA concepts, which are used to build our notation. This notation is described in Section 3. Section 4 describes the development process used for engineering complex business processes. Section 5 concludes and presents future work.

2 Coordinated Atomic Actions and COALA

The CAA concept [5], [6] facilitates the design of reliable distributed systems. It allows several threads (or *roles*) to perform a set of operations cooperatively using the conversation concept [11], [12] through *internal objects*. CAAs may be nested. Several nested CAA may be started concurrently and may be in competition for shared resources. To maintain the consistency of these resources in the presence of failures and competitive concurrency, the nested transaction model [13] is used. Moreover, CAAs integrate exception handling, in order to perform coordinated error

recovery involving all the threads, or to propagate an exception to the enclosing CAA (possibly after having undone all its effects through the transaction system). The COALA language [9] provides a well-defined semantics to CAA in order to carry out dynamic properties verifications.

Fig. 1, below, illustrates an enclosing CAA involving three roles of which two carry out a nested CAA. This section will detail progressively the elements of the figure.

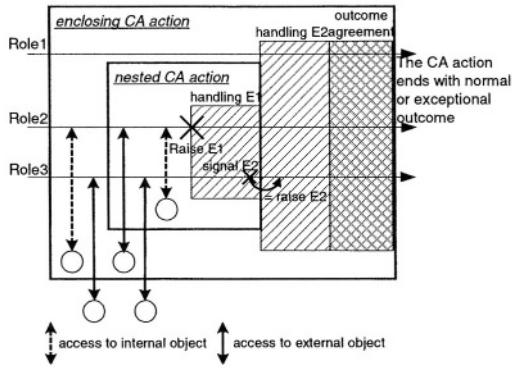


Fig. 1. Coordinated Atomic Actions

2.1 Fundamental Concepts of CAA

Roles. A CAA is composed of one or more *roles* (Fig. 1 shows three roles) executed in parallel to perform cooperatively a transaction fulfilling the ACID properties. The ACID model is one of the oldest and most important concepts of database theory. It sets forward four goals that every database management system must strive to achieve: Atomicity, Consistency, Isolation and Durability. The goal of a CAA consists in coordinating its roles. Each thread that wants to participate in a CAA takes a role by activating it. The CAA cannot start until all the roles are activated, i.e. roles start together.

Objects. The CAA concept defines two kinds of objects. *Internal objects* (shown as circles inside the CAA in Fig. 1) are internal to the CAA and are used to support cooperative concurrency allowing the roles to agree upon the set of operations they wish to perform on the external objects. *External objects* (shown as circles outside of the CAA in Fig. 1) may be accessed by other CAAs executed concurrently, by using a transaction system ensuring the ACID properties.

Outcomes. All the roles exit a CAA at the same time, after each role has voted for its outcome. The four possible outcomes are:

- *Normal.* Indicates that the CAA was terminated and committed its results correctly while satisfying the ACID properties during execution.
- *Exceptional.* Indicates that the ACID properties were satisfied, but an exception was signaled to the enclosing CAA.

- *Abort.* Indicates that the CAA has aborted and undone all its operations while satisfying the ACID properties
- *Failure.* Indicates that an error occurred and the ACID properties could not be satisfied.

Nesting. A CAA can be nested into another one (as shown in Fig. 1). According to the nested transactions model, the effects of a CAA only become permanent when the top level enclosing CAA terminates. Nevertheless the effects of a nested CAA are visible to its enclosing CAA as soon as the nested CAA terminates.

Exceptions and Handlers. A CAA may contain two kinds of exceptions: *internal* and *interface* exceptions. In Fig. 1, exception E1 is internal to the nested CAA, and E2 is an interface exception of the nested CAA while it is internal to the enclosing CAA in order to handle it.

Internal exceptions are totally managed in the CAA, whereas interface exceptions correspond to the list of exceptions that the CAA may signal to its enclosing CAA (E2 is propagated to the enclosing CAA that handles it). By default, two interface exceptions exist: *abort* and *fail* (corresponding respectively to the abort and failure outcomes).

Due to the concurrent nature of the targeted systems, several exceptions may occur at the same time, and an exception requires all the roles to perform the recovery actions cooperatively. CAAs apply the following rules to handle exceptions:

- If internal exceptions are raised concurrently, an exception graph [14] is used to determine which handler to activate.
- When an internal exception is raised, all the roles activate the corresponding handler. If roles have entered a nested CAA, the resolution mechanism waits until the nested CAAs terminate (blocking method).
- If an interface exception is signaled by at least one role, all the roles signal the exception. If different interface exceptions are signaled, the CAA attempts to abort, if it fails the fail exception is signaled.
- If an interface exception is signaled while another role raises an internal exception, the raised exception is ignored.

2.2 COALA Language

COALA (Coordinated atomic Action LAnguage) [9] is a formal language used for the specification of CAAs. A CAA is defined within a COALA module, which comprises two sections:

- An *interface* section, which is visible to other CAAs. It declares the list of roles with their parameters (the external objects), and the list of interface exceptions with their parameters;
- A *body* section, which is private and hidden to other CAAs. This section contains:
  - The list of internal objects,
  - The list of parameterized internal exceptions,
  - A resolution graph which lists the combinations of internal exceptions that can be raised concurrently together with the resolved exception,

- A description of each role, consisting in the roles' behavior, the declaration of which handler must be activated for each resolved exception, and the behavior of each handler.

COALA provides additional features compared to the ones provided by the basic CAA concepts. A role is allowed to create new threads; however, a thread must terminate in the CAA where it was created. A role may be *asynchronous,* i.e. the role may enter the CAA during its execution and the CAA is allowed to start before the role is activated, however all the roles must synchronize at the end of the CAA and leave together. A role may require to be activated more than one time, possibly an indefinite number of times for asynchronous roles.

2.3 COALA's Semantics

COALA's semantics is defined as a translation from COALA programs into their formal description in COOPN/2 [15], which is an object-oriented specification language, based on Petri nets and algebraic data types. Three generic COOPN/2 classes are defined *(Caa, Role* and *Scheduler),* specifying the general behavior of CAA, roles, and access to external objects. The translation consists in generating a COOPN/2 class for each CAA and each role of the COALA program. Each of these classes inherits from one of the basic abstract classes *(Caa* or *Role)* and defines new axioms to specify their behavior.

3 FTT-UML: A UML-Profile for Fault-Tolerant Transactions

The proposed business process modeling language is based on the UML activity and class diagrams syntax and designed for automatic transformation into COALA, i.e. the modeling of COALA concepts, described in Section 2, with UML are eased with our notation: FTT-UML. We define our notation as a UML profile, i.e a customization and extension of existing UML elements. The syntactic elements used are *swimlanes* representing the participants of a business process (also called *roles*), *nodes* representing activities performed by a participant or representing exchanged data, and *edges* representing the execution flow (solid lines) or the data flow (dotted lines). In the following sections, the flows will be specified in terms of token flow through the nodes. A node has pre- and post-condition, which may be the followings:

- Pre-condition XOR-merge: the node is executed for each incoming token
- Pre-condition AND-join: the node is executed only when a token is present for all its incoming edges
- Post-condition XOR-choice: after execution, a token is offered to only one outgoing edge
- Post-condition AMD-fork: after execution, a token is offered to all the outgoing edges

3.1 Running Example

To illustrate our notation, we will use a simple running example (see Fig. 2) involving several organizations, namely a customer, a wholesaler and a manufacturer; the management of abnormal situations as payment failure or production failure; and nested processes, as shown in Fig. 3.

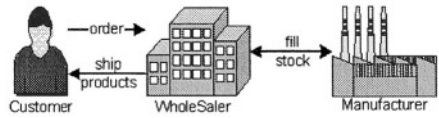


Fig. 2. Running example

When a customer sends an order to the wholesaler, the wholesaler launches a nested process *OrderTreatment* (see Fig. 3) consisting in two concurrent sets of activities for the wholesaler:

- Calculate the price of the order and launch the nested process *Payment* for the payment of the customer to the wholesaler;
- Check the stock. In case of an insufficient stock it sends a production order to the manufacturer and launches the nested process *Production.*

Two failures may occur:

- Production failure, if the manufacturer is unable to produce ordered products;
- Payment failure, if the customer is unable to pay the wholesaler. If the production fails, the wholesaler needs to abort the payment.

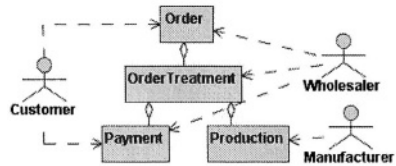


Fig. 3. Nested processes

3.2 Roles

A role in a business process represents a participating organizational unit. For satisfying the reusability requirement, a role has to be independent of the context where the process is executed, i.e. a role does not represent a physical organization, but the role that an organization has to take in the process (for example, the role *wholesaler* may be taken by different physical companies in different contexts).

Roles are represented by swimlanes, which are vertical lines dividing an activity diagram, as shown in Fig. 4. Each swimlane contains the activities performed by the corresponding role. From the COALA concepts, the execution of a process starts when all its roles are activated, excepted for asynchronous roles (represented by a stereotyped swimlane «asynch»), which may be activated during the execution of the

process (for example, we could imagine that during the production of the manufacturer, an expert comes to perform verifications after a product was detected with a missing piece). However all the roles must synchronize at the end of the process and leave together with the same outcome.

The designer can specify that a synchronous role must be activated *n* times before the process starts, by adding <*n*> at the end of the role's name, or that an asynchronous role can be activated for an indefinite number of times by adding <*> (e.g. if the wholesaler performs an invitation to tender for the production, an indefinite number of manufacturers may join the process to make propositions).

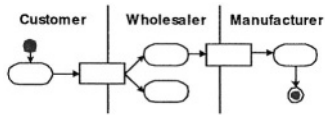


Fig. 4. Roles' notation

3.3 Data

Data structures are modeled using class diagrams (Fig. 5) and the corresponding data are used in activity diagrams using object nodes (Fig. 6) represented by rectangles. Each time a data structure is produced (a token comes in an object node: XOR-merge), a copy of this set of data is made available to all the activities that need it (a token is offered to all the outgoing edges of the object node: AND-fork). For example, each order sent by a customer will be transmitted to all the wholesaler's departments that need it.



Fig. 5. Class diagram modeling data structures

Fig. 6, below, illustrates data operations. In this figure, we model an agreement between the customer and the wholesaler: the customer sends an order, the wholesaler asks the price for the delivery and the production, then sends back the total price to the customer. At this moment, the customer can choose to make a new order replacing the old one, accept the order, or cancel the process. The process provides as output the order accepted by the customer.

In order to provide data manipulation primitives, the notation includes the following operations:

- Assign an expression to an object (or data structure) or an attribute. This is specified by an edge going to an object node. This edge has the stereotype «assign» and is labeled either with *expression* or *attribute=expression,* respectively to assign an expression to the object or an attribute of the object. For example, in Fig. 6 for assigning a value to the *total price* object.

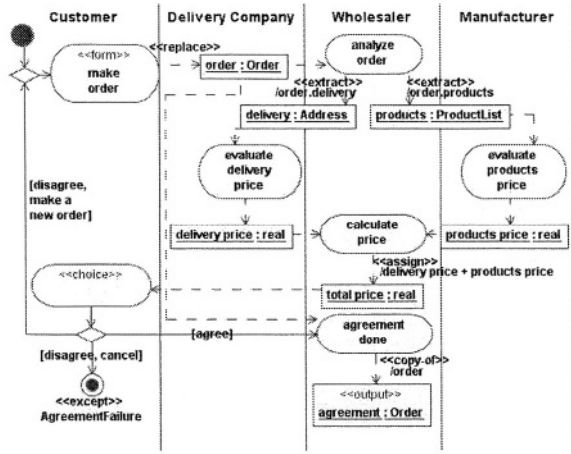


Fig. 6. Order agreement

- Copy the content of an object or an attribute into another object of same type, specified by an edge from the source object to an action node, and an edge from the action node to the target object. For copying an object, the latter edge has the stereotype «copy-of» and is labeled by the name of the source object, while for extracting an attribute the edge has the stereotype «extract» and its label has the format *object.attribute.* The copy is performed at the end of the action node execution. For example, in Fig. 6 for extracting the delivery address and the products from the order.
- Replace the content of an object that is not used yet. If a token still exists in an outgoing edge of the object node while a new one is coming, instead of queuing the new token, it replaces the existing one. This is specified by adding the stereotype «replace». For example to replace the *order* object for the action *agreement done* waiting the customer agreement (Fig. 6).

3.4 Actions

A role performs actions, represented by action nodes (corner-rounded rectangles). An action is executed when all its pre-conditions are satisfied (AND-join), and offers a token to all its outgoing edges (AND-fork). For example, in Fig. 6 the action *calculate price* is executed once the delivery price and the products price are available. An action without specific stereotype represents a business activity, however the two following stereotypes may be used:

- «form» for information supplying. Incoming objects represent data giving details about the information to supply, while outgoing objects are supplied information. For example, in the action *make order,* the customer fills in the information related to his order.

- «choice» for taking a decision. Incoming objects represent data giving details about the decision to take. A decision node having guards representing possible choices must immediately follow such an action node. For example in Fig. 6, the customer uses the *total price* to choose either if he is *agree* or *disagree.* Stereotypes «form» and «choice» may be combined.

A diamond represents either a decision node if its outgoing edges have guards (in Fig. 6 the diamond under the action «choice») or a merge node if not (the diamond under the black dot). A decision node performs an AND-join / XOR-choice, while a merge node performs a XOR-merge / AND-fork.

3.5 Starting Point and Inputs

A process may take data as input. Object nodes having the stereotype «input» represent such data. The starting point node is either an initial node (represented by a black dot) or the only node depending only on «input» nodes (i.e. a node having only incoming edges from object nodes «input»). There must be a unique starting point amongst all the synchronous roles, which receives a token when the process starts. The asynchronous roles may have an initial node, which receives a token when the role enters the process.

3.6 Outcomes

A process terminates with a normal outcome if its objective is reached, or signal an exception to the enclosing process if a problem occurs. Moreover a process may produce outputs. A normal outcome is represented either by a final node (a bull's eye) if the process does not produce outputs, or by object nodes with stereotype «output». In the same way, an exceptional outcome is represented either by a final node with stereotype «except» and labeled with the exception's name, or by an object node with stereotype «except» named by the exception's name and representing the exception parameter. Fig. 6 shows a normal outcome with an output (agreement:Order) and an exceptional outcome without parameter (AgreementFailure).

3.7 Nested Processes

A sub-activity node (represented by an action node with an icon in the lower right corner) models a nested process. A nested process being able to involve several roles, the roles of the enclosing process have to specify which role they want to take in the nested process. There are two possibilities for that:

- In the enclosing process, a role participating to the nested process has an edge going to the sub-activity node and labeled with the name of the role to take (Fig. 7 shows the *Wholesaler* taking the role *Paid* in *Payment*). If the role in the enclosing process does not contain the sub-activity node, it has an action node with the stereotype «invited» in its swimlane (Fig. 7 shows the *Customer* taking the role *Payer* in *Payment*).

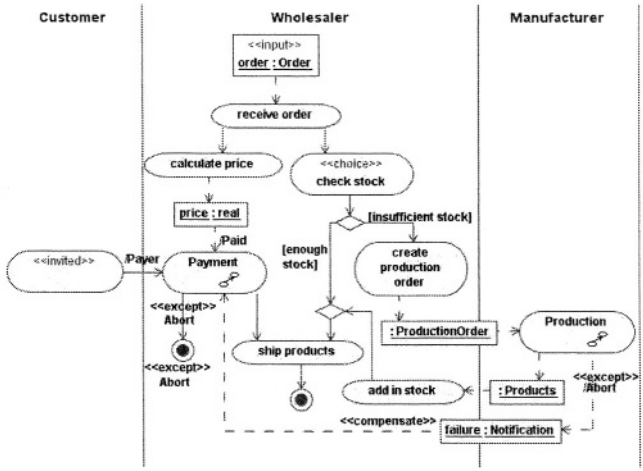


Fig. 7. OrderTreatment process

- If the name of the role is the same in the enclosing process as in the nested process, and there is no edge to the sub-activity node labeled with the role's name, then it is considered that the role of the enclosing process takes the same role in the nested process. This in order to have clearer diagrams.

Providing inputs to a nested process is modeled by object nodes having an edge to the sub-activity node (object *price* for the process *Payment* in Fig. 7). However, if the process has several inputs having the same type, there is an ambiguity to know which incoming object corresponds to which input. To remove this ambiguity, the edge from an object to the sub-activity may have the stereotype «input» and be labeled with the name of the corresponding nested process' input.

After the nested process was terminated, the execution flow follows the outgoing edges corresponding to the outcome: edges having the stereotype «except» and labeled by the name of the signaled exception or edges without the stereotype «except» for a normal outcome. Such edges go into object nodes if the exception has a parameter or if the nested process produces outputs (in Fig. 7, *Production* may have a normal outcome producing *Products* or signal an exception *Abort* with a parameter of type *Notification*). To remove the ambiguity when several outputs have the same type, the edges may have the stereotype «output» and be labeled with the name of the nested process' outputs.

Undone the effects of a nested process is modeled by an edge with the stereotype «compensate» going to the nested process' sub-activity node (in Fig. 7 the edge from the object *failure* to the sub-activity *Payment* indicates that if the nested process *Production* fails the effects of the nested process *Payment* must be undone). If the nested process is running, it interrupts its activities, undoes all its effects and signals the *Abort* exception. If the nested process is already terminated, there are two possibilities:

- It it succeed, a compensating nested process (modeled by an activity diagram having the stereotype «compensate») is used to obliterate its effects.
- If it signaled the *Abort* exception, nothing is done.

Then the execution flow continues on the outgoing edge corresponding to the *Abort* exception.

3.8 Internal Exceptions

As a CAA's role, a role of a business process may raise an internal exception by using an action node with stereotype «raise» and named by the exception to raise. Such a node must not have outgoing edges. For resolving concurrent exceptions into a single exception to raise, an exception graph may be specified within a swimlane having the stereotype «ExceptionResolution». Such a swimlane contains an oriented graph, where action nodes represent exceptions, as shown in Fig. 8. A role may have a handler for each resolved exception, which is a sub-set of the role's activities starting from an action node with the stereotype «handler» and named by the exception's name. For a parameterized exception, the action «raise» has an incoming edge from an object node and the action «handler» has an outgoing edge to an object node representing the parameter.

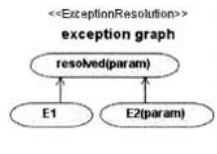


Fig. 8. Exception graph

4 Validating and Verifying Business Process Models

Our work is part of the E-fficient project, which aims at providing the business experts with a tool set for the modeling and the validation of e-business transactions. However, we believe that our approach may be integrated into other methods involving business process modeling and validation. The first phase of an E-fficient project consists in the identification by business analysts of the need for e-business transactions. These needs arise from business opportunities that bring some added value to one or more organizations. These needs are represented within UML use case diagrams together with a class diagram representing the structure and the relations of the entities involved in the e-business transactions. Once the need of business processes has been clearly identified, their behavior is specified using our notation.

The modeling phase is followed by a validation phase by business analysts through animation of the business process model in a workflow environment. This animation is performed using a translation of the activity diagrams into a XPDL file [16] and class diagrams into XML Schemas [17]. These files are then used by a workflow engine, which allows business experts to cooperatively validate the business process model. Each expert plays one or more roles in the processes and participates in the execution of the processes by receiving messages and sending answers. Thus, business experts validate the processes by playing different possible scenarios. Validating a specification by animating it, is widely recognized to be a useful technique for business experts. But the successful application of this technique depends on the relevance of the chosen scenarios.

The validation by animation allows detecting errors and inconsistencies, but it cannot prove their absences. We suggest to complement this technique with an

approach based on formal proofs that allows verifying a property for all possible scenarios. The properties to verify may be exhibited from errors discovered by business experts during the validation phase. [18] identifies four categories of properties to verify:

- Structural properties, which allow checking that diagrams are well formed according to the rules presented in section 3. This can be done by using the model checking tool USE to verify class and activity diagrams. The properties are specified with the OCL language.
- Static properties, which are related to a single state of the process.
- Dynamic properties, which are related to two or more states of the process. For verifying such properties together with static properties, we investigate the relevance of the use of COOPN. Actually, we perform an automatic transformation of business processes into a COALA program, providing its specification in COOPN language. The COOPN language is based on Petri nets and is provided with a prototyping tool for simulating the specification [19].
- Real time properties, which aim at evaluating the process in term of time response. Such properties are very useful in business processes since these processes are often time-critical (for example, a payment deadline or a maximum execution time for an activity or a process). [20] presents an approach for specifying real time constraints on an UML activity diagram by specializing the UML profile RT-UML [21], and for specifying real time properties by using a pseudo-english language translated into TCTL formulas [22]. This approach transforms an activity diagram into a timed automaton and use KRONOS tool for verifying properties expressed with TCTL formulas.

5 Conclusion

In this paper, we have presented an approach for modeling business processes using UML and CAA. We have shown how business analysts can develop business processes by using our extension to UML: FTT-UML. We have also shown that models done with our UML notation are specifically designed to be transformed into COALA models, which can then be verified with formal tools thanks to COALA's semantics based on the COOPN/2 formal specification language. CAA brings concepts for modeling cooperative and competitive concurrent processes with fault-tolerance, while UML provides a well-adapted syntax and interoperability through the standardized XMI format. We have defined an UML profile specializing activity diagrams for the business processes' behavior specification and class diagram for describing data structures.

The modeled business processes are validated by business experts through animation in a workflow environment and automatically transformed into a COALA program for future verification of static and dynamic properties.

Next step is to use COOPN/2 tools [23] to perform these verifications. The integration of our notation with [18] and [20] is also in progress in order to verify the four property categories: structural, static, dynamic and real-time properties. In [18] the verification of structural properties is based on the formal approach promoted by

USE tool; in [20] KRONOS tool is used to verify temporal properties specified in pseudo-english.

BPMN [24] (Business Process Modeling Notation) has been proposed by BPMI and allows the generation of executable BPEL4WS [25]. This notation is very close to UML activity diagrams, and we believe that a UML profile provide the same capabilities. Moreover, BPMI and OMG plan to integrate the two notations in the future. [26] proposes a notation based on UML activity diagram for workflow modeling supported by a verification tool, but nested processes and exception handling are not integrated in this work. Our approach brings transactional and fault-tolerance capabilities through the CAA concepts, and aims at allowing validation and automatic verification.

References

[1] OMG, *Unified Modeling Language Specification Version 1.5,* 2003

[2] H.-E. Eriksson, M. Penker, *Business Modeling with UML: Business Patterns at Work,* John Wiley & Sons, 2000

[3] R. Eshuis, *Semantics and Verification of UML Activity Diagrams for Workflow Modelling,* Ph.D. Thesis, University of Twente, 2002

[4] M. Dumas, A.H. Hofstede, *UML Activity Diagrams as a Workflow Specification Language,* 4th International Conference on the Unified Modeling Manguage, Modeling Languages, Concepts, and Tools, vol.2185, Toronto, Canada, Springer-Verlag, pp.76-90, 2001

[5] J. Xu, B. Randell, A. Romanovsky, C.M. Rubira et al. , *Fault Tolerance in Concurrent Object-Oriented Software through Coordinated Error Recovery,* Proceedings of the 25th International Symposium on Fault-Tolerant Computing, Pasadena, IEEE Computer Society, pp.499-508, 1995

[6] B. Randell, A. Romanovsky, R. Stroud, A. Zorzo, *Coordinated Atomic Actions: from Concept to Implementation,* Technical Report n°TR 595, Department of Computing, University of Newcastle upon Tyne, 1997

[7] P. Bernstein, V. Hadzilakos, N. Goodman, *Concurrency Control and Recovery in Database Systems,* Addison Wesley, 1987

[8] K. Ramamritham, P.K. Chrysanthis, *Advances in Concurrency Control and Transaction Processing,* Executive Briefing Serie, IEEE Computer Society Press, 1997

[9] J. Vachon, *CO ALA: A Design Language for Reliable Distributed Systems,* Ph.D. Thesis, Swiss Federal Institute of Technology, Lausanne, Switzerland, #2302, 2000

[10] A. Zorzo, R. Stroud, *An Object-Oriented Framework for Dependable Multiparty Interactions,* OOPSLA-99, pp.435-446, 1999

[11] B. Randell, *System Structure for Software Fault Tolerance,* IEEE Transactions on Software Engineering, vol.SE-1, pp.220-232, 1975

[12] B. Randell, J. Xu, *The Evolution of the Recovery Block Concept,* Software Fault Tolerance, pp. 1-21, 1995

[13] J. Moss, *Nested transactions: an approach to reliable distributed computing,* Massachusetts Institute of Technology, 1981

[14] R.H. Campbell, B. Randell, *Error recovery in asynchronous systems,* IEEE Transactions on Software Engineering, vol. 12, IEEE Press, pp.811-826, 1986

[15] O. Biberstein, D. Buchs, N. Guelfi, *Object-Oriented Nets with Algebraic Specifications: The CO-OPN/2 Formalism,* Advances in Petri Nets on Object-Orientation, Lecture Notes in Computer Science, Springer-Verlag, 2001

[16] WfMC, *XML Processing Description Language,* http://www.wfmc.org/standards/docs/TC-1025_10_xpdl_102502.pdf, 2002
[17] W3C, *XML Schema Type Formats,* http://www.w3.org/TR/xmlschema-2/,
[18] N. Guelfi, A. Mammar, B. Ries, *A Formal Approach for the Specification and the Verification of UML Structural Properties: Application to E-Business Domain,* Submitted to the 6th International Conference on Formal Engineering Methods (ICEFM), Seattle, WA, USA, 2004
[19] S. Chachkov, D. Buchs, *From Formal Specifications to Ready-to-Use Software Components: The Concurrent Object Oriented Petri Net Approach,* International Conference on Application of Concurrency to System Design, Newcastle, IEEE Press, pp.99-110,2001
[20] R. Annonier, N. Guelfi, A. Mammar, *Verification of Real-Time e-Business Transactions using RT-UML and KRONOS,* Submitted to, 2004
[21] OMG, *UML Profile for Schedulability, Performance and Time,* Specification OMG, 2003
[22] R.A, C.C, D.L. Dill, *Model-Checking in Dense Real-Time,* Information and Computation 104(1), pp.2-34, 1993
[23] C. Péraire, S. Barbey, D. Buchs, *Test Selection for Object-Oriented Software Based on Formal Specifications,* IFIP Working Conference on Programming Concepts and Methods (PROCOMET'98), Shelter Island, New York, USA, pp.385-403, 1998
[24] BPMI, *Business Process Modeling Notation (BPMN) 1.0,* BPMI.org, 2004
[25] Microsoft, IBM, Siebel, BEA, SAP, *Business Process Execution Language for Web Services,* 2003
[26] R. Eshuis, R. Wieringa, *Tool Support for Verifying UML Activity Diagrams,* vol.30, IEEE Press, pp.437-447, 2004

Cross-Enterprise Business Process Management Architecture – Methods and Tools for Flexible Collaboration

Sven Zang, Anja Hofer, and Otmar Adam

Institute for Information Systems (IWi) at the German Research Center
for Artificial Intelligence, Saarbruecken, Germany
{Zang, Hofer, Adam}@iwi.uni-sb.de
http://www.iwi.uni-sb.de

Abstract. New forms of cooperation like collaborative business scenarios require a deep but flexible integration of enterprises. To manage cross-organizational business processes existing concepts for business process management need to be adapted and extended. In this paper an architecture is presented, that shows how cross-enterprise processes can be planned, implemented and controlled. The architecture is based on the differentiation of global knowledge within the network and local knowledge of each participating company. Another important part of it is the process lifecycle model that serves as a guideline for the process-oriented setting-up and operation of cooperations. By the use of graphic representations of BPM-models and intuitive metaphor based model-generation and -visualization tools a broad acceptance for the inter-organizational BPM-effort within the network can be achieved.

1 Innovation Through Collaborative Business

The growing importance of cooperation is a result of globalization in combination with the disappearance of political borders and above all technological advances caused mainly by the Internet [1], [2]. Thus, enterprises have to react to the raised innovation pressure and facilitate flexible collaboration on a global scale by aligning their business processes.

The borderless enterprise has been the subject of scientific discussion for years [3], [4] and the collaborative production of goods and services has been established as a crucial factor in the consciousness of economic entities. The opening of the organizations' borders is no longer regarded as a necessary evil, but rather as a chance with strategic importance. [3]

Current approaches that address solutions to specific problems of dynamically interacting organizations are summarized under the term **"Business Integration";** the field of investigation is referred to as **"Collaborative Business (C-Business)"** [5]. C-Business describes the Internet-based interlinked collaboration of all participants in an added-value network – from the raw material supplier to the end-consumer [6]. It allows a comprehensive information exchange not only between employees but also between departments and even between enterprises and encourages creative coopera-

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 483–494, 2004.
© Springer-Verlag Berlin Heidelberg 2004

tions at all levels. As first case-studies show, the increase in added-value is out of proportion to the amount of participants in the network. Unlike former concepts, as e.g. E-Procurement, which focused only on small parts of the value chain, C-Business incorporates all stages of added value [7].

While the technological implementation [8] on the one hand and the lifecycle of cooperation [9] on the other hand have already been intensively researched, too little consideration is given to the interconnecting business management concepts. A rethinking from the pure technology-driven implementation or profit-driven business model discussion to an integrated view that spans from the conceptual level to the system blueprint is needed. From a conceptual point of view business processes have proven to be the ideal design item in conjunction with the use of graphical methods. These models can then be transformed into Information and Communication Technology (ICT) -based specifications. With the use of open, standardized technologies, such as web services, they enable Business Process Automation, i.e. the automatic negotiation of process interfaces.

For a detailed and systematic analysis and redesign of interorganizational processes, enterprises need an architecture that offers a set of integrated methods from the business concept level up to the implementation into ICT-systems. The appropriate graphic representation of these contents and user-friendly, intuitive tools that ensure the flawless connection of the different levels are of great importance in order to support the exchange of ideas and the reconciliation of interests between the different recipients within the network.

Finding consensus and decision-making between the different stakeholders is considerably facilitated by a general methodology; it encompasses not only the common conception of a collaborative business process but also the system-side implementation in an existing IT application landscape. Besides, more and more work-flow-aware application systems are used in mission-critical environments, e.g. as an extension of ERP-systems. Thus the main challenge concerning technology is the open implementation and interoperability of these process-sensitive applications in order to integrate preceding and following process steps dynamically [10]. To achieve a truly flexible integration and interoperability, standards like BPML [11], ebXML [12] or RosettaNet [13] have to be consolidated within the architecture to ensure a common procedure within the business network.

For these purposes a proposal for a Cross-enterprise Business Process Management Architecture is developed in this paper. Furthermore, collaborative modeling tools are presented to demonstrate how theory can be put into practice.

2 Architecture

Compared to traditional business processes, the complexity of interorganizational processes has risen considerably as a result of the numerous possibilities of interaction as well as the strategic, structural and cultural differences between the partners [14]. The allocation of performances and resources of the business partners, the determination of responsibilities for material and financial exchange relationships as well as the information and data exchange over interfaces have to be planned, arranged and “lived” together. Thus the demands on Business Process Management (BPM) increase.

Existing BPM methods and phase models are used as a foundation in the architecture presented here, which had to be adapted to the specifications of collaborative scenarios. Especially because of its completeness of vision and its proven practicability, both in the scientific and the economic context, the **"ARIS House"** [15] is accepted as a generic framework for business process management and serves as a basis for further considerations. The ARIS House describes a business process, assigning equal importance to the questions of organization, functionality and the required documentation. First, it isolates these questions for separate treatment, in order to reduce the complexity of the field of description, but then all the relationships are restored using the Control View introduced for this purpose.

The **Cross-enterprise Business Process Management Architecture** is presented here in a three-tier framework that is connected through control loops, following the concept of business process excellence of Scheer [16], which consists of a model to track a complete lifecycle model of business process management, including modeling, real-time control and monitoring of business processes. The first layer focuses on the **collaboration strategy.** In the centre of the second layer, **C-Business Process Engineering,** there are design, optimization and controlling of both enterprise spanning and internal processes. The third layer, **C-Business Execution,** deals with the (operational) implementation of business processes in value-added networks as well as their support through information and communication technologies. The structure of the layer model is clarified in Figure 1.

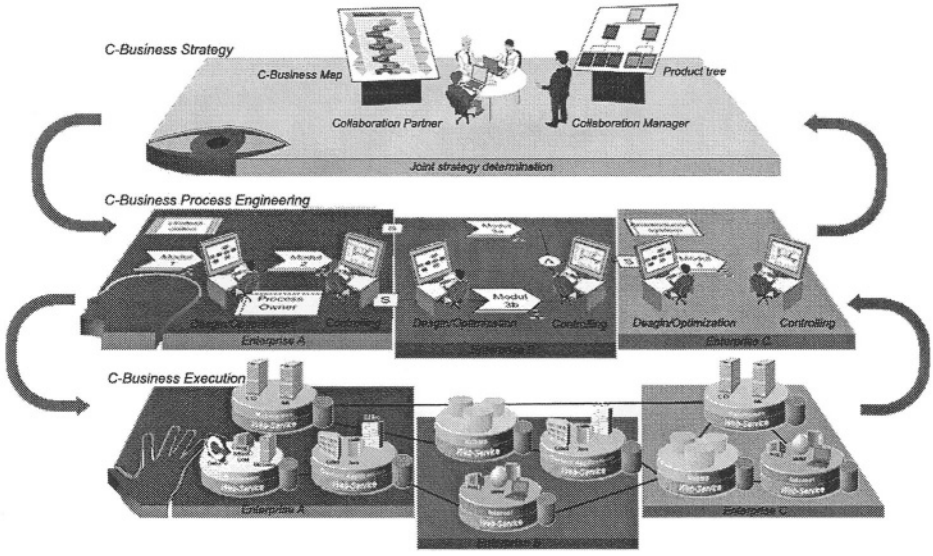


Fig. 1. Architecture for Collaborative Business Process Management

2.1 Views on Business Process Models

As described above, the framework is based on the ARIS House and divides it into a vertical axis of global knowledge of all collaboration partners and a horizontal axis of local knowledge of the single participants (cf. Fig. 2). The organisation view and the output view are global knowledge because a goal-oriented collaboration is impossible without them.

At the time the interaction occurs between two partners, local knowledge is shared (bilaterally) between the partners, i.e. additional information, like data structures and semantics, are exchanged. Updates of the local knowledge do not influence the network as network knowledge has to be available for all partners. This information is stored in the description of interfaces between the process modules of the partners (cf. section 2.3). Changes in the global network knowledge and as a consequence changes in the output and organization view have to be accessible to all partners immediately, for example if a company leaves the network or if a product or service is no longer available within the network.

Global and local knowledge merge gradually in the step-by-step development of C-Business process engineering. Following the distinction between global and local knowledge, a language is needed for the exchange of these knowledge fragments. Because the necessary detail functions and data schemes of the respective enterprise are determined in the data and the function view, these are treated from a micro perspective. They are characterized by an intensive internal interdependence, whereas externally a standardized encapsulation has to be provided. Interfaces of the data and function views to other network participants become visible in the process view in form of attribute correlations to process modules and concern the technological field of the cooperation during the realisation much more intensely than the conceptual one.

This technique enables the generation of **public** (visible to network partners) and **private** (enterprise-internal) **views** and levels of detail for management, process owner and IT-experts out of a C-Business model.

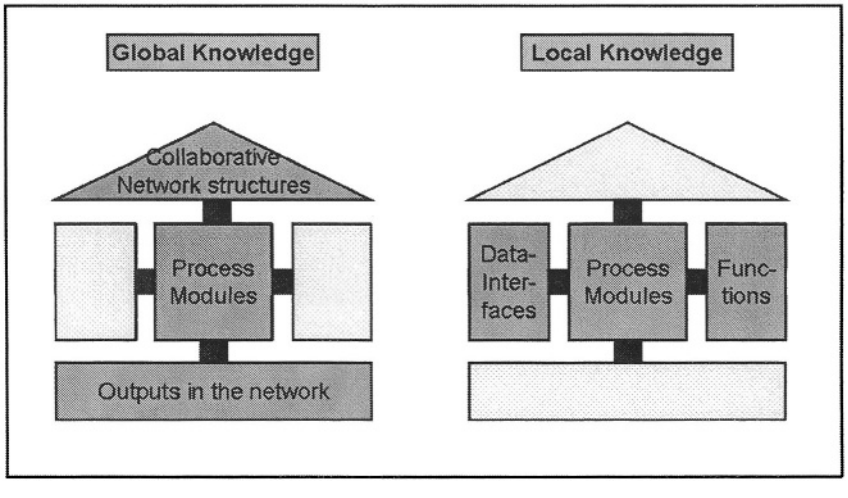


Fig. 2. Global and local knowledge in value-added networks

2.2 C-Business Strategy

Enterprise spanning business processes are not planned in detail at the strategic level, but are designed as concentrated, high-level process modules. Thus, they combine the public knowledge about the collaborative processes that is shared by all participants. C-Business scenario-diagrams that are used e. g. by SAP Ltd. for the description of my-SAP.com collaboration scenarios, aim at the representation of the cooperation of different enterprises and participants by means of an easily understandable method and the documentation of the value-added potentials resulting from it [17]. The responsibility for each process step, indicated by swimlanes, is of central importance to the determination of the scenario. This method is integrated into the ARIS concept and combined with methods of (classical) business process and data modeling used at the C-Business Process Engineering layer.

The question of core competences in the enterprises is directly associated with the question which processes remain in the enterprise and which are supposed to be assigned to partner enterprises or collaboratively operated [18].

2.3 C-Business Process Engineering

On this layer each partner considers their part in the inter-enterprise process. Each party models its own internal processes. The event-driven process chain (EPC) [15] is used for the design of the process flow within an enterprise (**local view**). A possibility to reduce complexity and to focus on special aspects is the use of different views like data, organizational, function or output view.

The **public view** on the collaborative process is generated in order to manage the common processes and to reduce the complexity of integrating the participating organizational units into one virtual unit. In doing so it is important that the partners provide access to all relevant information described as global knowledge beforehand and at the same are able to hide their business secrets. Both aims are achieved by enhancing the EPC with a new construct, the **process module** [19]. It serves as an abstraction for more detailed sub-processes that contain the local knowledge and thus encapsulates crucial process information. Additionally the object-type **interface**, which is marked by the letter "I" is introduced. The interface contains all relevant organizational and output information in order to connect internal process modules to external ones and vice versa.

The view concept presented in section 2.1 allows generating different views on the collaborative process depending on the degree of collaboration. In some cases closely interlinked collaboration partners allow each other inspection of their internal EPC models or parts thereof. By consolidating the processes at their interfaces a detailed collaborative process that can be used for further optimization and implementation emerges. More often the partners expose only parts of their individual processes to the partners because of the reasons above-mentioned. Thus, they grant access to their process knowledge at the process module level. A high-level public view of the collaborative process consisting of process modules and their interfaces is created. The information whether the detailed EPC or the high-level process module should be disclosed in a certain public view is stored as an attribute in each process module object.

The collaboration partners have to continuously compare the result of the implementation with their goals and adjust deviations. Hitherto the management has obtained its knowledge about the company's success from figures of the past, e.g. cash-flow, trading volume or profit made. The causes for fluctuations, requiring immediate counter measures, were not discernible. Until the problem was recognized, valuable time has elapsed. Therefore new measurement categories, which allow a reliable and contemporary evaluation of process efficiency, are required. The information needed cannot be extracted from the record and transaction oriented applications alone. **Key performance-indicators** must be defined based on records, logfiles, time stamps etc. These can be measured and analysed by means of intelligent tools [18]. The controlling function is a must when there is a high degree of uncertainty as with C-Business projects. The management can permanently control the implementation of the strategic collaboration configuration and promptly evaluate whether the expected added-value potentials have been reached.

2.3.1 Tool-Based, Intuitive Generation of Process Models

Discussing business processes means discussing abstract and mental formations as well. When the process-owner has to communicate the process to co-workers or even to partners within the network, the notional construct has to be visualized. This task is called **process visualization;** an effort to refine the mental representation of a process by using available media like images, texts or animations.

The main concern of process visualization is to achieve a common understanding of collaborative processes among all persons involved in the business process. Additionally, in collaborative environments the use of views as described above is supported effectively by a visualization system. An innovative approach to this is to record the process while reenacting it. This can be achieved with the aid of tools, e.g. the INTERACTIVE Process Modeler^{VR} from Interactive Software Solutions. The tool provides an intuitive Internet-based communication platform to record business processes interactively and in a decentralized way. In particular functionally responsible employees describe the business processes by playing them through within a virtual environment (cf. Fig. 3).

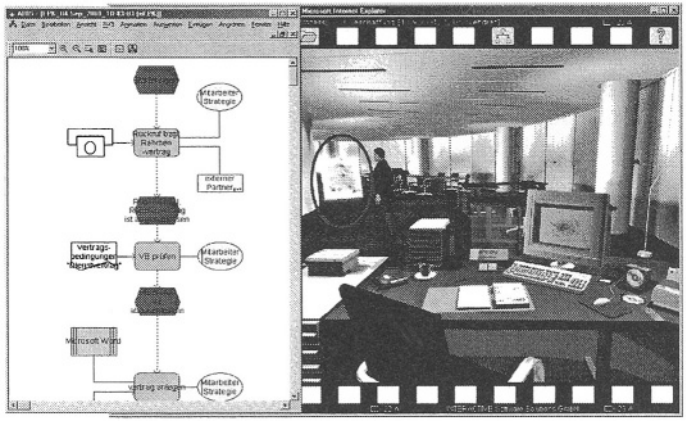


Fig. 3. Automatic generation of a process model

The recording of the employee's activities in the virtual working environment is transformed into a semi-formal process models needed for the analysis and evaluation. The automatically provided process models are based on the method EPC and can be handed over to a process modeling repository, for example the one of the ARIS Toolset, to process the models.

By doing so information loss due to communication errors between knowledge carriers and method experts is reduced to a minimum. In addition, modeling errors can be recognized and eliminated easier and faster on a much broader basis and errors resulting from formalisation of the surveyed process by method experts are avoided. Furthermore, expensive method training courses and the use of modelling experts can be reduced to a large extent.

2.3.2 Tool-Based Communication of Process Models

Within the process execution phase the implementation of target processes, the evaluation of the existing processes and the permanent optimization of processes require an adequate visualization of the processes in order to establish a common understanding of the processes among all persons involved. Otherwise, specialized divisions' employees with knowledge of the operations often take a rather passive role. This holds especially true for interorganizational processes.

In order to achieve the demanded participation by employees in validation and discussion of the process concepts produced in reorganization projects the business processes are visualized close-to-reality. By the intuitive representation weak points and opportunities for improvements can be identified and used for the construction of optimized target processes.The new form of business process visualization can serve to reach a quality improvement of conventional semi-formal business process modeling methods.

When the optimization should go across corporate frontiers the challenge gets more demanding, because of the higher complexity, as described in section 2. The distributed modeling approach, combined with the use of close-to-reality metaphors can cause an immense boost for the success of business process management within distributed modeling environments.

2.4 C-Business Process Execution

Instead of closed systems that have been used so far, C-Business requires the integration of different applications. Component based architectures that are process-driven and rely on fully developed standards and interfaces can be seen as a state-of-the-art approach to overcome these problems [20]. The term "process-driven" emphasizes the importance of the process models created on the preliminary layer. On the execution layer these models are used e. g. for the orchestration of web services. Orchestration in this context describes the composition of business objects in a process flow. In detail it defines the complex interaction between business objects, including the business logic and execution order of the interactions and the exchange of messages between them.

3 Collaborative Business Process Management Lifecycle

The lifecycle -model presented in this section serves as a manual for the process-oriented setting-up and operation of cooperations. Using a consistent phase model and standardized modeling methods increases transparency and structuring of cooperations and creates a basis for communication between participants, including management that lays down strategies, process-owners in the departments and IT-experts that integrate the different application systems.

Despite the increased complexity of network processes in comparison to internal processes, those involved have to adapt to constantly occurring changes in a fast and flexible way. The model is a fusion of classic phase-models with lifecycle -models of virtual enterprises [21]. The resulting dynamic model is consistent with the structure-oriented Cross-enterprise Business Process Management Architecture (cf. section 2) and represents a cyclical approach.

Protecting internal know-how is of paramount importance to the network participants, even though the business process knowledge has to be used jointly. Following the view concept presented in paragraph 2.1, this implies that the lifecycle alternates between phases that focus on global and on local issues in order to reach a coherent solution (cf. Fig. 4).

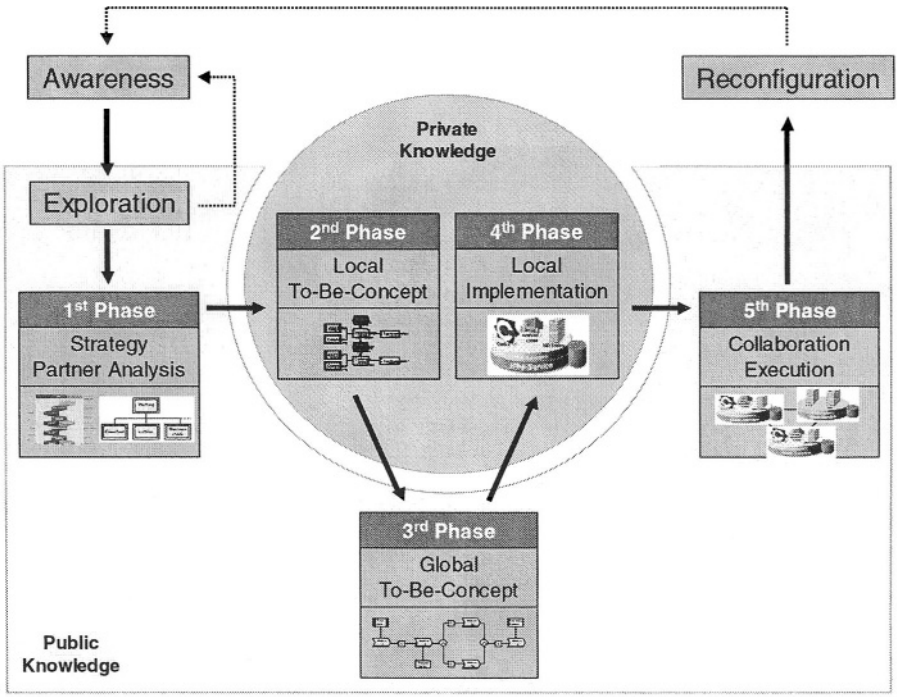


Fig. 4. Collaborative Business Process Management Lifecycle

3.1 Pre-phase and Reconfiguration

Prior to the use of the architecture is the **awareness** of one or more enterprises that they can profit by collaboration with complementary core competence partners. The decision if and with which enterprises a C-Business scenario should be implemented is taken by every single enterprise individually and rationally; for this reason it depends highly on the expected economical profit of the individual partner. In this model, it is assumed, that a set of potential network participants is given.

After conducting the cooperation the companies regroup or split and **reconfigurate** themselves. The lifecycle returns to its starting position "awareness".

3.2 Main-Phases

In the first phase **Strategy Partner Analysis** or formation phase, also referred to as initiation and agreement of the enterprise network, the collaboration partners are determined by the shared goals of the collaboration and the aspired win-win situation of all partners. The joint aims of the collaboration have to be defined as synthesis of the individual aims.

To facilitate the collaborative service or product delivery, graphical methods, like product models, are used in this stage for the determination of a common service or product bundle. They simplify and put the often implicit objectives into concrete terms. In addition to the characteristic features of a service or a product over its entire lifecycle , the organizational units participating in the production are contained in a product model [22]. By means of product trees enterprises can conceal detailed service descriptions in an internal view that puts special focus on the organizational aspects of the product offered by the partners. In an external view they just provide the information required for the configuration of the common service bundle in form of product bundle models [23].

Having completed the strategy finding, in the second phase, **Local To-Be-Concept,** an existing or a new (local) as-is model and the (global) to-be concepts are compared. According to predefined conditions about collective product creation, intra-organizational business processes can be derived. Each partner considers their part in the inter-enterprise process. Starting with process modelling and optimisation over process controlling up to implementation, the processes involved are aligned with the requirements of the collaborative scenario agreed on in the former phase.

In the third phase **Global To-Be-Concept** coordinated public parts are allocated over the network, establishing a collective to-be concept. Every partner is able to connect their own private model with every other public process model. Every partner gains their partial view of the collaborative process or in other words a virtual process chain of the whole collaboration is designed. The Business Process Modeling Language (BPML) can be considered as an appropriate exchange-language. Global knowledge is described in a public interface, which can be provided by a BPMN representation. The public processes as well as the message formats and contents can be formally defined by B2B protocols like RosettaNet or ebXML. Furthermore the semantic combination of models of the different partners is necessary. As long as ontology-based approaches don't reach a productive state this combination process is a manual action.

The integrated collaborative business process model enables all partners to configure their application systems locally in a fourth phase called **Local Implementation.** Reference systems for interfaces are provided by interface definitions of the collective to-be concept.

Now every partner is prepared for the execution of interactions within the collaborative framework. That is the transition to the fifth phase **Collaboration Execution.** Based on bilateral bases interacting information systems are able to communicate by using the standardized protocols and interfaces. The transactions are arranged and executed. The aim of this phase is to support collaboration through the appropriate use of ICT. That requires primarily the configuration of interfaces and the implementation of interorganizational workflows; at the same time the permanent monitoring and adaption of the collaboration, based on business ratio defined in the conception phase, must be assured. [6]

In order to automate inter-organizational processes the conceptual models are transformed into formal models that are used as configuration data for the orchestration of business objects. The applications of the partners have to communicate bilaterally to negotiate the interface specifications based on the formal models, defined in the repository. The local knowledge is generated by this negotiation for a certain situation. After this collaboration task has ended no updates of configuration changes etc. are reported to any other party except at the time when a new direct interaction occurs. In this context multiagent systems offer a solution to achieve an automated or at least semi-automated interface-configuration [24], [25].

With the use of XML the technological basis for interoperability has been established, the interoperability between the semantic business process definitions however is still missing. Efforts like BPMI's Business Process Modeling Language (BPML) promise standardization for the management of inter-organizational business processes that involve different applications, departments and business partners [11]. This standard, which is based on XML, complements existing B2B protocols like RosettaNet, BizTalk and ebXML. On the one hand BPML acts as an intermediary between business process modeling tools and IT. On the other hand BPML enables the interoperability between modeling tools. Besides, a wide acceptance of the Business Process Execution Language for Web Services (BPEL4WS) by BEA, IBM, and Microsoft as well as the newly finalized specification of the Web Services Choreography Interface (WSCI) mainly driven by BEA, Intalio, SAP and Sun show the importance of such standardization efforts for interoperability [26]. While BPML is seen as more conceptually-oriented, the latter two focus on the transformation into the system-level by orchestrating web services.

4 Towards an Intuitive Cross-Enterprise Business Process Management

The vision of this paper is to develop an architecture and a lifecycle model that provides a generic solution concept, which transfers business recommendations into ICT-solutions based on a holistic business process management approach and supported by intuitive tools. The foundation stone is laid as the varied methods presented in the paper clearly show.

The greatest demand for further research can be seen in the formulation of transformation methods especially in a methodologically sound transfer of process models into ICT-configurations. Another aspect that requires further research is the use of supporting tools that ease the task of exchanging process models between different enterprises and to distinguish between private and public knowledge. User-specific views on the business process models will enable new user groups to use BP-models, as the example of intuitive metaphor based process modeling points out. Moreover ICT can support actively business process management by checking, verifying or even automatically negotiating consistency and interoperability of models.

The described conceptual design of inter-enterprise business processes and the open research questions are currently elaborated in the research project "Architecture for Collaborative Scenarios (ArKoS)", funded by the German Federal Ministry of Education and Research (BMBF). As a proof of concept the presented architecture and methods will be implemented in a software prototype and will be used in real-life showcases.

References

1. Scheer, A.-W., Erbach, F., Thomas, O.: E-Business – Wer geht? Wer bleibt? Wer kommt?. In: Scheer, A.-W. (ed.): E-Business – Wer geht? Wer bleibt? Wer kommt?. 21. Saarbrücker Arbeitstagung 2000 für Industrie, Dienstleistung und Verwaltung. Physica-Verlag, Heidelberg (2000) 3-45
2. Naisbitt, J.: Megatrends 1986. Ten New Directions Transforming Our Lives. 6th ed., Warner Books, New York
3. Kanter, R. M.: Transcending Business Boundaries: 12,000 World Managers View Change. In: Harvard Business Review 69 (1991) 3, 151-164
4. Picot, A., Wigand, R., Reichwald, R.: Information, Organization and Management – Expanding Markets and Corporate Boundaries. Wiley, Chichester
5. Scheer, A.-W., Grieble, O., Hans, S., Zang, S.: Geschäftsprozessmanagement – The 2nd wave. In: Scheer, A.-W. (ed.): IM Information Management & Consulting 17 (2002) Sonderausgabe, 9-15, 10 et seq.
6. Scheer, A.-W., Grieble O., Zang, S.: Collaborative Business Management. In: Kersten, W. (ed.): E-Collaboration - Prozessoptimierung in der Wertschöpfungskette. Deutscher Universitäts-Verlag, Wiesbaden, 30 et seq.
7. Scheer, A.-W.; Feld, T., Zang, S.: Vitamin C für Unternehmen – Collaborative Business. In Küting, K., Noack, Chr. (eds.): Der große BWL-Führer. Die 50 wichtigsten Strategien und Instrumente zur Unternehmensführung. F.A.Z.-Institut, Frankfurt, pp. 123-129, 124 et seq.
8. Linthicum, D. S.: Enterprise Application Integration. 4. ed., Addison-Wesley, Boston et al., 2003
9. Liebhart, U. E.: Strategische Kooperationsnetzwerke: Entwicklung, Gestaltung und Steuerung. Dt. Univ.-Verl., Wiesbaden, 2002
10. zur Muehlen, M., Nickerson, J. V., Stohr, E. A.: Process Integration – From Workflow Models to Web Service Choreography. Istanbul 2003 Euro/Informs, http://www.workflow-research.de/Publications/YVAN.MIZU-ISECON(2003).pdf
11. Arkin, A.: Business Process Modeling Language. Working Draft.
12. Kotok, A.; Webber, D. R. R.: ebXML: The New Global Standard for Doing Business over the Internet. New Riders, Indianapolis, 2002
13. RosettaNet (Edt.): RosettaNet Implementation Framework: Core Specification. http://xml.coverpages.org/RNIF-Spec020000.pdf

14. Scheer, A.-W., Beinhauer, M., Habermann, F.: Integrierte E-Prozessmodellierung. In: Industrie Management 16 (2000) 3, 19-26, 20 et seqq.
15. Scheer, A.-W.: ARIS – Vom Geschäftsprozess zum Anwendungssystem. 4th ed. (2002) Springer, Berlin
16. Scheer, A.-W., Borowsky, R.: Supply Chain Management – die Antwort auf neue Logistikanforderungen. In: Kopfer, H., Bierwirth, C. (eds.): Logistik Management – Intelligente I+K Technologien. Springer, Berlin, 3-14
17. Hack, S.: Collaborative Business Scenarios – Wertschöpfung in der Internetökonomie. In: Scheer, A.-W. (ed.), E-Business – Wer geht? Wer bleibt? Wer kommt?. 21. Saarbrücker Arbeitstagung für Industrie, Dienstleistung und Verwaltung., Physica-Verlag, Heidelberg 85-100, 88 et seqq.
18. Jost, W., Scheer, A.-W.: Geschäftsprozessmanagement: Kernaufgabe einer jeden Unternehmensorganisation. In: Jost, W, Scheer, A.-W. (eds.): ARIS in der Praxis: Gestaltung, Implementierung und Optimierung von Geschäftsprozessen. Springer, Berlin, 33-44, 38, 42 et seqq.
19. Grieble, O., Klein, R., Scheer, A.-W.: Modellbasiertes Dienstleistungsmanagement. In: Scheer, A.-W. (ed.): Veröffentlichungen des Instituts für Wirtschaftsinformatik. No. 171, Saarbrücken, 22
20. McMichael, C.: Business process integration may eclipse EDI, EAI. In: HP Chronicle 17 (2003) 6, 1, 6
21. Mertens, P., Faisst, W.: Virtuelle Unternehmen – eine Organisationsstruktur für die Zukunft?. In: technologic & management 44 (1995), 61-68
22. Genderka, M.: Objektorientierte Methode zur Entwicklung von Produktmodellen als Basis Integrierter Ingenieursysteme. Shaker, Aachen, 13
23. Scheer, A.-W., Herrmann, K., Klein, R.: Modellgestütztes Service Engineering – Entwicklung und Design neuer Dienstleistungen. In: Bruhn, M., Stauss, B. (eds.): Dienstleistungsinnovationen: Dienstleistungsmanagement Jahrbuch 2004. Gabler, Wiesbaden, in print
24. Blake, M. B.: Coordinating multiple agents for workflow-oriented process orchestration. http://www.cs.georgetown.edu/~blakeb/pubs/blake_ISEB2003.pdf
25. Denti, E., Ricci, A., Rubino, R. : Integrating and orchestrating services upon a MAS coordination infrastructure.
http://www.ai.univie.ac.at/~paolo/conf/ ESAW03/preproc/E001 1.pdf
26. Shapiro, R.: A Comparison of XPDL, BPML and BPEL4WS: Cape Visions. Rough Draft, 1-17

From the Analysis of Cooperation Within Organizational Environments to the Design of Cooperative Information Systems: An Agent-Based Approach

Djamel Benmerzoug, Zizette Boufaida, and Mahmoud Boufaida

LIRE Laboratory, Computer Science Department,
Mentouri University of Constantine 25000, Algeria
Tel.:+213 31 63 90 10
benmerzougdj@yahoo.fr, boufaida@hotmail.com

Abstract. This paper presents an agent-oriented method and a generic agent-based architecture for the development of Cooperative Information Systems (CISs). The proposed method allows mastering the complexity of the cooperation processes and the difficulty of setting up effective CISs, by the analysis and the modelling of the cooperation in two levels of abstraction. The first level produces an understanding about organizational relationships and the rationales behind them. In this level, we use the **i*** framework, in which organizations and work processes are modeled in terms of dependency relationships among strategic actors. The second level focuses on the cooperation between software agents, where we use the AUML language for agents' system modelling. The proposed architecture implemented in our work using the JADE platform lays a foundation for a possible approach to solve CISs related problems.

1 Introduction

Agent technology has received a great deal of attention in the last few years and, as a result, the industry is beginning to get interested in using this technology to develop its own products. In spite of the different developed agent theories, languages, architectures and the successful agent-based applications, very little work for specifying (and applying) techniques to develop applications using agent technology has been done. The agent-oriented methodologies should cover all the phases of each agent-based application life cycle. This paper is about the exploitation of agent technology for the development of Cooperative Information Systems (CISs). More specifically, it presents an agent oriented method and a generic agent-based architecture for supporting the information systems cooperation.

Recently, the widespread use of information technology and the availability of networking services have enabled new types of applications, characterized by several geographically distributed interacting organizations. In particular, the term CISs is used to denote distributed information systems that are employed by users of different organizations under a common goal [5]. Indeed, several companies that depended on Information Systems (IS) -based applications have showed many interests to the integration of these techniques. However, the challenge in the CISs development will

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 495–506, 2004.
© Springer-Verlag Berlin Heidelberg 2004

shift towards the understanding of organizational environments and needs, and how to make decisions involving technical systems to address those needs and concerns. To this end, we need a clearer understanding of what it means for systems to be "cooperative". So, the notion of cooperation cannot be fully addressed unless the goals and desires of agents are explicitly modeled and argued about. Indeed, our method serves for mastering the complexity of the cooperation processes and the difficulty of setting up effective CISs. The proposed method consists of three phases of development that operate in two levels of abstraction. The *Analysis* phase produces an understanding about organizational relationships and the rationales behind them. The *System Design* and *Agent Design* phases focus on the system specification, according to the requirements resulting from the above phase. Both levels of modelling correspond to a progress in conceptualization, for which we will use different models and framework rising from IS and Multi-Agents Systems (MAS) design methods. Moreover, we will describe a generic agent-based architecture for supporting the cooperation of IS. We will also show the mapping that may exist between the basic concepts proposed by our method for agents specification and agents interactions and those provided by JADE platform for agents implementation.

The remainder of this paper is structured as follows. Section 2 provides an overview of some suggested approaches and architectures in CISs. Section 3 presents our point of view concerning the design of CISs, while the section 4 illustrates our methodology through a case study. The proposed generic agent-based architecture and the implementation phase are described respectively in sections 5 and 6. Finally, concluding remarks and future work directions are given in section 7.

2 Overview of CISs

It seems that the notion of CISs is used to mean different things: cooperative systems, federated, interoperable, multi-bases, etc. So the cooperation of the IS does not indicate the same thing according to the authors. In the context of our study, we consider the concept of CISs as an architectural framework that maintains consistency among a variety of computer-based systems, user groups, and organizational objectives as they all evolve over time [5]. Several research work within this field are developing quickly. We can classify them into two categories. Some work are oriented to technical aspects, by modelling the CISs using an agent approach. The others are oriented towards organizational aspects. Their aim is to address global organizational concerns, including organizational objectives and business goals, policies, regulations, and resulting workflow or project plans [1], [10]. Concerning the work about technical aspects, the proposed architectures consider CISs as a set of distributed information agents that work in a synergistic manner by exchanging information and expertise, coordinating their activities and negotiating how to solve parts of a common information-intensive problem [6], [11].

The major drawback of these approaches is the lack of the methodological aspect for the design of CISs. It is then frequent to certify that the developed systems put numerous problems. They do not always satisfy the users' needs, and are often badly adapted to the organization work. So, our contribution carries at the proposition of a methodological step in order to guide the design of CISs. The methodology adopts ideas from requirements engineering, where agents and goals have been used heavily

for requirements analysis [9], [10]. We are also adopting some ideas from agent-oriented method for the modelling of the agents system [4], [8].

3 Towards an Agent Oriented Method for Designing CISs

Increasingly, the IS development occurs in the context of existing systems and established organizational processes. Indeed, the cooperation among many "agents" (human or computer based) is crucial in order to attain these organizational goals. IS can be viewed as being cooperative to the extent that they contribute to the achievement of organizational goals. It means that after the development of the CISs, it is necessary to have an understanding of the organizational environment and goals, so that the resulting systems will work together with human agents to achieve overall goals [10]. According to these perspectives, we develop an agent-oriented method that represents a methodological framework in order to guide the design of CISs. It aims to analyse then to model the cooperative processes among a set of IS in measure to answer their needs. Indeed, the proposed method identifies three phases of design, which operate in two levels of abstraction: (Figure 1).

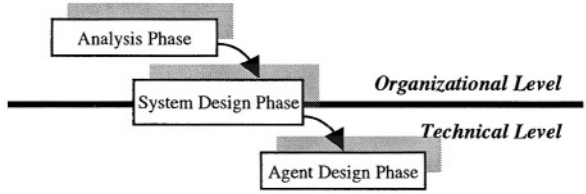


Fig. 1. Different phases of the proposed method

- ***Organizational** level:* This level corresponds to the modelling of the cooperation during the description of business frame. That is, the modelling of the cooperation among the various organizations in term of a set of actors, which depend on each other for goals to be achieved, tasks to be performed, and resources to be furnished. By clearly defining these dependencies, it is then possible to state the why, beside how and which are the actors entering in an operation requiring their cooperation. This level defines the system's global architecture in terms of subsystems, interconnected through data and control flows.
- ***Technical level:*** This level deals with the specification of the agents' micro level. Then, capabilities, plans, and the communication among agents are specified in detail.

The following section presents the various phases of the proposed method through a case study.

4 Presentation of the Proposed Method with a Case Study

This section briefly illustrates how the proposed method can be applied, through a case study by using the example presented in [8]. For reasons of brevity, we omit

some details, and aim instead to give a general flavour of the analysis and design. The particular application is providing customers with a quote for installing a network to deliver a particular type of telecommunications service. This activity involves the following departments: the *customer service division* (CSD), the *design division* (DD), the *legal division* (LD) and the various organizations, which provide the out-sourced service of *vetting customers* (VC). The process is initiated by a customer contacting the CSD with a set of requirements. In parallel to capturing the requirements, the CSD gets the customer vetted. If the customer fails the vetting procedure, the quote process terminates. Assuming the customer is satisfactory, their requirements are mapped against the service portfolio. If they can be met by a standard off-the-shelf item then an immediate quote can be offered. In the case of bespoke services, however, the process is more complex. DD starts to design a solution to satisfy the customer's requirements and whilst this is occurring LD checks the legality of the proposed service. If the desired service is illegal, the quote process terminates. Assuming the requested service is legal, the design will eventually be completed and costed. DD then informs CSD of the quote. CSD, in turn, informs the customer. The business process then terminates.

In that case, the customer can negotiate price and delivery date with the customer service. Besides and in order to enhance competition of the product, the DD Group should give the customer a good design, including good performance, good price, and short delivering time. Good performance means the product should sufficiently meet the need of the customer. The following section shows the use of the proposed method through the example presented above.

4.1 Analysis Phase

The objective of this phase is to obtain an understanding about organizational relationships and the rationales behind them. Indeed, we propose to use the notion of strategic actor based on the **i*** framework [9] to represent the cooperation issue of CISs.

In the **i*** framework, organizations are viewed as consisting of social actors who have freedom of action, but depend on each other for goals to be achieved, tasks to be performed, and resources to be furnished. The ***i**** framework is very useful to provide a higher level modelling method comparing to conventional modelling techniques such as data flow diagramming and object-oriented analysis. It includes the *strategic dependency (SD)* model for describing the network of relationships among actors, as well as the *strategic rationale (SR)* model *for* describing and supporting the reasoning that each actor has about its relationships with other actors [9].

This phase is constituted of three steps. The first step tries to identify the working groups, while the last two steps, each of them takes charge of the construction of a model. We begin with the construction of the SD model, and then we pass to the construction of the SR model.

4.1.1 Identification of the Working Groups

The working groups often correspond to the current structural division in the company when it exists. Every working group represents an independent entity, i.e., it performs a precise role, which requires certain amount of information and produces

others. In our example, we find four working groups: *CSD, DD, LD,* and the *VC service.* These groups correspond to organizational separations noticed in these departments. The CSD's behaviour has two distinct roles: one acting as an interface to the customer, and one overseeing the process inside the organization. The VC group is charged to examine customer's situations, while the LD group verifies the legality of the service asked by the customer. The DD group falls to design a solution to answer the requirements of the customer.

4.1.2 Construction of the SD Model

After the identification of the various working groups, we attempt in the following step to define the cooperation between these groups to attain global objective. The Figure 2 shows the SD model for the example of our case. The SD model is a graph where each node represents an *actor,* and each link between two actors indicates that one actor depends on the other for something in order that the former may attain some goal. In [9], the depending actor is called the *depender,* and the actor who is depended upon is called the *dependee.* The object around which the dependency relationship centers is called *dependum.* Authors distinguish among four types of dependencies, based on the type of the dependum: *hardgoal dependency, softgoal dependency, task dependency* and *resource dependency.* The softgoal having no clear-cut definition and/or criteria for deciding whether they are satisfied or not. Softgoal are typically used to model non-functional requirements. For simplicity, in the rest of the paper goal refer to hardgoal when there is no danger of confusion.

In our case, every working group is represented by an actor. In figure 2, the customer depends on the CSD to obtain a quote for installing a network to deliver a particular type of telecommunications service. It can also negotiate the price and the delivering time (with the same service).

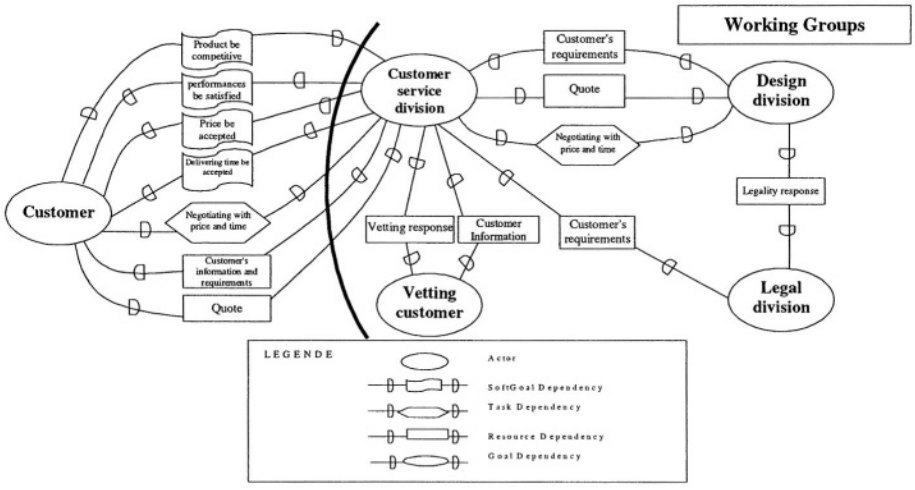


Fig. 2. The strategic dependency model

4.1.3 Construction of the SR Model

The SR model describes a more detailed level of an actor in order to model its internal intentional relationship and support to reason of each actor about its intentional relationships. The intentional elements, such as goals, tasks, resources and softgoals, appear in SR model not only as external dependencies, but also as internal elements arranged into a hierarchy of means-ends and task-decompositions relationships. A goal may be associated through means-ends with multiple, alternative ways to achieve it, which are usually represented as tasks. Task decomposition links hierarchically decompose task into four types: sub-tasks, subgoals, resources, and softgoals [9]. In figure 3, the goals and internal intentional relationships of the customer and the CSD group are elaborated.

The goal of the customer is to get a personalized quote and is accomplished by "Requirements of Service Be Decided". This task can be decomposed of three sub-tasks, such as "Performances Be Decided", "Price Be Decided", and "Deadline Be Decided". For the CSD, there are two ways possible to carry out the demand of the customer. If it is about a standard service, then an immediate quote can be offered. In the case of bespoke services, the CSD depends on the DD to conceive a new quote. DD, in turn, depends on the VD to checks the legality of the proposed service.

In the SR model, each possible way to accomplish a goal has different implications for the qualities of goals or softgoals. The contributions to softgoals can be positive or negative. In figure 3, the "Offer Standard Quote" goal is based on the existed designs, so the designing process will be quick and low price (positive effect). The "Offer New Quote" goal has an innovative designing method and may make sufficient positive effect to good performance, but it needs more time and high price (negative effect). So the SR model provides a systematic way to explore the possible alternative in the design process. This would be help for the cooperators to decide which way they should take and the effect of each way.

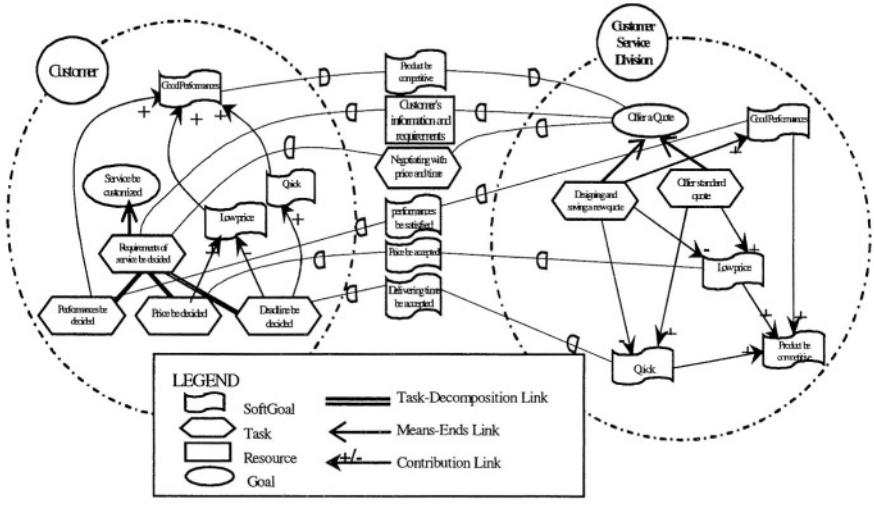


Fig. 3. The strategic rationale model

4.2 System Design Phase

In this phase, the developer begins the design of its system. He starts with a global design without entering in details. This phase is divided into two steps. The first step consists of defining a set of agent type, while the second step serves for identifying capacities and acquaintances of each agent.

4.2.1 The Actor-Agent Mapping

This step makes a *simple* mapping of system actors to a set of software agents. The proposed method allows associating a software agent for each working group. Indeed, in our example, the CSD's actor falls into the CustomerService agent, thus, the VC's, the LD's, and the DD's actors are covered respectively by the agents CustomerVetter, LegalAdvisor, and NetworkDesigner.

4.2.2 Identification of Capacities and Acquaintances of Each Agent

This step serves for identifying the capacities of each agent, and the links among them (acquaintances model).

Identification of Capacities of Each Agent

This step consists of the identification of the capacities needed by the agents to fulfill their goals and satisfy their dependencies. Following the approach described in [4], capacities can be easily identified by analysing the SR model. In particular, each dependency relationship can give place to one or more capacity trigged by external events. To give an intuitive idea of this process let's focus on a specific actor of the SR model, such as the CSD (which corresponds to the CustomerService agent), and we consider all the in-going and out-sorting dependencies, as shown in figure 3. Each dependency is mapped to a capacity. So, for instance, the dependency for the resource "Customer Requirements" calls the capacity "Get Customer Requirements", and so on.

Construction of the Acquaintances Model

According to [8], an agent acquaintance model (figure 4) is simply a graph, with the nodes corresponding to agent types loosened in the first stage of this phase, and arcs in the graph corresponding to communication pathways. Agent acquaintance models are directed graphs, and so an arc $a \rightarrow b$ indicates that a will send messages to b, but not necessarily that b will send messages to a. An acquaintance model may be derived in a straightforward way from the SR model.

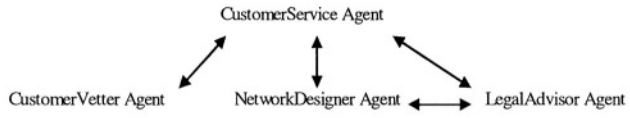


Fig. 4. The acquaintance model

4.3 Agent Design Phase

This phase allows better to detail initial design and to refine the system design. Like in [4], it is constituted of two steps, each of them taking charge of the construction of a model, the interactions model and the plans model. During this phase, we adopt a subset of the AUML (for Agent Unified Modelling Language) sequence diagrams proposed in [2] for specifying interactions among agents (interaction model) and the activity diagrams for representing agents plans (plans model).

4.3.1 Construction of the Interaction Model

In AUML sequence diagrams, agents correspond to objects, whose life-line is independent from the specific interaction to be modelled (in UML an object can be created or destroyed during the interaction); communication acts between agents correspond to asynchronous message arcs [2].

Figure 5 shows a simple part of the interaction model. In particular, the diagram models the interaction among the costumer and the CustomerService agent. The interaction starts with a request by the customer, and ends with the results presentation by the CustomerService agent to the customer.

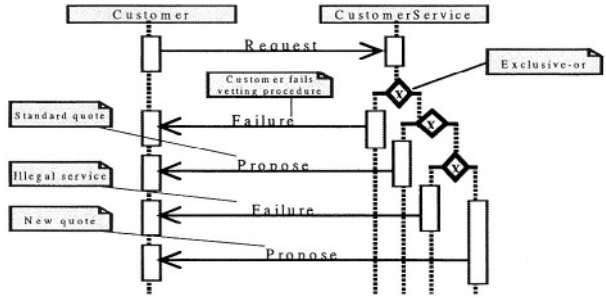


Fig. 5. AUML interaction diagram among the customer and the CustomerService agent

4.3.2 Construction of the Plans Model

The plans model consists of a set of plans, each being specified with an activity diagram. This one expresses how an agent has to behave to react to an event. The figure 6 presents the Evaluate_Vetting_Response diagram plan of the CustomerService

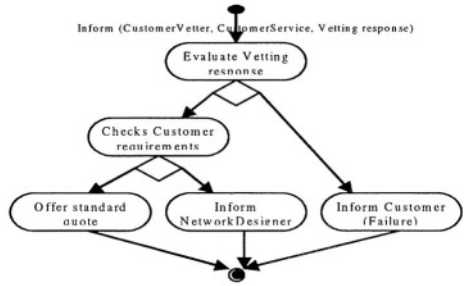


Fig. 6. The Evaluate_Vetting_Response Plan

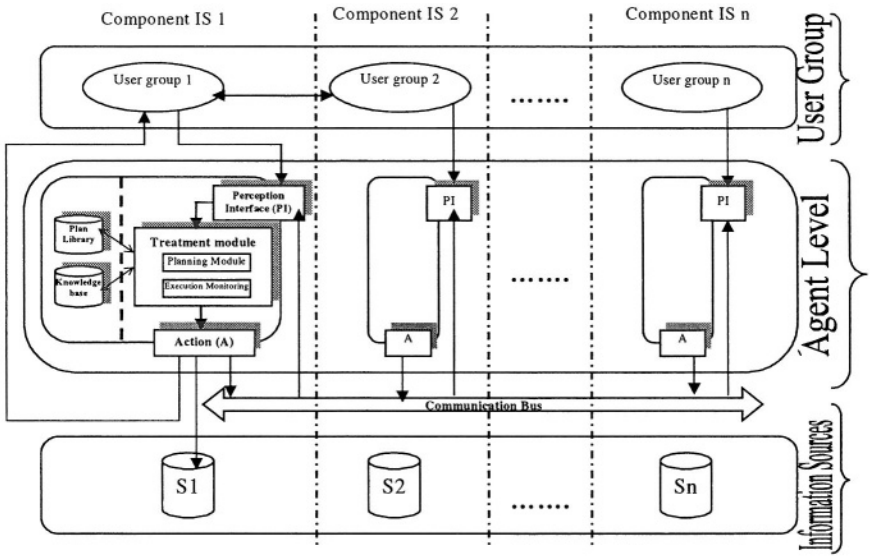


Fig. 7. A generic agent-based architecture

agent when it receives an event from the CustomerVetter agent. In this diagram ovals represent capabilities and arcs represent internal or external events.

In the following section of this paper, we describe the implementation phase of our framework by the proposition of a generic agent-based architecture, which can be implemented using the JADE platform.

5 A Generic Agent-Based Architecture

In this section, we describe the agent system architecture that supports inter-agent cooperation. In this work, the key general, characteristics of agents include social ability and pro-activity. The social ability is taken into account since an agent can interact with other ones. Agents are pro-active in the sense that they can exhibit a goal-directed behaviour by taking the initiative. They are also informational since they perform the role of managing, manipulating or collating information from many distributed sources. The conceptual architecture, shown in figure 7, lays a foundation for a possible approach to solving CISs related problems. In our architecture, we find three levels: *the user group level, information sources level* and *the agent one.*

5.1 User Group Level

The proposed architecture illustrates that each of the component IS interfaces to a software agent, which provides the means for working groups (working groups are identified in the analysis phase) to exchange meaningful messages. Each of these groups has its own local information source.

5.2 Information Sources Level

This level accommodates existing information, workflow and other computer-based systems. These systems are developed in terms of conventional technologies such as programming languages, database management and workflow systems. These systems were intended to serve local needs.

5.3 Agent Level

The choice of agent's model is made essentially according to the needs that are expressed for this agent. All the agents have a communications of high level and a decision-taking capacity. Indeed, each agent has a perception interface that assures environment or users' perception. The task of this interface is to listen to incoming messages and interprets it into a set of goals (see implementation phase for more detail). A treatment module allows determining the chain of protocols according to the role of the agent. Actions correspond to the sending messages towards other agents or towards the user. The treatment module holds two parts: the Planning Module and the Execution Monitoring. The first part takes as input a set of goals (the results of message interpretation by perception interface) and produces a plan that satisfies the goals. According to [7], the agent planning component is a simple plan retrieval mechanism for each goal (plans of each agent are identified in the agent design phase). The second one prepares, monitors, and completes each plan step. The agent's plan library contains skeletal plans and plan fragments that are indexed by goals and can be retrieved according to the current input parameters. The knowledge base represents the internal states of the agent and the representation of the other agents (acquaintances).

6 Some Implementation Aspects

The platform chosen for the implementation is Java Agent Development framework (JADE) [3]. JADE is a software development framework, fully implemented in Java, which aims at the development of multi-agent systems and applications that comply with FIPA standard. To achieve such a goal, JADE offers the following list of features to the agent programmer:

- Distributed agent platform. The agent platform can be distributed on several hosts, each one of them executes one Java Virtual Machine.
- FIPA-Compliant agent platform, which includes the Agent Management System, the Directory Facilitator and the Agent Communication Channel.
- Efficient transport of ACL messages between agents.

6.1 Mapping Architecture Concepts into JADE Code

Creating a JADE agent is as simple as defining a class extending the *Jade. core.Agent* class and implementing the *setup( )* method. The *setup( )* method is intended to include agent initializations. The actual job an agent has to do is presented as JADE

behaviours. Besides, the predefined behaviour `PerceptionInterface` is started by *setup()* as shown in the JADE code below.

```
import jade.core.*;
public class MyAgent extends Agent  {
  // Possible variables
  protected void setup {
      // to do: add necessary start-up code
         addBehaviour(new PerceptionInterface(this));
    }
  }
```

In our case, we propose to implement capabilities resulting from system design phase as *SimpleBehaviour* and plan resulting from agent design phase as *FSMBehaviour* class. The *FSMBehaviour* class is a *CompositeBehaviour* that executes its children (sub behaviour that are going to be used by *FSMBehaviours*) according to a Finite State Machine defined by the user.

Perception Interface (PI) presented in our generic architecture is derived from JADE class *CyclicBehaviour* and will therefore run continuously. If no messages have arrived, the behaviour will block and restart after a new message has arrived. If a message has arrived, the PI has to interpret this message into a set of goals, starts `PlanRetrival` behaviour and resumes waiting for incoming messages. The `PerceptionInterface` behaviour code is shown below.

```
import java.util.*;
import jade.core.*;
import jade.core.behaviour.*;
import jade.lang.acl.ACLMessage.*;
class PeceptionInterface extends CyclicBehaviour {
  public PeceptionInterface (Agent a) {super(a);}
  Vector goals = new Vector();
  public void action() {
    // wait for message
    ACLMessage received = myAgent.receive();
    if(ACLMessage == null) {block();}
    else {
      // message interpretation
      .....
      // Start PlanRetrieval Behaviour
      addBehaviour(new PlanRetrieval(this,goals));
         }
      }
    }
```

The `PlanRetrival` behaviour is derived from JADE class *SimpleBehaviour*. This behaviour takes as parameters a set of goals (the results of message interpretation by PI) and finds the adequate plan from the plans library.

7 Conclusion and Future Work

This article presents our point of view concerning the design of CISs, by the exploitation of agent technology. Indeed, agent concepts are used throughout requirements analysis, design, architecture, and implementation. The proposed method that represents a methodological framework, serves to master the complexity of the cooperation processes and the difficulty of setting up effective CISs. Our method of design identifies two levels of abstraction: an organizational level and a technical one. The first level aims to analyse then to model cooperative processes among various organizations, in term of strategic and rational relations. The technical level deals with the specification of the agents' micro level. On the other hand, our aim was to reveal the mapping that may exist between the basic concepts proposed by our method for agents specification and agents interactions and those provided by JADE for agents implementation. Our primary future work direction concerns research on technical ways for managing large and complex data that are distributed and heterogeneous in nature, and support cooperative use of legacy IS and carriers in our framework. Finally, further work needs to be done on the definition of semantics and syntax of data and knowledge conveyed among our agents.

References

1. Agostini, A., De Michells, D., Grasso, M.A., Patriarca, S.: Reengineering a Business Process with an Innovative Workflow Management System: a Case Study. COOCS'93-11/93/CA, USA, (1993) 154-165
2. Bauer, B., Müller, J., Odell, J.: Agent UML: A Formalism for Specifying Multiagent Interaction. Agent-Oriented Software Engineering, Springer, Berlin, (2001) 91-103
3. Bellifemine, F., Caire, G., Trucco, T., Rimassa, G.: Jade Programmer's Guide. JADE 3.0bl. Available at http://sharon.cselt.it/projects/jade/, (2003)
4. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A.: TROPOS: An Agent-Oriented Software Development Methodology. Accepted in JAAMAS, (2003)
5. Giorgio, M., Dubois, E., Jarke, M., Matthes, F., Mylopoulos, J., Papazoglou, M., Pohl, K., Schmidt, J., Woo, C., Yu, E.: Cooperative Information Systems: A Manifesto. CIS: Trends & Directions, Mike P. Papazoglou and Gunter Schlageter, Aacdemic-Press, (1997)
6. Papazoglou, P., Laufrman, C., Sellis, K.: An organizational framework for Cooperating Intelligent Information Systems" Int. journal of ICIS Vol. 1, N° 1 (1992) 169-202
7. Sycara, K., Zeng, D.: Coordination of multiple intelligent software agents. Ineternational journal of Cooperative Information Systems, (1996)
8. Wooldridge, M., Jennings, N., Kinny, D.: The GAIA Methodology for Agent-Oriented Analysis and Design. Journal of Autonomous Agents and MAS, (2000) 285-312
9. Yu, E.: Agent Orientation as a Modelling Paradigm. Wirtschaftsinformatic. 43(2), (2001) 123-132
10. Yu, E., Du Bois, P., Dubois, E., Mylopoulos, J.: From Organization Models to System Requirements A "Cooperating Agents" Approach. Proc third Int. Conf. Cooperative Information Systems (CoopIS95), Vienna, Austria, (1995)
11. Zarour, N., Boufaida, M., Seinturier, L.: Using information technologies for managing cooperative information agent-based systems. Proc in ICEIS'03, (2003) 430-437

Transforming UMM Business Collaboration Models to BPEL

Birgit Hofreiter and Christian Huemer

University of Vienna, Dept. of Computer Science, Liebiggasse 4, 1010 Vienna, Austria
Research Studios Austria, Digital MemoryEngineering, Thurngasse 8/20, 1090 Vienna
{birgit.hofreiter,christian.huemer}@univie.ac.at

Abstract. UN/CEFACT's Modeling Methodology (UMM) has been developed to analyze and design B2B business processes independent of the underlying exchange technology. It became the methodology of choice for developing ebXML business processes. Another technology for realizing B2B partnerships is Web Services. Currently, the business process execution languages (BPEL) seems to be the winner amongst the Web Services languages for orchestration and choreography. If Web Services is used as underlying exchange technology for B2B, the semantics of UMM business processes must be represented in BPEL. The goal of this paper is to verify whether BPEL is appropriate to capture UMM business collaborations or not. For this purpose we describe a transformation from UMM to BPEL.

1 Introduction

The OMG's Model Driven Architecture (MDA) is an effort to create applications by transforming models to executable software. MDA starts with the well-known and long established idea of separating the specification of the operation of a system from the details of the way that system uses the capabilities of its platform. The MDA Guide [9] identifies 4 very high level steps: (1) specifying a system independently of the platform that supports it, (2) specifying platforms, (3) choosing a particular platform for the system, and (4) transforming the system specification into one for a particular platform. The MDA approach should lead to a re-use of analysis & design artefacts as well as implementations. This should reduce the complexity of software development and result in lower costs. Furthermore, the approach of separating specification and implementation enables the portability of applications to existing and future technologies. Another goal of MDA is interoperability even across platforms. Strict methods guarantee that applications based on different technologies implement the same business logic.

In first place, the MDA approach focuses on the development of enterprise systems. However, we feel that there are no restrictions to follow a similar approach to interconnect systems of different enterprises in B2B e-Commerce. This is also in-line with the Open-edi reference model, which was developed in the electronic data interchange community and which became an ISO standard in 1997 [5]. Open-edi distinguishes a business operational view (BOV) and a functional service view (FSV). BOV related standards provide the tools for formal business description(s) of

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 507–519, 2004.
© Springer-Verlag Berlin Heidelberg 2004

the external behavior of organizations, as seen by other organizations, in view of achieving a business goal. These will address the rules for inter-organizational business processes, like operational conventions, agreements, or mutual obligations, as well as the semantics of business data exchanged. The FSV standards address the supporting services meeting the mechanistic needs of Open-edi. Consequently, FSV standards focus on the IT infrastructure for inter-organizational systems.

UN/CEFACT's Modeling Methodology (UMM) defines a development process for BOV standards on top of the UML [13]. It provides a UML profile for modeling B2B. In terms of the MDA, UMM fits into step (1) specifying a system independently of the platform. Candidate B2B platforms on the FSV and for MDA step (2) are typically implemented by the concepts of UN/EDIFACT, ebXML, and Web Services. In step (3) one has to select one of these standards for implementing the B2B system. The MDA requires a well-defined set of rules and techniques for mapping the platform independent model to the target technology in step (4). Thus, the definition of these rules and techniques is a critical task for the success of the MDA.

We are mainly interested in the choreography of business collaborations. Therefore, it is our goal to define the mapping from UMM business collaborations to the different choreography languages for B2B platforms. In other publications we concentrated on the mapping between UMM and ebXML choreography language BPSS [11]. A critical evaluation of these two standards is included in this conference proceedings [6]. Although UMM is not a mandatory part of ebXML, BPSS might be considered as "XML-ification" of parts of the UMM meta model. Owing to the close relationship between UMM and BPSS, the mapping rules are rather simple [4].

Currently, the adoption of Web Services standards by industry is rather quick. It seems that industry support for Web Services is much higher than for ebXML. Therefore, it is important to define a mapping between UMM and Web Services. The winner amongst the different choreography / orchestration languages seems to be the business process execution language (BPEL) [1], which combines WSFL [7] and XLANG [10]. This paper presents a first approach for a mapping from UMM version 12 [13] (that is based on UML 1.4) to BPEL 1.1. Before we go into the details of this transformation in Section 3, Section 2 briefly describes those UMM concepts that are relevant for the run-time systems.

2 UMM Concepts for Run-Time Systems

The UMM meta model consists of 4 views in order to describe business collaboration models. Due to space limitations we will not go into the details of each view. We concentrate on the business transaction view (BTV) which captures most of the concepts relevant for a run-time system: business transaction and business collaboration protocol. The interested reader is referred to the UMM Meta Model [12] and the UMM User Guide [13], and our overview article [3].

A *business process* is defined as an organized group of related activities that together create customer value [2]. A *business collaboration* is a special type of a *business process* that involves two or more partners. A *business collaboration* is about aligning the information systems of the business partners, i.e. keeping all relevant business objects (e.g. purchase orders, line items, etc.) in an aligned state. If a business partner recognizes an event that changes the state of a business object, it

initiates a communication to synchronize with the collaborating business partner. It follows that this communication is an atomic unit that leads to a synchronized state in both information systems. Hence, this special type of *business collaboration* is called a *business transaction.* It is the basic building block for more complex *business collaborations.* Although UMM is a top-down process, we start our explanation bottom-up from *business transactions* for educational purposes.

2.1 Business Transaction

The requirements of a business transaction are documented by a business transaction use case. Each business transaction use case results in exactly one business transaction that is represented by an activity graph. The activity graph of a business transaction is always built by exactly two business actions, a requesting business activity and a responding activity. Each business action is performed by exactly one authorized role executed by a business partner. The assignment of the business action to an authorized role is realized by the UML concept of swimlanes. The requesting business activity is assigned to the initiating role and the responding activity is assigned to the reacting role. The exchange of business information is shown by an object flow. One business action sets an information object flow state that is consumed by the other business action. An information object flow state refers to an information envelope exchanged between the business actions. Since we concentrate on the choreography in this paper, we do not further discuss the assembly of the content within the information envelope.

We distinguish two types of one-way transactions - information distribution and notification - as well as four types of two-way transactions - query/response, request/confirm, request/response, and commercial transaction. These six types cover all known legally binding interactions between two decision making applications as defined in Open-edi. The different types of business transaction patterns differ in the default values for the attributes that characterize *business actions: is authorization required, is non-repudiation required, time to perform, time to acknowledge receipt, and time to acknowledge acceptance.* The values for *is non-repudiation of receipt required* and for *retry count* are only defined for the *requesting business activity.* Most of these attributes are self-explanatory. An acknowledgment of receipt is usually sent after grammar validation, sequence validation, and schema validation. However, if the *is intelligible check required* flag is set *to false,* the acknowledgment is sent immediately after receipt without any validation. An acknowledgment of acceptance is sent after validating the content against additional rules to ensure that the content is processable by the target application. *Retry count* is the number of retries in case of time-outs.

Accordingly, a business transaction follows always a rather strict pattern. The activity graph of register customer depicted in Fig. 1 is a typical example of a two-way transaction. In case of a one-way transaction the responding object flow is omitted. A business expert provides the characteristic of register customer in a business transaction use case. The buyer starts a *request registration* activity. This activity creates a *registration request envelope* that triggers the *perform registration* activity of the seller. According to UMM business transaction semantics, *request registration* does not end after sending the envelope - it is still alive. The *perform registration*

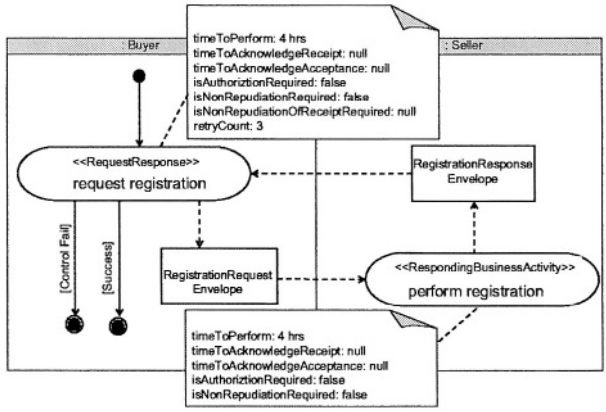


Fig. 1. Business Transaction "Register Customer"

activity outputs the *registration response envelope,* which is returned to the *request registration* activity. It becomes obvious that the activity graph of a business transaction shows only the exchange of business information in the corresponding envelopes, but does not show any business signals for acknowledgements. The need for acknowledgements is documented in the tagged values only. Register customer does not need any acknowledgements.

2.2 Business Collaboration Protocol

The business transaction is the basic building block for more complex business collaborations. Consequently, a business collaboration covers a number of business transactions. It is important that the business collaboration defines an execution order for the business transactions, i.e. a business collaboration choreography. Each *business collaboration protocol use case* specifies the requirements for the choreography that is defined by an activity graph called *business collaboration protocol.* Currently, all activities of the *business collaboration protocol* must be stereotyped as *business transaction activity.* A *business transaction activity* must be refined by the activity graph of a *business transaction.* A *business collaboration protocol* might describe the choreography of multi-party collaborations. Each *business transaction activity* is executed by exactly two parties. Since a *business collaboration protocol* describes the contractual obligations between business partners and since most contracts are bilateral, a *business collaboration protocol* usually defines a binary collaboration.

The transition from one *business transaction activity* to another is triggered by two types of events: the completion of the previous business transaction and (in addition) the availability of a business object in a certain state. In addition to *business transaction activities* a business collaboration protocol includes the pseudo states used in UML activity diagrams: *initial state, final state, decisions* and *merge,* as well as *fork* and *join*.

Fig. 2 shows the business collaboration protocol for a very simple order management between a buyer and a seller. It is an over-simplified example showing only

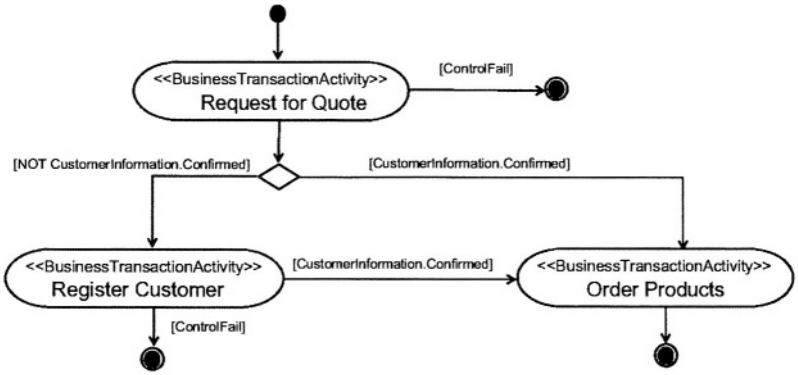


Fig. 2. Business Collaboration Protocol "Order Management"

those aspects that are needed to understand the main concepts of a business collaboration protocol and, later, its transformation to BPEL. Thus, we have omitted any complex exceptional cases. The business collaboration starts with a *request for quote.* After the quote is made, i.e. the transaction is completed, a registered buyer can initiate *order products.* If the buyer is not registered, *register customer* is required prior to *order products.* Control failures of *request for quote* and *register customer* terminate the business collaboration. The register customer business transaction activity is refined by the activity graph in Fig. 1. The refinements of the other activities follow later in Fig. 3 and Fig. 4.

3 Transformation to BPEL

UMM defines the business semantics of a (usually binary) business collaboration without considering any specific exchange technology. In this section we concentrate on mapping the choreography of the business collaboration from UMM to BPEL. As the acronym indicates, BPEL defines executable business processes within a company. However, BPEL can be used for specifying so-called business protocols. A business protocol specifies the potential sequencing of messages exchanged by one particular partner with its other partners to achieve a business goal [8]. According to this definition, a business protocol is able to capture the B2B choreography defined in UMM. BPEL uses the concept of an abstract process to realize a business protocol. Abstract processes use all the concepts of BPEL but data handling might be more abstract handling only protocol-relevant data[1].

3.1 Transformation to WSDL

A business process defines how to coordinate the interactions with a business partner. In a Web Service environment this means a BPEL process definition coordinates activities representing the execution of Web Services. An activity is either receiving a message from a business partner via one's own Web Service or invoking a partner's Web Service. The interface of each Web Service is described by the means of WSDL.

```
[1]<portType name="BuyerPortType">
[2]   <operation name="ReceiveResponseForRequestRegistration">
[3]     <input message="RegistrationResponseEnvelope"/>
[4]   </operation>
[5]   <operation name="ReceiveResponseForPlaceOrder">
[6]     <input message="PurchaseOrderResponseEnvelope"/>
[7]   </operation>
[8]   <operation name="AckReceipt">
[9]     <input message="BusinessSignalAckReceipt"/>
[10]  </operation>
[11]  <operation name="AckAcceptance">
[12]    <input message="BusinessSignalAckAcceptance"/>
[13]  </operation>
[14]  <operation name="ControlFailure">
[15]    <input message="BusinessSignalControlFailure"/>
[16]  </operation>
[17]</portType>
```

```
[18]<portType name="SellerPortType">
[19]  <operation name="calculateQuote">
[20]    <input message="RequestForQuoteEnvelope"/>
[21]    <output message="QuoteEnvelope"/>
[22]  </operation>
[23]  <operation name="performRegistration">
[24]    <input message="RegistrationRequestEnvelope"/>
[25]  </operation>
[26]  <operation name="processOrder">
[27]    <input message="PurchaseOrderEnvelope"/>
[28]    <output message="PurchaseOrderResponseEnvelope"/>
[29]  </operation>
[30]  <operation name="AckReceipt">
[31]    <input message="BusinessSignalAckReceipt"/>
[32]  </operation>
[33]  <operation name="AckAcceptance">
[34]    <input message="BusinessSignalAckAcceptance"/>
[35]  </operation>
[36]  <operation name="ControlFailure">
[37]    <input message="BusinessSignalControlFailure"/>
[38]  </operation>
[39]</portType>
```

It follows that a BPEL process references Web Services interfaces defined in a WSDL file.

A UMM model defines which type of services are expected from which type of business partner. Therefore, a WSDL file—including port types and operations as well as messages used as input/output to the operations—can be derived from a UMM model. Each business partner of the collaboration results in its own port type. In our order management example a buyer collaborates with a seller. Hence, we create a *buyer port type* (lines 1 to 17) and a *seller port type* (lines 18 to 39).

In a UMM business transaction each partner performs exactly one activity. This activity sends and/or receives message envelopes. In case of asynchronous messaging (c.f. subsections 3.4 register customer and 3.5 order product) each activity that receives a message becomes an operation of the corresponding partner's port type. The operation is called exactly as the acitivity. In case of an requesting activity we add *receive response for...* at the beginning of the name. In case of synchronous messaging (c.f. subsection 3.3 request quote) only the responding activity becomes an operation. Additionally, each port type needs operations to receive acknowledgment of receipt, acknowledgement of acceptance and control failure messages (c.f. subsection 3.5).

3.2 Transformation of Partner Links

A BPEL file defines the exchanges with different partners - in case of a binary collaboration it is exactly one partner. The partner is connected to the process by a partner link type. A partner link type defines a relationship between two services and their roles within. Each role refers to a port type of the WSDL file. A binary collaboration requires exactly one partner link type (lines 40 to 43). Our example defines the partner link type *buyer seller link type* (line 40). The *buyer* role (line 41) refers to the *buyer port type* (line 1) and the *seller* role (line 42) to the *seller port type* (line 18).

```
[40] <partnerLinkType name="BuyerSellerLinkType">
[41]   <role name="Buyer">   <portType name="BuyerPortType"/>   </role>
[42]   <role name="Seller">   <portType name="SellerPortType"/>   </role>
[43] </partnerLinkType>
```

A UMM model describes a business collaboration between business partners from an overall point of view. BPEL describes a business process always from the point of view of a particular partner. Consequently, a UMM model results in multiple BPEL processes, one for each business partner involved. Of course these BPEL processes must be compliant to each other. *Partner links* define the relationships of the process owner with its partners. In the process definition for the buyer the *partner link* is defined in lines 44 to 46. The partner link (line 45) refers to the *buyer seller link type* (line 40). Since the buyer is the owner of the process, the *myRole* attribute refers to the *buyer* role (line 41) of the partner link type and the *partnerRole* attribute to the *seller* role (line 42). The partner link in lines 47 to 49 defines the pendant for the process owned by the seller.

```
[44] <partnerLinks>
[45]    <partnerLink name="LinkToSeller"
                partnerLinkType="BuyerSellerLinkType"
                myRole="Buyer" partnerRole="Seller" />
[46] </partnerLinks>
```

```
[47] <partnerLinks>
[48]    <partnerLink name="LinkToBuyer"
                partnerLinkType="BuyerSellerLinkType"
                myRole="Seller" partnerRole="Buyer" />
[49]    </partnerLinks>
```

3.3 Transformation of Request for Quote (Synchronous)

In subsection 3.1 we already explained that the Web Services operations are derived from the requesting and responding activities of UMM business transactions. The basic activities in a BPEL process refer to these operations. The activity <*invoke*> is used to call an operation of a partner's port type and the activity <*receive*> for receiving messages from a partner via an operation of one's own port type. In addition, <*reply*> is used to specify a synchronous response following a <*receive*> activity.

In UMM, a business transaction is the basic building block for interactions between the business partners. Hence, we transfer each business transaction into a set of BPEL activities, that will be a building block for assembling a BPEL process (c.f. subsection 3.6). We demonstrate a mapping for the three business transactions of our example, each representing a different type of complexity. Fig. 3 depicts the *request for quote* transaction which is the one of the lowest complexity. It is stereotyped

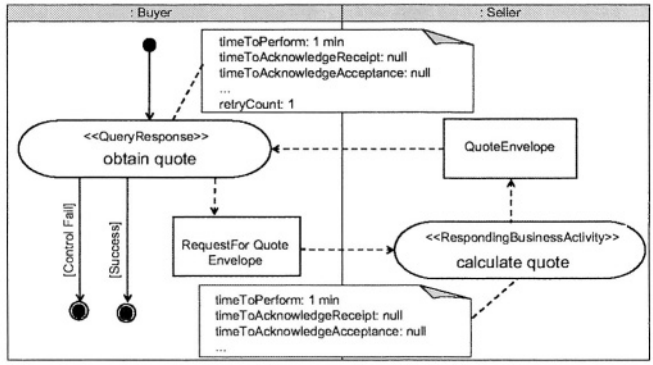


Fig. 3. Business Transaction "Request for Quote"

as *query/ response* business transaction. This means the responding business partner has the information available prior to the request. No acknowledgments are needed. Due to this characteristics the request for quote is realized by a synchronous interaction in BPEL.

The synchronous interaction is realized by a single operation *calculate quote* on the *seller port type.* This operation expects a *request for quote envelope* as input and outputs a *quote envelope.* In the BPEL file for the buyer process the interaction is defined by the simple activity <*invoke*> (lines 50 to 53). It references the partnership with the seller defined in the *partner link* named *link to seller* (line 45). Furthermore, <*invoke*> references the *calculate quote* operation (line 19) of the *seller port type* (lines 18 to 39). The *inputVariable* and the *outputVariable* are used for data handling of sent and received messages in executable processes. They are optional in abstract processes. Nevertheless, we include them to hint on the type of input/output message. Additionally, the presence of an *outputVariable* signifies a synchronous invocation, because an asynchronous invocation needs an input but no output. The <*source*> statements in line 51 and 52 are explained later in subsection 3.6.

```
[50] <invoke partnerLink="LinkToSeller" portType="SellerPortType" operation="calculateQuote"
            inputVariable="SentRequestForQuote" outputVariable="ReceivedQuote">
[51]   <source linkName="RequestForQuote2RegisterCustomer"
         transitionCondition="bpws:getVariableData('CustomerInformationConfimed') = 0"/> <!--ref line 118 -->
[52]   <source linkName="RequestForQuote2OrderProduct"
         transitionCondition="bpws:getVariableData('CustomerInformationConfimed') = 1"/> <!--ref line 119 -->
[53] </invoke>
```

The BPEL file for the seller's process includes the activity <sequence> (line 54) that is built by a <*receive*> activity (line 55) and a <*reply*> activity (line 56). Both included activities must reference the same operation: *calculate quote* (line 19). Consequently, the values for *partnerLink, portType* and *operation* must be the same in both cases (c.f. line 55 and 56).

```
[54] <sequence>
[55]   <receive partnerLink="LinkToBuyer" portType="SellerPortType" operation="calculateQuote"
                variable="ReceivedRequestForQuote"/>
[56]   <reply partnerLink="LinkToBuyer" portType="SellerPortType" operation="calculateQuote"
                variable="SentQuote"/>
[57] </sequence>
```

3.4 Transformation of Register Customer (Asynchronous)

Register customer is the next business transaction we consider. We already used *register customer* (Fig. 1) to illustrate UMM business transactions in section 2.1. It is stereotyped as *request/response* transaction. This means a decision process at seller side is stimulated by the request and must be finished before a response. Therefore, the request and the response are defined as two asynchronous interchanges. *Register customer* does not need any acknowledgements. According to the retry count, the buyer has to restart the transaction three times if he does not receive an response. If the buyer does not succeed after three trials, UMM requires him to send a control failure message.

```
[58]  <sequence>
[59]    <target linkName="RequestForQuote2RegisterCustomer"/> <!-- references line 118 -->
[60]    <source linkName="RegisterCustomer2OrderProduct"/> <!-- references line 120 -->
[61]    <assign>
[62]      <copy>
[63]        <from expression="3"/>
[64]        <to variable="PerformRegistrationRetryCount"/>
[65]      </copy>
[66]    </assign>
[67]    <while condition="bpws:getVariableData('PerformRegistrationRetryCount') &gt; 0 AND
                          bpws:getVariableData('PerformRegistrationRetryCount') = NULL">
[68]      <sequence>
[69]        <invoke ... portType="SellerPortType"operation="performRegistration" ... />
[70]        <pick>
[71]          <onMessage ... portType="BuyerPortType" operation="ReceiveResponseForRequestRegistration"
                          variable="ReceivedRegistrationResponse">
[72]            <empty/>
[73]          </onMessage>
[74]          <onAlarm for="PT4H">
[75]            <assign> <!-- decrement Perform RegistrationRetryCount --> </assign>
[76]          </onAlarm>
[77]        </pick>
[78]      </sequence>
[79]    </while>
[80]    <switch>
[81]      <case condition="bpws:getVariableData('PerformRegistrationRetryCount') = 0">
[82]        <throw faultName="RequestForQuoteControlFail"/>
[83]      </case>
[84]    </switch>
[85]  </sequence>
```

Owing to space limitations we only illustrate the buyer's process in lines 58 to 85. Note, in this code we do not show partner links and message variables anymore. An <*invoke*> activity (line 69) is used to call the *perform registration* operation (line 23). Next the buyer expects to receive a message via its *receive response for request registration* operation. We use a <*pick*> activity that awaits the occurrence of one of a set of events and then performs the activity associated with the event that occurred. The first event is given in the <*onMessage*> element (line 71) receiving a message from the *receive response for request registration* (line 2). The other event is stated in the <*onAlarm*> element (line 74) which is effective if a time frame of 4 hours is exceeded. In this case one retry is consumed. Hence, the *perform registration retry count* variable - which is initiated at the beginning of the transaction in lines 61 to 66 - must be decremented (line 75). The <*while*> loop (lines 67 to 79) continues if no message was received and the maimum number of retrys is not reached. If the while loop is stopped because no retrys are left, the condition of the <*case*> statement (line 81) in the <*switch*> (lines 80 to 84) holds, and a control failure exception is thrown (line 82).

3.5 Transformation of Order Product (with Acks)

Finally, we take a look at the *order product* business transaction, which is the most complex one. It is stereotyped as *commercial transaction.* In UMM this means that the business transaction results in a residual obligation between the business partners to fulfill terms of a contract. A commercial transaction requires acknowledgments to

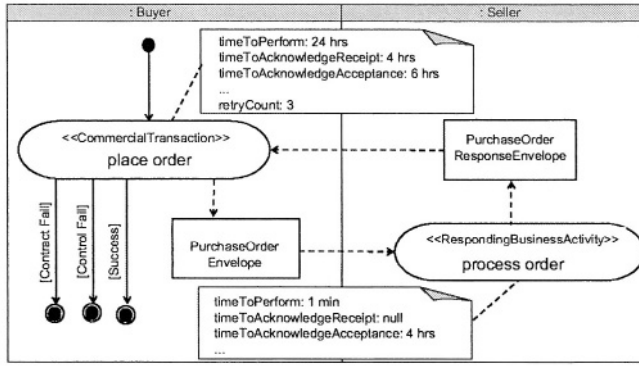


Fig. 4. Business Transaction "Order Product"

be sent. These acknowledgements are not the ones on the network level. An acknowledgment of receipt is sent after grammar validation, sequence validation, and schema validation. An acknowledgment of acceptance is sent after validating the content against additional rules to ensure that the content is processable by the target application.

The code in lines 82 to 112 represent the order product business transaction in the buyer's process. We do not detail the transaction for the seller's process. The idea of this code pattern is similar to the one used in the previous subsection. This time the request results in three return messages, two acknowledgements and the response. A control failure must be raised, if any one of these is not received in the permitted time frame. As a consequence, we recursively nest <*pick*> elements (lines 95 and 97) within the <*onMessage*> elements of the previously "picked" message.

```
[86] <sequence joinCondition="bpws:getLinkStatus('RequestForQuote2OrderProduct')
                              OR bpws:getLinkStatus('RegisterCustomer2OrderProduct')">
[87]    <target linkName="RequestForQuote2OrderProduct"/> <!-- references line 119 -->
[88]    <target linkName="RegisterCustomer2OrderProduct"/> <!-- references line 120 -->
[89]    <assign> <!-- initiate OrderProductRetryCount --> </assign>
[90]    <while condition="bpws:getVariableData('OrderProductRetryCount') &gt; 0 AND
                          bpws:getVariableData('ReceivedPurchaseOrderResponse') = NULL">
[91]      <sequence>
[92]        <invoke partnerLink="LinkToSeller" portType="SellerPortType"
                    operation="processOrder" inputVariable="SentPurchaseOrder" />
[93]        <pick>
[94]          <onMessage ... portType="BuyerPortType"
        operation="AckReceipt" ... >
[95]            <pick>
[96]              <onMessage ... portType="BuyerPortType"
                  operation="AckAcceptance" ...>
[97]                <pick>
[98]                  <onMessage ... portType="BuyerPortType" operation=
                    "ReceiveResponseForPlaceOrder">
[99]                    <invoke ... portType="SellerPortType" operation="AckAcceptance" ... />
[100]                   </onMessage>
[101]                   <onAlarm for="PT18H"> <!-- decrement OrderProductRetryCount --> </onAlarm>
[102]                 </pick>
[103]               </onMessage>
[104]               <onAlarm for="PT2H"> <!-- decrement OrderProductRetryCount --></onAlarm>
[105]             </pick>
```

```
[106]              </onMessage>
[107]              <onAlarm for="PT4H"> <!-- decrement OrderProductRetryCount --> </onAlarm>
[108]            </pick>
[109]          </sequence>
[110]        </while>
[111]        <switch> <!-- Throw exception if OrderProductRetryCount = 0 --> </switch>
[112]      </sequence>
```

3.6 Transformation of Business Collaboration Protocol

A UMM business collaboration protocol choreographs the business transactions of the business collaboration. So far we detailed the choreography within a business transaction. Now we define the flow between the business transactions in order to finalize the BPEL process. For this purpose BPEL provides the structured activity called flow. A flow is a group of activities. It completes when all of the activities in the flow have completed. Completion of an activity in a flow includes the possibility that it will be skipped if its enabling condition turns out to be false [1]. Additionally, the flow specifies synchronization dependencies between activities. A link element is used to express a transition from one activity to the other. The link has a unique name within the process. The activity that is the source for the link includes a *source* element that references the link by name. Similarly, the target activity includes a target element that references the corresponding link by name. Furthermore, the *source* element may specify a transition condition. If an activity includes more than one *target* element, it is useful to specify a join condition for its activation. By default the status of at least one incoming link has to be positive to start an activity.

Code lines 113 to 133 represent the buyer's process of the *order management* collaboration shown in Fig. 2. After a starting <*process*> element the partner link to the seller (copy of lines 44 to 46) and a definition of all variables follow. Next, the main part groups all activities of the business collaboration into a <*flow*> element (lines 116 to 125). All sub-processes representing the business transactions *request for quote* (lines 50 to 53), *register customer* (lines 58 to 85) and *order product* (lines 85 to 112) are defined as part of the flow.

Prior to these sub-processes all the transitions between the transactions are defined. According to Fig. 2, three synchronization dependencies exist: (1) from *request for quote* to *register customer* (line 118), (2) from *request for quote* to *order product* (line 119), and (3) from *register customer* to *order product* (line 120). The first two transitions start from the request for quote sub-process (lines 50 to 53). Consequently, the <*invoke*> activity covering the whole sub-process includes two <*source*> elements (lines 51 and 52) referencing the two links. Since the two links are mutually exclusive depending on the status of the customer information, corresponding transition conditions are included in the <*source*> elements. The third transition starts from the *register customer* sub-process that includes a related <*source*> element (line 60). Since this activity is also the destination of the first transition, it includes a corresponding <*target*> element in line 59. Both the second and the third transition end at the *order product* sub-process, which is marked by the <*target*> elements in lines 87 and 88. If any of the two transitions becomes active, order product should start. Therefore, the join condition of the <*sequence*> element (line 88) is a Boolean *OR*.

```
[113]<process>
[114]   <!-- insert p artner links; lines 44 - 46 -->
[115]   <!-- insert varaible definitions, not shown in this p    aper -->
[116]   <flow>
[117]     <links>
[118]       <link name="RequestForQuote2RegisterCustomer"/>
[119]       <link name="RequestForQuote2OrderProduct"/>
[120]       <link name="RegisterCustomer2OrderProduct"/>
[121]     </links>
[122]     <!-- insert invoke st atement of T ransaction Request For Quote lines 50 - 53 -->
[123]     <!-- insert sequence st atement of T ransaction Register Customer lines 58 - 85 -->
[124]     <!-- insert sequence st atement of T ransaction Order Product lines 84 - 108 -->
[125]   </flow>
[126]   <faultHandlers>
[127]     <catch faultName="RequestForQuoteControlFail">
[128]       <invoke ... portT ype="SellerPortT ype" operation="ControlFailure" ... />
[129]       <terminate/>
[130]     </catch>
[131]     <catch faultName="OrderProductControlFail"> ... </catch>
[132]   </faultHandlers>
[133]</process>
```

The flow automatically starts with the *request for quote* sub-process, because it does not include an incoming synchronization dependency. The process automatically ends after the *order product* sub-process, because all other activities were active before. Note, *register customer* might be skipped according to the transition condition. The control failures thrown in lines 82 and 111 also terminate the process. These control failures are subject to the fault handlers in lines 126 to 132.

4 Summary

UN/CEFACT's modeling methodology (UMM) is a UML-based methodology for capturing the business semantics according to the BOV of Open-edi. ebXML and Web Services are to popular technologies for implementing B2B processes on the FSV of Open-edi. The Open-edi concept is very similar to the MDA - it separates the specification from the implementation. This approach guarantees reusability, portability, and interoperability. A key success factor is a well-defined set of rules and techniques to map from the UMM model to the different B2B implementation technologies. Our current research is directed towards the B2B choreography. In our previous work we concentrated on the mapping from UMM to ebXML's BPSS. Since BPSS is based on the UMM meta model the mapping is comparatively easy. Owing to the growing popularity of Web Services, BPEL is gaining more and more acceptance by industry. Hence, there is a desire to transform UMM models to BPEL processes. This paper presents a first evaluation to what extent UMM models can be transformed to BPEL. Future work will include a detailed analysis on the strengths and limitations of both BPSS and BPEL to express business collaborations.

In this paper we show how the UMM artefacts business transaction and business collaboration protocol are represented in BPEL. Whereas a transformation from UMM to BPSS results in a single process definition for all partners, a transformation to BPEL results in a different process definitions for each partner. BPSS is able to capture UMM's security requirements on the activities and exchanged messages. BPEL is not. Instead other Web Services standards must capture the security semantics. Nevertheless, we are able to represent all the UMM choreography

information in BPEL. We demonstrated the transformation by means of a simple example. Since UMM business transactions have a very strict structure, the resulting transformations provide useful patterns. Furthermore, the transformation of the business process protocol follows a clear approach based on grouping all business transaction activities to BPEL flow structure and transforming the UMM transition to BPEL link elements.

References

1. Andrews, T., Curbera, F., Dholakia, H., Goland Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.; Business Process Execution Language for Web Services, Version 1.1, May 2003 http://msdn.microsoft.com/library/-default.asp?url=/library/en-us/dnbizspec/html/bpel1-1.asp
2. Hammer, M., Champy, J.; Reengineering the Corporation: Manifesto for Business Revolution; Harper Business; 1993
3. Hofreiter, B., Huemer, C.; Modeling Business Collaborations in Context; Proceedings of On The Move to Meaningful Internet Systems 2003: OTM 2003 Workshops; Springer LNCS; November 2003
4. Hofreiter, B., Huemer, C.; ebXML Business Processes -Defined both in UMM and BPSS; Proceedings of the 1 st GI Workshop on XML Interchange Formats for Business Process Management at 7th GI Conference Modellierung 2004, Marburg Germany, March 2004 http://wi.wu-wien.ac.at/~mendling/XML4BPM/xml4bpm-2004-proceedings-bpss.pdf
5. ISO; Open-edi Reference Model. ISO/IEC JTC 1/SC30 ISO Standard 14662. ISO; 1997
6. Kim, J.-H., Huemer, C.; Analysis, Transformation and Improvements of ebXML Choreographies based on Workflow Patterns; Proceedings of On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE; Springer LNCS; October 2004
7. Leymann, F.; Web Services Flow Language (WSFL 1.0); May 2001 http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.pdf
8. Leymann, F., Roller, D.; Modeling Business Processes with BPEL4WS; Proceedings of the 1st GI Workshop on XML Interchange Formats for Business Process Management at 7th GI Conference Modellierung 2004, Marburg Germany, March 2004 http://wi.wu-wien.ac.at/~mendling/XML4BPM/xml4bpm-2004-proceedings-bpel4ws.pdf
9. Miller, J., Mukerji, J.; MDA Guide Version 1.0.1; OMG omg/2003-06-01, http://www.omg.org/docs/omg/03-06-01.pdf
10. Thatte, S.; XLANG - Web Services for Business Process Design; June 2001; http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm
11. UN/CEFACT; ebXML - Business Process Specification Schema v1.10; October 2003; http://www.untmg.org/downloads/General/approved/ebBPSS-v1pt10.zip
12. UN/CEFACT; UMM Meta Model, Revision 12; January 2003; http://www.untmg.org/downloads/General/approved/UMM-MM-V20030117.zip
13. UN/CEFACT; UMM User Guide, Revision 12; September 2003; http://www.untmg.org/downloads/General/approved/UMM-UG-V20030922.zip

Using BPEL Process Descriptions for Building Up Strategic Models of Inter-organizational Networks

Dominik Schmitz[1], Gerhard Lakemeyer[1], Günter Gans[1], and Matthias Jarke[1,2]

[1] RWTH Aachen, Informatik V, Ahornstr. 55, 52056 Aachen, Germany
{schmitz,lakemeyer,gans,jarke}@cs.rwth-aachen.de
[2] Fraunhofer FIT, Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abstract. In previous work, we proposed the prototype environment SNet for the representation and dynamic evaluation of agent-based designs for inter-organizational networks. A key feature of SNet is the automatic translation of extended i* models into the action language ConGolog. An issue we have not yet considered is how to arrive at the foundational and hopefully realistic i* model. Currently there is no support to incorporate information from already existing descriptions of business processes in an enterprise. BPEL is expected to play an important role in the future by enabling interoperability of different partners' business processes – not only in the web service domain. Once standardized a wide-spread availability of BPEL-based process descriptions can be expected. In this paper we suggest how to map BPEL descriptions into i* descriptions, thus opening the door to generating SNet simulations of business processes from BPEL descriptions.

1 Introduction

In previous work, we proposed the prototype environment SNet to model strategic inter-organizational networks, which are comprised of human, organizational, and technological actors [1]. A crucial aspect of these networks are the interdependencies among the various actors, which result, for example, from the need to delegate certain activities, which in turn requires a certain level of trust between the (human) members of the network. The agent-based graphical modeling language i* [2], which was developed for early requirements engineering, has proven to be particularly suitable as a modeling means in this context, because it explicitly deals with dependency relations, besides other notions like actors, goals, resources, and tasks. To capture the dynamic aspects of agent networks we proposed to amalgamate i* and the action formalism ConGolog [3]. To bridge the gap between the two formalisms we extended i* by features to describe task preconditions and effects. These extended i* diagrams are automatically translated into executable ConGolog programs, supported by the metadata manager ConceptBase [4]. Running simulations for different scenarios within a network is useful for analyzing its properties and can provide the foundation of a decision-support tool for network members.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 520–532, 2004.
© Springer-Verlag Berlin Heidelberg 2004

In recent work [5] we introduced a decision-theoretic planning component for each network representative to run even more realistic simulations and improved the modeling facilities among other things by introducing a role concept [6]. The main current research effort concentrates on a closer contact with the real world: in [6] we considered already a real world entrepreneurship network. In this paper, we investigate now the possibility to use BPEL (Business Process Execution Language for Web services) [7] process descriptions as a starting point for building up the detailed i* model we need for analyzing strategic inter-organizational networks. The consideration of BPEL process descriptions results from the insight that it is very unlikely that a modeler will start the modeling of an inter-organizational network and the processes therein from scratch. He will base his work on existing process models and more general all experiences he has had so far.

Although BPEL is not undisputed (see [8],[9] for competing efforts) it must be considered as the most promising candidate for describing business processes – esp. in an inter-organizational setting – in a standardized way. Additionally there are already suggestions to combine agent technology and web services to finally realize the promise of a virtual enterprise [10]. Once the standardization effort is successful and if the trend towards web services continues this leads to the expectation that in the near future many business software products will be able to produce BPEL descriptions as an output. Thus providing an interface to BPEL is a natural step to support building up realistic i* models of an inter-organizational setting and the first step to incorporate already existing information on business processes in an enterprise in a systematic way. In this paper, we suggest how to map BPEL descriptions into the i* model. The goal is to eventually turn this into a completely formal mapping which can be automated.

The rest of the paper is organized as follows. In Sect. 2 we introduce our SNet simulation and modeling tool. In the following section (Sect. 3) we provide a short introduction into the usage of BPEL. The main contribution is then presented in Sect. 4 where we describe how and which parts of BPEL process descriptions can be mapped to extended i*. After shortly discussing some related work Sect. 5 ends the paper with a conclusion and a brief outlook on future work.

2 The Modeling and Simulation Environment SNet

2.1 The Architecture of the SNet Tool

We base our modeling and simulation environment SNet for inter-organizational networks on a combination of two formalisms: *i** – a graphical modeling language originally intended for describing early requirements – for statically modeling the network and *ConGolo*g – a logic-based high-level programming language – for simulations so that dynamic aspects such as trust can be analyzed. We take an agent-oriented view in that each actor of an inter-organizational network is represented by a deliberative agent. We will discuss the features of the two formalisms in more detail later on. First we give a short overview of their overall interplay. The SNet architecture is depicted in Fig. 1.

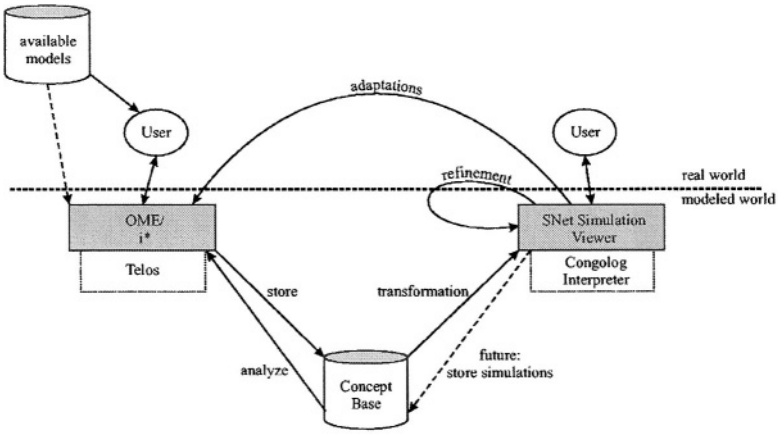


Fig. 1. SNet Architecture

We use *OME3* (Organization Modeling Environment) – a graphical model editor developed at the University of Toronto [11] – to build up static models of inter-organizational networks in the modeling language i* [2]. The semantics of i* is defined in the knowledge representation language Telos [12] which is also the formalism underlying *ConceptBase* [4], a deductive metadata repository. The ConceptBase query-language can be used for static analyses and especially for the transformation into ConGolog. The execution of the resulting ConGolog program is shown in a step by step view by the simulation viewer, which also provides access to control the simulation run, i. e. the user creates scenarios by simulating the pro-activity of network members and investigates how this and resulting delegations affect relationships (especially trust). Conclusions derived by the user from such simulations might lead to modifications of the model or scenario conditions which provide the basis for new simulation runs.

2.2 An Extended Version of i*

The i* framework is a graphical language and includes the *strategic dependency (SD)* model for describing the network of relationships among actors and the *strategic rationale (SR)* model, which, roughly, describes the internal structure of an actor in terms of tasks, goals, resources, etc. Compared to Yu's original formulation we added a few new features to SR models such as task preconditions. Figure 2 shows part of an extended SR model from the entrepreneurship domain with a focus on the roles *Venture Capitalist, Entrepreneur,* and *Faculty Member.*

The venture capitalist's task *choose_promising_entrepreneur* is decomposed into three subtasks, which are ordered using *sequence links* (an easy to use form of a task precondition). The task *suggest_business_idea* is delegated to the actor *Entrepreneur.* Goals like *ask_evaluation* provide a means to specify alternatives from which the instantiated agent (respectively represented network member)

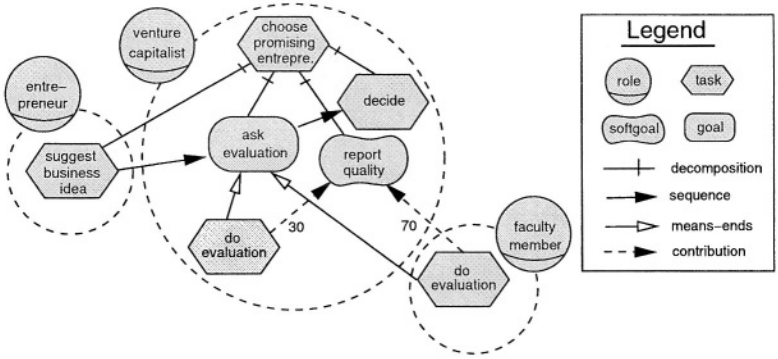


Fig. 2. Modeling in i*/SNet

can choose at run-time. In this example the venture capitalist can choose to do the evaluation himself or delegate this to a *Faculty Member.* Softgoals are used to specify the criteria on which the deliberative component bases its decision, e. g. *report_quality.* Tasks can have preconditions and effects, represented by their own model element denoted as a triangle (but not occurring in the example above), and produce or consume resources.

2.3 Mapping the i* Model to a ConGolog Program

ConGolog based on the situation calculus [13] is a language for specifying complex actions (high-level plans). For this purpose constructs like sequence, procedure, *if-then-else,* but also non-deterministic and concurrency constructs are provided. ConGolog comes equipped with an interpreter which maps these plans into sequences of atomic actions assuming a description of the initial state of the world, action precondition axioms, and successor state axioms for each fluent. For details see [3].

To enable simulations the abstract roles defined in the i* model have to be instantiated. The instances of a role (agents) differ in details as for example duration of tasks and softgoal contributions. We have to omit the details of the mapping here but simply mention that in addition to the ConGolog interpreter we provide an environment which equips each agent with a decision theoretic planning component to reason about which alternative (or partner) to choose according to the specified criteria (softgoals). See [5] for details.

3 BPEL

BPEL [7] the Business Process Execution Language for managing interoperability of business processes is currently under evaluation at the standardizing committee Oasis. It builds upon the web service stack (including SOAP, WSDL, etc.). BPEL's main idea is that while WSDL allows for the functional description

of web services, this is not enough if it comes to supporting business processes. Besides the interface of each web service the orchestration of services (requiring some stateful knowledge) has to be enabled. Accordingly BPEL provides the means to combine several web services into a new web service with all the necessary considerations of states and information exchange while avoiding to explicitly refer to instances of web services within this description. A suitable instance is supposed to be chosen at run-time (e. g. with the help of UDDI).

In the following we provide a brief introduction into BPEL by considering a small example. Due to limited space we cannot cope with all BPEL constructs. See Sect. 4.3 for some considerations about omitted parts.

The Loan Service Example. In the example the approval of a loan is provided as a web service. A customer enters some personal information and the amount of money he needs, the web service evaluates the request with the help of other web services, and eventually approves or rejects the request. To fulfill its task the loan approval service can make use of two other services: one for simple risk assessment and one for a more detailed approval analysis. Whether and which of these services are invoked for a given request at run-time depends on the request itself and exactly this is described in the business process description. So if the requested amount is higher than $ 10.000 the approval service is contacted. If it is below or equal to $ 10.000 first the simple risk assessment service is fed. If this service assumes a low risk the loan approval service immediately approves the request whereas in case a medium or high risk level is identified the approval service is contacted again.[1]

```
 1:<process name="loanApprovalProcess" ...>
 2:  <partnerLinks>
 3:    <partnerLink name="customer" myRole="loanService"
 4:                 partnerLinkType="loanPartnerLinkType"/>
 5:    <partnerLink name="approver" partnerRole="approver"
 6:                 partnerLinkType="loanApprovalLinkType"/>
 7:    <partnerLink name="assessor" partnerRole="assessor"
 8:                 partnerLinkType="riskAssessmentLinkType"/>
 9:  </partnerLinks>
10:  <variables>...</variables>
11:  <faultHandlers>...</faultHandlers>
12:
13:  <flow>
14:    <links><link name="receive-to-assess"/>
15:           ...omitted...</links>
16:    <receive partnerLink="customer" portType="loanServicePT"
17:             operation="request" variable="request"
18:             createInstance="yes">
```

[1] BPEL assumes a specification of the appropriate services (port type and operation) together with a suitable set of <message>s to be defined in WSDL.

```
19:       <source linkName="receive-to-assess"
20:               transitionCondition="amount=<10000"/>
21:       <source linkName="receive-to-approval"
22:               transitionCondition="amount>10000"/>
23:     </receive>
24:     <invoke partnerLink="assessor" portType="riskAssessmentPT"
25:             operation="check" inputVariable="request"
26:             outputVariable="risk">
27:       <target linkName="receive-to-assess"/>
28:       <source linkName="assess-to-setMessage"
29:               transitionCondition="risk-level=low"/>
30:       <source linkName="assess-to-approval"
31:               transitionCondition="risk-level!=low"/>
32:     </invoke>
33:     <assign>
34:       <target linkName="assess-to-setMessage"/>
35:       <source linkName="setMessage-to-reply"/>
36:       <copy><from expression="'yes'"/>
37:             <to variable="approval" part="accept"/></copy>
38:     </assign>
39:     <invoke partnerLink="approver" ...>
40:       <target linkName="receive-to-approval"/>
41:       <target linkName="assess-to-approval"/>
42:       <source linkName="approval-to-reply"/>
43:     </invoke>
44:     <reply partnerLink="customer" portType="loanServicePT"
45:            operation="request" variable="approval">
46:       <target linkName="setMessage-to-reply"/>
47:       <target linkName="approval-to-reply"/>
48:     </reply>
49:   </flow>
50:</process>
```

Fig. 3. BPEL Description of Loan Service

Separately from the above process description BPEL allows for the specification of <partnerLinkType>s by specifying at maximum two roles (minimum 1 role) which are involved in a partnership. A <role> refers to the port type of a web service. If there is only one role specified this means that there is no constraint placed on the other partner. In the example the following <partnerLinkType>s exist: *loanPartnerLinkType, loanApprovalLinkType,* and *riskAssessmentLinkType*. The details of the first are given below:

```
<partnerLinkType name="loanPartnerLinkType">
  <role name="loanService"><portType name="loanServicePT"/></role>
</partnerLinkType>
```

Fig. 4. <partnerLinkType> Definition

In the process definition these <partnerLinkType>s are instantiated (see Fig. 3, lines 2–9). But with instantiation it is not meant here to assign a dedicated service but to specify which of the roles is played by the described business process. In our example this is the role *loanService* concerning the <partnerLinkType> *loanPartnerLinkType* whereas concerning the two other <partnerLinkType>s the business process is only client and the partners play the only specified roles *approver* resp. *assessor.*

Providing and Invoking Web Services. Concerning communication i. e. web service interaction the three most important activities are <invoke>, <receive>, and <reply>.<invoke> is used to start a request to another service (e. g. Fig. 3, lines 24–26). BPEL allows for synchronous as well as asynchronous requests. A synchronous invocation is blocking, i. e. the process does not proceed any further until the requested service has provided its answer. In the example it is assumed that all services run fast enough to let synchronous requests be sufficient. In case an asynchronous request needs to be modeled the reply is gathered by a callback interface on which the process waits via a <receive> statement. The <receive> construct is also used for the provision of a service, i. e. a process description usually starts with a <receive> construct (Fig. 3, lines 16–18).[2] The <reply> construct is needed to answer a synchronous request (Fig. 3, lines 44–45).

Flow Control. A key feature of BPEL is the possibility to describe how the different invocations of services and answers to requests are timely related to each other via so called *structured activities.* Activities grouped together in a <sequence> construct are executed one after another. In contrast to this all activities combined in a <flow> (see example) are in general executed in parallel. But it is possible to specify arbitrary sequential relationships between these activities via <link>s. Every activity can be source or target of such a link and conditions can be associated with the sources which enable or disable the corresponding links accordingly.[3] The results of the link semantics as specified in Fig. 3 will become clear when the mapping of the example process is considered (see Sect. 4).

For simplicity and due to reasons of space we concentrate here on the constructs occurring in the example. BPEL provides some more constructs like <switch>, <while>, another communication construct (<pick>) as well as event, fault, and compensation handlers where the latter two belong to BPEL's sophisticated means to specify fault behavior.

[2] The setting of the createInstance attribute reflects that for each request a new process instance is created.

[3] The *transitionCondition* in the example is simplified since normally this should be a XPath1.0 expression referring to variables, message properties etc.

4 Mapping BPEL to Extended i*

Since BPEL claims to be sufficiently detailed so that an engine able to execute the BPEL process description can provide the described service as a new combined service, it has to cope with many nasty details which are not relevant to a more strategic investigation we are interested in. For example only message parts which have strategic effects esp. on the quality of a service must be regarded. Thus in the following we want to analyze which aspects of a BPEL process description provides valuable information for creating i* models.

4.1 Deriving Actors from BPEL Descriptions

The most high-level elements in i* are actors specialized into agents, roles and positions. Since agents refer to individuals they correspond to an instance of a service and thus are not relevant here neither on the i* modeling level nor in the BPEL descriptions. In contrast to this the abstract definition of a web service (via WSDL as a port type with an operation) can be mapped to an i* role since it comprises a self-contained functionality. For the naming we suggest to refer to the `<role>` specification within a `<partnerLinkType>` (see Fig. 4). Thus for the example we arrive at the roles *loan service, assessor,* and *approver.*

Additionally the `<partnerLinkType>` definitions describe relationships between different roles on an SD model like level without detailing the type of dependency (task, resource, goal, softgoal). While in general a task dependency will be most suitable it is up to the modeler to semantically analyze how much freedom is left to the delegatee and thus choose an appropriate dependency type. For the example we assume a task dependency to be adequate for all relationships. Altogether we arrive at the situation pictured in Fig. 5. Note that we added the role of a *customer* not explicitly mentioned as the partner role for a *loanPartnerLinkType.*

Introducing a role for each web service might be too detailed, but a later merging is not excluded so its a good starting point. Especially in case of repeated communication between business partners a continuous business process on both sides must be assumed which must be reflected in the model (re- and back-delegations). Another type of grouping which also exists in BPEL does not occur in the example but can also nicely be mapped to i*. Using the `<partners>`

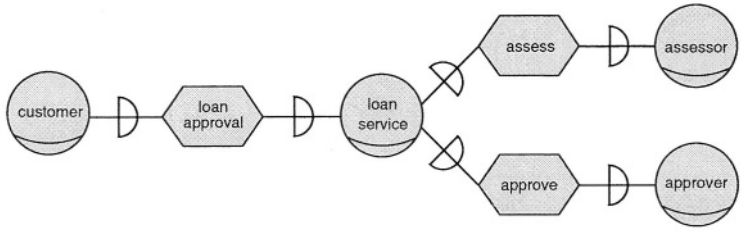


Fig. 5. Strategic Dependency Diagram

element several <partnerLink>s (recall in a <process> <partnerLinkType>s are instantiated) can be grouped together to put an additional constraint on the web service chosen at run-time in that it must fulfill all the partner roles in the specified <partnerLink>s. This is much in the same spirit as i* combines several roles into a position if they are commonly played together.

4.2 Mapping the Process Body

After we have got now our basic actors available, we can focus on detailing their internals. Since a BPEL document describes only the details of one of the business partners involved we must in general expect several of such detailed descriptions to be combined into one i* model. Naming conflicts are not expected since we assume an i* model to be built out of the subjective view of one of the business partners. Accordingly the knowledge about the other actors is limited since they will not provide him with all the details about their internals. Instead he himself has to revise the models of their behavior according to his knowledge and experiences so far. Thus he has got to a large extent control over the naming and can avoid naming conflicts.

The body of the process has to be mapped onto the specification of the internal behavior of a role. To subsume all the activities we first have to generate a top level task element whose name can originate from the name attribute of the <process> element. For our example we have chosen *loan approval.*

Mapping of Basic Activities. Concerning inter-service communication only the <invoke> activity automatically results in an i* task element because due to the higher modeling level the receipt of a message and the provision of a reply need not be considered in i* models separately. Thus the starting <receive> activity is not mapped to an SR model element, but instead strategically relevant message parts are mapped to parameters of the top level task element. Accordingly the corresponding <reply> activity is also not mapped to an SR model element. But since SNet currently does not support return values for delegated tasks[4] if the return value is needed for the process description it must be mapped to a precondition/effect element which then can again be used as a precondition for other tasks.

The <invoke> activity is mapped to a task, but following its semantics this task is not associated with the current role, but with the one of the invoked service. Thus this is a delegation in the SNet naming and it is graphically represented by a decomposition link which crosses role boundaries. If – as suggested – a more detailed process description for this business process at the business partner is given this is used to detail it. If such a description does not exist the resulting top level task element is simply primitive and the naming must be derived from the associated operation attribute.

[4] It can only be decided whether the delegation was successful or not and possibly its contributions on accompanying softgoals.

The effects (and sequential relationships) of a <receive> activity corresponding to an asynchronous request must be associated with the task element resulting from the transformation of the matching <invoke> activity. Notice that SNet innately assumes asynchronous i. e. non-blocking communication. To emulate synchronous communication a suitable set of sequence links needs to be added.

The means to specify internal local activities in BPEL is limited – BPEL focuses virtually exclusively on orchestration. Only the <assign> activity which sets the value of a variable (see Fig. 3, lines 33–38) can be seen as some internal activity. But whether it should be mapped to its one primitive task depends on whether it has any strategic impact. In the example the local decision to approve a loan if the amount is below $ 10.000 and risk is low is mapped onto a primitive task. A name for this task can be derived from the destination variable possibly in connection with the names of links arriving at or leaving this activity. In the example we simply suggest *setMessage.*

Mapping of Structured Activities. Equally important to the transformation of basic activities is their timely relation to each other as specified using the <sequence> and the <flow> construct together with <link>s. A <sequence> relationship between activities can simply be reflected on the side of i* via usage of sequence links which have exactly the same meaning. Attention must only be paid if there are activities which do not have an equivalent on the side of i* (see above <receive>, <reply>). In this case the destination of the sequence links must be adjusted accordingly.

The default semantics of the <flow> construct is also the default of the combination of sub tasks (or goals) on the side of i*, i. e. everything is executed in parallel. Unconditioned links can again be mapped onto sequence links whereas conditioned links should be mapped to the more general precondition/effect element. This means the source activity has got some effect and the described condition is checked as a precondition for the target activity. In fact the modeler should make use of the separate specification of conditions to avoid doubling conditions as in the example: the check whether the amount exceeds $ 10.000 or not should be represented only as one precondition/effect element and a negated precondition link can be used as the precondition for the invocation of the simple risk assess service.

The above description does not yet reflect one property which makes the <link> semantics different from the <sequence> semantics. This is if the joinCondition of an activity (which per default combines incoming <link>s via OR) is not fulfilled the activity is simply skipped.[5] This corresponds to an 'if needed' style of task decomposition (denoted by dashed decomposition links) we already proposed in earlier versions (OR-task decomposition).

[5] In fact we assume here, that the suppressJoinFailure attribute of <process> is set to yes. If this is not the case a lot of fault behavior invocation has to be regarded which we omit here (see Sect. 4.3).

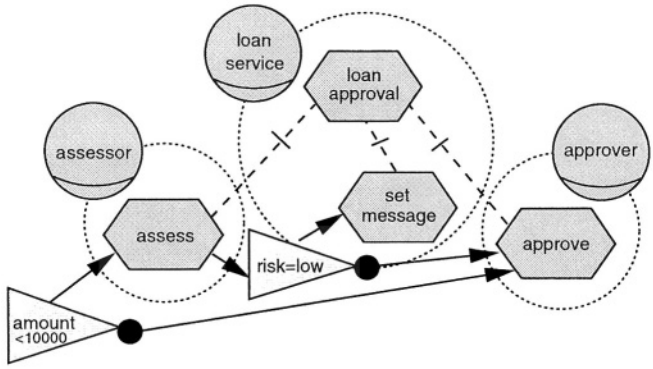


Fig. 6. Strategic Rationale Diagram Resulting from Transformation

The result of the mapping of the loan service description to an SR model is given in Fig. 6.

It is important to mention, that it is not a necessity that every basic activity within a process description becomes a direct sub task of the top level task. Indeed it is much nicer to preserve the groupings resulting from the usage of structured activities like `<sequence>` and `<flow>`. In case a name is given (via the corresponding default attribute) it can be used to name this complex sub task. Otherwise we suggest to generate a dummy name (e. g. *flow_23*) and leave it to the modeler to choose a more suitable one later on.

4.3 Preliminary Evaluation of the Suitability of BPEL

As the previous presentation already suggests, BPEL is to some extent too detailed (e. g. explicit `<receive>` and `<reply>` activities) and thus parts of it are and should not be mapped to the strategic models. A major disadvantage are the poor means to describe the (local) internals of a process. Their effect can only be represented via an `<assign>` activity and its duration via a `<wait>` activity. Thus we expect the modeler to detail the internal behavior manually if such knowledge is available. In particular BPEL does not support the soft choice of process paths. If there are different paths to choose from the conditions are deterministic. Something comparable to the i* goal element is missing. If such considerations do make sense for the described process the modeler has to add them manually. It might also be the case that some deterministic choices can be relaxed to such more free choices. The lack of a goal-like construct causes also the lack of softgoal-like aspects and corresponding contributions.

The other way round we find some constructs in BPEL which are most suitable but currently not supported by SNet. Especially all the sophisticated means to handle (run-time) fault behavior which we had to omit here. We are looking forward to incorporate this and especially connect these behavioral rules to our monitoring component (cf. [6]). Concerning the `<while>` construct we are currently investigating whether we shall incorporate this already on the i* level. On

the side of ConGolog a corresponding construct (with the same name) of course already exists.[6]

Eventually we want to emphasize a nice property of BPEL which is the inherent subjective view from which the processes are described. While this can also cause some problems for deriving a combined model it nicely stresses that the whole modeling procedure is subjective: the business partner who is modeling the scenario can include only as many details about its partners and other enterprises involved as he knows.

5 Discussion and Conclusion

There exists already an approach [14] to map BPEL descriptions to Petri nets with the intended goal to allow for a detailed analysis. Although we think that a formal and exact transformation of BPEL into ConGolog is possible and comparable to the one into Petri nets we concentrated here on the strategic and hence more abstract level of extended i*. The reason for this is that we do not want to debug BPEL descriptions but to use them for building up models to run long-term prospective studies of the evolution of inter-organizational networks.

Another branch of work concerning the area of "adapting Golog for composition of semantic web services" is carried out by Sheila McIlraith and others [15]. They have shown that Golog might be a suitable candidate to solve the planning problems occurring when services are to be combined dynamically at run-time. Additionally they related their work also to BPEL(4WS) [16] explicitly by stating that the semantic web efforts in the research area are disconnected from the seamless interaction efforts of industry and thus propose to "take a bottom-up approach to integrating Semantic Web technology into Web services". But they mainly focus on introducing a semantic discovery service and facilitate semantic translations.

To summarize we have seen that it is indeed possible to use BPEL process descriptions for building up extended i* models. BPEL provides several constructs which are similar to the ones used in i* as for example service definitions correlate with roles, `<partnerLinkType>`s with strategic dependencies, `<partner>`s with positions. Furthermore the structuring of the process body via `<sequence>`, `<flow>` etc. can be mapped to ordering constraints of sub tasks in ext. i*. The modeler might have to provide additional information to nicely group tasks to complex sub tasks and neither goals nor softgoals with contributions result from a BPEL description. Such more strategic considerations must be added to the very detailed, executable process description. Other aspects like fault behavior and the `<while>` construct are not yet covered in SNet but would fit in.

For the future we expect to be able to provide a better support for deriving i* models from BPEL descriptions by generating corresponding Telos frames (textual representation of i* models) automatically. And of course we are looking forward to apply the transformation procedure to real world BPEL descriptions.

[6] The `<switch>` construct can be mapped to a task decomposition of 'if needed' sub tasks with suitable precondition/effect elements checking the conditions.

Additionally, we have to watch out also for the rivals of BPEL as there are BPML, BPMN, WS-CDL, etc. maybe also enabling to derive useful information from descriptions written in these languages.

Acknowledgment. This work was supported in part by the Deutsche Forschungsgemeinschaft in its Priority Program on Socionics, and its Graduate School 643 "Software for Mobile Communication Systems".

References

1. Gans, G., Jarke, M., Kethers, S., Lakemeyer, G.: Continuous requirements management for organization networks: A (dis)trust-based approach. Requirements Engineering Journal, Special Issue RE'01, Springer **8** (2003) 4–22
2. Yu, E.: Modelling Strategic Relationships for Process Reengineering. PhD thesis, University of Toronto (1995)
3. de Giacomo, G., Lespérance, Y., Levesque, H.: ConGolog, a concurrent programming language based on the situation calculus: language and implementation. Artificial Intelligence **121** (2000) 109–169
4. Jarke, M., Eherer, S., Gallersdörfer, R., Jeusfeld, M.A., Staudt, M.: ConceptBase - a deductive object base for meta data management. Journal of Intelligent Information Systems, Special Issue **4** (1995) 167–192
5. Gans, G., Jarke, M., Lakemeyer, G., Schmitz, D.: Deliberation in a modeling and simulation environment for inter-organizational networks. In: Proc. of CAiSE03. LNCS 2681, Klagenfurt, Austria (2003) 242–257
6. Gans, G., Schmitz, D., Jarke, M., Lakemeyer, G.: SNet reloaded: Roles, monitoring and agent evolution. In: Proc. of AOIS@AAMAS-2004 Workshop, (to appear)
7. Andrews, T., et al.: Business process execution language for web services, IBM, version 1.1, 2nd public draft release, www.ibm.com/ developerworks/ webservices/ library/ ws-bpel (2003)
8. Arkin, A.: Business process modeling language 1.0. Technical report, BPMI Consortium, http://www.bpmi.org/ (2002)
9. Kavantzas, N., Burdett, D., Ritzinger, G.: Web services choreography description language 1.0. working draft. http://www.w3.org/TR/ws-cdl-10/ (2004)
10. Petrie, C., Bussler, C.: Service agents and virtual enterprises: A survey. IEEE Internet Computing (2003) 68–78
11. Liu, L., Yu, E.: Organziation Modeling Environment (OME). WWW ([Accessed 2004/08/18]) http://www.cs.toronto.edu/km/ome.
12. Mylopoulos, J., Borgida, A., Jarke, M., Koubarakis, M.: Telos - representing knowledge about information systems. ACM Transactions on Information Systems **8** (1990) 325–362
13. McCarthy, J.: Situations, actions and causal laws. Tech. report, Stanford (1963) Reprinted 1968 in Minsky, M.(ed.): Semantic Information Processing, MIT Press.
14. Vidal, J.M., Buhler, P., Stahl, C.: Multiagent systems with workflows. IEEE Internet Computing (2004) 76–82
15. McIlraith, S., Son, T.C.: Adapting Golog for composition of semantic web services. In: Proc. of the 8th Int. Conf. on Knowledge Representation and Reasoning (KR2002). (2002) 482–493
16. Mandell, D.J., McIlraith, S.A.: Adapting BPEL4WS for the semantic web: The bottom-up approach to web service interoperation. In: Proc. of the 2nd Int. Semantic Web Conf. (ISWC). (2003)

A Meta Ontology for Organizations

Jan L.G. Dietz and Nathalie Habing

Delft University of Technology
P.O. Box 5031, NL-2600GA Delft
j.l.g.dietz@ewi.tudelft.nl, nhabing@worldonline.nl

Abstract. The research reported upon in this paper aims at developing a meta ontology for organizations, i.e. a schema for devising the ontology of a particular organization or a reference ontology for a class of organizations. The prospect of such a meta ontology is provided by the Ψ-theory that underlies the DEMO methodology. General requirements regarding such a meta ontology are stated. The proposed meta ontology satisfies all of them. It has the added advantage that there are socionomic laws at its core; this reinforces the ontological quality. It has been applied in several health care cases in practice, out of which a reference model for health care organizations has arisen.

1 Introduction

It is becoming increasingly important for enterprises to collaborate in temporary or permanent networks, in order to survive in the growingly competitive global market. Collaboration requires at the operational level that the business processes of the participating enterprises are integrated. Interorganizational information systems enable and support this integration. The design of these systems however comes after the integration of the business processes into cross-enterprise business processes. A prerequisite for integrating business processes is that their essential nature is well understood. This is what ontology is about. The notion of ontology is rather new in the field designing and engineering business systems and information systems. In its practical effects, ontology is a kind of standardization. That is why it has got the particular interest of the Semantic Web community [2, 25]. Ontology is about what reality is, what the things one observes really are. So, it must be about the apparent and inherent essence of something, it cannot be some possible view on it, next to other possible views. We define the ontology of an organization as the specification of the conceptual model of the essence of the organization, independent from its implementation. What we exactly mean by essence and by implementation independent will become clear shortly. Consequently, there is only one ontology for an organization. It shows the essential business activities, the participating actors, and the products and services that they are about or deal with.

We will present and discuss a meta ontology for organizations. The adjective 'meta' must be understood in the same way as it is understood in conceptual modeling. An ontology is comparable to a conceptual schema of a Universe of Discourse (UoD) [11]. A *meta ontology* is a schema for ontologies. This meta ontology should serve to develop ontologies in such a way that the next requirements

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 533–543, 2004.
© Springer-Verlag Berlin Heidelberg 2004

are met. The rationale behind these requirements is that they cover important properties of an ontology while taking optimal advantage of work that has already been done. It is not an exhaustive list:

1. Full advantage is taken of the achievements in conceptual modeling, in particular of the achievements by the natural language based modelling approaches (cf. [14, 15]), since they map directly to logic [24].
2. An ontology should not include lexical issues such as the naming of things and the denotation of values.
3. A clear distinction is made, based on a sound theoretical foundation, between a world (states and events) on the one hand, and the causes of change in this world (actors and acts) on the other hand.
4. An ontology should only be concerned with the essential aspects of production and communication in an organization, not e.g. with how actors communicate. This is a matter of implementation.
5. The meta ontology should be specific for organizations. Ideally, it is founded on the socionomic laws that govern human collaboration, in much the same way as physical laws govern the behavior of physical objects.

As has been explained in e.g. [6, 7, 8, 17], the scientific roots of DEMO are the philosophy of Mario Bunge, in particular the volumes on ontology [4, 5], the language/action perspective [1, 11, 20, 21, 22], and semiotics [16, 19], but also general systems theory [3] and logic [24]. The knowledge from these sources has been combined in the **Ψ-theory** about organizations that underlies the methodology; Ψ (to be pronounced as PSI) stands for Performance in Social Interaction. It has proven, in many practical projects, including large civil engineering projects and health care projects [13], to be an exceptionally appropriate paradigm for (re) designing and (re) engineering organizations.

The outline of the paper is as follows. In section 2, a concise summary of the **Ψ-theory** is provided, just enough for understanding the remainder of the paper. The proposed meta ontology is introduced in Section 3, which also contains the formal definitions of the two ontological aspect models on which we focus in this paper. In section 4, the ontology of an case, namely a library, is developed, by applying the proposed meta ontology. Discussions of the findings as well as the conclusions that can be drawn are provided in section 5.

2 Concise Summary of the Ψ-Theory

The ontological definition of an organization is that it is a system in the category of social systems [5]. This means that the elements are socialindividuals, i.e. human beings in their ability of entering into and complyingwith commitments about the things that have to be produced in collaboration.The **Ψ-theory** provides an explanation of the construction and the operation of organizations, regardless their particular kind or branch. It is based on fouraxioms, which are presented hereafter.

The Construction Axiom
An organization consists of *actors* (human beings fulfilling an actor role) who perform two kinds of acts. By performing *production acts,* the actors bring about the

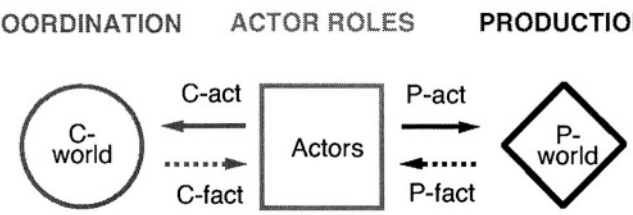


Fig. 1. The white-box model of an organization

mission of the organization. A production act (P-act for short) may be material (e.g. a manufacturing or transportation act) or immaterial (e.g. deciding, judging, diagnosing). By performing *coordination acts* (C-acts for short), actors enter into and comply with commitments. In doing so, they initiate and coordinate the execution of production acts. An *actor role* is defined as a particular, atomic 'amount' of authority, viz. the authority needed to perform precisely one type of P-act. The result of successfully performing a P-act is a *production fact* or P-fact. P-fact types in a library, for example, include "membership M has started to exist" and "the late return fine for loan L is paid". The variables M and L denote an instance of membership and loan respectively. Examples of C-acts are reqesting and promising a P-fact (e.g. requesting to become member of the library).

The result of successfully performing a C-act is a *coordination fact* or Cfact (e.g. the being requested of the P-fact "membership #387 has started to exist"). Just as we distinguish between P-acts and C-acts, we also distinguish between two worlds in which these kinds of acts have effect: the *production world* or P-world and the *coordination world* or C-world respectively (see Figure 1). At any moment, the C-world and the P-world are in a particular state, simply defined as a set of C-facts or P-facts respectively. While being active, actors take the current state of the P-world and the C-world into account (indicated by the dotted arrows in Figure 1). The P-world and the Cworld collectively constitute the UoD of an organization.

The Operation Axiom

C-facts serve as agenda[1] for actors. The operational principle of organizations is that actors feel committed towards each other to adequately and timely deal with their agenda. This principle is elegantly explained by the Habermas' theory of social action [11]; it is therefore a socionomic law. Actors interact by means of creating and dealing with C-facts. For every type of C-act there is a particular action rule. Such a rule specifies which C-fact(s) may be created and which information is needed in order to decide on creating a C-fact. In principle action rules serve as guidelines for an actor. This means that for some agendum at some instance of time, an actor may deviate from the rule because of current circumstances. Ultimately, it is the actor who is held responsible for taking appropriate action(s).

The Transaction Axiom

P-acts and C-acts appear to occur in generic recurrent patterns, called *transactions.* A transaction goes off in three phases: the order phase (Ophase), the execution phase (E-

[1] Agenda is the plural form of the Latin word 'agendum' which means 'thing to do'.

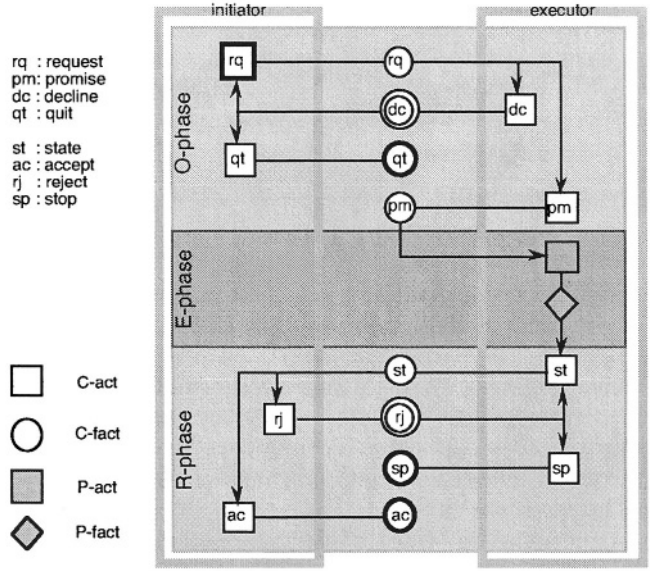


Fig. 2. The standard transaction pattern

phase), and the result phase (R-phase). It is carried through by two actors, who alternately perform acts (cf. Figure 2). The actor who starts the transaction and eventually completes it, is called the *initiator*. The other one, who actually performs the production act, is called the *executor*. The O-phase is a conversation that starts with a request by the initiator and ends (if successfully) with a promise by the executor. The Rphase is a conversation that starts with a statement by the executor and ends (if successfully) with an acceptance by the initiator. In between is the E-phase in which the executor performs the P-act. The so-called standard pattern (Figure 2) must always be passed through for establishing a new P-fact. A few comments are in place however. First, performing a C-act does not necessarily mean that there is oral or written communication. Every (physical) act may count as a C-act (cf. [21]). Second, C-acts may be performed *tacitly,* i.e. without any signs being produced. In particular the promise and the acceptance are often performed tacitly (according to the rule "no news is good news"). The complete transaction pattern consists of the standard pattern plus four cancellations patterns. Every *transaction process* is some path through this complete pattern, and every *business process* in every organization is a structure of such transaction processes [9]. Therefore, the transaction pattern must be taken as a socionomic law: people always and everywhere conduct business (of whatever kind) in conformity with this pattern.

The Abstraction Axiom

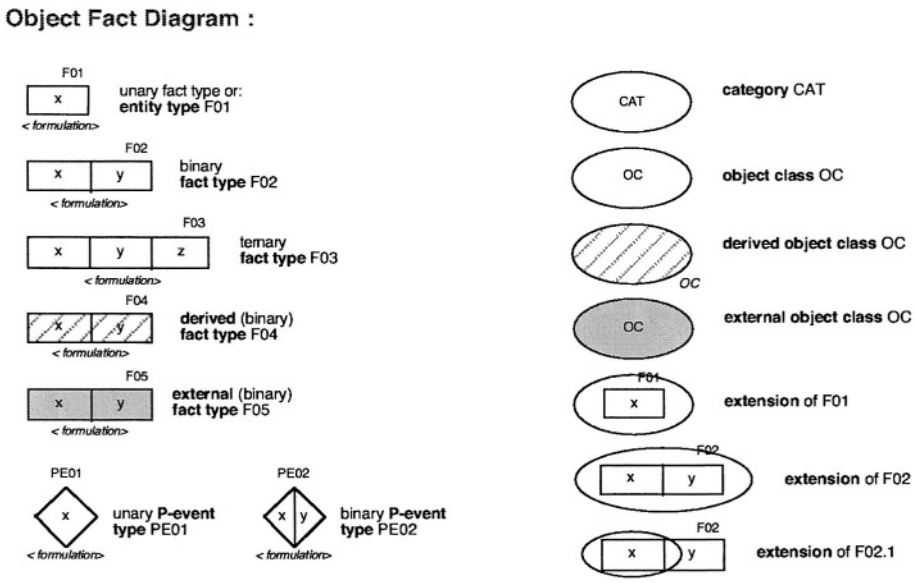
Three human abilities play a significant role in performing C-acts. They are called forma, informa and performa respectively. The *forma* ability concerns being able to produce and perceive sentences (the atomic unit of information). The forma ability coincides with the semiotic layers syntactics and empirics. The *informa* ability concerns being able to formulate thoughts into sentences and to interpret sentences.

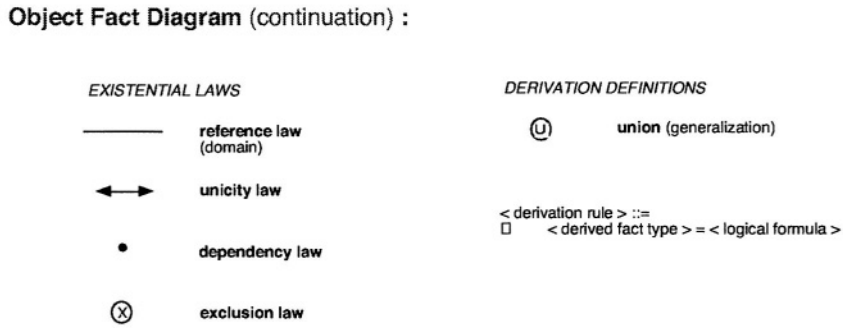
The term 'thought' is used in the most general sense. It may be a fact, a wish, an emotion etc. The informa ability coincides with the semiotic layers semantics and pragmatics. The *performa* ability concerns being able to engage into commitments, either as performer or as addressee of a coordination act. It coincides with the (organizational) semiotic layer socialics [19]. This ability may be considered as the *essential* human ability for doing business (of any kind). A similar distinction in three levels of abstraction can be made on the production side. The *forma* ability now concerns being able to deal with recorded sentences, called documents (Note. The term 'document' is used here to refer in a most general sense to the forma aspect of information). The *informa* ability on the production side concerns being able to reason, to compute, derive etc. Lastly, the *performa* ability concerns being able to establish original new things, like creating material products or making decisions. Because these acts are at the core of doing business (on the production side), they are called the *essential* production acts.

3 The Meta Ontology

In this section, we propose a meta ontology and investigate the extent to which it satisfies the requirements, as stated in section 1. We define the *ontology* of an organization as the model of the organization at the essential level of abstraction, according to the Ψ-theory. Thus, only the essential production matters are considered, not the informational and documental ones that (only) serve to support the bringing about of these essential ones. Also, only the essential level of coordination is taken into account. Thus only the performative acts, the entering into and the complying with commitments, are considered; not the supporting acts of expressing and inducing knowledge and of uttering and perceiving information. The complete ontology of an organization consists of four aspect models. The Construction Model (CM) specifies the composition, the environment and the structure of an organization (according to [5]): the identified transaction types and the associated actor roles). The Action Model (AM) specifies the action rules that serve as guidelines for the actors in dealing with their agenda: there is an action rule for every type of agendum. The State Model (SM) specifies the lawful states of the C-world and the P-world: the object classes, the fact types and the ontological coexistence rules. The Process Model (PM) specifies the lawful sequences of events in the C-world and the P-world: the atomic process steps and their causal and conditional relationships.

In this paper, only the CM and the SM are presented and discussed. The CM is expressed in the Actor Transaction Diagram (ATD) and the Transaction Result Table (TRT); the SM is expressed in the Object Fact Diagram (OFD) and the Object Property Table (OPT). The ATD and the OFD are formally defined in a language of twhich the legend is provided in Figure 3. It is a variant of the ORM language [14]. Through this property, requirement 1 (as stated in section 1) is satisfied. Space limitations prohibited us to exhibit these formal definitions. From the Ψ-theory it follows that a clear distinction is made between the cause and the effect of a change in a UoD: a C-event is the result of successfully performing a C-act and a P-event is the result of the successful completion of a transaction. Moreover, this distinction is well





Object Property Table :

< object property table > ::= { < property type > < domain > < range > ("O" | "D") }
< domain > ::= < object class >
<range> ::= < scale type > < scale kind >

Note. "O" stands for "Original", "D" stands for "Derived"

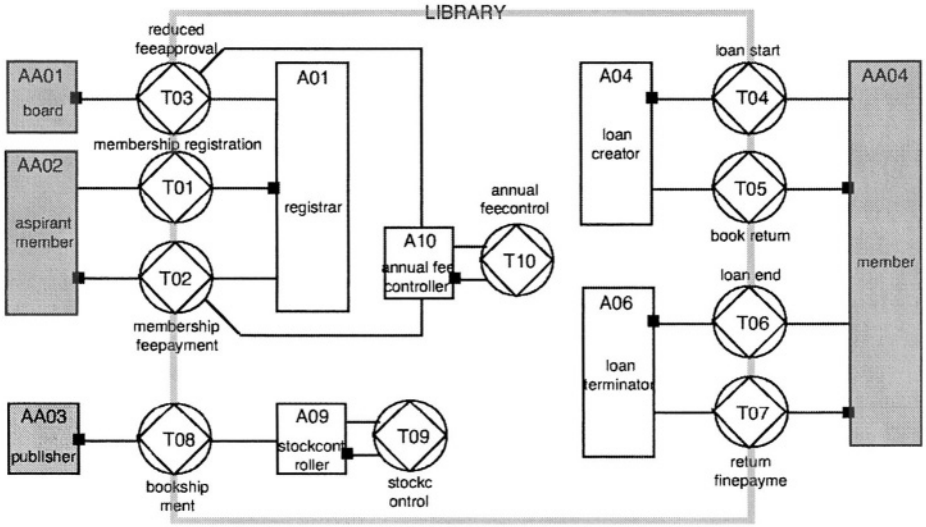
Fig. 3. Legend of the OFD and the OPT

founded in the Ψ-theory. Therefore, also requirement 3 is satisfied. Lastly, the fact that the CM and the SM are based on the Ψ-theory, which incorporates socionomic laws like the operational principle of organizations and the transaction pattern, ensures that the meta ontology satisfies requirement 5. The OPT is actually just a concise notation of all those fact types in the SM that are purely mathematical functions. These functions map to measurement scales. The scale type (Categorical,

Ordinal, Interval, Ratio, Absolute) determines the lawful mathematical operations on the scale values.

4 The Library Case

To illustrate how the ontology of an organization looks like and how it is developed, we use the library as the example organization, the same one as was taken in [10]. For a full narrative description, the reader is referred to that article. Figure 4 shows the ATD of the library case: the actor roles, transaction types, and the relationships between them. An actor role is represented by a box; the transaction symbol is a diamond (representing production) in a disk (representing coordination). A small black box on the edge of an actor box indicates that this actor role is the executor of the connected transaction type. The boundary of the considered part of the library is represented by the graylined open box. All actor roles inside the boundary are elementary: they are executor of exactly one transaction type. Actor roles outside the



transaction type		resulting P-event type	
T01	membership_registration	PE01	*membership M has started to exist*
T02	membership_fee_payment	PE02	*the fee for membership M in year Y has been paid*
T03	reduced_fee_approval	PE03	*the reduced fee for membership M in year Y has been approved*
T04	loan_start	PE04	*loan L has started to exist*
T05	book_return	PE05	*book copy C has been returned*
T06	loan_end	PE06	*loan L has ended to exist*
T07	return_fine_payment	PE07	*the late return fine for loan L has been paid*
T08	book_shipment	PE08	*shipment S has been performed*
T09	stock_control	PE09	*the stock_control for period P has been done*
T10	annual_fee_control	PE10	*the annual_fee_control for year Y has been done*

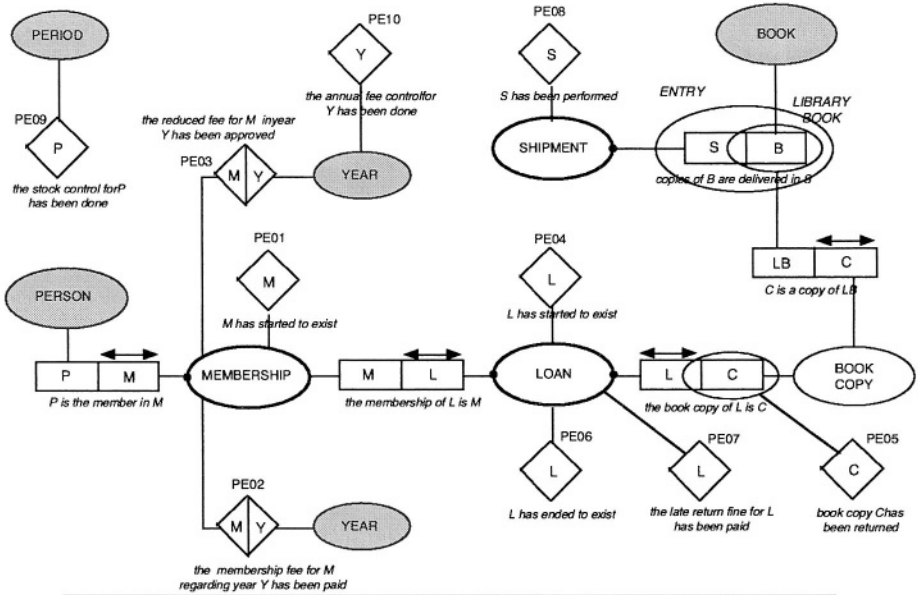
Fig. 4. The ATD and the TRT of the Library

boundary are (in principle) non-elementary. Therefore they are called aggregate actor roles; they are colored gray. The TRT below the ATD lists all transaction types and specifies for each of them the resulting P-event type. Space limitations prohibit us to provide an explanation of the CM of the library. For such an explanation the reader is referred to [10]. No information systems or flows are exhibited in the ATD. Also these things are ontologically irrelevant. Instead, the Ψ-theory states that every C-fact and every P-fact is knowable at any point in time to the actor(s) who have the right to know them, according to their action rules. For similar reasons the computation of (derived) information is not included in an ontology.

Figure 5 shows the SM corresponding to the CM of Figure 4. The OFD and the corresponding OPT specify all object types and fact types that are ontologically relevant. These are the (only) object types and fact types that appear in the AM (the action rules). Otherwise said, the SM of (a part of) an organization is an *ontological* conceptual schema of its UoD: it describes the types of things and facts (relationships) that are essential, that are and must always be there, as well as the laws that appear to hold for the co-existence of these things and facts (Note. We leave out the state model of the C-world. This is generic for all organizations because of the socionomic laws that shape the interaction (cf. section 2)). As said earlier, Figure 3 contains the legend of the OFD and the OPT. The gray-colored boxes depict external object classes. They contain objects that play a role in the business processes, but their existence is determined by transactions other than those in Figure 4. The white-colored boxes depict internal object classes. The objects in these classes are created in the mentioned transactions. For the classes Membership, Loan, and Shipment, this is obvious. For BookCopy, these are the books delivered in shipments to the library.

The diamond shaped fact types are the production fact (event) types that also appear in the Transaction Result Table of Figure 4. These fact types link the conceptual schema of the P-world to the transactions that change its state. Consider e.g. the creation and termination of loans. There are two binary fact types: “the membership of L is M” and “the book copy of L is C”. A unicity law holds for the role of the loan in both fact types: a loan always relates to at most one membership and one book copy. Also, a dependency law holds for Loan in both fact types. Hence a loan always coexists with exactly one membership and one book copy. Lastly, a new loan can be conceived of (and put in a database) but that doesn’t mean that it actually exists yet. In order to come into being, an event of type PE04 is needed. This event has a time stamp (the point in time at which it occurs). By definition this is the point in time at which the transaction T04 concerning L has successfully been completed [8]. The loan ends its existence by an event of type PE06. During the lifetime of the loan, an event of type PE07 may occur (late return fine payment).

From Figure 5 it is also apparent that no lexical object types or fact types are included in the ontology. No name classes are associated with object types and no denotations of values are associated with scales. This illustrates the satisfying of requirement 2 (in section 1). Next, there are also no fact types regarding the way of communicating between actors, like the postal address of persons. Postal (or other kinds of) addresses are ontologically not relevant; they are only needed for realizing the communication between persons. This illustrates that the ontology satisfies also requirement 4.



property type	object class	scale type + scale kind		int. / ext.
date_of_birth	PERSON	JULIAN_DATE	I	E
age (*)	PERSON	NUMBER	A	-
#days_overdue (*)	LOAN	NUMBER	A	-
incurred_fine (*)	LOAN	EURO	R	-
minimal_age	YEAR	NUMBER	A	E
standard_fee	YEAR	EURO	R	E
reduced_fee	YEAR	EURO	R	E
normal_loan_period	YEAR	NUMBER	A	E
max_#copies_in_loan	YEAR	NUMBER	A	E (=5)
daily_late_fine	YEAR	EURO	R	E
control_increment	YEAR	NUMBER	A	E
#books_of_line_item	LINE_ITEM	NUMBER	A	I
#books_in_loan (*)	MEMBERSHIP	NUMBER	A	-

age (P) = (current_date - date_of_birth (P)) {Note: age in days}
#days_overdue (L) = < (start date of L) + (normal_loan_period) - (current date) >
#books_in_loan (M) = < the sum of book copiess in loans of M that are not yet ended >
incurred_fine (L) = #days_overdue (L) * daily_late_fine (current_year)

Fig. 5. The OFD and the OPT of the Library

5 Discussion and Conclusions

The notion of ontology, as presented and elaborated in this paper, is largely a systematic, mathematical and science-oriented notion, contrary to the highly verbal or even esoteric notions that go around. We think that such a rigorously defined notion is necessary for the field of application we have in mind: the (re) designing and (re) engineering of organizations of any kind, all over and across the world. Although this notion heavily relies on the work of Mario Bunge [5], it transcends this basic and important work through the addition of socionomic laws (the operational principle of

organizations and the transaction pattern) and of the three human abilities: forma, informa and performa. These additions make our work not easily comparable with the other work in ontology that is also based on Bunge's ontology [23]. This also holds for the criticisms regarding [23], like those in [18]. An important difference is that our work is mainly based on [5] while the work of Wand and Weber is mainly based on [4].

The concept of ontology as commonly understood nowadays [25] is like a conceptual schema of a UoD. Such a schema is itself an instance of a meta schema (which by definition is an instance of itself, cf. [14, 15]). Likewise, an ontology is an instance of a meta ontology. We have stated five requirements for ontologies and meta ontologies that we consider indispensable. They make the notion of ontology something that is clearly distinguished from the notion of conceptual model, as applied in information systems design. The aspect models of the DEMO methodology, which is fully based on the Ψ-theory, appear to constitute an appropriate meta ontology for organizations, as we have demonstrated in section 3. It turned out that this meta ontology and the the notion of ontology, as we defined it, satisfy all requirements.

The proposed meta ontology has been tested in numerous practical projects. A particular extensive and meticulous evaluation has been performed in three health care organizations [13]. These experiences have lead to the conception of a reference ontology for health care providing institutions. Likewise, the ontology we have developed in section 4 for the library case may turn out to be a reference ontology for libraries. It contains (most of) the peculiarities of a library and it is abstracted fully from all implementation issues.

References

1. Austin,J.L., *How to do things with words,* Harvard University Press, Cambridge MA, 1962
2. Berners-Lee, T., J. Hendler, O. Lasilla, The Semantic Web, *Scientific American,* May 2001.
3. Bertalanffy, L. von (1968), *General Systems Theory,* Braziller, New York.
4. Bunge, M.A., *Treatise on Basic Philosophy,* vol.3, *The Furniture of the World,* D. Reidel Publishing Company, Dordrecht, The Netherlands, 1977
5. Bunge, M.A., Treatise on Basic Philosophy, vol.4, A World of Systems, D. Reidel Publishing Company, Dordrecht, The Netherlands, 1979
6. Dietz, J.L.G., Understanding and Modelling Business Processes with DEMO, Proc. 18th International Conference on Conceptual Modeling (ER'99), Paris, 1999
7. Dietz, J.L.G., Deriving Use Cases from Business Process Models, in: Song, Liddle, Ling, Scheuermann (Eds.), Conceptual Modeling – ER2003, Lecture Notes in Computer Science 2813, Springer Verlag, 2003, ISBN 3-540-20299-4
8. Dietz, J.L.G., The Atoms, Molecules and Fibers of Organizations, *Data and Knowledge Engineering,* vol. 47, pp 301-325, 2003
9. Dietz, J. L. G., Generic recurrent patterns in business processes. In: Aalst, W. van der, Hofstede, A. ter, & Weske, M. (Eds.), *Business Process Management,* LNCS 2678. Springer-Verlag, 2003.
10. Dietz, J.L.G., T.A. Halpin, Using DEMO and ORM in concert – A Case Study, in: Siau, K. (ed), *Advanced Topics in Database Research,* vol. 3, IDEA Group Publishing, London, 2004

11. Falkenberg, E.D., (ed.), *A Framework for Information Systems Concepts,* IFIP 1998 (available as web edition from www.wi.leidenuniv.nl/~verrynst/frisco.html)
12. Habermas, J., *Theorie des Kommunikatives Handelns,* Erster Band, Suhrkamp Verlag, Frankfurt am Main, 1981
13. Habing, N., J.L.G. Dietz, P.J. Toussaint, J.H.M. Zwetsloot, A transaction-based generic model of care processes, in: *Methods of Information in Medicine,* 2004 (forthcoming).
14. Halpin, T.A., *Information Modeling and Relational Databases.* San Francisco: Morgan Kaufmann, 2001
15. Nijssen, G.M., and T.C. Halpin, *Conceptual Schema and Relational Database Design,* Prentice Hall, Sydney, 1989.
16. Peirce, C., 1958, *Collected Papers of Charles Sanders Peirce,* Cambridge Mass. 17. Reijswoud, V.E. van, J.B.F. Mulder, J.L.G. Dietz, Speech Act Based Business Process and Information Modeling with DEMO, *Information Systems Journal,* 1999
18. Rosemann, M., P. Green, Developing a meta model for the Bunge-Wand-Weber ontological constructs, *Information Systems,* 27, pp 75-91, 2002
19. Stamper, R., Liu, K., Hafkamp, M. & Ades, Y. (2000) "Understanding the Roles of Signs and Norms in Organizations", *Journal of Behavior and Information Technology.*
20. Searle, J.R., *Speech Acts, an Essay in the Philosophy of Language,* Cambridge University Press, Cambridge MA, 1969
21. Searle, J.R., *The Construction of Social Reality,* Allen Lane, The Penguin Press, London, 1995
22. Winograd, T, F. Flores, 1986. *Understanding Computers and Cognition: A New Foundation for Design.* Ablex, Norwood NJ.
23. Wand, Y., R. Weber, Mario Bunge's Ontology as a formal foundation for information systems concepts, in: P. Weingartner, G.J.W. Dorn (Eds.), *Studies on Mario Bunge's Treatise,* Rodopi, Atlanta, 1990, pp. 123-149.
24. Wittgenstein, L., *Tractatus logico-philosophicus* (German text with an English translation by C.K. Ogden), Routledge & Kegan Paul Ltd, London, 1922
25. W3C, *OWL, Web Ontology Language Overview,* http://www.w3.org/TR/2004/RECowl-features-20040210/

A Model-Driven Approach to Designing Cross-Enterprise Business Processes

Bernhard Bauer[1], Jörg P. Müller[2], and Stephan Roser[1]

[1] Programming of Distributed Systems,
Institute of Computer Science, University of Augsburg, D-86135 Augsburg, Germany
{bauer | roser}@informatik.uni-augsburg.de
[2] Siemens AG Corporate Technology, Intelligent Autonomous Systems,
Otto-Hahn-Ring 6, D-81739 München, Germany
joerg.p.mueller@siemens.com

Abstract. Modeling and managing business processes that span multiple organizations involves new challenges, mainly regarding the ability to cope with change, decentralization, and the required support for interoperability. In this paper, we present an approach to modeling cross-enterprise business processes based on the model-driven architecture (MDA). Starting from the ATHENA interoperability architecture, we propose a conceptual architecture for cross-enterprise business processes. Then, we present a methodical approach towards designing cross-enterprise business processes based on a model-driven architecture. The core contribution of the paper is a set of original mappings at and across different layers of the model-driven architecture, including a mapping from ARIS to UML and the Business Process Definition Metamodel (BPDM).

1 Introduction

A key to maintain competitiveness is the ability of an enterprise to describe, standardize, and adapt the way it interacts with suppliers, partners, competitors, and customers. In the context of process orientation, enterprises today describe these procedures and interactions in terms of business processes, and invest huge efforts to describe and standardize these processes. The near future will bring an extension of these efforts towards cross-enterprise business processes. Modeling and managing business processes that span multiple organizations involves new challenges, mainly the ability to cope with change, decentralization, and the support for interoperability. Parts of the work reported on in this paper are motivated from the European Integrated Project ATHENA [1]. ATHENA addresses the vision of seamless interoperation of distributed enterprises across and beyond Europe, focusing on interoperability[1], but also covering aspects such as cross-enterprise business process modeling and architectures and platforms for business process management and enactment (see also Section 3 for more details).

[1] In the ATHENA context, interoperability is "the ability of two or more systems or components to exchange information and to use the information that has been exchanged" [4].

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 544–555, 2004.
© Springer-Verlag Berlin Heidelberg 2004

The focus of this paper is on business process modeling, as opposed to run-time business process management or enactment. However, given the ultimate goal to support end-to-end business processes, we need to provide a process of gradual transition from abstract conceptual descriptions of business process to concrete, executable business processes. Our approach towards this end is *model-driven development and architecture* (MDA) as promoted by the Object Management Group (OMG). Within MDA the software development process is driven by the activity of modeling the business software system. One of the major differences between MDA and traditional development processes lies in the nature of the artifacts that are created during the development process. These artifacts are formal models, i.e. models that can be understood by computers and finally be transformed into a representation that lends itself to execution which can effectively supported, e.g. by a web services ICT infrastructure.

In this paper, starting from the ATHENA interoperability architecture, we propose a conceptual architecture for cross-enterprise business processes. Then, we present a methodical approach towards designing cross-enterprise business processes based on a model-driven architecture. The core contribution of the paper is a set of original mappings at and across different layers of the model-driven architecture. In particular, the model transformations we describe are: (1) Mapping from ARIS to an UML2 business process representation (adherent to the Business Process Definition Metamodel specification [5]) at the computation-independent model (CIM); (2) mapping from an UML2 representation at the CIM level to an UML2 representation at the platform-independent model (PIM) (also adhering to BPDM).

2 Background

The next two sections outline the state-of-the-art in business-process related software architecture and IT infrastructure. They present related work and standards in business process modeling, and outline the ATHENA interoperability definition and architecture underlying our work.

The Model Driven Architecture (MDA) is a framework for software development driven by the Object Management Group (OMG). The following three models are at the core of the MDA: (1) *Computation Independent Model (CIM):* This is the most abstract model within MDA independent of computational technology; (2) *Platform Independent Model (PIM):* This model is defined at a high level of abstraction; it is independent of any implementation technology. It describes a software system that supports some business. Within a PIM, the system is modeled from the viewpoint of how it best supports the business; (3) *Platform Specific Model (PSM):* In the next bin terms of the implementation constructs available in one specific implementation technology. A PIM is transformed into a PSM for each specific technology platform. Most systems today span several technologies; therefore it is common to have many PSMs with one PIM. The final step in the development is the transformation of each PSM to code. Because a PSM fits its technology rather closely, this transformation is relatively straightforward.

Today, end-to-end business processes are the focus of internal and cross-company integration. To support this diagramming approaches that can adequately represent the process modeling requirements are needed. The following are main standards addressing business process modeling:

- **ARIS** (Architecture of integrated Information Systems) (see [8]) is a framework for development and optimization of integrated information systems. In this context the ARIS concept serves as model for creating, analyzing, and evaluating business management process chains. Thus ARIS allows the description of business processes and the complexity is reduced by decomposing them into different views. ARIS is commonly used by in specifying the business view of processes.
- **EDOC** – Enterprise Distributed Object Computing (EDOC) is an OMG-supported effort to simplify the development of component based systems by means of a modelling framework, based on UML 1.4 and conforming to the MDA. The business process modelling features of EDOC are now addressed by UML2 diagrams and therefore EDOC is not considered in this paper.
- **BPMN** – The Business Process Modelling Notation (BPMN) specification, produced by BPMI provides a graphical notation for expressing business processes in a Business Process Diagram. The objective is to support process management by both technical users and business users by providing a notation that is intuitive to business users yet able to represent complex process semantics. UML2 added many of the diagramming elements from BPMN to the UML2 diagram family.
- **UML2.0** – The Unified Modeling Language is a language for visualizing, specifying, constructing and documenting software artifacts. It is a general-purpose modeling language that can be used with all major object and component methods and applied to all application domains. Its extensive use has raised numerous application and implementation issues by modelers and vendors. UML2 was produced to address many of these issues — including business process modeling [10].
- **BPDM** – Resulting from an OMG Request for Submission, the primary objective of the Business Process Definition Metamodel (BPDM) initiative is to provide an abstract model for the definition of business processes (see [5]). BPDM is specified as a UML2 profile enabling generic UML tools to both author or consume business models. As BPDM provides basic concepts from business process modelling, such as processes, tasks, rules, transactions, workers, and organizations, as first-class citizens, and additionally provides support for the modelling of collaboration, it appears a promising approach to combine the openness and genericity of UML with the expressiveness and vocabulary required for business process modelling. Mappings from a business-level model directly to runtime model like J2EE or BPEL4WS need to be defined and supported by tools. There are numerous activities towards this end, some of them carried through within the ATHENA project.

3 The ATHENA Interoperability Architecture

The ATHENA project [1] attempts to contribute towards the vision of seamless interoperation of distributed enterprises across and beyond Europe, focusing on the prob-

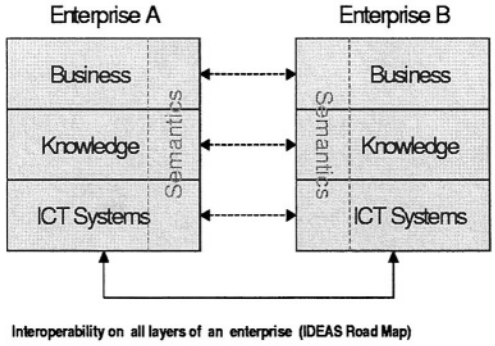


Fig. 1. The ATHENA business interoperability architecture

lem of interoperability, but also addressing aspects such as cross-enterprise business process modeling as well as architectures and platforms for business process management. Fig. 1 illustrates the conceptual architecture of ATHENA. The architecture addresses cross-organizational interoperability at three related levels: The *business level,* the *knowledge level,* and the *information and communication technology (ICT) level.* At the business level, all issues related to the organization and the operations of an enterprise are addressed, i.e. the way an enterprise is organized, how it operates to produce value, how it manages its relationships, etc.. Interoperability at business level is the organizational and operational ability of an enterprise to cooperate with other organizations. The knowledge level deals with acquiring a deep and wide knowledge of the enterprise. This includes knowledge of internal aspects such as products or the way the administration operates and controls as well as knowledge of external aspects such as partners and suppliers or laws and regulations. Furthermore, speed of changes tends to increase and the knowledge of the environment, in its widest accepted meaning, becomes more important, and sometimes even vital for the success of the business. Finally, the ICT Systems level focuses on the ICT solutions that allow an enterprise to operate, make decisions, and exchange information within and outside its boundaries. Interoperability at ICT Systems level should be seen as the ability of an enterprise's ICT systems to cooperate with those of other, external organisations.

In this paper, we focus on the business and ICT systems level. We regard the knowledge level as somewhat orthogonal to business and ICT systems level, in that different abstractions and representations of knowledge will be required at business and ICT level respectively.

In the following section, we discuss options for a conceptual modeling architecture before we describe our methodical approach and the MDA mappings in Section 5.

4 Modeling Architecture for Cross-Enterprise Business Processes

In order to enable business processes to collaborate with partners and to facilitate the composition of business processes, we apply the paradigm of service-oriented archi-

as components which provide services to and consume services from other business process components. Interacting business processes form a network of interconnected processes where conversations are conducted. The same process can appear in multiple solutions and can be connected to different partners in each case. The service-oriented approach to business process modeling provides a natural view to distributed business systems and lends itself to a mapping into an ICT perspective.

In describing cross-enterprise business processes, [2] suggests distinguishing between an *internal* and an *external view.*

- The *internal view* models the 'how' of a business process. In [3] processes, which model process flow as a set of partially ordered *tasks,* are called *executable processes.* There the flow of the processes' interactions is described from the point of view of a single process, which coordinates participating sub-process. This kind of process composition is referred to as process *orchestration.*
- The *external view* models the 'what' of a business process. Each process specifies its roles but does not give any indication about its own realization. Beside the definition of interfaces it includes a description which process types can invoke the interfaces' operations as well as which process types are invoked through the service itself. This *external view* on such business process components is described in *abstract processes.* Since a*bstract processes* only provide interfaces for business process invocation, the interactions between processes are modeled by *business protocols,* specifying conversations of business processes as interaction patterns between their roles from the viewpoint of an external observer. Thereby only interactions visible to an external observer, i.e. the cross-enterprise process, are modeled. [2] refers to this as *collaborative processes.*

Business protocols can be realized in multiple ways, which differ in how the business protocols' conversation flow is coordinated. Our architecture for modeling cross-enterprise business processes relies on the broker approach. An intermediary acts as a global observer process coordinating the partners, which take part in the cross-enterprise business process. There are several reasons for applying a broker pattern:

- Without a broker one executable process per partner has to be implemented, which coordinates the partners' activity in the business protocol. This is problematic since control flow logic of one cross-enterprise business process is divided into different executable process. Due to the mutual exchange of messages these processes depend on one another. Changing the business protocol would result in changing multiple executable processes.
- This disadvantage doesn't appear by implementing a broker coordinating the conversation between the communication partners. Each business protocol is realized through one of the broker's executable processes. Moreover this approach is more convincing, since cross-enterprise business process, i.e. *collaborative processes,* are modeled as separate processes from the viewpoint of an external observer.

5 Model-Driven Business Process Design

In this section, we describe the core contributions of this paper: A methodical approach to designing business processes via model-driven architecture and two mappings that implement essential parts of the model-driven approach.

5.1 Methodical Approach

Due to the fact that different kinds of experts, like economists and computer scientists, did business process modeling from different angles, the existence of different approaches is not surprising. In addition to the differences in the models used, we can distinguish between a top-down and a bottom-up approach in the context of MDA:

- In the top-down approach business processes are first modeled in a computational independent fashion. Corporate as well as cross-enterprise business processes are modeled by process consultants with a business background. A very common modeling methodology for this kind of business process modeling is ARIS.
- Computer scientists tend to chose a more bottom-up approach to business process modeling, starting from platform specific models, which allow automated process execution (e.g. BPEL4WS, BPML, etc.). Hence models for higher level descriptions of business processes, which can be mapped to those process execution languages, like the Business Process Definition Metamodel (BPDM) are emerging. Since for most IT systems object-oriented methods have become accepted as a standard, it is obvious that these systems are modeled in UML. In addition, UML offers the advantage of openness and (to a large extent) vendor independence.

In such an environment it is important for developers of IT systems to ensure that the ARIS models of economists are consistent with the BDPM models of computer scientists. The change of modeling methods is a very crucial point in the development of an IT system, since mistakes in process modeling made here are rarely found before the system is deployed.

In this paper we introduce a mapping between ARIS and BPDM. This mapping makes it possible to deduce BPDM-Models at PIM level from ARIS-Models at CIM-level. Processes at PIM level shall be described in such a way, that they can be transformed to process execution languages on PSM level. The choice of an UML-based business process representation at the CIM and PIM levels addresses the tendency towards improving interoperability through choosing open standards for business process representation. At the same time, supporting a mapping from ARIS into BPDM takes the fact into account that ARIS is the leading modelling business process methodology in industry and tries to improve the impact of the work done in ATHENA by widening its scope, hence making it easier for enterprises to migrate their business process models into a model-driven framework. Finally, choosing BPDM as a vendor-independent representation will allow to leverage the support for modelling collaboration which will be provided in BPDM and makes our approach particularly suitable for the modelling of decentralized, cross-enterprise processes.

The mapping is performed in two stages:

- Transformation of ARIS models to object-oriented BPDM models, which are as far as possible equivalent, whereas static and dynamic structures have to be considered. The focus is on the transformation of the process-oriented event-driven process chains, which are used for the representation of dynamic aspects in ARIS.
- Generation of BPDM models at PIM level out of BPDM models at CIM level. Tools which implement the MDA approach will provide implementations to perform this task automatically.

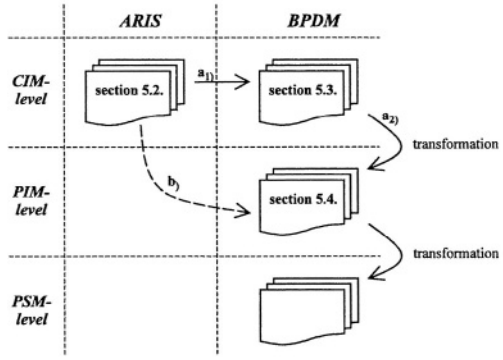


Fig. 2. Model-driven business process design approach

As illustrated in Fig. 2, there are two alternatives in developing the ARIS-BPDM mapping. In alternative a) each of the two identified tasks is carried out in a separate step. Alternative b) suggest performing the mapping, i.e. both the transformation from ARIS to BPDM and generation of PIM models from CIM models, in a single step. In the following, we shall briefly discuss advantages and disadvantages of these alternatives.

An advantage of alternative b) is that the mapping is conducted in one single step. It avoids the development and therefore the existence of a (redundant) second model at CIM level. Though it would be possible to realize the mapping in one single step, there are a number of aspects we consider problematic. A parallel and nested execution of the mapping, in reference to the mapping of ARIS to BPDM and the transformation of CIM level to PIM level, leads to a large, monolithic and poorly comprehensible mapping. Moreover, two different and insufficiently integrated modelling languages are used for describing the development of one ICT system, though integrating different models is one of the main points required by the MDA.

These disadvantages do not occur in alternative a), where the mapping is separated into two tasks as described above. Hence, the mapping becomes more transparent and easier to understand for the user. It is easier to track which model elements have their origin in ARIS diagrams and which model elements were generated through the transformation to PIM level. Furthermore, it is possible to use completely integrated models for system development. Even the existence of two models for one system at CIM level need not be a disadvantage since different models may appeal to different types of users (i.e. ARIS to business users and BPDM to users with an ICT perspective).

For the development of an ARIS-BPDM mapping we choose alternative a) and sum up the correlations and differences between various model types. ARIS models are common for describing corporations and their business processes from an economical point of view at CIM level. With BPDM models system developers describe the business processes and corporation's aspects from an object-oriented point of view at CIM level. At PIM level BPDM models describe business processes and aspects of the corporation, which are relevant for the ICT system in development, in a more detailed way. Finally on PSM level runtime models, like for example automatable business processes models, are described.

In the remainder of this section we present a mapping from ARIS models at CIM level to BPDM models at PIM level. The mapping is showed by an example. The example follows the architecture for modeling cross-enterprise business processes in Section 4. Since the focus of the mapping is the transformation from process-oriented to object-oriented descriptions of the control flow, the examples cover only models which are relevant for the description of control flows.

5.2 ARIS Model at CIM Level

Process structures can be modeled as value-added chain diagrams (VACDs) in ARIS. The control flow is modeled by event-driven process chains (EPCs) attached to the respective processes. EPCs consist of *functions* representing tasks which contribute to the corporation's objectives; *events* representing states, which are conditions under which a function can be executed or hold when a function was terminated; *process interfaces* indicating the passing of the control flow from one process to another one. Therefore a process interface at the beginning of an EPC specifies the process types by which the process can be invoked. Process interfaces at the end of an EPC specify which process types the process intends to invoke.

In the example (see figure 3) the processes of a buyer and a seller conducting cross-enterprise business processes are modeled. The *buy* process is partitioned into an *identify demand* and a *verify offer* process. The *sell* process consists of a *calculate offer* and a *coordinate offerings* process.

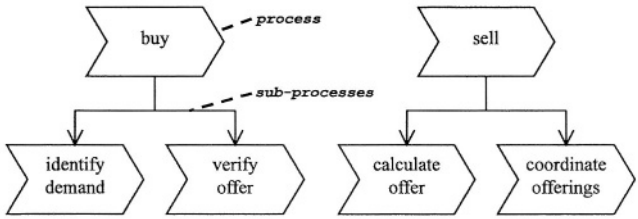


Fig. 3. Example of an ARIS value-added chain diagram

Figure 4 shows the refined EPCs for the example processes. When demand is identified by the buyer, the *identify demand* process finally invokes the seller's *calculate offer* process. This process calculates an offer and returns the result to the buyer.

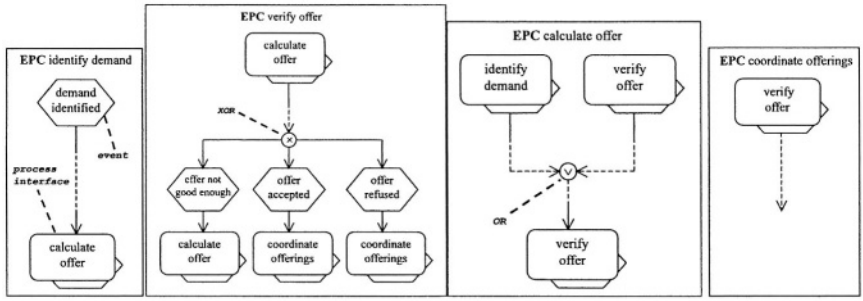


Fig. 4. Example of ARIS event-driven process chains

There the *verify offer* process decides whether the offer can be accepted or not. When the offer is accepted the process informs the seller's *coordinate offerings* process about this. In the case the offer cannot be accepted the *verify offer* process can invoke the *calculate offer* process for a new offer or inform the *coordinate offerings* process that the offer is refused.

5.3 Mapping to BPDM Model at CIM Level

Since BPDM is an extension of UML2 it is allowed to use UML2 concepts in BPDM models. In BPDM we model static aspects in class diagrams. Dynamic aspects of the collaboration between the processes of the buyer and the seller are modeled as *business protocols.* Therefore we use sequence diagrams.

Mapping rules for modeling static aspects are as follows:

- Each process from the ARIS VACD is mapped to a class with the stereotype <<*process*>> defined by BPDM (see figure 5).
- The process hierarchy from the ARIS model is realized through the *sub-process concept* specified in BPDM [5]. For modeling we use directed associations.

While information about the cross-enterprise processes in the ARIS model is contained in the EPC, i.e. through process interfaces, we use sequence diagrams to represent the business protocol interactions in UML. In figure 6 the corresponding business protocol is modeled as the *MyContractNet Protocol.*

- The processes (agents) representing buyer and seller respectively conduct a negotiation. They are modeled as *lifelines.*
- In the sequence diagram internal behavior of e.g. the seller process is not visible.

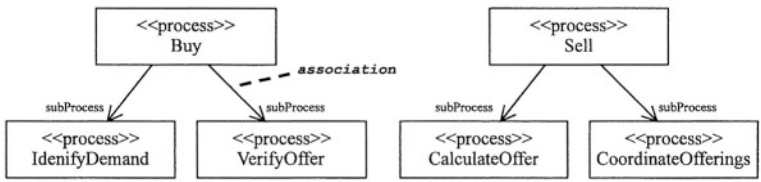


Fig. 5. BPDM – class diagram at CIM level

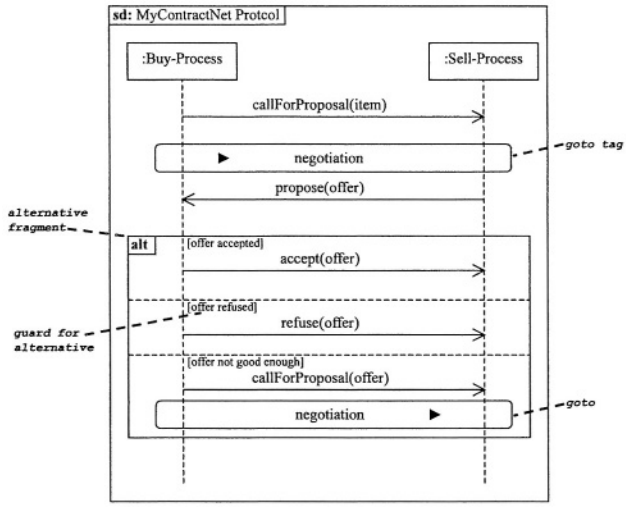


Fig. 6. BPDM – business protocol modeled in sequence diagram

- *Messages* are derived from the *process interfaces* of the EPCs in ARIS. Parameters of the messages, which are not visible in the discussed ARIS diagrams, are added to obtain a more complete model.
- Decisions in the control flow are mapped to *combined fragments,* e.g. the *alternative fragment* which models whether an offer is accepted or not.
- *Guards* are derived from the *events* in the EPCs and specify conditions for the alternatives to be taken.

5.4 BPDM Model at PIM Level

At PIM level business processes und business protocols are modeled from the IT implementation point of view. According to our modeling architecture of cross-enterprise business processes above, a broker (see figure 7) is introduced as a collaborative process coordinating the control flow of the business protocols.

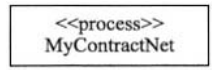


Fig. 7. BPDM – process class of the broker

For modeling the broker process's internal behavior as an executable process we use an activity diagram. Following constructs can be found in the activity diagram:

- The *control flow* identifies sequencing of activities. *Data flow* identifies the flow of data objects between activities. Data objects being in- and outputs for actions are modeled as *PINs.* The names of the PINs specify the type of the data objects exchanged.

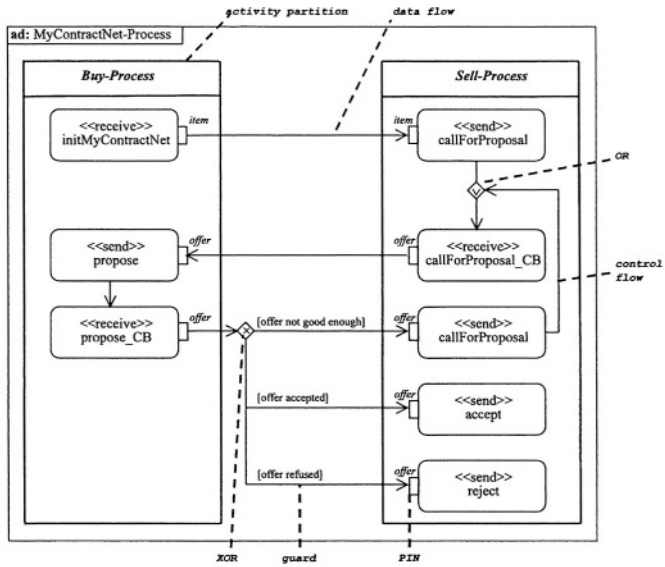


Fig. 8. BPDM – modeling cross-enterprise business process control flow in an activity diagram

- BPDM [2] describes actions stereotyped <<*send*>> or <<*receive*>> to model inter-process communication. The corresponding partner processes are identified by *activity partitions* in which these actions are modeled.
- *Decision nodes* and *merge nodes* are used to model alternatives of the control flow. Which alternatives have to be taken is specified with *guards.*

Although the activity diagram models the internal view of the broker process, i.e. the executable process, elements used to model the external view of a process are also present, e.g. the stereotyped actions are also used to describe abstract processes.

In Fig. 8, the *MyContractNet* protocol is depicted in an activity diagram as an executable process. We give a brief description of main tasks performed by transforming the sequence to an activity diagram:

- Lifelines of a sequence diagram become activity partitions in an activity diagram.
- Sequence diagram messages lead to stereotyped actions in an activity diagram. In the example the initial receive action *initMyContractNet* starts the broker's process. Due to the *callForProposal*-message of the sequence diagram the broker process sends a *callForProposal* to the sell agent's process. Since the *callForProposal*-message contains the parameter *item,* a data flow between the two actions is modeled. As we use asynchronous messaging, the *callForProposal*-message's results, i.e. an *order,* are passed to the broker's process in the next receive action *callForProposal_CB.* The broker's process sends the *propose* to the sell agent's process.
- The *alternative fragment* of a sequence diagram, which models whether an order is accepted or not, is realized as a *decision node* in the activity diagram.

6 Conclusions and Outlook

The contribution of this paper is threefold: First, we present a conceptual architecture for modeling cross-enterprise business processes based on a model-driven architecture; second, we propose a design approach suitable to MDA, and third, we provide specifications of two model transformations (mappings) to implement the design approach, thus enabling the smooth transition from an ARIS model via a computation-independent BPDM model to a platform-independent BPDM model.

Future work will complete the approach by adding mappings to selected platform-specific models including the BPEL4WS language [3]. We will also explore richer models of cross-organizational business processes, such as that proposed by [9], to address a wider range of central and decentralized collaboration architectures. A third strand of work will be to enhance adaptability of runtime process infrastructures by a Robust Planning and Execution Layer that combines dynamic service discovery and composition as well as richer transaction and compensation in business process execution. Preliminary work in this direction has been published in [7].

Acknowledgements. Part of this work reported in this paper has been funded by the ATHENA IP under the European grant FP6-IST-507849. This paper is partly funded by the E.C. through the Athena IP. It does not represent the view of E.C. nor that of other consortium members, and authors are responsible for the paper's content.

References

1. ATHENA. Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks and their Applications. Europ. IP FP6-IST-507849. http://www.athena-ip.org
2. Frank, Gardner, Johnston: Business Process Definition Metamodel – Concepts and Overview, IBM, 2004
3. IBM, Business Process Execution Language for Web Services Version 1.1, http://www.ibm.com/developerworks/library/ws-bpel/
4. IDEAS. Interoperability Developments for Enterprise Application and Software– Roadmaps. European Project IST-2001-37368. http://www.ideas-roadmaps.net
5. Iyengar et al. Business Process Definition Metamodel. Revised Submission to BEI RFP bei/2003-01-06. 2004
6. Kleppe, Warmer, Bast. MDA Explained – The Model Driven Architecture: Practice and Promise, Addison Wesley, 2003
7. Müller, Bauer, Friese. Programming software agents as designing executable business processes: a model-driven perspective. Vol. 3067 of Lecture Notes in AI, Springer, 2004
8. Scheer. ARIS: Modellierungsmethoden, Metamodelle, Anwendungen, Springer 1998
9. Schulz, Orlowska. Architectural Issues for Cross-Organisational B2B Interactions, International Workshop on Distributed Dynamic Multiservice Architectures at ICDCS-21, IEEE Computer Society Press, 2001
10. UML Homepage. The Object Management Group. http://www.omg.org/uml/
11. W3C Web Services glossary. http://www.w3.org/TR/ws-gloss/

A Component-Based Architecture for Protocol Vector Conversion in Inter-organizational Systems

Bettina Bazijanec, Christian Winnewisser, Antonia Albani, and Klaus Turowski

Business Informatics and Systems Engineering
Business Faculty, University of Augsburg,
Universitätsstraße 16, 86135 Augsburg, Germany
{bettina.bazijanec, christian.winnewisser, antonia.albani,
klaus.turowski}@wiwi.uni-augsburg.de

Abstract. Inter-organizational systems (IOS) that enable cross-enterprise value creation have moved in the focus of both the scientific and business community. However, a lack of formal methodological backing, and the absence of proven collaboration concepts and architectures have up to now limited the impact of IOS in practical application. In this paper, we focus on external coordination requirements in IOS based on an analysis of B2B protocol standards, and system theory based classification of IOS elements. A protocol vector is introduced to represent the set of protocols used in a particular collaboration scenario. A flexible component-based architecture is proposed in order to implement the protocol vector. Therefore, a process for derivation of required components is introduced, and a selected part of the architecture is presented.

1 Introduction

Economic enterprises are facing an increasingly complex competitive landscape. Decisive factors such as innovative pressure in products, services and processes, globalization of markets and companies, decreasing customer loyalty and scarce resources are forcing companies to undergo a drastic transformation of business processes as well as organizational and managerial structures [5]. Traditional improvement programs based on the assumption of a stable environment and focused on the redesign of internal processes turn out to be insufficient to cope with the challenges as characterized above. Instead, companies have increasingly turned to advancements in information and communication technology (ICT), e.g., global infrastructure, semantic standards, distributed application systems and componentized architectures, to overcome those obstacles. At the core of the application of these new ICT is their potential to enable companies to co-operate on an inter-organizational level. The benefits of inter-organizational collaboration, especially with respect to flexibility and efficiency gains as prerequisites to maintain competitiveness in an unstable environment, have been well established (e.g. [29], [16], [24]) and new ICT have been identified as a key-enabler for collaborative value creation in business networks that cross company boundaries (e.g. [23]). In analogy to ERP systems in an intra-company context, this kind of collaboration supports inter-organizational business processes, thereby exceeding the range of application of traditional EDI, which is primarily

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 556–567, 2004.
© Springer-Verlag Berlin Heidelberg 2004

limited to inter-organizational data interchange. The use of ICT for cross-enterprise collaboration in the broader context as characterized above has been subsumed under the term *Inter-organizational System* (IOS). Based on previous work in the field of cross-enterprise collaboration (see [2]) , the authors define IOS as a computer-aided, network-based, distributed, cross-enterprise information system, that

- Enables collaborative value creation in business networks,
- Reaches beyond enterprise boundaries, thereby reshaping the structure of companies and industries,
- Supports cross-enterprise business processes,
- Utilizes modern communication infrastructures and open networks,
- Increases the efficiency and effectiveness of cross-enterprise collaboration by exploiting the potential of computer-aided information systems,
- Increases the flexibility to adapt to fast-changing competitive environment.

However, in the practical application of the IOS concept, some major problems remain, predominantly in the ICT-supported inter-organizational coordination of business processes and the subsequent handling of corresponding internal application flows (see [30], [27]). [9] and [20] demonstrated in extensive literature reviews that those deficiencies are partly rooted in a lack of operationalization of the IOS concept, especially with respect to proven collaboration concepts and architectures. A reason for that can be found in insufficient backing by systems theory to define the constituting elements of an IOS as a basis for respective collaboration concepts.

In this paper, we introduce the concept of a component-based architecture for IOS, based on a systems theory backed description of IOS, to address the coordination problem as described above. In section 2, the constituting elements of an IOS are defined and affected system objects for the IOS architecture are isolated. In section 3, types of B2B protocols are identified that are required to coordinate distributed systems. The concept of a protocol vector is introduced to denote a particular set of protocols valid in a collaboration scenario. Section 4 proposes a domain-specific component modeling process to derive a component-based architecture that allows to flexibly adopt to different protocol vectors. In section 5 this process is exemplarily applied to inter-organizational message exchange in order to derive components for the protocol conversion task. In section 6 conclusions are drawn and an outlook is given.

2 Identification of IOS System Elements

Figure 1 illustrates the definition of an IOS based on Ropohl's technological systems theory [21]. The technological systems theory is partially based on findings from operations research, the general systems theory and cybernetics and is proven as a means of specifying socio-technical systems. An inter-organizational system is a socio-technical system that consists of several sub-systems. The general specification (e.g. range, participation structure, cooperation concept) of an IOS is defined in the overall target system. Apart from the overall target system, an IOS consists of several socio-technical sub-systems that represent the enterprises that participate in an IOS and a sub-system that represents the interlinking between the participating enterprises.

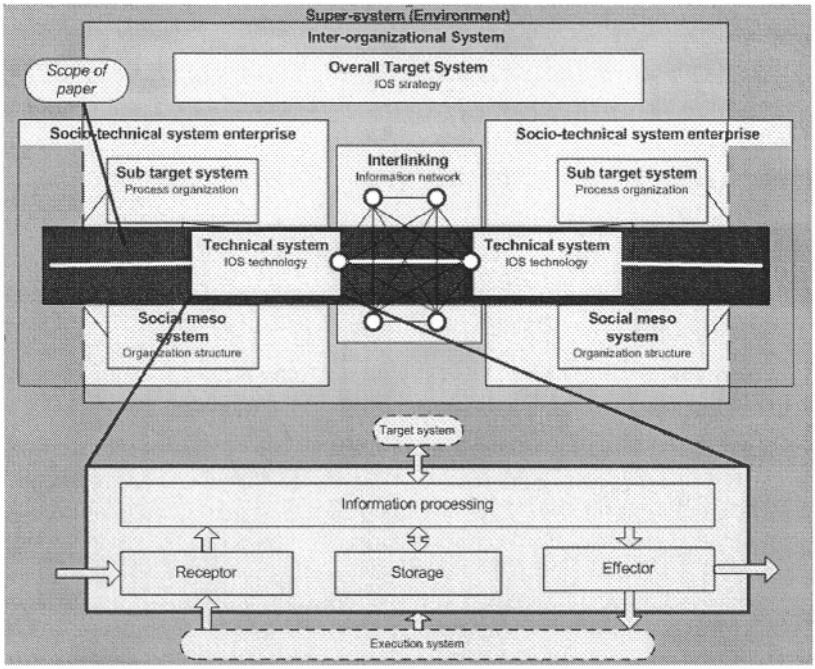


Fig. 1. Inter-organizational system elements

It is important to notice that not all sub-systems within an enterprise, which are by themselves necessary to describe the system enterprise in total, are actually participating in the cross-enterprise value creation. Therefore, only the sub-systems that actively participate in the IOS are represented in the description of an IOS, whereas the remaining sub-systems of the enterprise can be viewed as a black box, to which the IOS sub-systems have intra-enterprise linkages. Among the IOS related sub-systems within an enterprise are the sub target system, representing the process organization, the social meso system, representing the organization structure and the technical system, representing the IOS technology. The enterprises participating in the IOS are linked with each other via their technical systems.

For the scope of this paper, the external linkage between the technical systems of the participating enterprises and the internal linkage between the technical system and the black-box elements of the respective enterprise are relevant to address the inter-organizational coordination problem and to develop an IOS architecture. Therefore, in this paper we focus on the technical systems that provide connectivity between the business partners and hence allow information exchange that is necessary to coordinate inter-organizational business processes. Information exchange in terms of the technical systems theory takes place when one system sends out information as output to its environment and another system receives this information in form of an input [21]. To coordinate an inter-organizational process this input has to be processed by the receiving system and information has to be send back. This processing is done by the transformation of inputs into outputs dependent on specific internal states, including states that represent goals of the target system [22]. Figure 1 shows the general architecture of IOS technology that consists of four sub-systems: *information process-*

ing, receptor, storage, and *effector.* A receptor is needed to receive the inputs that are then processed or stored by the sub-systems information processing respectively storage. Storage is also needed for preserving internal states. Generated outputs can be sent out by the effector. As already mentioned, IOS technology can exchange information with two types of communication partners: first, with internal enterprise systems, namely application systems, second with the business partners' technical systems over the information network. Therefore, there are two coordination problems to be solved by the technical sub-system of an IOS: the external coordination problem caused by interaction with one or more business partners with potentially different process implementations, and the internal coordination problem caused by the usage of existing application systems for implementing business processes (see figure 2). The goal is to solve both problems simultaneously.

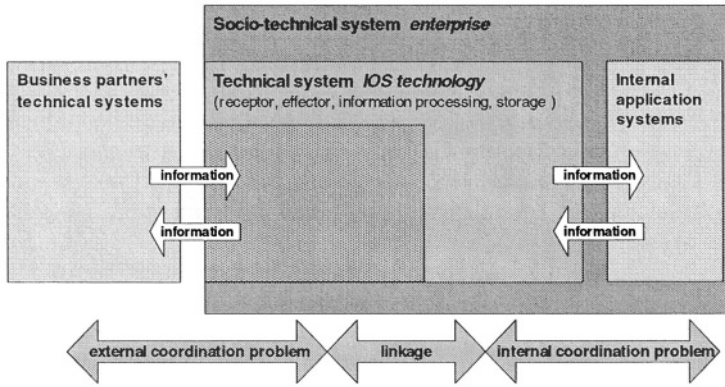


Fig. 2. External and internal coordination problems

This paper primarily focuses on concepts relevant to the external coordination problem and parts of its linkage to the internal coordination problem. The connection of internal applications to IOS technology and coordination to represent the internal business process are not within the scope of this paper. Therefore, the defining characteristics of inter-organizational collaboration will be discussed next to identify particular external coordination requirements.

3 External Coordination Requirements

The main goal of business collaboration is to interconnect internal business processes of participating enterprises in a way that common goals can be achieved but own process details do not have to be disclosed to preserve autonomy. As no direct access to internal application systems will be allowed for business partners, linking of their technical sub-systems is achieved by message exchange. It is crucial that messages reach business partners on time and that they contain exactly the information necessary to complete the next internal process step. Different assumptions about the message exchange may cause interoperability problems that make collaboration impossible. Conflicting assumptions, derived from a more general consideration of interacting systems in [14], can help to explain problems in a collaboration scenario:

- *Assumptions about the presence of partners or communication connectors:* Problems may occur if sender and receiver do not support the same communication channels or if it is not known to which business partner or which particular address messages have to be sent.
- *Assumptions about incoming and outgoing data:* Since every enterprise has its own representation of business data (i.e. different syntax or semantics), exchanged information might not necessarily be understood by business partners.
- *Assumptions about patterns of interaction:* If the exchange sequence of messages does not match it may be that information is received that is not expected by receiver's internal business process, or messages are expected which are not sent. This may lead to deadlock situations.

Collaborating enterprises therefore have to agree on *protocols.* In general, protocols describe all interaction steps between interacting systems and contain a description of syntax and semantics of data that is exchanged. Based on the protocol's message exchange sequences a *public process* can be identified that describes the sequence of sending and receiving (business) information types from one business partner's point of view (see [28], [8]). Hence, public processes describe external coordination steps whereas *private processes* coordinate applications within one organization. The linkage between public and private processes is called *binding* [8] which encompasses integration tasks such as compliance checks, data transformation, message routing, and process wrapping.

Protocols can be found on different levels of abstraction. B2B protocols define message exchange sequences for inter-organizational business processes whereas communication protocols describe exchange sequences that are used to establish a network connection and transport of payload data. Since business partners expect to exchange their data in a trusted and reliable communication, additional protocols have to be included into a B2B and communication protocol infrastructure to ensure these requirements. Many authors have discussed additional requirements for distributed protocol execution (see [13], [11], [19], [7]). Regarding different requirements for external collaboration of inter-organizational collaboration following protocol types can be identified:

- *Communication protocols:* These protocols are mostly standardized and do not affect higher-level protocols. IOS technology must be capable to send and receive data over different channels that implement these protocols (e.g. HTTP).
- *Reliability protocols:* These protocols ensure only-once delivery of messages. This includes additional acknowledgment messages and checking of time-out values.
- *Conversation protocols:* In order to be able to determine which messages belong together in a sense of conversation it is necessary to mark these messages with an identifier. Roles that are played by the conversation participants need to be defined and communicated between them.
- *Security protocols:* There are different aspects of security in a collaboration scenario, namely identification, authentication, authorization, confidentiality, and integrity. Besides knowing utilized algorithms security requires sharing of certain data between all participants e.g. public keys.

- *Transaction protocols:* These protocols provide mechanisms for enabling transaction management in the collaboration scenario to ensure a reliable outcome of a collaborative business process. They allow for example initiating rollback processes in case of failure.
- *Business protocols:* These protocols describe the exchange sequence of business data. They therefore can be mapped to the public processes as described above. According to our definition of protocols this one should not include any information about other requirements like transport or security.
- *Further protocols:* Special business protocols that represent a more general process like search, negotiation or payment.

Many of the protocol classes mentioned above are standardized. However this also means that several specifications are published providing possible solutions for a specific problem [6] (e.g. BTP and WS-Transaction in the area of distributed transactions). Together with the fact that within a standard there are several protocol alternatives or methods to generate and process protocol specific data, there is a relatively high number of protocol combinations possible. To express possible combinations we propose the concept of a *protocol vector* that represents one specific combination of protocols used for a certain collaboration scenario. An example vector is shown in figure 3:

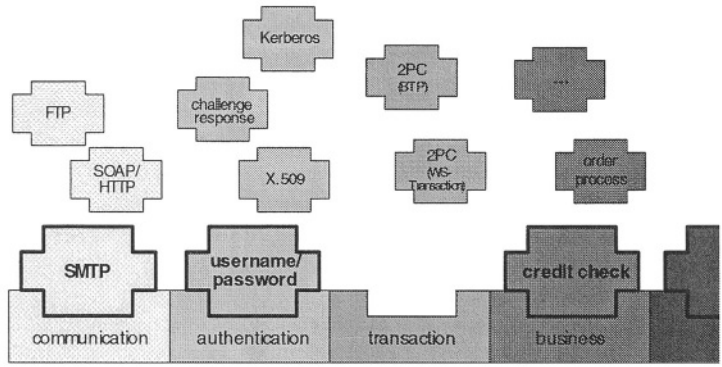


Fig. 3. Protocol vector and protocol assignment

Also, enterprises typically have a high number of potential business partners and support many business processes. Actual business relationships are built dynamically according to the current strategy and market situation. Inter-organizational collaboration is therefore subject to continuous variation. When looking at the protocol vector this implies one vector for each potential business partner and most of the vectors are different from each other. Thus, for external coordination IOS technology has to provide protocol vector conversion functionality that adapts to every possible protocol combination. Since a high degree of automation is desired for external coordination this adaption to different protocol vectors should flexibly be possible through IOS architecture. The protocol vector description represents a possible basis for such an automatic system configuration since this formalization enables dynamic negotiation of protocol combinations e.g. as already shown for communication protocols with ServiceAP [14]. Requirements to implement such a solution will be discussed next.

4 Component-Based Coordination Architecture

As described in section 3 inter-organizational collaboration is highly variable due to the variety of possible business partners with different requirements and representations of data and processes. Therefore IOS technology has to cope with many protocol implementations. An architecture is needed that provides a high degree of flexibility and reusability which allows to easily customize the IOS technology according to a given protocol vector. A well-known approach is the use of software components that allow configuration of particular systems through compositional plug-and-play-like reuse of black box components [1]. A rich literature on the general advantage of such systems exists (cf. [4], [25]). In the field of inter-organizational systems component-based architectures have been proposed ([7], [17], [10]). However, a systematic derivation has not been provided. According to [12], we define the term component as follows: *A component consists of different* (*software-*) *artifacts. It is reusable, self-contained and marketable, provides services through well-defined interfaces, hides its implementation and can be deployed in configurations unknown at the time of development.* With regard to component based application systems two types of components can be identified: system components and business components. Business components offer application specific functions of a given business domain whereas system components provide generic functions like database access [26]. A component-based architecture consists of business as well as system components. The set of joined and customizable system components, offering application invariant services to business components, is called a *component system framework.*

In an inter-organizational scenario communication and coordination tasks have to be handled by IOS technology in order to allow processing of business data by application systems. The IOS technology, responsible for communication and coordination tasks, is part of the component system framework in a component-based architecture. Applications like SCM or CRM that also may consist of several business components use services from such a component system framework (see figure 4):

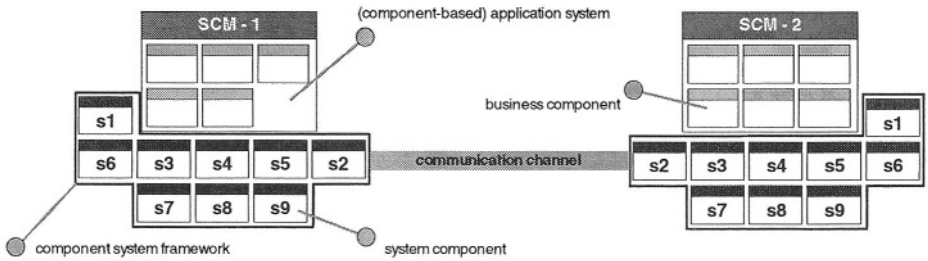


Fig. 4. Component-based IOS technology

In order to use this component-based approach for solving the described external coordination problem with the help of an protocol vector, component interfaces for each protocol class have to be identified that allow all possible protocol implementations to be plugged in as sub-components, i.e. replacing different security or communication protocols should be possible for one business process depending on business partners or other parameters. For example, a request for quote can be performed by e-

mail (SMTP) or Web Service (SOAP over HTTP) with or without transactional context or authentication. To obtain a suitable component model for IOS technology, i.e. the identification of components and relations between them, a well defined derivation process is necessary. Such a derivation process is known for business components where based on a (business) domain analysis of functions and data a business identification methodology can be applied. This process is called Business Component Modelling process (BCM) and has been introduced in [1]. We propose an adaptation of the BCM process for the identification of system components called *Domain-specific Component Modelling* (*DCM*) process. Instead of targeting at a business domain like strategic sourcing or warehouse management, a more technical domain, namely protocol vector conversion, is analyzed. Components, identified by this DCM process form an integration framework that can be considered as a sub-system of the component system framework introduced earlier.

5 Application of the DCM Process to Protocol Vector Conversion

In this section DCM is used to find components for protocol vector conversion and further tasks of B2B integration. The resulting abstract component model will refine the very general model of technical sub-systems in section 1. It shows how receptors, effectors, storage and information processing components can be disassembled to finer-grained components in order to form the information processing sub-system. In order to derive an integration framework for inter-organizational collaboration the domain of protocol vector conversion has to be analyzed. Several authors discussed this domain and related fields in their work dedicated to B2B integration. The most comprehensive work was done by the authors of [8] and [3], who identified concepts and problems relevant to B2B integration. Other authors focus more on special problems of integration such as data transformation (see [31], [18]) or mapping of public and private processes [28]. Research also exists with focus on integration of application systems in intra- and inter-organizational scenarios that describe problems and solutions for messaging based coupling systems [15].

According to the DCM process, tasks have to be identified and refined to get a function decomposition diagram. Tasks relevant to the domain of protocol conversion are presented in form of a generic process for external message reception in figure 5. Relevant data in the generic process is attached to the corresponding activity, i.e. this data is either used or generated by the activity. The process starts with *receive message* where an external message enters the system. Received messages have to be checked *(check processing data),* i.e. sender address, signature, possibly existent context data, timestamp and payload type are compared with stored protocol vector information that at runtime not only contains information about used protocol standards but also actual parameters values.

If the message passes this check it is further processed. Otherwise it will be discarded *(discard message).* If the message payload contains business data it will be decoded according to some rule *(decode payload),* i.e. decrypted and decompressed if necessary. Other messages belong to protocols that will be handled by specific protocol handler processes and routed to these processes either before *(hand over processing data to protocol-specific process)* or after decoding *(hand over payload to proto-*

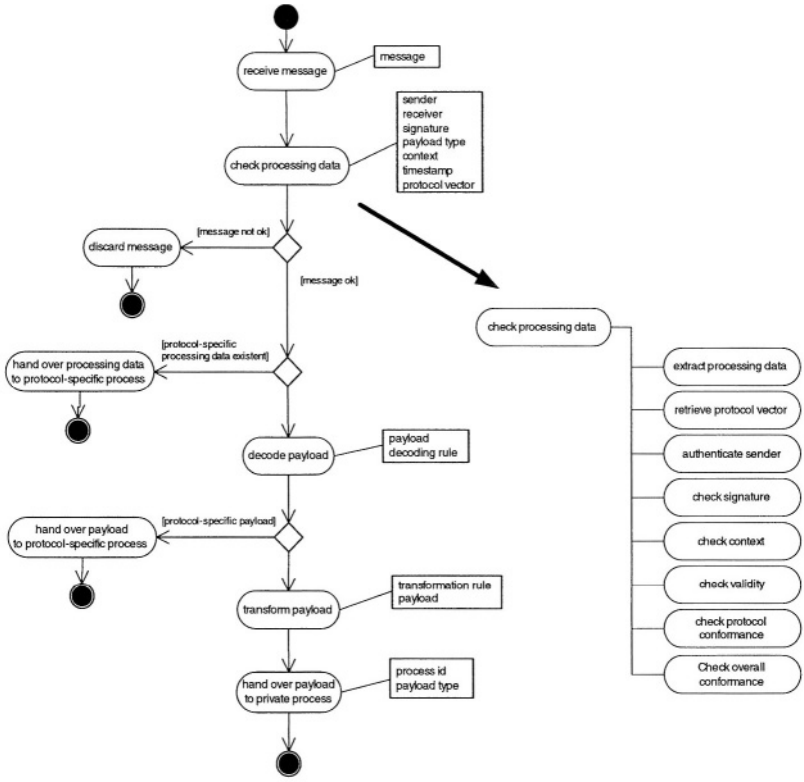


Fig. 5. Generic process of external message reception and functional decomposition of the activity *check processing data*

col-specific process), depending on the positioning of this data in the payload or not encoded processing data. Business related payload passes transformation *(transform payload)* where syntactic and/or semantic conversion is performed by applying the corresponding transformation rule. Finally, the receiving process is determined according to the payload type and the data is handed over to the private process *(hand over payload to private process).* As already mentioned, the focus of this paper is not to describe following steps like coordination of internal applications according to the private process. After having defined this generic process the refinement phase starts where each activity has to be decomposed. In figure 5 this is exemplarily shown with the activity *check processing data.* Sub-activities have been identified as depicted. This process has to be carried out until no further refinement is possible. Then other relevant system framework processes have to be identified (e.g. send message) and refined. After that, the relationships between data elements have to be analyzed. Figure 6 shows the corresponding semantic data model to the generic message reception process described above.

This semantic data model becomes more detailed with every refinement step. If all tasks in the domain have been described, the next step of DCM, namely the component identification, can be performed. Elementary functions in the decomposition diagram are put into rows of a matrix whereas all data elements of the overall seman-

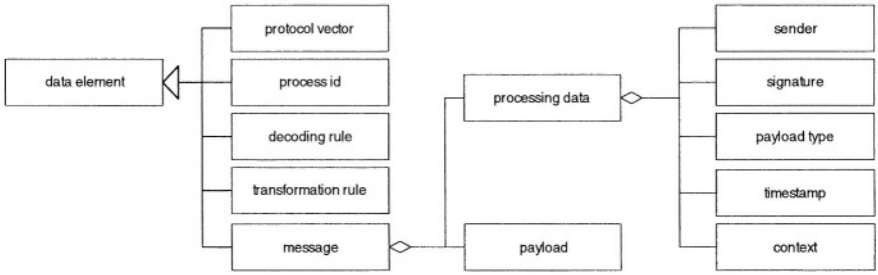


Fig. 6. Semantic data model of the generic message reception process

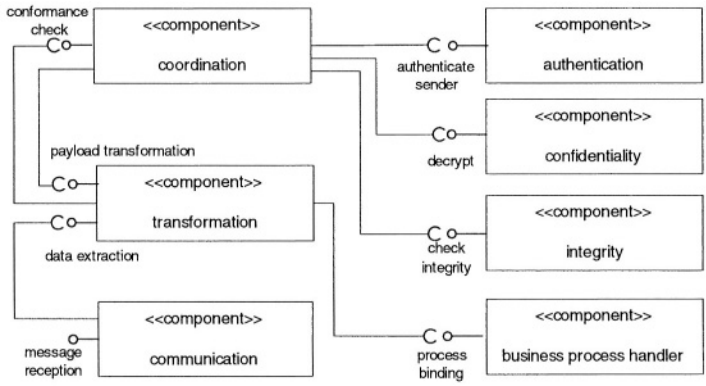


Fig. 7. Part of an abstract component model

tic model will be found in the columns. Then, according to their relationship, entries are made in the matrix to show which data element is input/output for a function or has to be retrieved by this function from storage. In changing the order of data elements and functions, groups of relationships can be discovered which represent potential components. Relations between these components can also be identified by data elements that can be provided by one component and that are needed by another. Figure 7 exemplarily shows a part of an abstract component model that was derived by the identification step.

It focuses on those relations that are established at reception of a business process related message. The communication component sends the message to the transformation component that is able to extract processing information according to the used message envelope. The coordination component then checks message conformance with the help of the protocol vector that is managed by this component. Those parts of the message that are related to authentication are routed to the authentication component for further processing. This might include sending additional messages e.g. to some authentication authority (interfaces and relations for this task are not depicted). Decryption and integrity checks are also done by special components. Message payload has to be routed back to the transformation component from where it is handed over to the business process handler.

6 Conclusion and Future Work

Having identified constituting elements of an IOS in a first step, IOS technology was further discussed with respect to the operationalization of the IOS concept. Two coordination problems in the context of inter-organizational collaboration were identified, namely external coordination of business partner interactions and internal coordination of application systems. The external coordination problem was further discussed and the concept of a protocol vector was introduced. To cope with the variability of inter-organizational settings expressed by the high number of possible protocol vector variations, a component-based architecture was proposed and the DCM process to find suitable components was introduced. Next steps in the work of defining a component-based architecture for IOS will be the refinement of the component model in order to get the desired protocol class interfaces that allow automatic system configuration based on a protocol vector instance. With an already existing prototypical implementation of the component framework it will be tested how different protocol implementations can be plugged into the framework to get an integration platform that can cope with the given variability of inter-organizational collaboration.

References

1. Albani, A., Keiblinger, A., Turowski, K., Winnewisser, C., Domain Based Identification and Modelling of Business Component Applications, 7th East-European Conference on Advances in Databases and Informations Systems, September 2003, pp. 30-45.
2. Albani, A., Keiblinger, A., Turowski, K., Winnewisser, C., Dynamic Modelling of Strategic Supply Chains, E-Commerce and Web Technologies: 4th International Conference, EC-Web 2003 Prague, Czech Republic, September 2003, pp. 403-413.
3. Alonso, G., Casati, F., Harumi, K., Machiraju, V., Web Services - Concepts, Architectures and Applications, Springer-Verlag, Berlin, 2004.
4. Baldwin, C. Y., Clark, K., Design Rules: The Power of Modularity, The MIT Press, Cambridge (Mass.), London, 2000.
5. Burtler, P., Hall, T. W., Hanna, A. M., Mendonca, L., Auguste, B., Manyika, J., Sahay, A., A Revolution in Interaction, The McKinsey Quarterly, 1 (1997), pp. 4-23.
6. Bussler, C., B2B Protocol Standards and their Role in Semantic B2B Integration Engines, Bulletin of the Technical Committee on Data Engineering, 24 (2001).
7. Bussler, C., The Role of B2B Engines in B2B Integration Architectures, SIGMOD Record, 31 (2002), pp. 67-72.
8. Bussler, C., The Role of B2B Protocols in Inter-enterprise Process Execution, Workshop on Technologies for E-Services (TES 2001), September, 2001, pp. 16-29.
9. Chatterjee, D., Ravichandran, T., Inter-organizational Information Systems Research: A Critical Review and Integrative Framework, Proceedings of the 37th International Conference on System Sciences (HICSS-37) 2004.
10. Damm, D., Eine IS-Plattform zur Unterstützung kooperativer interorganisationaler Netzwerke, Wirtschftswissenschaftliche Fakultät, Zürich, 2003.
11. Erl, T., Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services, Prentice Hall, Upper Saddle River, 2004.
12. Fellner, K., Turowski, K., Classification Framework for Business Components, Proceedings of the 33rd Annual Hawaii International Conference On System Sciences 2000.
13. Ferguson, D. F., Storey, T., Lovering, B., Shewchuk, J., Secure, Reliable, Transacted Web Services: Architecture and Composition, IBM / Microsoft, 2003.

14. Garlan, D., Allen, R., Ockerbloom, J., Architecture Mismatch: Why Reuse is so Hard, IEEE Software, 12 (1995), pp. 17-26.
15. Hohpe, G., Woolf, B., Enterprise integration patterns, Addison-Wesley, Boston, 2004.
16. Malone, T. W., Lautbacher, R. J., Dawn of the E-Lance Economy, Harvard Business Review, 9/10 (1998), pp. 144 -152.
17. Medjahed, B., Benatallah, B., Bouguettaya, A., Ngu, A. H. H., Elmagarmid, A. K., Business-to-business interactions: issues and enabling technologies, The VLDB Journal, 12 (2003), pp. 59-85.
18. Omelayenko, B., Fensel, D., An Analysis Of B2B Catalogue Integration Problems - Content and Document Integration, Proceedings of the International Conference on Enterprise Information Systems (ICEIS-2001), July 7-10,, 2001.
19. O'Riordan, D., Business Process Standards For Web Services, Tect, Chicago, 2002.
20. Ramanathan, S., Rose, J., Rationalizing, Probing, Understanding: the Evolution of the Inter-Organizational Systems Adoption Field, 36th Hawaii International Conference on System Sciences (HICSS - 36) 2003, pp. 7.
21. Ropohl, G., Allgemeine Technologie, Hanser, München, 1999.
22. Ropohl, G., Philosophy of Socio-Technical Systems, Techné: Research in Philosophy and Technology, 4 (1999), pp. 59-71.
23. Sauter, M., Die Revolution und ihre Kinder - Einsatzmöglichkeiten der IKT im Produktionsnetzwerk, in H.-J. Warnecke and J. Braun: Vom Fraktal zum Produktionsnetzwerk. Unternehmenskooperationen erolgreich gestalten., Springer, Berlin, 1999, pp. 159 - 170.
24. Sawhney, M., Zabin, J., The Seven Steps to Nirvana, McGraw-Hill, New York, 2001.
25. Schilling, M. A., Toward a general modular systems theory and its application to interfirm product modularity, Academy of Management Review, 25 (2000), pp. 312-334.
26. Turowski, K., ed., Standardized Specification of Business Components, University of Augsburg, Augsburg, 2002.
27. van der Aalst, W. M. P., Anyanwu, K., Inheritance of Interorganizational Workflows to Enable Business-to-Business E-commerce, Proceedings of the Second International Conference on Telecommunications and Electronic Commerce (ICTEC'99), pp. 141-157.
28. van der Aalst, W. M. P., Weske, M., The P2P Approach to Interorganizational Workflows, Proceedings of the 13th International Conference on Advanced Information Systems Engineering 2001, pp. 140-156.
29. Warnecke, H. J., Vom Fraktal zum Produktionsnetzwerk. Unternehmenskooperationen erfolgreich gestalten, Springer Verlag, Berlin, 1999.
30. Wise, R., Morrison, D., Beyond the Exchange: The Future of B2B, Harvard Business Review, 78 (2000), pp. 86-96.
31. Wüstner, E., Hotzel, T., Buxmann, P., Converting Business Documents: A Classification of Problems and Solutions using XML/XSLT, 4th IEEE Int'l Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS 2002).

Workshop on Regulatory Ontologies (WORM) PC Co-chairs' Message

This proceedings reflects the issues raised and presented during the second international workshop on regulatory ontologies (WORM 2004). WORM aims to gather researchers and practitioners from multiple disciplines, such as Law, Business, Ontologies, Regulatory metadata, Information extraction. Regulatory ontologies typically involve the description of rules and regulations within the social world. Modeling and deploying regulatory knowledge has indeed some specifics that differentiate it out from other kinds of knowledge modeling, such as reasoning methods and application scenarios, the legal weight (/order) of regulations, parsing legal texts, the sensitivity in cross-boarder regulations, etc.

A total of 28 papers were submitted to WORM 2004. Each submission was reviewed by at least two experts. The papers were judged according to their originality, validity, significance to theory and practice, readability and organization, and relevancy to the workshop topics and beyond. This resulted in the selection of 9 papers for presentation at the workshop and publication in this proceedings.

In particular, we would like to express our deep appreciations to the program committee members: Andrè Valente, Andrew Stranieri, Bart Verheij, Cecilia Magnusson Sjoberg, Giancarlo Guizzardi, Jaime Delgado, Jerome Euzenat, Joost Breuker, Leonardo Lesmo, Mariano Fernandez Lopez, Nicklas Lundblad, Richard Benjamins, Rita Temmerman, Said Tabet, Valentina Tamma, Youssef Amghar, for their reviewing efforts and advice on organizational details. We thank the authors of the submitted papers and all the workshop attendees for their dedication and assistance in creating our program and turning the workshop into a success. Producing this book would not have been possible without the much appreciated contribution of Kwong Lai. Finally, we gratefully acknowledge the partial support of the EU project Knowledge Web (FP6-507482).

August 2004

Mustafa Jarrar
Aldo Gangemi
(WORM CoRe 2004 PC Co-chairs)

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, p. 568, 2004.
© Springer-Verlag Berlin Heidelberg 2004

Ontology-Based E-government Thematic Services Based on Topic Maps

Ioannis Tsampoulatidis[1], Dimitrios Tzovaras[1], and Michael G. Strintzis[1,2]

[1] Informatics & Telematics Institute
1st Km Thermi-Panorama Road, PO Box 361,
GR-57001 Thermi-Thessaloniki, Greece
{itsam, tzovaras}@iti.gr
http://www.iti.gr
[2] Electrical and Computer Engineering Department
Aristotle University of Thessaloniki
GR-54006 Thessaloniki, Greece
strintzi@eng.auth.gr

Abstract. This paper describes a novel ontology representation scheme for emerging e-Government systems, based on XML Topic Maps (XTMs). This scheme supports the provision of various services to the citizens guaranteeing independence of the knowledge between the actual sources of information and the data representation. Ontology is considered as an autonomous system, which acts as a linkage among the rendered services offered by an e-Government portal by providing and respecting the terminologies, metadata and rules defined by the State functionaries. The proposed ontology representation is supported by a Knowledge Base, which includes data about repositories, document types, domains, citizen profiles, themes for search and retrieval and service catalogues. The XTMs are created using a novel Topic Map Generator tool that is used to establish the necessary rules and constraints in the system. Finally, a use case is described in detail, for urban planning applications showing the advantages of the proposed approach.

1 Introduction

Over the last ten years the term "ontology" and the technology that lies behind it along with the fundamental data model for knowledge systems have appeared. Ontologies introduce various advanced functions like easy search and are the basis of the semantic web. "Ontology" is the name given to a structure of knowledge, used as a means of knowledge sharing within a community [1].

An ontology can be considered as a "data skeleton" that can be developed for a variety of data-based and knowledge-based systems. It consists of a collection of concepts and relationships between concepts where relations represent a type of interaction between these concepts [2]. Generally, each concept is represented by a favored term (word or phrase). The ontology might include principle synonyms (abbreviations, common misspellings) for each term or it could contain only a single type of

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 569–580, 2004.
© Springer-Verlag Berlin Heidelberg 2004

relationship between terms; e.g., a "published-by" relationship, such as "Birth certificate is published-by the Registry Office". Generally, an ontology includes basic knowledge relationships that a variety of types of knowledge applications might use.

Of course, the Ontologies cannot be used in real time applications if there isn't a mechanism to extract the semantics and separate the content. Such a mechanism for thematic web services is presented in this paper based on Topic Maps and empowered by the SOAP protocol [3] and the general concept of web services.

According to T. Bernes-Lee et. al. [4], the "Semantic Web" is defined as an extension of the existing Web, in which information is given well-defined meaning and as stated by [5] ontologies describe domain theories for the explicit representation of the semantics of the data in the context of the Semantic Web. It aims at providing machine-processable services and information on the Web. Efforts towards enabling the envisioned Semantic Web are gaining momentum. The proposed approach uses the emerging concept of web services in a great extend since it is slated to be the backbone of tomorrow's Web as stated by S.A. Mclraith et. al. [6]. A Web service is a set of related functionalities that can be programmatically accessed through the Web [7]. One of the benefits of Web services is that users need no longer think in terms of data but rather of the services they wish to receive. Adopting the statement of B. Medjahed et. al. [8], this powerful concept is well suited for e-Government applications where citizens are not expected to know about or understand the data infrastructure behind the provided services.

The semantics of Web services is crucial to the development of a Semantic Web enabled e-government. E-Government agencies would automatically discover, "understand", and share each other services. Ontologies are expected to play a central role to empower Web services with semantics [4]. In this respect, an ontology is a shared conceptualization based on the semantic proximity of terms in a specific domain of interest [6].

The proposed Thematic Services Ontology System is the backbone of an e-Government system as it controls, to a great extent, the integrity and the logic of the system and provides a set of rules to be complied by the rendered services. It consists of a Knowledge Base and a number of tools that perform management on the Knowledge Base data. Technically, it can be perceived as the entire system processor, where all the system metadata are stored, processed and sent out to the governmental portal to be accessed by the citizens. Most of the data stored in the Knowledge Base are Topic Maps, thus the implementation of a tool for XML Topic Maps (XTM) authoring was more than requisite.

State functionaries, Municipalities, state councils, tax officers and urban planning officers are only a few examples of the prospective users of the "Topic Maps Generator" authoring tool whose main target is the enrichment of the e-Government Ontology. One of the major features of the tool is the avoidance of user errors causing incompatibilities with the XTM specifications by restricting fault entries and validating the generated XTM with the corresponding Document Type Definition. The major purpose of adopting the XML Topic Maps technology is not to cover or overlap the existing XML-based metadata standards that are often used by e-Government

systems but to strengthen and enrich them. The main gaining advance to use Topic Maps is the ability of the standard to define themes of information.

2 Ontology-Based E-government System

Ontology is central to the proposed e-Government portal architecture as shown in Figure 1, since it stores and manipulates XML metadata and other information that is necessary for the proper functionality of the e-government portal.

Through the e-government administration module, the system expert can manage the ontology system by calling SOAP methods. In the proposed system, a web-based tool that takes advantage of the web-services is also provided for easiness sake. In addition, if necessary, a programmer could use the SOAP methods in order to built tailored made applications for specific governmental services.

The administration module also hosts the Topic Maps Generator – a tool for authoring and visualizing XML Topic Maps [9], which is directly connected with the Ontology's Knowledge Base. The occurrences of the topics that are defined in the XTMs can reference any document, image and any other data that are located in the governmental data repository or even other external links. The citizen accesses the e-government portal, which in turn is using the search and retrieval mechanism that composed by several different technologies in order to take advantage of the topic maps that relied on the ontology system.

The proposed system could be used for any of the services listed in Table 1 by providing the appropriate Topic Maps for each one.

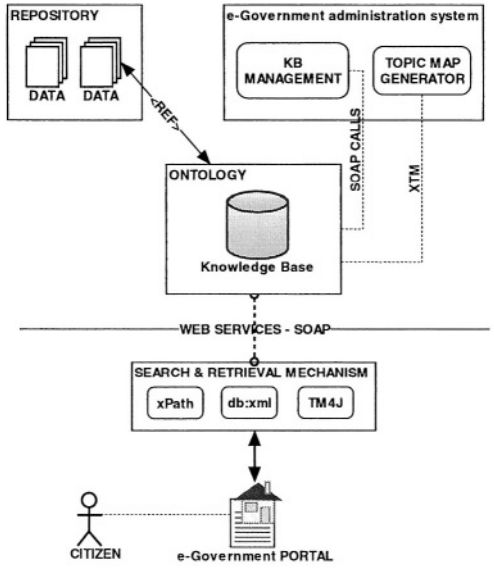


Fig. 1. Overall Thematic Services e-Government System.

Table 1. List of indicative public services that could be offered by an on-line e-Government system according to e-Europe 2002; Impact and Priorities, COM 2001 140 final, Brussels 13.3.2001

Services for the Citizens	Services for Business
Income taxes	Social contribution for employees
Job search	Corporate tax
Social security benefits	VAT
Personal documents	Registration of a new company
Car registration	Submission of statistical data
Application for building permission	Customs declaration
Declaration to the police	Environment-related permits
Public libraries	Public procurement
Birth and Marriage certificates	
Enrolment in higher education	
Announcement of moving	
Health-related services	

For the aforementioned services, different approaches and ontology-based techniques are used in conjunction with Topic Maps in order to achieve the maximum benefits for the citizens and business. Of course the system provides such mechanisms that allow the reusability of metadata in the Knowledge Base by merging several topic maps in order to be used for various e-Government services.

One of the major reasons choosing XTM as the mean for knowledge representation is the fact that the Topic Maps were designed from the beginning for ease of merging in mind. The interoperability offered by this feature is a great advantage for e-government systems, since pieces of government information are used repeatedly in multiple stages. The concept of *scope* in Topic Maps is also a great merit because it solves the problem of capturing contextual validity. This characteristic is very useful for filtering information and supporting multilingual topic names and is indeed easier to be defined than other data models like RDF and its constraints languages and vocabularies (DAML, OIL, OWL). Also Topic Maps are more expressive in the definition of *associations* due to the notion of 'association roles' which make clear the kind of role played by each participant in a relationship [10]. This means that relationships are inherently two-way in Topic Maps. RDF could manage many of the features provided by Topic Maps probably as well, especially with DAML and OIL schema languages that enrich the data model, but XTM provide these advantages without external vocabularies. For example the definition of non-binary relationships (very useful for e-government ontologies) are difficult to be defined in RDF in comparison with Topic Maps. *"As for the semantics added by DAML to RDF, topic maps already have most of these"* as stated by Lars Marius Garshol [11], co-editor of the Topic Map data model.

3 Knowledge Base

Knowledge management deals with the representation, organization, acquisition, creation, usage and evolution of knowledge in its many forms. To build effective technologies for knowledge management, further understanding of how individuals, groups, organizations and governments are using the knowledge is needed. Given that more and more knowledge available in computer-readable forms, building tools which can effectively search through databases, files, web sites and the like, to extract information, capture its meaning, organize it and make it available, is also needed [12].

The Ontology System that wraps the Knowledge Base can be thought of as constructed in two parts: a) a database used to store the metadata and b) a set of tools and functions, which manipulate these data. The metadata that are being used by many e-Government systems are based on various metadata and framework standards like e-GIF and Dublin Core and the XML Topic Maps by no means are going to replace that. The purpose of the Topic Maps is to manage, organize, connect and access the information and the knowledge of e-government data that most likely are held in different governmental repositories and departments.

The database has been named Knowledge Base since it provides the "knowledge" of the system. Concerning the technology used in the Knowledge Base it has been decided that the implementing native XML database will be the Apache Xindice [13], because it is open-source, specific to XML and reliable enough.

The tools supported by the Ontology include functions which check the validity of XML documents, SOAP methods for the synchronization of various e-Government portals, procedures to classify the files according to their format and a mechanism that produces unique keys to be used for proper naming of documents and files that are stored in the e-Government system.

3.1 Data Stored in the Knowledge Base

Knowledge Base mainly stores metadata information as well as other general administrative information, such as citizen profiles containing personal information, passwords, citizen's past requests and so on. These data are mostly formatted with XML based metadata standards. It is necessary for the e-Government Portal and probably the Portal mirrors to have direct access to the metadata: the Portal for example needs citizen profiles to verify the citizen's access to the system, the search and retrieval mechanism that is embodied in the portal needs the Topic Maps to search among the metadata and so on. Summarizing, there are six major categories of data stored in the Knowledge Base: Topic Maps (XTM), XSL Style sheets, Citizen/Terminal profiles, Query predefinitions, Document Type Definitions / XML Schemas and Administrative information.

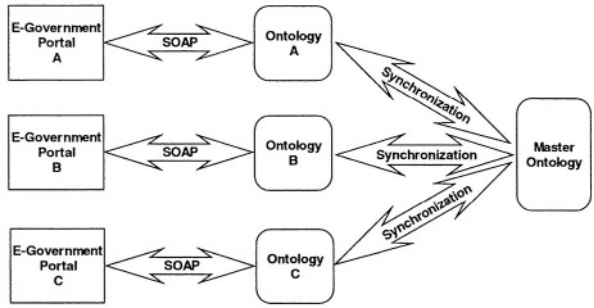


Fig. 2. Synchronization between various e-Government portals with the use of Master Ontology and SOAP-based methods.

3.2 Knowledge Base Interconnections

An e-Government system may consist of several sub-portals that compose a complete functional system. The various governmental portals need to be interconnected with each other in order to achieve homogeneity of the data and that is due to the reason that the information needs to be synchronized in every portal. To achieve the synchronization of the actual data and consequently of the metadata, a Master Ontology is created. As shown in Figure 2 the e-Government System infrastructure consists of several e-Government Portals and each one of them contains its Knowledge Base. The data stored in every Knowledge Base are mirrored with each other with the aim of the Master Ontology node, which provides automated SOAP-based web services that synchronize every sub-system, as shown in Figure 2.

The synchronization is actually a very simple process. As previously stated, the ontologies (in XTM format) are stored in native XML databases which are handled through SOAP and DB:XML API calls. For every remote procedure calls on any deployed method on the SOAP server of each portal, a special method is also called automatically (or sometimes on demand, depending on method arguments) that sends the same request to the master ontology, where in turn another deployed method listens and calls the appropriate method from every other SOAP server in the system.

Practically, this means that a citizen's profile is available from every e-Government Portal irrelevantly of the one that is initially created. The definition "Portal" does not necessary mean a public web site. A specialized version for PDAs or a private version for use by the government authorities with enhanced capabilities can be considered "Portal" as well. Going further, the synchronization of various parts of the e-Government Ontologies can lead to an easier interconnection between e-Government Portals on different states or even different countries.

3.3 Knowledge Base Management

The management of the Knowledge Base is feasible with two different ways: Either through a web-based graphical user interface or through communication via the SOAP protocol.

The web-based tool for manipulating the structure and the content of the Knowledge Base is implemented with the use of XML:DB API [14]. It could be installed in every governmental portal and helps the system administrator to have a quick view of the documents and the collections contained in the Knowledge Base. The system administrator is able to add and delete collections and documents and also to perform advanced searches by using queries, which are based on Xpath [15] - a language for addressing parts of an XML document. he system includes an enhanced search mechanism for Topic Maps with the use of drop-down menus. The role of the web-based GUI for manipulating the Knowledge Base, is supplementary and its absence does not disqualify the proper functionality of the Ontology.

Simple Object Access Protocol [3] is an XML/HTTP-based protocol for accessing services, objects and servers in a platform-independent manner. This is the reason for choosing SOAP for the integration of the access mechanism of governmental data from a programmer's point of view. The solution of using SOAP based classes for manipulating the Knowledge Base is more flexible and robust in comparison with other protocols since it provides an easy way for fast and reliable interconnection between the e-Government portal and the Ontology. Further, the SOAP specification mandates a small number of HTTP headers that facilitate firewall/proxy filtering which is extremely important for further upgrades of the e-government systems. The Apache SOAP [16] has been selected as the SOAP server for the Ontology Systems because it is widely used and it is available under the Apache Software License. The use of Topic Maps in conjunction with the SOAP protocol offers great flexibility to the web portal programmer. For example a list of Frequently Asked Questions (FAQ) can be constructed simply by creating a batch process that traverse the Topic Map and extract the necessary topics. Topic Map can be considered as a navigation mechanism that gives access to data by topic and also enables the direct navigation between related topics.

4 Topic Maps Generator

The development of a web-based Authoring Tool for XML Topic Maps was essential for the implementation and the definition of governmental rules since it facilitates one of the main tasks that the systems administrators have to undertake. Creation, manipulation and processing of Topic Map metadata documents into the Ontology Knowledge Base are the rendered services. Creation and maintenance of topic maps without the help of a tool introduces the possibility of inconsistency, thus, the "Topic Maps Generator" created for this purpose and tailored specifically for governmental services offers all the functions needed. Apart from the pure technical features the web-authoring tool has got extra features to make the creation and storage of Topic Maps more convenient, since it has direct access to the Ontology Knowledge Base where the Topic Maps are stored. The system administrator is able to:

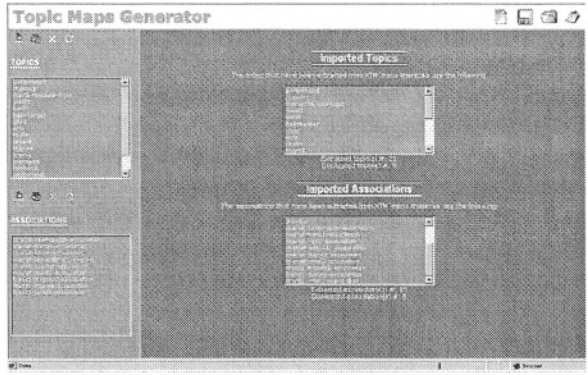


Fig. 3. Snapshot of the "Topic Maps Generator" tool. Extracting topics and associations from a topic map that is stored in the knowledge base of the e-government system

- Make concurrent requests, since multiple sessions are supported
- Open existing XTM files from the KB and re-use its topics and associations. This is a very useful feature that saves the system administrators from rewriting topics that are already defined elsewhere
- Show the current XTM file produced onscreen at anytime during processing
- Save the created file to the KB so that it can be readily accessible from everywhere inside the e-Government portal or save it to a file for downloading
- Import an XTM file to the Knowledge Base for further processing

Figure 3 shows the procedure of opening and extracting the topics and the associations of a stored topic map. The SOAP protocol is hidden in the background of every connection of the tool with the Knowledge Base.

Reasonably, the topic map standard has only a very limited set of mechanisms for defining constraints on the map [17]. The Topic Maps Generator provides such a mechanism to avoid omissions and mistakes from users by blocking fault entries. Furthermore duplicate topic ids are not allowed.

It must be signified that that the produced XTM file is fully compatible with the XTM 1.0 specifications [9].

4.1 Architecture of the Authoring Tool

The tool is implemented with the Java programming language, it is running on Apache Jakarta Tomcat [18] as Java Servlet and it is based on the TM4J [19], which is a Topic Map engine for Java. TM4J is an open-source project to develop topic map processing tools and applications. The current focus of the TM4J project is on the development of a topic map "engine" which processes files conforming to the XML Topic Maps (XTM) specification and stores them either in memory or in a persistent store, providing access via a Java API. The persistent store is a native XML database. As already stated, the "Topic Maps Generator" tool is directly connected to the Knowledge Base of the e-Government system and that is the reason, which makes the

tool innovative. No other effort is required from both the state employee and the programmer point of view, in order to make the data accessible. In Figure 4 it is shown the architecture of the tool graphically:

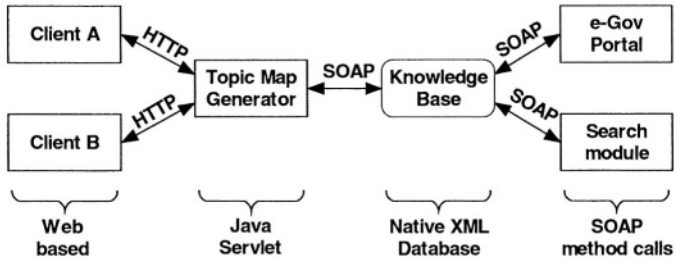


Fig. 4. Architecture of the Topic Maps Generator

More precisely:

- Clients representing the system administrators accessing the Topic Map Generator Tool through an Internet Browser using the HTTP protocol
- The web pages are produced at Jakarta Tomcat Server and send back to clients
- The Topic Map Generator communicates with the Knowledge Base (Xindice in our case) to store and request XTM files
- Every module of the e-Government portal has access to the Knowledge base through the SOAP protocol

5 Use Case: Urban Planning

The grouping of pieces of governmental information into a thematic context is beneficial not only for the citizen-side but also for the Government-side as well since it provides a substantial added value in terms of better understanding of the individually addressed topics of interest. The maintenance of information with the use of Topic Maps is easier than before because even if the data are held in non-homogenous structures and formats, as is the case in most e-Government systems that handle the demanding needs of on-line urban planning services, the Topic Maps are acting as a middleware between the actual data and their representation in a common user friendly interface.

A successful e-Government system is the one that offers what the citizen needs. Many systems do that, but their functions and services are left hidden due to the fact that is hard for the citizens to find easily and quickly the pieces of information they are looking for. The solution to this weakness is a smart and 'live' mechanism for search and retrieval of urban planning-related information where the citizens will rely on, to complete their requests no matter how complex they are.

The selection of the urban planning use case was based not only on the complexity of the data involved in such a demanding service, but also on the different repositories where the data relied as well.

Although the urban planning framework is the same for each state, the actual data (electronic maps, applications, extra justifications, etc) concerning the urban planning services in most e-Government systems are different in every geographical department or municipality. This lack of homogeneity makes the use of Topic Maps requisite.

In the proposed system for every urban planning related data, two additional data types, the "fragments" and the "themes", are added. Fragments are delivering supplementary and contextual information related to an object or a set of objects (e.g. justifications for a new building). Moreover, these fragments are set up in a way that they can be reused for different contextual views and situations. A theme is a collection of thematic texts that have one or more characteristics (keyword) in common.

The fragments could be considered as thematic texts and along with the themes and the collection of similar topics can be referred as the thematic approach of urban-planning information within the e-Government system.

For example, if a citizen seeks a list of justifications for getting a building redecoration license, the system could provide information about the permitted building restorations in a specific area, which is surely the kind of information that the citizen will need in a later stage.

One can easily understand that with this approach the theme concept is quite flexible and that a single thematic text can show up in different contexts. Of course the "knowledge" of the system depends on the proper use of context and scope attributes that can really narrow the searching results to the exact level that a citizen requires.

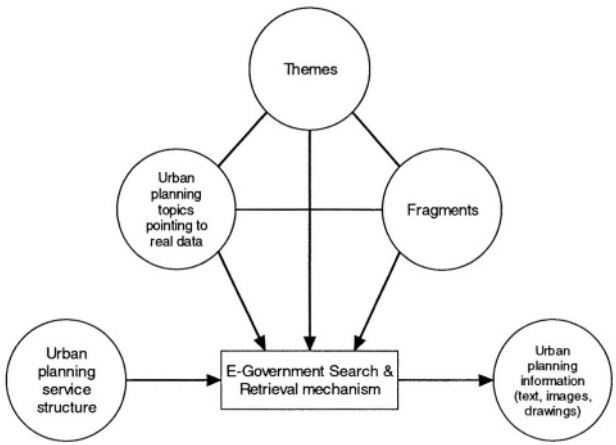


Fig. 5. The search and retrieval mechanism for urban planning service enhanced with topic maps technology.

As shown in Figure 5 the search and retrieval actions can be originated on three levels: a) urban planning topics pointing to digitized maps, text descriptions, etc., b) fragments and c) themes. Once one of these levels is selected, further searches can be done within the same level or shifted towards all other levels, where more appropriate searches can be carried out. If the citizen has no preference, the search will be per-

formed on all levels and presented in the following priority: Themes – Fragments – Urban planning topics. Going further, the system supports customized searches according to the citizen profile that contains information about the citizen's last visits in the portal and the fields of interest.

Practically, using Topic Maps to classify thematically the urban planning content could be difficult due to the multiple levels of data categorization. Defining and setting up the appropriate topic maps to cover every detail of information is a little bit cumbersome and time consuming. Estimation of the size of the ontology is difficult to be made since it is clearly depends on the level of complexity of the urban planning rendered services. A general assessment can be made though, corresponding one service to 15-20 topics. Expanding a whole theme (e.g. "Metropolitan Park and Urban Development Information Section") requires about 140-180 topics including associations. All topics of course are not held in a single XTM file since it is proved by experience that keeping several small files containing all topics, associations, occurrences, etc., is better than keeping them in one huge file, in terms of efficiency.

Also, an alternative method was adopted in order to link semantically the information. Since there are hundreds of HTML4 documents relied on the urban planning service repositories, a batch file was implemented in order to transform the existing documents from HTML4 into XHTML1.1 format. The cleaning of the HTML code took place with the use of the HTML Tidy [20] library. The next step was to develop a simple application that parses every document and automatically adds <meta> tags containing information about the content of the document and some keywords in addition. This way, the administrator can group together several documents with similar knowledge information and refer to them by using the resource reference identifier <resourceRef>. Of course, this type of reference should be carefully discerned from the other reference types, since the resource itself is the subject and not what the resource might contain or indicate by its contents. It is obvious that this method can serve as a temporary solution until the creation of the topic map and its occurrences is completed.

6 Conclusions

A novel ontology representation scheme was presented for emerging e-Government systems, based on XML Topic Maps (XTMs). This scheme supports the provision of various thematic services to the citizens guaranteeing independence of the knowledge between the actual sources of information and the data representation. The XML Topic Maps, selected as the ontology data representation form, were created using a novel Topic Map Generator tool that is used to establish the necessary rules and constraints in the system. A use case was also described for urban planning applications supporting search and retrieval on three levels: a) urban planning topics pointing to digitized maps, text descriptions, etc., b) fragments and c) themes. Using Topic Maps instead of other data models, proved to be reliable and efficient enough. On the other hand, the flexibility offered by Topic Maps makes a little hard for people to build them because there are many 'philosophical' decisions to be made and the deep un-

derstanding of how information related to other fragments of information is mandatory. Also the use of the XTM Generator tool demands time for users to adjust to the new approach of Topic Map's philosophy and the 'everything is a topic' way of thinking.

Acknowledgements. This work was supported by the IST-2003-511298 ASK-IT – 'Ambient Intelligence System of Agents for Knowledge-based and Integrated Services for Mobility Impaired users', the IST- 2002-507860 INTELCITIES – 'Intelligent Cities' and the Greek General Secretariat for Research and Technology C-30 REACH – 'New Forms of Distributed Organization and Access to Cultural Heritage'.

References

1. TopQuadrant White Paper, "Ontology Myth or Magic", USA, 2003
2. O. Gorcho and A.G. Perez, "Evaluating Knowledge representation and reasoning capabilities of ontology specification languages", ECAI 2000, Berlin, 2000
3. The SOAP protocol http://www.w3.org/TR/SOAP/
4. T. Berners-Lee, J. Hendler, and O. Lassila. "The Semantic Web", Scientific American, May 2001
5. A. Maedche, V. Pekar and S.Staab, "Web Intelligence - Ontology Learning Part One – On Discovering Taxonomic Relations from the Web", Springer Verlag, 2002
6. S. A. Mclraith, T. C. Son, and H. Zeng. "Semantic Web Services", IEEE Intelligent Systems, March 2001
7. B. Medjahed, A. Rezgui, A. Bouguettaya, and M. Ouzzani. Infrastructure for E-Government Web Services, IEEE Internet Computing, 7(1), January 2003.
8. B. Medjahed, A. Bouguettaya, and M. Ouzzani. "Semantic Web Enabled E-Government Services". The dg.o 2003 NSF Conference for Digital Government Research, Boston, USA, May 2003
9. XML Topic Map standard - http://www.topicmaps.org/xtm/1.0/
10. S. Pepper, Ontopia, "Ten Theses on Topic Maps and RDF", August 2002
11. L. M. Garshol, "Topic Maps, RDF, DAML, OIL", XML 2001, Orlando, USA, 2001
12. Jurisica, Mylopoulos and Yu, "Using Ontologies for Knowledge Management: An Information Systems Perspective", ASIS '99, 1999, Washington, D.C. pp. 482-296
13. Xindice – native XML database - http://xml.apache.org/xindice/
14. XML:DB Application Program Interface - http://www.xmldb.org/xapi/
15. xPath query language - http://www.w3.org/TR/xpath
16. SOAP protocol - http://xml.apache.org/soap/
17. G. Ove Grunmo, "Creating semantically valid topic maps", XML Europe 2000, 2000
18. Apache Jakarta - http://jakarta.apache.org/
19. TM4J Application Program Interface - http://www.tm4j.org
20. HTML Tidy - http://tidy.sourceforge.net

Workflow Description of Digital Rights Management Systems

Silvia Llorente, Eva Rodríguez, and Jaime Delgado

Universitat Pompeu Fabra (UPF), Departament de Tecnologia,
Pg. Circumval·lació 8, E-08003 Barcelona, Spain
{silvia.llorente, eva.rodriguez, jaime.delgado}@upf.edu
http://dmag.upf.edu

Abstract. Digital Rights Management (DRM) is becoming a key issue in our highly networked world. Piracy of digital goods of any kind (music, software, video) is growing day by day. In this scenario, many companies, organisations and administration-funded projects provide solutions for the implementation of digital rights management (DRM) systems. Nevertheless, although these solutions have several points in common, they are incompatible in terms of architecture and system components. This paper analyses some of these solutions, focusing on the description of their data flow, one area where common points can be found. We propose the use of workflow modelling in order to find commonalities among data flow of DRM systems, that would allow easier implementation of new ones. The selected language for performing this modelling is OWL-S (Ontology Web Language for Services). The use of an ontology language will allow us to combine workflow modelling with ontologies defining DRM concepts.

1 Introduction

Many companies, organisations and projects supported by European administrations provide solutions for the implementation of digital rights management (DRM) systems. Each of them proposes its own system and architecture for the protection of digital goods in an increasingly networked environment.

In this paper, we present some of these initiatives, making a distinction between the ones offered by companies and the ones offered by organisations or being a result of a project funded by an administration.

Afterwards, the concept of workflow is introduced, together with process modelling. The context where these concepts are used is that of services provided by electronic means [1]. In this area, OWL-S (Ontology Web Language for Services) [2], was selected.

Then, we relate the concept of workflow for the description of processes with DRM systems. In this sense, we give some preliminary ideas on the modelling of generic DRM systems, specially the underlying workflow controlling transactions in such systems.

Next, we give some examples of modelling for existing DRM systems using the approach proposed. The use of an ontology language (OWL [3]) for performing the

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 581–592, 2004.
© Springer-Verlag Berlin Heidelberg 2004

modelling should allow the integration of other existing ontologies (or newly created ones), dealing with the DRM domain.

Finally, we present some conclusions and future lines of research.

2 Current Trends in DRM Systems

There are many ways of providing DRM systems. In particular, each company offering a DRM system has its own solution, usually closed and proprietary. There is also the other way around, the one related to projects, at European or national level, where the results are given to the public domain in a more or less detailed way.

Nevertheless, there are some common points between the different DRM systems proposed. One of them is the use of licenses for protecting content. A protected content is separated from the license or licenses governing its usage. A user first accesses to a content and then purchases the rights to use it and the keys to unprotect it.

2.1 DRM Systems Offered by Companies

In this section we present existing proprietary DRM technologies, some of them under development, provided by relevant companies in this area, such as Microsoft [4], RealNetworks [5] or Trymedia [6].

2.1.1 Microsoft

The DRM system [4] provided by Microsoft is tied to Microsoft Windows platforms. The main features of Microsoft DRM systems are that the resources are delivered in encrypted form, the licenses are not attached to the content and usually they are also delivered in encrypted form because they contain the key to unlock the encrypted resources.

The data flow in Microsoft DRM is the following. First, the content owner creates a packaged file with the content locked with a key. If a user wants to use the content, he must request a license. Then, the license clearing house generates a license containing the key that unlocks the packaged file and this license is downloaded to the user PC.

The licenses in Windows Media Rights Manager contain the rights and rules that govern the use of the digital media file and the key to unlock it. The content owner is who sets these rights in order to determine which rights can be exercised against the governed content. These licenses can be delivered in different ways depending on the business model being used.

2.1.2 RealNetworks

The DRM that RealNetworks [5] offers shares several similarities with the Microsoft DRM system described in previous section. Real's DRM main feature is that the content access authentication is performed by the RealPlayer just before the playback.

The data flow in Real DRM system is described next. First, the RealSystem Packager generates a secured media file (*.rms). Furthermore, it generates a globally unique identifier (GUI) and a secured key for the content file that are imported into

the retailer's database and the secured media file is made available to consumers. Then, users contact the retail Web server to obtain a license to play the secured media file and this server requests rights from RealSystem License Server, that generates and encrypts a license that is delivered to the trusted client. Finally, the trusted client retrieves the content file. It checks its secured license database to ensure it has received the rights to play the file. Then, the secured media file is decrypted and played.

2.1.3 Trymedia

Trymedia's ActiveMARK DRM System [6] was designed specifically for decentralised networks, such as P2P exchanges or e-mail. It follows the idea of separating content and licenses like the DRM systems described above. The main difference with the other systems is that it is not provided as a set of tools, but as a service by Trymedia Systems [6].

Trymedia's ActiveMARK DRM System has two main features; it is file independent and Player/Viewer independent. This means that it can protect any kind of content and works with every player/viewer in the market.

The license terms are described in Perl [7] and rights can be associated with users and devices.

2.2 DRM Systems Defined in Projects and Organisations

There are organisations and projects working in defining its own DRM systems. In this section we present some of the most relevant DRM systems developed thus far.

2.2.1 Eurescom Project Opera

The OPERA project [8] had the objective of specifying an open DRM architecture that addressed the needs of content providers, DRM system operators and customers. With respect to rights languages, the OPERA project used a proprietary rights language.

The data flow in this system is described next. First, the user selects the protected content in the content shop he wants to purchase and he buys a license that has been already registered in the Opera Server, which is responsible for the management of the rights the users have obtained. Then, the browser requests the license to the Opera Server that generates a challenge key which is sent by SMS to the user mobile phone. Then, the user device sends this challenge key to the Opera Server that validates the user's rights to the content and generates a one-time usage license. Finally, the License Server (e.g. Real DRM) sends this license to the device and the user can decrypt the protected content and reproduce it.

2.2.2 OMA

OMA DRM v1.0 [9] specification defines three DRM methods: forward-lock, combined delivery and separate delivery. The data flow for each one of the methods is specified below.

In the forward-lock method, the media object is wrapped into a DRM message and delivered to the device, that can render the content but not to forward it to other devices. The device does not support messages with rights objects.

In the combined delivery method, a rights object and a media object is wrapped into a DRM message and delivered to the device that must enforce the rights expressions, based on Open Digital Rights Language (ODRL) [10], when consuming the content.

In the separate delivery method, the media object is always encrypted and converted into the DRM Content Format (DCF) [11]. The DCF object is downloaded to the device using OMA Download [12], after which the rights object is separately delivered to the device using Wireless Application Protocol (WAP) push technology as defined in "Push Over The Air (OTA) Protocol" specification [13]. After receiving the rights object the device may render the media object. In this method the device is allowed to super-distribute the media object, but not the rights object to another device. Then, the receiving device must acquire rights for the media object from the rights issuing service.

The main features and functionalities that OMA DRM v2.0 [14] offers are the concept of domain, a group of devices (owned by a user) that will be allowed to share rights objects, and the possibility of content super-distribution. This system will be able to bind the rights to the user identity, will support downloading and streaming and will have better security based on public key infrastructure.

The data flow in OMA DRM 2.0 is described below. First the user browses to a web site and downloads the media object. Then, the content issuer transfers the content encryption key to the rights issuer. During the consumer purchase transaction, the rights issuer establishes trust with the user device and delivers the rights object to it. Finally, the user can reproduce the media object, share it within his domain and super-distribute it to a friend, which has to purchase the rights object to be able to use this media object.

2.2.3 OpenSDRM

Open and Secure Digital Rights Management (OpenSDRM) [15] has been developed in the MOSES (MPEG Open Security for Embedded Systems) project [16].

The data flow when the users download existing content is the following. First, the user downloads protected content, for example a protected song. Then, the License Server generates a license granting to that user the right to listen the song according to the conditions selected. Finally, when the user wants to listen the song, a connection between the MOSES player and the License Server is established, and the license is downloaded. Then, the protected song is decrypted using the key extracted from the license and it is finally played.

3 Need of Harmonisation of DRM Systems

The existence of several initiatives in DRM systems, both commercial and non-commercial, makes difficult their wide adoption, as concepts, rules and processes controlling them have very diverse features usually incompatible.

The creation of a common model for the description of DRM systems will help in the harmonisation of them. From a commercial point of view, this may not be very feasible, as each product provider wants its solution to be selected by content owners (or providers) as the one to provide protected content. Nevertheless, from a research point of view and having into account the ease of integration of existing DRM solutions into newly implemented systems, which may provide their own DRM solution or use one or more of the existing ones, this is clearly an interesting approach.

One way to provide a common model for the description of DRM systems is the definition of common ontologies describing the different concepts, rules and processes present inside them. To do so we can take a top-down or a bottom-up approach. If we take a top-down approach, we should first define (or look for) an ontology (or group of ontologies) with general DRM concepts and then try to apply them to existing systems. If we take the bottom-up approach, then we should define each DRM system, and then try to extract the common concepts among them, in order to generate a general ontology.

Moreover, inside the description of the elements of DRM systems by means of ontologies, we could find different levels of development. We already have ontologies, like IPROnto [17], that describe some aspects of DRM systems, specially general concepts (rights, actors, etc), but we probably do not have ontologies describing the lifecycle of a digital content since its creation until the moment it is provided inside a DRM system. In this case, we may describe the lifecycle by identifying steps or phases forming part of it. To do so, we propose the use of workflow in order to control this lifecycle. It is explained in more detail in the next section.

4 Workflow Inside DRM Systems

Workflow Management Coalition (WfMC) [18] provides the following definition for workflow: the automation of a business process, in whole or part, during which documents, information or asks are passed from one participant to another for action, according to a set of procedural rules.

Relating this definition with DRM systems, we can describe the lifecycle of a digital content as a process where digital content and information (like cryptographic keys) are passed from one participant to another for action, according to a set of procedural rules. The workflow of a DRM system represents the different steps or phases through which the content passes from its initial creation (or protection, if we are not the content owners but the content providers or protectors) to its final distribution and purchase. During the digital content lifecycle, several actors, information and processes may be involved.

Although we have already done some work in the definition of protected content lifecycle [19, 20] based on IPROnto [17], an ontology which models the domain of intellectual property rights, we would like to represent content lifecycle using a different approach. This approach is based on the work previously done in the modelling of services offered by electronic means [1, 21]. We can establish some parallelism between services offered by electronic means and DRM systems, so the concepts defined for the modelling of services apply, as explained in next section. It

does not mean that we forget the lessons learnt from IPROnto, as some parts of content lifecycle were already described in this ontology where actors and roles were defined, but that we want to continue the work done in the DRM field from another perspective.

4.1 Modelling of Services Offered by Electronic Means

In [22, 1] we presented and refined a methodology for describing services to be provided by electronic means. This methodology had several components, defining each of them different aspects of the services. One of the components of this methodology was the definition of the service workflow. To do so, we first identified the phases of the service, the users involved on each phase and the information interchanged among users participating in a phase of a service. Once this identification is done, we are able to define the service using OWL-S [2], a language built on Ontology Web Language (OWL [3]) for the definition of services [23]. Some examples of definition of services using OWL-S can be found in [1, 21]. Other languages for the definition of workflow and process modelling like XPDL [24], BPEL4WS [25] or ebXML [26] were studied during the definition of the service description methodology.

DRM systems share some features with the services we have modelled with our methodology as they also involve several users, different phases and interchange of information, for instance, digital content, licenses, keys, etc.

4.2 Description of DRM Systems Workflow

Commercial and non-commercial DRM systems describe their operation in terms of phases. In order to show the feasibility of our approach, we are going to describe in a preliminary way the workflow of two of the existing DRM systems introduced in section 2.

Microsoft DRM

Microsoft DRM is a proprietary product based on Windows platforms [4]. In this system, the protected content and licenses are separated, and a user that wants to play a protected content must purchase a license.

The workflow present in this system can be separated into two main phases: Content protection and content purchasing. The content protection phase describes the way a content owner can protect its content in order to permit the purchase of it in a secure way. The content purchasing phase describes how a final user can purchase a protected content and the license associated to consume it.

Figure 1 shows the content protection workflow phase. In this phase, the license clearing house is referenced from protected content and the proper keys to generate licenses associated to content are transmitted to the corresponding license clearing house. The digital content protection phase involves the two subsequent phases, protected content distribution and key for license creation. They are separated because involve different users and information.

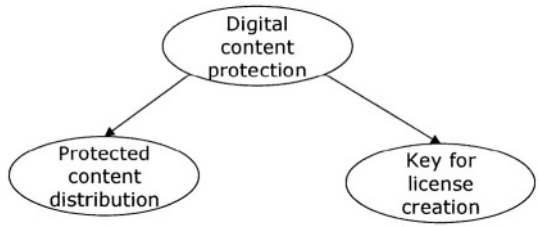


Fig. 1. Content protection phase

On the other hand, figure 2 shows the content purchasing workflow phase, that involves the purchasing of the protected content, the purchase of the corresponding license and, finally, the use of the protected content.

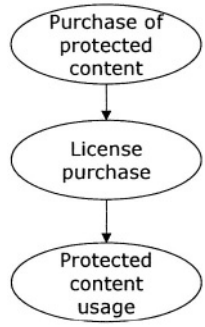


Fig. 2. Content purchasing phase

These workflows may seem very simple, as they show consecutive phases. We have to take into account that they correspond to a commercial system with limited access to its specification.

Figure 3 shows the refinement of the digital content protection phase in the form of a OWL-S process. In this refinement we describe the different components of the DigitalContentProtection process in terms of OWL-S inputs, outputs, conditions and effects [2].

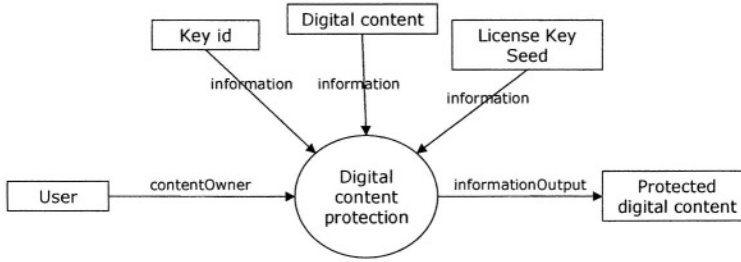


Fig. 3. Digital content protection process

Figure 4 shows a fragment of the OWL-S serialisation of the Digital-ContentProtection process. Several elements are common concepts in a DRM system.

```
<process:AtomicProcess rdf:ID="DigitalContentProtection">
  <process:hasParticipant rdf:resource="#User"/>
  <process:hasInput rdf:resource="#DigitalContent"/>
  <process:hasInput rdf:resource="#KeyId"/>
  <process:hasInput rdf:resource="#LicenseKeySeed"/>
  <process:hasOutput rdf:resource="#ProtectedDigitalContent"/>
</process:AtomicProcess>

<process:Input rdf:ID="DigitalContent">
  <process:parameterType rdf:resource="&DRMconcepts;#Content"/>
</process:Input>

<process:Input rdf:ID="KeyId">
  <process:parameterType rdf:resource="&DRMconcepts;#Key"/>
</process:Input>

<process:Input rdf:ID="LicenseKeySeed">
  <process:parameterType rdf:resource="&DRMconcepts;#Seed"/>
</process:Input>

<process:UnConditionalOutput rdf:ID="ProtectedDigitalContent">
  <process:parameterType rdf:resource="&DRMconcepts;#ProtContent"/>
</process:UnConditionalOutput>
...
```

Fig. 4. Fragment of OWL-S serialisation of digital content protection process

For this reason, they could be referenced from one or more external ontologies, that we have generically represented by the DRMconcepts entity.

Open Mobile Alliance (OMA) DRM phase 2

Although the specification of OMA DRM 2.0 has not been published yet, some of the scenarios that will be supported by it have been already presented [14]. Based on this example, we have done a preliminary approach of a workflow inside OMA DRM 2.0. The workflow present in this system can be also separated into two main phases: Content protection and content purchasing. The content protection phase describes the way a content owner can protect its content in order to purchase it in a secure way. The content purchasing phase describes how a final user can purchase a protected content and the license associated to consume it. It is also considered the super-distribution, where a user can give a protected content to another user, which afterwards purchases the corresponding license to access to it.

Figure 5 shows the content protection workflow phase. In this phase, the content is encrypted and packaged into the DRM Content Format, which can be later purchased by a user.

Figure 6 shows the content purchasing workflow phase including super-distribution. The dotted lines represent the optional part of this workflow, that mainly refers to the super-distribution. The arrow that goes back to license purchase phase from the super-distribution of content phase represents that the user that has received the super-distributed protected content from another user, has to purchase the corresponding license in order to be able to use the protected content.

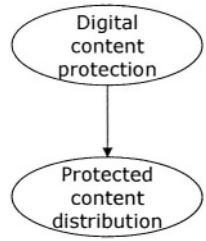


Fig. 5. Content protection phase

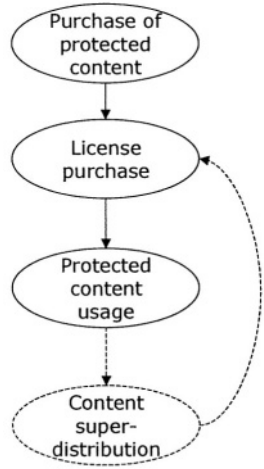


Fig. 6. Content purchasing phase, including super-distribution

Figure 7 shows the refinement of the content super-distribution phase in the form of a OWL-S process. In this refinement we describe the different components of the ContentSuperDistribution process in terms of OWL-S inputs, outputs, conditions and effects.

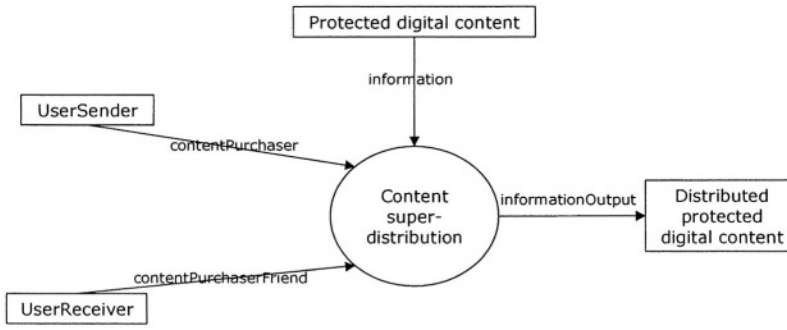


Fig. 7. Content super-distribution phase refinement

```
<process:AtomicProcess rdf:ID="ContentSuperDistribution">
  <process:hasParticipant rdf:resource="#UserSender"/>
  <process:hasParticipant rdf:resource="#UserReceiver"/>
  <process:hasInput rdf:resource="#ProtectedDigitalContent"/>
  <process:hasOutput
  rdf:resource="#DistributedProtectedDigitalContent"/>
</process:AtomicProcess>

<process:Input rdf:ID="ProtectedDigitalContent">
  <process:parameterType rdf:resource="&DRMconcepts;#ProtContent"/>
</process:Input>

<process:UnConditionalOutput
rdf:ID="DistributedProtectedDigitalContent">
  <process:parameterType rdf:resource="&DRMconcepts;#ProtContent"/>
</process:UnConditionalOutput>
...
```

Fig. 8. Fragment of OWL-S serialisation of content super-distribution process

Figure 8 shows a fragment of the OWL-S serialisation of the Content-SuperDistribution process. Several elements are common concepts in a DRM system. For this reason, they could be referenced from one or more external ontologies, that we have again represented by the DRMconcepts entity.

5 Conclusions and Future Lines

Digital Rights Management systems are being described and/or developed by companies, organisations and projects supported by administrations. Each of them proposes its own architecture and way of working.

In order to facilitate the integration among these systems (at least at a functional level), we propose the description of their workflow using a process modelling approach. The language selected to describe the processes was OWL-S (Ontology Web Language for Services), as we already used it for the description of services offered by electronic means. As OWL-S is an ontology language it allows the use of other ontologies describing concepts related to DRM or any general concept needed. The description of the processes conforming DRM systems will facilitate their understanding, the comparison between them and their possible (and desirable from a customer point of view) integration.

We have started this activity by specifying existing DRM systems following our workflow approach (see section 4.2). Our next step is to develop more general workflows, based if possible on standards, like MPEG-21 [27]. The current activities we are carrying out on MPEG-21 [28, 29] and ODRL [30] are a good environment where to start with.

Acknowledgements. This work has been partly supported by the Spanish administration (AgentWeb project, TIC 2002-01336) and is being developed within VISNET, a European Network of Excellence (http://www.visnet-noe.org), funded under the European Commission IST FP6 program.

References

1. Llorente, S. and Delgado, J. Using workflow-based systems for e-services provision. Towards the knowledge society - eCommerce, eBusiness and eGovernment, Kluwer Academic Publishers, ISBN 1-4020-7239-2. October 2002.
2. The OWL Services Coalition. OWL-S: Semantic Markup for Web Services. http://www.daml.org/services/owl-s/1.0/owl-s.pdf
3. World Wide Web Consortium (W3C). Web Ontology Language (OWL). http://www.w3.org/2004/OWL/
4. Microsoft Corporation. Architecture of Windows Media Rights Manager. May 2004. http://www.microsoft.com/windows/windowsmedia/howto/articles/drmarchitecture.aspx
5. RealNetworks, Inc. Media Commerce suite. http://www.realnetworks.com/products/commerce/description.html
6. Trymedia Systems. ActiveMARK Digital Rights Management. http://www.trymedia.com/services/security _drm.shtml
7. Perl language. http://www.perl.com
8. EURESCOM project OPERA (1207). Public deliverables. http://www.eurescom.de/public/projects/P1200-series/P1207/default.asp
9. Open Mobile Alliance. Digital Rights Management - Candidate Version 1.0. October 2003. http://www.openmobilealliance.org/release_program/docs/DRM/OMA-Download-DRM-V1_0-20031031-C.pdf
10. The Open Digital Rights Language Initiative. Open Digital Rights Language (ODRL) Version 1.1. August 2002. http://www.odrl.net/1.1/ODRL-11.pdf
11. Open Mobile Alliance. DRM Content Format – Candidate Version 1.0. November 2003. http://www.openmobilealliance.org/release_program/docs/DRM/OMA-Download-DRMCF-V1_0-20031113-C.pdf
12. Open Mobile Alliance. Generic Content Download Over The Air Specification. February 2003. http://www.openmobilealliance.org/release_program/docs/Download/OMA-Download-OTA-v1_0-20030221-C.pdf
13. Push OTA Protocol. Wireless Application Protocol. April 2001. http://www.wmlclub.com/docs/especwap2.0/WAP-235-PushOTA-20010425-a.pdf
14. Bushe, W. OMA Secure Content Delivery for the Mobile World. ODRL Workshop Keynote speech. April 2004. http://odrl.net/workshop2004/prez/odrl-buhse-prez.ppt
15. Serrao, C. et al. Open SDRM – An open and secure digital rights management solution. November 2003. http://www.crl.co.uk/projects/moses/Public/docs/IADIS03No74.pdf
16. MPEG Open Security for Embedded Systems (MOSES) project. http://www.crl.co.uk/projects/moses/
17. IPROnto. http://dmag.upf.edu/ontologies/ipronto
18. Workflow Management Coalition (WfMC). http://www.wfmc.org/
19. Delgado, J., Gallego, I., Llorente S. and García, R. Regulatory Ontologies: An Intellectual Property Rights approach. Workshop on Regulatory ontologies and the modeling of complaint regulations (WORM CoRe 2003). Lecture Notes in Computer Science (LNCS) 2889, ISBN 3-540-20494-6. November 2003.
20. Delgado, J., Gallego, I., Llorente S. and García, R. IPROnto: An Ontology for Digital Rights Management. The 16th Annual Conference on Legal Knowledge and Information Systems (JURIX-03). IOS Press, ISBN 1-58603-398-0. December 2003.
21. Llorente, S. and Delgado, J. Dynamic e-Services for Collaborative Applications. 23rd International Conference on Distributed Computing Systems – Workshops (ICDCS 2003 Workshops), IEEE Computer Society, ISBN 0-7695-1921-0. May 2003.
22. Llorente, S., Delgado, J. and Polo, J. A methodology for the development of workflow based distributed systems. IEEE ICT 2001 Proceedings, Vol. 2, GEOMA, ISBN 973-99995-1-4. June 2001.

23. Darpa Agent Markup Language (DAML) Services (DAML-S, OWL-S) http://www.daml.org/services/
24. Workflow Management Coalition. XML Process Definition Language (XPDL). October 2002 http://www.wfmc.org/standards/docs/TC 1025_10_xpdl_102502.pdf
25. Business Process Execution Language for Web Services (BPEL4WS) Version 1.1. May 2003. http://www 106.ibm.com/developerworks/webservices/library/ws-bpel/
26. Electronic Business using eXtensible Markup Language (ebXML). http://www.ebxml.org/
27. ISO/IEC. ISO/IEC TR 21000-1 Information Technology - Multimedia Framework (MPEG-21) - Part 1: Vision, Technologies and Strategy. December 2001.
28. Rodriguez, E., Llorente, S. and Delgado, J. DMAG answer to MPEG-21 Intellectual Property Management and Protection Call for Proposals ISO/IEC JTC 1/SC 29/WG 11/M10832. June 2004. http://hayek.upf.es/dmag/DMAGMPEG21Tools/m10832.pdf
29. Torres, V., Delgado, J. and Rodriguez, E. MPEG-21 DIP Core Experiments: A contribution to the implementation of DIBOs for REL ISO/IEC JTC 1/SC 29/WG 11/M10873. June 2004. http://hayek.upf.es/dmag/DMAGMPEG21Tools/ml0873.pdf
30. Delgado, J., Polo, J. and Prados, J. Interworking of Rights Expression Languages for Secure Music Distribution. 4th International Conference on Web Delivering of Music (Wedelmusic 2004). To be published. September 2004.

Automatic Classification and Analysis of Provisions in Italian Legal Texts: A Case Study

Roberto Bartolini[1], Alessandro Lenci[2], Simonetta Montemagni[1], Vito Pirrelli[1], and Claudia Soria[1]

[1] Istituto di Linguistica Computazionale - CNR, Via Moruzzi 1, 56100 Pisa, Italy
[2] Università di Pisa, Dipartimento di Linguistica, Via S. Maria 36, 56100 Pisa, Italy

Abstract. In this paper we address the problem of automatically enriching legal texts with semantic annotation, an essential pre–requisite to effective indexing and retrieval of legal documents. This is done through illustration of SALEM (Semantic Annotation for LEgal Management), a computational system developed for automated semantic annotation of (Italian) law texts. SALEM is an incremental system using Natural Language Processing techniques to perform two tasks: i) classify law paragraphs according to their regulatory content, and ii) extract relevant text fragments corresponding to specific semantic roles that are relevant for the different types of regulatory content. The paper sketches the overall architecture of SALEM and reports results of a preliminary case study on a sample of Italian law texts.

1 Introduction

The huge amount of documents available in the legal domain calls for computational tools supporting efficient and intelligent search and filtering of information. Over the last several years, machine-learning oriented research in information retrieval and document classification has spawned a number of systems capable of handling structural content management, helping users to automatically or semi-automatically identify relevant structured portions of legal texts, such as paragraphs, chapters or intertextual references. However, while knowledge management systems can certainly profit from automated detection of structural properties of regulatory texts, advanced document indexing and retrieval functions are bound to require more granular and rich semantically-oriented representations of text content. Suppose you are interested in finding all regulations applying to a particular type of individual, say an employer. Searching legal texts by using "employer" as a keyword is likely to return many irrelevant text excerpts, as plain keyword matching is blind to the particular semantic role played by the concept encoded by the term in context. Alternatively, one might be interested in tracking down all the penalties that a citizen or a member of a particular category of people is subjected to in connection with a particular behaviour. Simply searching for "citizen" and "penalty" would not be enough,

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 593–604, 2004.
© Springer-Verlag Berlin Heidelberg 2004

since there are many other possible ways to express the same concepts, whose recognition requires advanced text understanding capabilities.

To successfully address all these issues, then, we expect regulatory texts to explicitly contain the sort of implicit information that human readers are naturally able to track down through reading, *e.g.* that a certain text expresses an obligation for an employer to provide a safe environment for his employees, or that another text encodes a penalty for doing or not doing something. The process of augmenting a text with labels expressing its semantic content is what we shall hereafter refer to as *semantic annotation.* In the past, indexing textual documents with semantic tags has been a manual chore, and methods to automate this process are highly desirable. Semantic annotation of unstructured natural language texts can significantly increases the value of text collections by promoting deployment of advanced services, well beyond traditional full-text search functionalities. In this paper we intend to squarely address the problem of automatically enriching legal texts with semantic tags through illustration of SALEM (Semantic Annotation for LEgal Management), an NLP system currently used as an advanced module of the NIR[1] legal editor (see Biagioli *et al.* [2]) to automatically tag the semantic structure of Italian law paragraphs through an integration of NLP and information extraction-inspired technology.

1.1 Previous Work

Functionalities for retrieving relevant documents on the basis of text queries are provided by most current legal knowledge management tools, as witnessed by the well–known LexisNexis© and WestLaw© systems. Surely, systems differ as to the types of information search queries are sensitive to. In most cases only low-level text structures can be searched for. For instance, the tool described in Bolioli *et al.* (see [3]) is used for the automatic recognition of structural elements of a law text, and allows for intra- and inter-textual browsing of documents. Fully automatic or semi-automatic systems that carry out semantic text analysis, thus providing content-based representations, are far less common. Notable exceptions are the DIAsDEM system [5] and the approach proposed by De Busser *et al.* [4], which is however not specialized for the legal domain. In DIAsDEM, unstructured texts are iteratively processed to yield a semantic representation of their content. Although developed for a different domain, for different purposes and with different techniques, the output of this system is in line with the one described in this paper: a text augmented with domain–specific tags explicitly representing portions of its content.

A technique more similar to the one presented here is adopted by Saias and Quaresma [6], who exploit NLP techniques to yield a syntactic annotation of law texts and populate a legal ontology. Their output representation allows users to retrieve a document on the basis of sophisticated queries such as, for instance, the presence of a certain action *X* in a document, the occurrence of individual *Y*

[1] NIR ("Norme in Rete", *Laws on the web*) is a national project sponsored by the Ministry of Justice for the free access by citizens to Italian jurisdiction.

as a subject of an unspecified action, or as the performer of *X*. SALEM identifies similar types of information in texts, but its semantic representation also contains the particular type of regulation expressed by a law paragraph, in addition to the entities and actions involved. In our view, this extra information represents an added value for effective indexing and retrieval of documents. Finally, the area of research in automatic construction and population of legal ontologies, albeit not specifically intended to address the task of semantic annotation as such, also shares many of the issues we are interested in here (see for instance the work of Lame [7] and Mommers [8], among the others).

2 Methodology and Motivations

2.1 The Legal Text

As textual units, (Italian) laws are typically organized into hierarchically structured sections, the smallest one being the so-called *law paragraph.* Law paragraphs are usually numbered sections of an article, as in Figure[2] 1 below:

Article 6.
1. The Commission shall be assisted by the committee set up by Article 5 of Directive 98/34/EC.
2. The representative of the Commission shall submit to the committee a draft of the measures to be taken.

Fig. 1. A typical article.

As to its content, a law paragraph is associated with a particular *legislative provision,* which could be seen as the illocutionary point of a law section. For instance, a paragraph expresses a permission or an obligation for some actor to perform or not to perform a certain course of action, as in Figures 2 and 3.

Directive. A Member State may provide that a legal body the head office of which is not in the Community may participate in the formation of an SCE provided that legal body is formed under the law of a Member State, has its registered office in that Member State and has a real and continuous link with a Member State's economy.

Fig. 2. A Permission.

Law paragraphs may also have an inter–textual content, *i.e.* they can contain some sort of amendments to existing laws. In this case they are said to be

[2] The examples provided in the paper are taken from EC laws. Every time the purpose of the example is to illustrate semantic, language-independent phenomena, we use the English text for the sake of clarity. We remind the reader, however, that SALEM works on Italian law texts only.

Licence applications shall be accompanied by proof of payment of the fee for the period of the licence's validity.

Fig. 3. An Obligation.

modifications. For instance, a paragraph may contain an insertion with respect to another law, or a replacement, or a repeal, as the Figures 4 and 5 illustrate.

The following point shall be inserted after point 2g (Council Directive 96/61/EC) in Annex XX to the Agreement: "2h. 399 D 0391: Commission Decision 1999/391/EC of 31 May 1999 concerning the questionnaire relating to Council Directive 96/61/EC concerning integrated pollution prevention and control (IPPC) (implementation of Council Directive 91/692/EEC) (OJ L 148, 15.6.1999, p. 39)."

Fig. 4. An Insertion.

The text of point 2eg (Commission Decision 95/365/EC) in Annex XX to the Agreement shall be replaced by the following: "399 D 0568: Commission Decision 1999/568/EC of 27 July 1999 establishing the ecological criteria for the award of the Community eco–label to light bulbs (OJ L 216, 14.8.1999, p. 18)."

Fig. 5. A Replacement.

2.2 SALEM Framework

SALEM has a twofold task: a) assign each law paragraph to a given *legislative provision* type; b) automatically tag the parts of the paragraph with domain-specific semantic roles identifying the *legal entities* (i.e. actors, actions and properties) referred to in the legislative provision.

The type of semantic annotation output by SALEM is closely related to the task of Information Extraction, defined as "the extraction of information from a text in the form of text strings and processed text strings which are placed into slots labelled to indicate the kind of information that can fill them" (MUC, Message Understanding Conference). Law text analysis in SALEM is driven by an *ontology* of legislative provisions types (e.g. obligation, insertion, etc.). Classes in the ontology are formally defined as *frames* with a fixed number of (possibly optional) *slots* corresponding to the semantic roles played by the legal entities specified by a given provision type. For instance, in Figure 1 above, which expresses an obligation, the relevant roles in the first sentence of paragraph 2 are the addressee of the obligation (i.e. *The representative of the Commission*), the action (i.e. what the addressee is obliged to do, in this case to *submit to the committee a draft of the measures to be taken*) and, optionally, a *thirdjparty* (i.e. the action recipient, here *the committee*). In a similar way, a modification such as an insertion can have up to four relevant roles: (1) the reference text being modified, or *rule* (in Figure 4 above, the text *(Council Directive 96/61/EC)*

in Annex XX to the Agreement), (2) the *position* where the new text is going to be inserted (here, *after point 2g*); (3) the new text or *novella* (here, the captioned text); (4) the verbatim text to be replaced by the novella (*novellato,* not occurring in the example above). The following example illustrates the frame for an obligation:

FRAME : obligation
ADDRESSEE: the member State
ACTION: pay the advance within 30 calendar days of submission of the application for advance payment
THIRD PARTY: –

while the following is an example of the frame for a replacement.

FRAME : replacement
RULE: (Commission Decision 95/365/EC) in Annex XX to the Agreement
POSITION: –
NOVELLATO: point 2eg
NOVELLA: 399 D 0568: Commission Decision 1999/568/EC of 27 July 1999 establishing the ecological criteria for the award of the Community eco-label to light bulbs (OJ L 216, 14.8.1999, p. 18).

Automatic identification of the provision type expressed by a law paragraph is important for effective management of law texts. Law databases could be queried through fine–grained "semantic" searches according to the type of legal provision reported by a law paragraph. Furthermore, automatic identification and extraction of text portions of law that are subject to modifications could enable (semi)automatic updating of law texts, or make it possible for the history of a law to be traced throughout all its modifications; the original referenced text could be imported and modified, etc. Finally, automatic assignment of the relevant paragraph parts to semantic slots is bound to have an impact on effective legal content management and search, allowing for fine–grained semantic indexing and query of legal texts, and paving the way to real–time analysis of legal corpora in terms of logical components or actors at the level of individual provisions. In the near future, it will be possible to search an on-line legislative corpus for all types of obligation concerning a specific subject, or to highlight all possible legislative provisions a given action or actor happens to be affected by.

3 SALEM Architecture

3.1 General Overview

Although legal language is considerably more constrained than ordinary language, its specific syntactic and lexical structures still pose a considerable challenge for state-of-the-art NLP tools. Nonetheless, if our goal is not a fully-fledged representation of their content, but only identification of specific information portions, legal texts are relatively predictable and hence tractable through NLP–based techniques.

SALEM is a suite of NLP tools for the analysis of Italian texts [1], specialized to cope with the specific stylistic conventions of the legal parlance, with the aim to automatically classify and semantically annotate law paragraphs. A first prototype of SALEM has just been brought to completion and its performance evaluated. The NLP technology used for SALEM is relatively simple, but powerful, also thanks to the comparative predictability of law texts. SALEM takes in input single law paragraphs in raw text and outputs a semantic tagging of the text, where its classification together with the semantic roles corresponding to different frame slots are rendered as XML tags. An output example (translated into English for the reader's convenience) is given in Figure 6, where the input paragraph is classified as an *obl(igation)* and portions of the text are identified as respectively denoting the *addressee* and the *action.*

```
<obligation><obl:addressee>The Member State</obl:addressee> shall
<obl:action>pay the advance within 30 calendar days of submission of the
application for advance payment </obl:action>.</obligation>
```

Fig. 6. SALEM output example.

SALEM approach to classification and semantic annotation of legal texts follows a two stage strategy. In the first step, a general purpose parsing system, hand-tuned to handle some idiosyncracies of Italian legal texts, pre–processes each law paragraph to provide a shallow syntactic analysis. In the second step, the syntactically pre–processed text is fed into the semantic annotation component proper, making explicit the information content implicitly conveyed by the provisions.

3.2 Syntactic Pre–processing

Syntactic pre–processing produces the data structures to which semantic annotation applies. At this stage, the input text is first tokenized and normalized for dates, abbreviations and multi–word expressions; the normalized text is then morphologically analyzed and lemmatized, using an Italian lexicon specialized for the analysis of legal language; finally, the text is POS-tagged and shallow parsed into non–recursive constituents called "chunks".

A sample chunked output is given in Figure 7. A chunk is a textual unit of adjacent word tokens sharing the property of being related through dependency relations (es. pre–modifier, auxiliary, determiner, etc.). Each chunk contains information about its type (e.g. a noun chunk (N_C), a verb chunk (FV_C), a prepositional chunk (P_C), an adjectival chunk (ADJ_C), etc.), its lexical head (identified by the label *potgov*) and any intervening modifier, causative or auxiliary verb, and preposition. A chunked sentence, however, does not give information about the nature and scope of inter–chunk dependencies. These dependencies, whenever relevant for semantic annotation, are identified at the ensuing processing stage (see section 3.3 below).

All'articolo 6 della legge 24_gennaio_1986 , n._17 , le parole: "entro centottanta giorni dalla pubblicazione della presente legge nella Gazzetta Ufficiale" sono soppresse.

1. [[CC: P_C] [PREP: A#E] [DET: LO#RD@MS] [AGR: @MS] [POTGOV: ARTICOLO#S@MS]] All' articolo
2. [[CC:ADJ_C] [POTGOV: 6#N]] 6
3. [[CC: di_C] [DET: LO#RD@FS] [AGR: @FS] [POTGOV: LEGGE#S@FS]] della legge
4. [[CC: ADJ_C] [POTGOV: 24_gennaio_1986#N]] 24_gennaio_1986
5. : [[CC: PUNC_C] [PUNCTYPE: ,#@]] ,
6. [[CC: ADJ_C] [POTGOV: n._17#N]] n._17
7. [[CC: PUNC_C] [PUNCTYPE: ,#@]] ,
8. [[CC: N_C] [DET: LO#RD@FP] [AGR: @FP-@FP] [POTGOV: PAROLA#S@FP PAROLE#S@FP]] le parole
9. ...

Fig. 7. A fragment of chunked text.

Although full text parsing may be suggested as an obvious candidate for adequate content processing, we contend that shallow syntactic parsing provides a useful intermediate representation for content analysis. First, at this stage information about low level textual features (e.g. punctuation) is still available and profitably usable, whereas it is typically lost at further stages of analysis. In this connection, it should be appreciated that correct analysis of modifications crucially depends on punctuation marks, and in particular on quotes and colons, which are used to identify the text of the amendment *(novella)* and the amending text *(novellato).* Secondly, chunked sentences naturally lend themselves to incrementally being used as the starting point for partial functional analysis, whereby the range of dependency relations that are instrumental for semantic annotation is detected. In particular, dependency information is heavily used for the mark–up of both modifications and obligations, which requires knowledge of the underlying syntactic structure. Finally, a third practical reason is that chunking yields a local level of syntactic annotation. As a result, it does not "balk" at domain–specific constructions that violate general grammar patterns; rather, parsing is carried on to detect the immediately following allowed structure, while ill–formed chunks are left behind, unspecified for their category.

3.3 Semantic Annotation

As mentioned above, the SALEM approach is closely inspired by mainstream techniques of Information Extraction. In particular, semantic annotation consists in the identification of all the instances of particular provision types in text. The frame defining a provision type can then be regarded as an *extraction template* whose slots are filled with the textual material matching the corresponding conceptual roles.

The semantic annotation component takes in input a chunked representation of each law paragraph and identifies semantically–relevant structures by applying domain–dependent, finite state techniques locating relevant patterns of chunks. Semantic mark–up is performed through a two–step process:

1. each paragraph is assigned to a frame (corresponding to the legislative provision expressed in the text);
2. slots of the frame identified at step (1) are turned into an extraction template and instantiated through the structural components (i.e. sentences, clauses, phrases) of the law paragraph.

The current version of the semantic annotation component is a specialized version of the ILC finite–state compiler of grammars for dependency syntactic analysis [9]. The SALEM version of the grammar compiler uses a specialized grammar including (i) a core group of syntactic rules for the identification of basic syntactic dependencies (e.g. subject and object), and (ii) a battery of specialized rules for the semantic annotation of the text.

All rules in the grammar are written according to the following template:

<chunk-based regular expression> WITH <battery of tests> => <actions>

The recognition of provision types and the subsequent extraction of information from the text are based on structural patterns which are combined with lexical conditions and other tests aimed at detecting low level textual features (such as punctuation) as well as specific syntactic structures (e.g. the specification of a given dependency relation). Structural patterns are expressed in terms of regular expressions over sequences of chunks, whereas all other conditions (*e.g.* lexical, syntactic, etc.) are checked through a battery of tests. The action type ranges from the identification of basic dependency relations (in the case of syntactic rules) to the semantic mark–up of the text (in the case of semantic annotation rules).

The assignment of a paragraph to a provision class is based on combinations of both syntactic and lexical criteria. As already mentioned above, a preliminary study of the linguistic features of law paragraphs revealed a strong association between classes of verbs and provision types: an obligation is typically expressed with the modal verb "dovere" *(shall/must)* or the verbal construction "essere obbligato/tenuto a" (*to be obliged to*); similarly, lexical cues for the identification of an insertion include verbs like "aggiungere" or "inserire" (*to add, to insert*). We will refer to these verbs as trigger verbs. The presence of a trigger verb in the text is only the first step towards the detection of a specific provision class which, in order to be completed, needs to be complemented with structural tests which include the role of the trigger verb as the sentence head and the types of dependency relations governed by it.

Reliable identification of dependency relations is also important for assigning structural elements to semantic roles, since the latter tend to be associated with specific syntactic functions (e.g. subject, object). To give the reader but one example, the addressee of an obligation typically corresponds to the syntactic

subject of the sentence, while the action (s)he is obliged to carry out is usually expressed as an infinitival clause, as in the example reported below:

> [[Il comitato misto]subj]addressee] è tenuto a raccomandare modifiche degli allegati secondo le modalità previste dal presente accordo
> [[*The Joint Committee*]subj]addressee] *shall be responsible for recommending amendments to the Annex as foreseen in this Agreement.*

Note, however, that this holds only when the verbal head of the infinitival clause is used in the active voice. By contrast, the syntactic subject can express the third–party if the action verb is used in the passive voice and is governed by specific lexical heads.

4 Case Study

We report the results of a case study carried out with SALEM on the basis of a small ontology covering 8 provision types. This section presents the ontology of provisions and illustrates the system's performance against a corpus of law paragraphs previously annotated by law experts.

4.1 The Frame–Based Legal Ontology

In this case study, we have used a small ontology designed by experts in the legal domain at ITTIG–CNR.[3] The ontology distinguishes three major categories of provisions: *obligations, definitions* and *modifications.* A main distinction can be made between obligations, addressing human actors, and modifications, which are rather aimed at modifying the textual content of pre–existing laws. Obligations in turn divide into the following classes: *obligation, prohibition, permission,* and *penalty.* In their turn, modifications are further subdivided into *replacement, insertion* and *repeal.* The taxonomical structure of the SALEM ontology is illustrated in Figure 8.

As mentioned above, law paragraphs are analyzed in SALEM not only according to the particular type of legislative provision they express, but also with respect to the main legal entities involved by the law. Consistently, each ontology class is formally defined as a *frame* with a fixed number of (possibly optional) *slots.* The slot types required for the description of the 8 bottom classes in the SALEM taxonomy are illustrated in Table 1.

4.2 Evaluation Results

SALEM preliminary results are very promising. The system was evaluated on a sample of 473 law paragraphs, covering seven ontology classes of Table 1. The test corpus was built by law experts at ITTIG-CNR, who also provided a hand–annotated version used as gold standard for evaluation. The aim of the evaluation

[3] Istituto di Teoria e Tecniche dell'Informazione Giuridica, CNR, Florence, Italy.

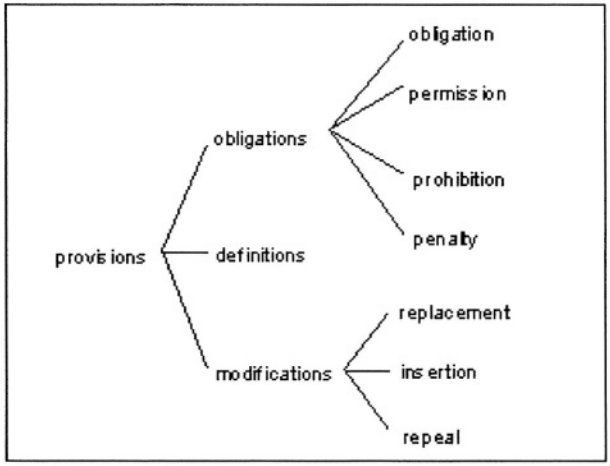


Fig. 8. Taxonomical structure of the SALEM ontology

Table 1. Frame-based description of the different provision types

Provision class	Slots
Obligation	Addressee, Action, Third-party
Permission	Addressee, Action, Third-party
Prohibition	Action, Third-party
Penalty	Addressee, Action, Object, Rule
Definition	Definiendum, Definiens
Repeal	Rule, Position, *Novellato*
Replacement	Rule, Position, *Novellato*, *Novella*
Insertion	Rule, Position, *Novella*

was to assess the system's performance on two tasks: classification of paragraphs according to the ontology (henceforth referred to as "classification task") and mark–up of semantic roles (henceforth "information extraction task").

Table 2 summarizes the results achieved for the paragraph classification task with reference to the seven bottom classes of provisions, where Precision is defined as the ratio of correctly classified provisions over all SALEM answers, and Recall refers to the ratio of correctly classified provisions over all provisions in the test corpus. Note that here a classification is valued as correct if the automatically assigned class and the manually assigned one are identical. The classification performance is even better if it is related to the corresponding first level ontology classes (i.e. obligations and modifications). In fact, in some cases, mostly penalties and permissions, multiple answers are given due to the fact that obligations bottom classes share a great deal of lexical and morpho-syntactic properties; yet, these answers are to be considered correct if classification is evaluated with respect to the first level ontology classes (i.e. obligations

Table 2. SALEM results for the classification task.

Class	Total	SALEM answers	OK	Precision	Recall
Obligation	19	19	18	0.95	0.95
Permission	15	18	15	0.83	1
Prohibition	15	15	14	0.93	0.93
Penalty	122	117	109	0.93	0.89
Repeal	70	69	69	1	0.99
Replacement	111	111	111	1	1
Insertion	121	119	119	1	0.98
Tot.	**473**	**468**	**455**	**0.97**	**0.96**

and modifications). On the other hand, when unambiguous linguistic patterns are used, the system easily reaches 1 for both Precision and Recall, as with the class of Modifications.

Table 3 illustrates the performance of the system in the information extraction task. The aim of the evaluation here was to assess the system's reliability in identifying, for each provision type or frame, all the semantic roles that are relevant for that frame and are instantiated in the text. For each class of provisions in the test corpus we thus counted the total number of semantic roles to be identified; this value was then compared with the number of semantic roles correctly identified by the system and the total number of answers given by the system. Here, Precision is scored as the number of correct answers returned by system over the total number of answers returned, while Recall is the ratio of correct answers returned by system over the number of expected answers.

Table 3. SALEM results for the information extraction task.

Class	OK	Expected answers	SALEM answers	Precision	Recall
Obligation	38	38	38	1	1
Permission	20	25	24	0.83	0.8
Prohibition	21	21	27	0.78	1
Penalty	303	388	330	0.92	0.78
Repeal	104	108	106	0.98	0.96
Replacement	303	309	306	0.99	0.98
Insertion	373	376	375	0.99	0.99
Tot.	**1162**	**1265**	**1206**	**0.96**	**0.92**

5 Conclusions and Future Work

In this paper we presented SALEM, an NLP–based system for classification and semantic annotation of law paragraphs. Although we follow the mainstream trend in state-of-the-art NLP architectures towards use of shallow parsing techniques, the novelty of our approach rests in the incremental composition of shallow parsing with higher levels of syntactic and semantic analysis, leading to

simultaneous, effective combination of low- and high-level text features for fine-grained content analysis. Besides, the cascaded effect of incremental composition led to a considerable simplification of the individual parsing modules, all of which are implemented as finite state automata. The effectiveness and performance of the system was tested with promising results on the basis of a small ontology of provision types, against a test-bed of text material hand-annotated by human experts.

We are presently working along several lines of development. On the one hand, we intend to make the system more robust by testing it on a larger sample of law texts. Although laws are quite stable as a language genre, they can also be stylistically variable depending on the personal inclinations of the author, the particular domain they apply to, not to mention variations determined by historical changes. Collection of a wider variety of text material is bound to have an impact on SALEM flexibility and coverage. On the other hand, we also aim at expanding the ontology of provisions for semantic annotation to cover new provision types.

References

1. Bartolini, R., Lenci, A., Montemagni, S., Pirrelli, V.: The lexicon-grammar balance in robust parsing of Italian. Proc. of 3rd International Conference on Language Resources and Evaluation. (2002)
2. Biagioli, C., Francesconi, E., Spinosa, P., Taddei, M.: The NIR project. Standards and tools for legislative drafting and legal document Web publication. Proc. of the International Conference of Artificial Intelligence and Law. Edinburgh, June 24 (2003)
3. Bolioli, A., Dini, L., Mercatali, P., Romano, F.: For the automated mark-up of Italian legislative texts in XML. Proc. of JURIX 2002, London, UK, 16-17 December (2002)
4. De Busser, R., Angheluta, R., Moens, M.-F.: Semantic Case Role Detection for Information Extraction. Proc. of COLING 2002. New Brunswick (2002) 1198–1202
5. Graubitz, H., Winkler, K., Spiliopoulou, M.: Semantic Tagging of Domain-Specific Text Documents with DIAsDEM. Proc. of the 1st International Workshop on Databases, Documents, and Information Fusion (DBFusion 2001). Gommern, Germany (2001) 61–72
6. Saias, J., Quaresma, P.: Using NLP techniques to create legal ontologies in a logic programming based web information retrieval system. Proc. of ICAIL 2003 Workshop on Legal Ontologies and Web Based Legal Information Management. Edinburgh, UK, June 24-28 (2003)
7. Lame, G.: Using text analysis techniques to identify legal ontologies' components. Proc. of ICAIL 2003 Workshop on Legal Ontologies and Web Based Legal Information Management. Edinburgh, UK, June 24-28 (2003)
8. Mommers, L.: A knowledge-based ontology for the legal domain. Proc. of the Second International Workshop on Legal Ontologies. December 13 (2001)
9. Bartolini R., Lenci A., Montemagni S., Pirrelli V.: Grammar and lexicon in the robust parsing of Italian: Towards a non-naïve interplay. Proc. of Coling 2002 Workshop on Grammar Engineering and Evaluation. Academia Sinica, Nankang, Taipei, Taiwan, 1st September (2002)

Engineering an Ontology of Financial Securities Fraud

Gang Zhao[1], John Kingston[2], Koen Kerremans[3], Frederick Coppens[4], Ruben Verlinden[1], Rita Temmerman[3], and Robert Meersman[1]

[1] STARLab, Computer Science, Vrije Universiteit Brussel, Belgium
{gzhao, rverlind,meersman}@vub.ac.be
[2] Joseph Bell Centre, University of Edinburgh, Scotland
j.kingston@ed.ac.uk
[3]CVC, Erasmushogeschool Brussel, Belgium
{koen.kerremans, rita.temmerman}@ehb.be
[4]Language & Computing NV, Belgium
frederik@landcglobal.com

Abstract. This paper discusses the approach of ontology-based knowledge engineering in *FF POIROT,* a project to explore the use of ontology technology in information systems against financial fraud. A fraud forensic ontology is being developed from laws, regulations and cases about illegal solicitation of financial products on the web. The knowledge development is based on the *DOGMA* ontology paradigm, and the derived ontology engineering methodology *AKEM.* The regulatory ontology engineering is a multi-disciplinary and distributed team work through a series of tasks and deliverables with emphasis on the traceability of decision making in the development. The machine ontology extraction and a manually constructed bilingual terminological database, is used to support the ontology modelling process.

1 Introduction

This paper discusses the approach and current results in *FF POIROT* (Financial Fraud Prevention Oriented Information Resources using Ontology Technology), a research and technology project to explore the use of ontology technology in information systems against financial fraud. The ontology targeted is one of securities fraud forensics. As one part of the ontology concerns the evidences of illegal conduct with respect to public regulations, the ontology is based on the readings of legislature scoped with real life cases of violation. The experience reported here is concerned with the illegal solicitation of financial instruments on the internet, in the light of regulations used by *Commissione Nazionale per le Società e la Borsa* (CONSOB), the authority regulator of security exchanges in Italy.

The ontology is intended to be used as baseline for web sites filtering and evidence recognition to assist investigators in their surveillance on securities frauds on the internet. One type of securities fraud on the internet is the unlawful solicitation of investment to the general public with intention to deceive or defraud. The law and

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 605–620, 2004.
© Springer-Verlag Berlin Heidelberg 2004

regulation provide an important part of the basis for identifying the evidence of frauds and justifying its prosecution.

We shall discuss how we are building the fraud forensic ontology from regulations in the light of the specific cases. Sect. 2 describes the nature of the problem, Sect. 3 introduces the ontology development framework and methodology, Sect. 4 discusses the ontology-based knowledge engineering in FF POIROT and Sect. 5 concludes on future development.

2 Problem Definition: Description of Application Domain

CONSOB's responsibility of regulating the Italian securities market is to ensure the transparency and correct behaviour of securities market participants, disclosure to the Italian investing public, complete and accurate information about listed companies, in compliance with Italian regulations on securities exchanges. As the internet becomes an important medium of illegal solicitation of financial instruments, an important aspect of CONSOB's work is to monitor websites offering online investment services to Italian residents to determine if there is any unauthorized or fraudulent solicitation of public savings.

The challenge faced by the CONSOB investigator is to cope with the vastness of the internet and narrow down their investigation space effectively on non-compliant or fraudulent websites. Context-insensitive and key word based search has proven unproductive and more so as the internet grows.

An ontology-driven approach is explored in order to filter web sites with specific concerns and read web pages for evidences to further short-list targets of examination by ontology-based information search and extraction.

3 Ontology Engineering

The FF POIROT ontology represents the concepts and relations underlying the expert knowledge of fraud forensics. The purpose of capturing this meta knowledge is to provide a baseline for

- Analyzing and modelling divergent fraud cases
- Developing applications where different knowledge bases such as lexicon for natural language parsing, production rules for inferencing are deployed

This section introduces the DOGMA ontology approach and a knowledge engineering process based on the ontology approach.

3.1 DOGMA Approach to Ontology Modelling

The DOGMA (**D**eveloping **O**ntology-**G**uided **M**ediation for **A**gents) approach to ontology advocates a two layered approach to ontology modelling [6], [12]: the *on-*

tology base and the *ontological commitments.* The ontology base represents the conceptualization of a subject domain valid for a family of applications. It is a set of lexons. The ontological commitments are the specific interpretation of lexons in the ontology base. They consist of lexons selected, organized, instantiated or constrained in the light of a specific task or application in a particular system context. The commitments serve to localise application specific interpretations thus facilitating the diverse use of lexons. They also lead to relative application-independence of lexons, which contributes to the consensus building and design scalability of the ontology base.

3.1.1 Lexons and Ontology Base

Lexons represent relationships between two concepts. They constitute the vocabulary for describing application semantics. A *lexon* is a quintuple $<\gamma, t_1, r_1, t_2, r_2>$, where $\gamma \in \Gamma$ is a context identifier, $t_1 \in T$ and $t_2 \in T$ are *terms* referring to conceptual entities, $r_1 \in R$ and $r_2 \in R$ are roles in the conceptual relationship. Γ, T and R are strings over an alphabet, $\Sigma+$. The lexon is a conceptual construct to encode fact types in the subject domain. The context identifier (indicates an ideational context as opposed to the application context. It symbolizes the conceptual scope or perspective for the intrinsic meanings of terms and roles in a lexon. The ideational context is externalized by a set of resources, such as documents, graphs, databases. Through such resources, the semantic extension of a lexon is established, communicated, documented and agreed upon among ontology developers.

The *ontology base* is a bag of lexons, unordered and unstructured. Formally it is a pair $<\Sigma, \Omega>$ where Σ is the alphabet and $\Omega = \Gamma \times T \times R \times T \times R$. It is a system of symbolic choices, from which the subset of lexons is selected, instantiated, constrained with axioms to describe the semantics of a particular application task.

3.1.2 Commitments

Lexons become fully specified in the application context on the commitment layer. The ontological *commitment* is application-specific interpretation of lexons. The commitments are a projection of the lexon base, in which the denotation of lexons becomes fully specified, consistent and unambiguous, in view of a particular task, application or service [2], [6], [16]. The processing agent for a given task commits to a selected set of lexons constrained and organized in particular networks.

3.2 AKEM

AKEM (**A**pplication **K**nowledge **E**ngineering **M**ethodology) [16], [17] is an ontology-based knowledge engineering method, translating the theoretical framework of the DOGMA into the engineering dimensions of time, task, deliverables and best practices. It stresses informal specification in a semi-formal approach, especially at the early stage of sharing, agreeing and managing requirements in a multi-disciplinary team of application users, domain experts, system analysts, knowledge

analysts, ontology engineers, linguistic engineesr, application developers possibly in a distributed environment.

3.2.1 Life Cycle Model

AKEM adopts the Rational Unified Process (RUP) for software engineering [5], [9], [10] as its lifecycle model emphasing on requirements management, iterative and component-based development, use-case based evolution. It has two tracks: system development and knowledge development. We shall concentrate on the latter here.

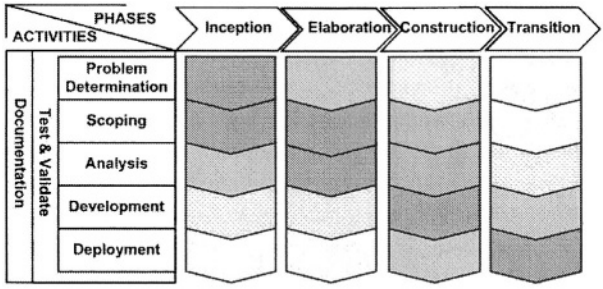


Fig. 1. AKEM Life Cycle Model

The inception phase defines problems and assesses suitability of knowledge-based solutions. Its focus is to collect the knowledge resources, analyse the problem space and scope necessary knowledge coverage. The elaboration phase consolidates the work on knowledge scopes and examines the knowledge structure involved in the scope. The construction phase takes over the specification of the knowledge analysis and describes the underlying conceptions to build the ontology base. The main issue of the transition phase is how the ontological commitments are specified with respect to processing tasks and translated into the required knowledge representation for deployment.

AKEM emphasizes the requirements management by knowledge. The explicit communication of the scope of coverage at particular points of knowledge modelling is necessary for productive modelling, team work and evolution management. Similar to use cases RUP and 'motivating scenarios' [4], *stories* are used for the scope specification. They are instruments to elicit the knowledge in reading legal and regulatory texts through cases in legal professions. They are thus mechanism for team communication and documentation in regulatory ontology engineering.

In the emphasis of the traceability of the decision-making in ontology engineering, AKEM seeks to organize the input and output of each activity in the form of natural language texts and creates links among them to document footprint of development, for natural language texts are the most effective medium of communicating semantics and its contexts in a multidisciplinary team. This emphasis puts us in line with effort in building ontologies from texts [1], [13].

In order to build ontologies of flexible architecture and topological adaptation, AKEM advocates iterative development as opposed to the prototyping development cycle, widely practiced in ontology engineering and expert system development. The

objective is to subject the feedback from application specific development to overall domain consideration in the upper stream activities. The rationale is to create ontology with 'minimal encoding bias' [3], late commitments to application tasks and processing paradigms.

3.2.2 Activities and Deliverables

This section summarizes the activities and deliverables. The illustration with examples will be left to Sect. 4. During each phase, seven activities are performed. The documentation and validation runs through the rest of five: problem/solution determination, scoping, analysis, development and deployment.

Scoping on the semantic space is less tangible than on information system functionalities. It is not only to identify the part of the semantics proper to focus on, but also to convey its domain contexts. The scoping activity in the domain perspective in AKEM produces two main deliverables: knowledge resources (documents, interview protocols) and stories (knowledge use cases). The analysis activity that follows aims at the knowledge constituent structure within the specified scope. The knowledge constituent structure shows the knowledge breakdown and the elaboration of each constituent. The development activity builds ontology. While the scoping and analysis is application-oriented, the development activity starts with abstracting away from application specifics.

Given the stories and knowledge constituent analysis in documents, the ontology modelling involves three subtasks: extraction, abstraction and organization. The extraction identifies key words and phrases in the documents. Two techniques are used: highlighting and paraphrasing, depending if the text input is too condensed or technical and if the ontology engineer needs to use paraphrases to confirm and validate his reading with domain experts and knowledge analysts. The abstraction encodes the meanings of the extraction in lexons. Best practice in the extraction and abstraction is to work exclusively on what is explicitly expressed in the text. The text is treated as explicit boundaries of conceptual scope. The third sub-activity, organization, lifts the text-bound restriction. It seeks to capture conceptions implicit in the document, import and integrate from existing ontologies and resources. The purpose is to structure and complete the conceptual coverage of lexons. The ontology modelling in AKEM thus takes text-based, bottom-up first approach. The deployment consists of two tasks: the specification of the ontological commitments in the light of specific processing tasks and transformation of the commitments into a particular knowledge specification [2], [17], such as OWL and KIF.

Since they impose particular formats and imply specific viewpoints, the deliverables also serve as instrument of activity management.

4 Ontology Engineering in FF POIROT

Since frauds are defined against the legislature, the text of laws and regulations are the prime sources of ontology modelling. As ontology is intended for filtering and

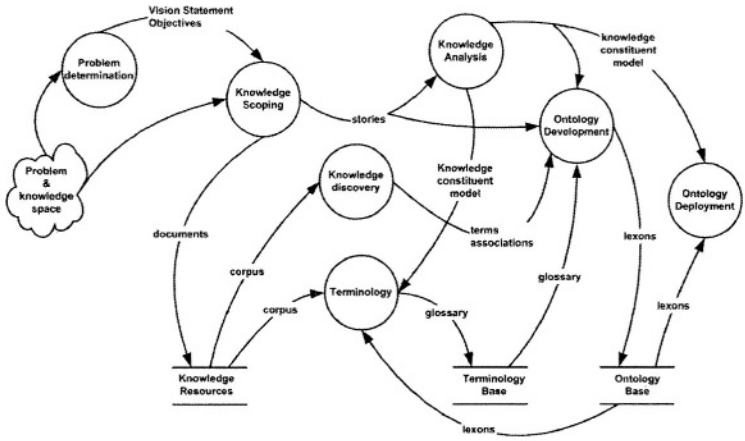


Fig. 2. Work Flow of Knowledge Engineering in FF POIROT

interpreting web pages for fraud evidences, the natural language text is also the medium of fraud forensics. This characteristic combines domain and language engineering in text-based ontology modelling for ontology-based text.

Figure 2 presents the AKEM work flow in FF POIROT. The methodology breaks down the process of development into small tasks towards specific deliverables in pipelines of work to support independent as well as collaborative team work. The ontology development takes stories and knowledge constituent analysis as main input. The extraction from texts is both manual and automatic. The automatic extraction takes place in the knowledge discovery, extracting terms and their associations on linguistic and statistic patterns (Sect. 4.4). The work on terminology (Sect. 4.5) also contributes to the ontology development, emphasising more the lexical semantics than the application semantics. Its work scope is represented by the knowledge resources, though it also considers stories and knowledge constituent analysis for validation.

4.1 Knowledge Scoping

The knowledge scoping delivers a collection of knowledge resources and knowledge use cases, stories. The former seeks to cover the domain knowledge with relevant documents and data, whereas the latter zooms on a particular application specific domain space in structured knowledge engineering.

4.1.1 Knowledge Resources

The knowledge resources about the subject domain pertain to finance, business, law and internet technologies. They are in the form of legislature, lexical databases, encyclopaedias, web sites of financial institutions and financial ontologies [8]. The knowledge resources are used for three main purposes in the project: knowledge analysis

and ontology modelling, construction of terminology of financial securities frauds and training and coding natural language technologies for knowledge discovery. Our focus of attention is on the regulations that govern both CONSOB's jurisdiction over a company and the legality of business activity. They are CONSOB Consolidated Law on Financial Intermediation, Legislative Decree 58/1998 and its implementing provisions, CONSOB Regulation 11522/1998 and 11971/1999 [11]. A number of cases of frauds, including their websites for forensic analysis are also part of the knowledge resources. Besides being used for devising ontology, they are also treated as linguistic corpora for training natural language processing systems or empirical terminography by the knowledge discovery and terminology compilation.

4.1.2 Stories

The story is used to describe the system's intelligent behaviour, the knowledge required and its context.

The main purpose is three fold. Firstly, it is an instrument for the knowledge analyst to validate his analysis with the user and domain expert through actual cases. Secondly, it defines, documents and communicates to knowledge/ontology engineers the scope of modelling and functionality under attention at a particular junction in the phases of development. Thirdly, it provides the basis for testing the system's knowledge coverage. In the context of regulatory ontology, it serves as summarization or generalization of cases with which regulations and laws are interpreted through case studies. In FF POIROT, it is used to describe prototypical fraud cases to indicate the scope and structure of the underlying knowledge, as illustrated in Figure 3. Though its diction is in natural language, the story has a structure to impose a conceptual frame of analysis, communication and documentation. The episodes can present events and process. They can be also declarative description of dimensions or aspects. Figure 3 illustrates a mixture of static and dynamic aspects of evidences to which the legislature applies.

4.2 Knowledge Analysis

The knowledge analysis consists of three tasks. Firstly, relevant documents, laws, degrees, and clauses, are selected in the light of stories from the knowledge resources. Secondly, the knowledge scoped by the story and substantiated by the relevant documents is analysed into a hierarchical structure. Thirdly, each section in the knowledge breakdown is elaborated. The resultant deliverable of this 3-step activity is the knowledge constituent analysis.

4.2.1 Knowledge Breakdown

The knowledge constituent structure is a directed graph, describing a model of structural composition of pieces of domain expertise.

STORY			
Title	[redacted] allegation 1	**ID**	C1
Theme	Unauthorized investment solicitation of investors and public offering by [redacted] Ltd on WWW		
Author	Joseph Bell Centre, University of Edinburgh		
Source	CONSOB, Italy	Date	8 August 03
Setting			
S1	The company [redacted] Ltd, headquartered in the British Virgin Islands, aims to become an unofficial 24-hour stock exchange offering stocks for sale by means of its webpage, [redacted]		
S2	Italian potential investors were solicited by this WWW Page advertising and offering financial instruments		
S3	Investors operating in the company's market must first become shareholders of [redacted] Ltd., paying a 'contribution of association'		
S4	Through the mentioned website, shares/units of the companies listed in the trading system are offered to the public		
S5	Through the mentioned website, shares of [redacted] Ltd are also offered to the public		
Episode			
E1.1	Foundation of the company [redacted] Ltd in British Virgin Islands		
E1.2	Creation of URL for [redacted] Ltd: [redacted]		
E1.3	[redacted] Ltd starts advertising, soliciting and offering investment products on this URL		
E2.1	CONSOB prohibited the selling of [redacted] shares addressed to potential investors who intend to operate in the organized exchanges system created by the same [redacted] Ltd and the offering of the shares/units of the companies that join, for their listing, the same system, whose activities are advertised through the mentioned Internet site [redacted] without previous procedure pursuant to legislative decree 58/1998		
E2.2	It is necessary to consider whether CONSOB has appropriate jurisdiction over E1.3. CONSOB considered that its jurisdiction was asserted as [redacted] targeted Italian residents according to CONSOB's jurisdiction criteria		
E2.3	The public offering on the website was not authorized by CONSOB as required under legislative Decree 58/1998		
E2.4	A prospectus related to these public offerings was not published in the specific Prospectus Archives of CONSOB (nor was a procedure for the approval of an informative prospectus started) as required under legislative Decree 58/1998		
E2.5	The soliciting operator gave also false statements on their web page		
Note			
AS1	[redacted] Ltd enters into a contract of participation with [redacted] Italia S.r.l. The latter will effectively carry out the enterprise activity as from 31 March 2001.		
AS5	Also participation certificates (deriving from the participation contract in AS1) are offered on the webpage		
AE1.1	At a certain point in time, [redacted] Ltd liquidated and resumed as a new company, this time incorporated in the Republic of Panama		
AE1.3	The products which are offered on the website, need to be qualified as 'financial products' according to CONSOB's criteria of what constitutes a financial product		
AE2.2	This must be considered separately from the regulator's ability to enforce its powers within such jurisdiction. For example, a securities regulator would first need to consider whether a security was being advertised or sold within its geographical jurisdiction and secondly, whether or not the person advertising the product was subject to their regulation.		
AE2.4	These are 'information requirements'		
AE2.5	This is understood as the capacity for dissemination of false and misleading information about securities on the Internet		

Fig. 3. Illustration of a story

Figure 4 is a graphical illustration of the knowledge constituent structure produced in FF POIROT based on the story in Figure 4. It is an extension of the Wigmore Chart used in legal studies [15], consisting of three layers: hypothesis, jurisdiction and evidence. The jurisdiction layer is symbolised by the white nodes. The names of the hypothesis and evidence nodes are prefixed with *H* and *E* respectively. The fourth

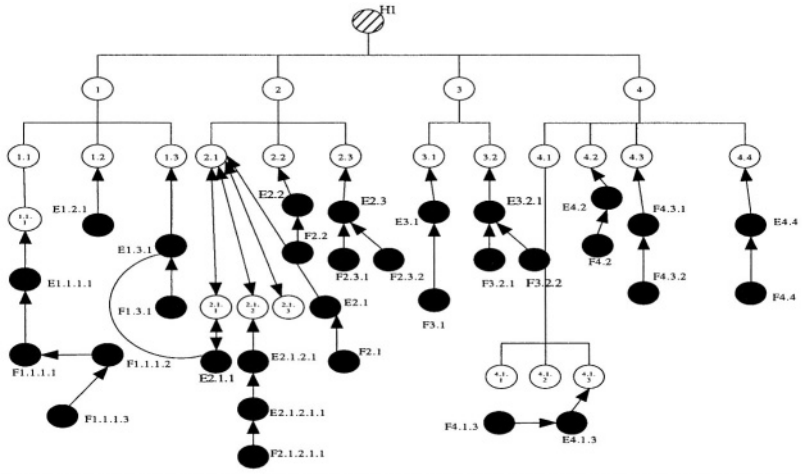


Fig. 4. Knowledge constituent structure in the Wigmore chart

layer of the knowledge constituent structure consists of nodes of facts interpreted as evidences. Such knowledge constituent structure is both an argumentation model for fraud prosecution and identification model for fraud detection. In addition, it also provides a structure or index of the knowledge resources. Each constituent refers to particular knowledge resources.

4.2.2 Knowledge Elaboration

The knowledge elaboration zooms onto the details of each constituent to specify logical definitions, axioms and statements to convey the rules and heuristics in the expert's knowledge. Figure 5 illustrates the elaboration of some of the knowledge constituents defined in Figure 4.

The elaboration statements are structured as the knowledge constituent structure and specified in a style of program design language, *Structured English* in simple

1. Unlawful **IF**
1.1 Organizer solicits investors on the WWW
1.1.1 Organizer manages a website that solicits investors
E1.1.1.1 Website is managed/registered by organizer
F1.1.1.1 Website states the name '[redacted] Ltd'
F1.1.1.2 Website registration details indicate '[redacted] Ltd' as registrant of website
F1.1.1.3 URL = www.[redacted].com
1.2 **AND** no advance notice of solicitation to CONSOB
E1.2.1 [redacted] did not give a notification to CONSOB re public offer to purchase
1.3 **AND** no related prospectus filed with CONSOB
E1.3.1 [redacted] Ltd did not drafted or filed a prospectus with CONSOB re public offer to purchase
F1.3.1 Consultation of Societa' and Intermediari sections on CONSOB website reveal that CONSOB did not received any prospectus from [redacted] Ltd.

Fig. 5. An example of knowledge elaboration

straightforward English combined with the reserved logical operators. The unary qualifiers include *not, true, false* and modality such as *possibility, probability.* The binary connectors include *and, or, if-then.* The context of these elaboration statements is the knowledge constituent structure and references of each constituent to the knowledge resources.

4.3 Ontology Development

Given the scope specification and knowledge constituent analysis, the activity of ontology development seeks for the underlying conceptual entity types and fact types. It focuses on the domain conceptualization rather than applications' intelligent behaviour and functionalities. Instead of going straight to build the knowledge base, AKEM builds an ontology of application semantics at a more abstract and generic level of meta knowledge underpinning the rules in the knowledge base.

4.3.1 Extraction

The extraction is a linguistic work in terms of its input and output. It works on natural language texts selected and generated in the knowledge scoping and analysis: knowledge resources, stories and knowledge constituent analysis. It requires linguistic knowledge, especially when dealing with complex legislative texts as well as domain knowledge. Within the framework set up by stories and knowledge constituent analysis it concentrates on spotting the elementary semantic elements rather than putting them into a coherent whole. Figure 5 shows a set of knowledge elaborations, read from CONSOB's Degree 58. It was produced by the knowledge analyst from legal profession. At the ontology development stage, the ontology engineer analysed it for ontology building, using key words highlighting techniques.

58.94.1
IF offerers who make a public offering did not give advance notice thereof to CONSOB, attaching the prospectus to be published
THEN The offerers are conducting unauthorized solicitation of investors
58.94.3-4
IF the public offering concerns financial products
AND NOT these products are listed **OR** widely distributed among the public (pursuant to Article 116)
AND NOT the publication of the prospectus has been authorized by CONSOB
AND NOT the publication of the prospectus has been authorized by a foreign regulator **AND** the necessary authorization has been obtained for the prospectus to be recognized in Italy
THEN The offerers are conducting unauthorized solicitation of investors

Fig. 6. Key word and phrase highlighting

The knowledge elaboration in layman's language proves to be effective specification interfacing legal professionals and knowledge engineers. Another technique of extraction is rewriting the knowledge elaboration in order to transform the meaning verbalization into a format closer to lexons, as illustrated in Figure 7. Both automatic ontology extraction and terminology compilation make significant contribution at this stage. We shall examine them in Sect. 4.4 and 4.5.

	Elaboration Statements	Paraphrases
H1	[redacted] ('Organizer') has made an unlawful public offer via WWW to the Italian public to buy financial instruments (unauthorized solicitation of investors: no prospectus & no notification to CONSOB). Legal rules and postulates: articles 1.1t), u), v); 32, 94, 99; and 102 CONSOB Decree 58; article 33.1a) and b) CONSOB Regulation 11971/1999.	[redacted] *is Organizer* *Organizer make public offer* *Public offer is unlawful* *Public offer made via WWW* *Public offer made to public* *Public is Italian* *Public offer to buy financial instruments*
1.	Unlawful IF	
1.1	Organizer solicits investors on the WWW	*Organizer solicits investors* *Organizer solicits on WWW* *Investors solicited on WWW*
1.1.1	Organizer manages a website that solicits investors	*Organizer manages website* *Organizer solicits investors* *Website solicits investors*
E1.1.1.1	Website is managed AND/OR registered by organizer	*Website managed by organizer* *Website registered by organizer*
F1.1.1.1	Website states the name '[redacted]'	*Website states name* *Name is* [redacted]

Fig. 7. Paraphrasing Knowledge Elaboration

4.3.2 Abstraction

The abstraction reads the highlights or paraphrases of the extraction into lexons. The task is still confined to the semantic scope bounded by the linguistic text. The background or presumed concepts not expressed explicitly in any linguistic form in the story and knowledge elaboration are not considered during this task. Table 1 shows lexons abstracted from the extraction results in Table 1. The context, in this case, indicates the clauses in Legislative Decree 58 of CONSOB.

Table 1. Example of abstractions from the extraction results

Context	Term	Role	Role	Term
58.94.1	Offerer	Make		PublicOffering
58.94.1	PublicOffering	SubypeOf		Offering
58.94.1	Offerer	Give		AdvanceNotice
58.94.1	AdvanceNotice	Concern		PublicOffering
58.94.1	Regulator	Receive	ReceivedBy	Notice
58.94.1	Notice	Contain	ContainedBy	Prospectus

4.3.3 Organization

The extraction and abstraction adhere closely to the semantics explicitly expressed in the text. A top-down view of the work done at the stage of organization is to streamline the model and integrating the existing conceptual frameworks in other ontologies, encyclopaedias and terminologies. The issues to pay attention to are

- Semantic redundancy, due to the lack of a super semantic category
- Unsystematic semantic sub-categorization, due to the limited view of the text under consideration at each point of development

- Different terms or lexons for the identical conception
- Overall structure with respect to top-level or foundational ontology.

Table 2. Example of proposed lexons to describe implicit assumptions

Context	Term	Role	Role	Term
58.94.1	Authority	Authorise	Authoris-edBy	Activity
58.94.1	Regulator	SubtypeOf	SupertypeOf	Authority
58.94.1	Solicitation	SubtypeOf	SupertypeOf	Activity
58.94.1	Solicitation	Target	TargetedBy	Investor

The task of ontology organization introduces conceptions for more economic and consistent generalization, encodes presumed conceptions in the common sense or background knowledge, groups lexons by relevance and for the ease of communication and the need of processing, aligns and integrates the existing existing ontologies or past models.

4.4 Automatic Lexon Extraction

The automatic knowledge discovery with linguistic tools from Language & Computing NV is used at the stage of ontology extraction and abstraction in a paradigm of machine-assisted human knowledge modelling. Though linguistic tools only make surface analysis of the linguistic text, there is significant correspondence between linguistic forms and semantic contents. Besides, a considerable part of application semantics is lexical semantics with linguistic structures and tokens. The automatic ontology extraction takes two steps. First, the relevant terms in lexons are extracted from the text. Next, relations between the terms are extracted to build lexons. The result obtained so far shows that it is a helpful as pre-processing for manual ontology development: suggestion from the robot team member.

4.4.1 Term Identification

The terms of lexons are identified based on two corpora. One is the small specific set of the documents selected and created in the knowledge scoping and analysis; the other is a general linguistic corpus of multiple domain texts. Automatic concept extraction starts with text preprocessing, marking paragraphs and sentences, morphological processing and tokenization of the specific corpus. The occurrence of lexical tokens is counted. Their average document frequency and average collection frequency are calculated and compared with those in the general corpus. Heuristics are applied to the result of comparison to determine if a given token should be treated as term of lexons and scores of reliability are calculated. The same procedure is separately applied to single-word and multiword tokens.

Table 3. Terms identified and their reliability scores

Context		Single-word Term	Reliability	Multi-word Term	Reliability
[redacted]	Stories	Article	0.50765	financial instrument	0.07221
[redacted]	Stories	company	0.50403	public offer	0.06401
[redacted]	Stories	investment	0.50229	investment firm	0.06036
[redacted]	Stories	Offer	0.50209	time limit	0.05738
[redacted]	Stories	market	0.50199	asset management	0.05638

4.4.2 Lexon Extraction

The lexon extraction is done in two steps. Firstly, multi-word terms are processed in order to identify other more elementary terms in the multi-word terms and establish subtyping relations among the terms through adjectival modification. The result is a set of lexons of subtyping relationships among terms ranked according to the frequency of newly identified terms in the text corpus.

Table 4. Automatically extracted subtyping relation

Context		Multi-word Term	Role	Role	Single-word Term	Occurrence
[redacted]	Stories	financial instrument	IS_A		instrument	268
[redacted]	Stories	collective investment	IS_A		investment	60
[redacted]	Stories	administrative sanction	IS_A		sanction	45
[redacted]	Stories	promotional message	IS_A		message	28
[redacted]	Stories	financial salesman	IS_A		salesman	28

The second step is syntactic parsing. The dependency syntactic parser of English of Language & Computing, NV, is used to identify semantic relationship through syntactic structures. The most important structure to process is the transitive verbs and their subject and object. The quality of the results is less satisfactory than that of term extraction and subtype relation extraction. This is not only due to the complex and dynamic nature of the subject-verb-object sentential structure, but also to the poor quality of sentence boundary marking.

Table 5. Automatically Extracted Lexons

Context		Term	Role	Role	Term
[redacted]	Stories	website	CONTAINS		hyperlink
[redacted]	Stories	Consob	HAS		jurisdiction
[redacted]	Stories	site	HAS		content
[redacted]	Stories	offerer	APPOINTS		person
[redacted]	Stories	prospectus	CONTAINS		information

4.5 Terminography in Knowledge Engineering

The semantics in terminography traditionally deals with how lexical resources of a language are used to represent the subject matter. The terminological work in FF POIROT is unique in two respects. Firstly, the study of lexical semantics is in the

context of modeling application semantics. The terminological database needs to pay attention to objects and relations of application knowledge relevant to tasks and applications of information systems from the perspective of lexical resources of language. Secondly, the terminological work is conducted for specific purposes and users, namely, ontology for information processing and its engineers in the development process. In this context, *Termontography* [7] is proposed as a method to combine the socio-cognitive approach to terminography [14] with the theory and practice of ontology engineering. It offers different but complementary perspectives to AKEM.

4.5.1 Knowledge Scoping and Analysis for Terminography

Three knowledge deliverables in AKEM are input to the application-oriented terminological database development: knowledge resources, stories and knowledge constituent analysis. The knowledge resources serve as linguistic corpuses about the subject domain in an empirical approach to terminography. The stories and knowledge constituent analysis provides a language-independent but application-specific viewpoint on the 'units of understanding' [14]. They are guidelines for the analysis of special purpose terminological usage and selection of terminological entries.

4.5.2 Use of Terminology for Ontology Engineering

The terminological resources produced by the method of Termontography in FF POIROT are analysed and referenced in the ontology development. Due to its targeted user group (ontology engineers), the terminology puts special emphasis on documenting semantic contexts by means of textual contexts. For the same reason, the entry also includes linguistic semantic descriptions, such as agent-predicate-patient/recipient links, and cross references among items of contents. As a result, the terminology adds an additional linguistic viewpoint of the application semantics. The viewpoint counterweighs the tendency of locking into a single-perspective application-specific modeling.

The application-oriented terminology is a content platform for ontology development in its three activities. Grouped by stories and indexed to the knowledge constituent analysis, the terminological entries can be input to the extraction of ontology development or part of the extraction deliverable for abstraction. During the abstraction of ontology development, the multi-lingual terminology is referenced to discover 'semantic gaps' across languages due to social and cultural differences [7] and facilitate consensus building in a multilingual team as FF POIROT.

A third important contribution is at the organization stage of the ontology development. Since terminological work is initially scoped on the subject domain, it covers a wider scope than any given modular attempt of ontology development focused on a specific story and knowledge constituent structure. The terminological resources are valuable in identifying implicit conceptions and presumed conceptual structure.

Entry Number 1272							
EN UK	**investment firm**						
	Domain	Investment					references
	Description	any legal person the regular occupation or business of which is the provision of investment services for third parties on a professional basis.					Dir-93-22_EN.txt-69
	Co-text	Whereas an **investment firm** should not be able to invoke this Directive in order to carry out spot or forward exchange transactions other than as services connected with the provision or investment services; whereas, therefore, the use of a branch solely for such foreign-exchange transactions would constitute misuse of the machinery of this Directive;					Dir-93-22_EN.txt-23
	Relation	investment firm	[agent]	provided	investment service	[patient]	Dir-93-22_EN.txt-50
		investment firm	[agent]	has	registered office	[patient]	Dir-93-22_EN.txt-137
		investment firm	[agent]	establishes	branch	[patient]	Dir-93-22_EN.txt-195
IT	**impresa di investimento**						
	Domain	Investment					
	Description	i soggetti (comprese le persone fisiche) che offrono consulenza in materia di investimenti come loro attività principale/esclusiva, dovranno ottenere l'autorizzazione ad operare in qualità di "**imprese di investimento**" ai sensi della dsi, in sostituzione dei regimi nazionali specifici a cui sono assoggettati attualmente					pro_0625_IT.txt-388
		qualsiasi persona giuridica la cui occupazione o attività abituale consiste nel prestare servizi di investimento a titolo professionale					pro_0625_IT.txt-658

Fig. 8. An Example of an Entry in FF POIROT Multilingual Terminology

5 Conclusion

This paper is a summary description of the effort to build ontology of fraud evidence from laws and regulations by a geographically distributed multi-disciplinary team. The ontology development is based on the DOGMA approach and put in the context of an application knowledge engineering process. It emphasizes the knowledge scoping and analysis. It introduces mechanism through which the footprint of development decision making is explicitly traceable. As legislature-based forensic evidence is only a part of the ontology of fraud forensics, the ontology development will look into 'red flags' and fraud plans. The same text-based engineering methodology will be followed. The ontology-based knowledge will be deployed in terms of commitments in application systems.

Acknowledgement. This work described is done in the FF POIROT project (www.ffpoirot.org), funded by the EC under the fifth framework IST programme,

IST 2001-38248. The authors wish to thank their colleagues from CONSOB for their indispensable and inspiring support.

References

1. Aussenac-Gilles, N., Bielow, B., Szulman, S.: Corpus Analysis for Conceptual Modelling. EKAW'00 Workshop on Ontologies and Texts, CEUR Workshop Proceedings 51 (2000).
2. Deray, T.,Verheyden, P.: Towards a Semantic Integration of Medical Relational Databases by Using Ontologies: a Case Study. On the Move to Meaningful Internet Systems 2003: OTM 2003 Workshops, Lecture Notes in Computer Science, Vol. 2889/2003. Springer-Verlag, Heidelberg, (2003) 137-150.
3. Gruber, T.: Towards Principles A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2) (1993) 199 – 220
4. Grüingger, M., Fox, M.S.: Methodology for the design and evaluation of ontologies. IJCAI 1995 Workshop on basic ontological issues in knowledge sharing. Montreal (1995).
5. Jacobson, I., Booch, G., Rumbaugh, J.: The Unified Software Development Process. Boston, Addison-Wesley, (1999).
6. Jarrar, M., Demey, J., Meersman, R.: On Reusing Conceptual Data Modeling for Ontology Engineering. Journal on Data Semantics, 1(1) (2003) 185 – 207
7. Kerremans, K., Temmerman, R., Tummers, J.: Representing Multilingual and Culture-Specific Knowledge in a VAT Regulatory Ontology: Support from the Termontography Approach. In: Robert Meersman & Zahir Tari (eds.) OTM 2003 Workshops. Tübingen: Springer Verlag (2003)
8. Kingston, J., Vandenberghe, W.: A Comparison of A Regulatory Ontology with Existing Legal Ontology Frameworks In: Robert Meersman & Zahir Tari (eds.) OTM 2003 Workshops. Tübingen: Springer Verlag (2003)
9. Kroll, P., Kruchten, P.: The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP, Boston, Addison-Wesley (2003).
10. Kruchten, P.: The Rational Unified Process: An Introduction, Boston, Addison Wesley (2000).
11. Legislative Decree 58 of 24 February 1998, Consolidated Law on Financial Intermediation Pursuant to Articles of Law 52 of 6 February 1996, as Amended by Legislative Decree 61 of 11 April 2002, http://www.consob.it/produzione/english/Regulations/fr_decree58.htm.
12. Meersman, R.: Reusing Certain Database Design Principles, Methods and Techniques for Ontology Theory, Construction and Methodology. STARLab Technical Report 01 (2000)
13. Nobecourt, J.: A Method to Build Formal Ontologies from Texts, Corpus Analysis for Conceptual Modelling, EKAW'00 Workshop on Ontologies and Texts. CEUR Workshop Proceedings 51 (2000)
14. Temmerman, R.: Towards New Ways of Terminology Description - the Sociocognitive Approach. Amsterdam: John Benjamins (2000)
15. Wigmore, J.H.: The Science of Judicial Proof as given by Logic, Psychology and General Experience. Boston, Little Brown (1937)
16. Zhao, G.: Application Semiotics Engineering Process. International Conference of Software Engineering and Knowledge Engineering (2004) 354–359
17. Zhao, G., Gao, Y., Meersman, R.: An Ontology-based Approach to Business Modelling. In: Proceedings of the International Conference of Knowledge Engineering and Decision Support (2004) 213–221

Extracting Legal Propositions from Appellate Decisions with Text Discourse Analysis Methods*

Woojin Paik[1] and Jee Yeon Lee[2]

[1] Dept. of Computer Science, Konkuk University,
322 Danwol-Dong, Chungju-Si, Chungcheongbuk-Do, 380-701, Korea
wjpaik@kku.ac.kr
[2] Dept. of Library and Information Science, Yonsei University,
134 Sinchon-Dong, Seodaemun-Gu, Seoul, 120-749, Korea
jlee01@yonsei.ac.kr

Abstract. Appellate decisions are the most important judicial documents in the Anglo-American legal system. Typically, judges write appellate opinions by including a summary of the facts of the case, identification of the issues of law raised in arguments by counsel for each of the parties, pronouncement of the legal propositions supported by the controlling authorities, and declaration of a decision that resolves the issues by applying the legal propositions to the facts of the case. The cited legal propositions are often concise summaries of certain aspects of previous cases or federal or state codes, which are applicable to the particular case in consideration. In this paper, we describe how a text discourse analysis program can be used to categorize each sentence in the appellate decisions as one or more of the discourse categories such as 'facts', 'issues', 'legal propositions', and/or 'decisions'. We also show how an information extraction program is applied to the sentences belonging to the 'legal proposition' category to build a visually browsable legal knowledge base. We expect the browsable knowledge base to aid both counsels and judges finding supporting legal propositions for their arguments during the early stage of preparing court briefs or the appellate decisions.

1 Motivation: Knowledge Extraction from Appellate Decisions

An appellate decision is a complex document, which embodies facts of the case in consideration, arguments by the counsels, controlling authorities' legal propositions supporting the arguments and/or the decisions, and finally the judges' decisions [1]. One of the primary goals of the previously developed legal information extraction systems was to convert legal cases to casebase entries for a Case Based Reasoning (CBR) system [2]. One such CBR system modeled the case-based argumentation according to twenty-six prototypical fact situations, which could have been favored by either plaintiff or defendant [3]. For this, an information extraction system needed

* This work was supported by the faculty research fund of Konkuk University in 2004.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 621–633, 2004.
© Springer-Verlag Berlin Heidelberg 2004

to scan the sentences in appellate decision, which match the pre-constructed case frame patterns, to extract relevant information such as the facts of the case and the arguments by the counsels. Another type of information extraction system targeted 'treatment history' to support a computer-assisted legal database update tool [4]. Treatment history describes one court's opinion regarding another case cited in the form of assertions and comments in the final decision section. In this context, the information extraction system found the cited cases to alert the human reviewers to determine the correctness of the found cases belonging to the treatment history. If the cases were indeed from the treatment history then the citatory database was updated. A list of keywords was used to find the treatment history sentences.

These information extraction systems targeted all parts of the appellate decision except the legal propositions supported by the controlling authorities. Facts of the case and arguments by the counsels were the information sources for the CBR systems. Treatment history, a part of the final decision, was used for the legal database update tool. On the contrary, our ongoing research aims to initially populate a case and code summary database with the information retrieved from the legal propositions supported by the controlling authorities. A judge applies the legal propositions to the facts to resolve the issues of the case in consideration and thus arrive at the final decisions. This means that the judges cite supporting previous cases to establish the legitimacy of their decisions. The legal propositions are essentially the summaries of the previous cases or federal codes or state codes. Our approach targets these carefully constructed summaries to populate the case database.

However, there can be one major drawback to our approach. The summaries written as the legal propositions belong to a type of summarization referred as purpose-oriented summarization. Purpose-oriented summarization does not address all aspects of the original document. It is a summarization created to meet a particular query or goal [5]. Similarly, the legal propositions summarize the previous cases only in light of a particular issue raised in the case in consideration. Therefore, users of the legal case summary database will not be able to grasp the full scope of a case if the case summary is from only one legal proposition. We believe this shortcoming will gradually diminish as our system accumulates more information retrieved from other citations referring to the same case. Our optimism is based on the observation that different citations often cover distinctive aspects of the same case and thus a collection of purpose-oriented summaries will eventually cover all aspects of one particular case.

The second part of our research effort is to convert the retrieved case summaries into the knowledge representation appropriate for a visual browsing. Our system converts the extracted information into Concept-Relation-Concept (CRC) triples, a simple type of knowledge representation. The CRC triples consist of a subject concept, a descriptive concept that describes the subject concept, and a relation that describes the relation between the subject and descriptive concepts. For instance, the subject concept may be the cause of an action that occurs, or it may be the recipient of an action or event. These are different relations that distinguish the connection between the same two concepts. The CRC triples are extracted from text as follows: 1) Identify boundaries between concepts and relation-revealing phrases using predetermined evidence sources such as punctuation marks, prepositional phrases or other

indicator words and linguistic structures; 2) Identify subject proper name and substitute full reference name, if necessary; 3) Identify descriptive information concepts; and 4) Identify relations using a rule-based detection and extraction module [6].

A visual knowledge base browser was designed to provide visual navigation capabilities to users so that they can search for information by moving from one concept to another via semantic relations, which link these concepts. The internal representation of the extracted information is linked CRC triples and the accumulation of the CRC triples forms a semantic network. Essentially, a collection of all documents in one database results in a semantic network. Thus, a visual knowledge base browser can be considered a tool for the users to review the information from many documents as one chunk without being restricted by document boundaries. The visual knowledge base browser also allows users to examine the extracted information in the most direct manner, and is expected to be useful to a person who is exploring topical interests without explicitly knowing what he/she is looking for.

In the legal process context, we expect a counsel to work without having specific queries to look for the previous cases, which might support or refute his/her arguments during the early stage of a court brief preparation. Thus, we decided to provide a search system, which is suitable for visually exploring a large number of interconnected cases by starting from a single concept.

2 Classifying Sub-texts in Appellate Decisions According to a Legal Text Discourse Model

A text discourse model specifies the necessary classes of knowledge to be identified in order to develop the skeletal conceptual structure for a class of entities. Based on a discourse linguistics observation [7], writers are often influenced by the schema of a particular text type if they repeatedly produce texts of that type. This means that the writers consider both specific content they wish to convey and usual structure for that type of text based on the purpose it is intended to serve.

The existence of such predictable structures in texts is consistent with findings in cognitive psychology which suggest that human cognitive processes are facilitated by the ability to 'chunk' the vast amount of information encountered in daily life into larger units of organized data [8]. Based on schema theories, humans recode individual units of perception into increasingly larger units, which will eventually reach at the level of a schema. It has also been argued that humans possess schema for a wide range of concepts, events, and situations [9]. In discourse linguistics, this schema theory was extended to suggest that schema exist for text-types that participate regularly in the shared communication of a particular community of users.

As the text structure of a particular text type is discovered, the text's discernible and predictable superstructure is also revealed. Superstructure is defined as the text-level syntactic organization of semantic content. It can be also referred to as the global schematic structure or the recognizable template that is filled with different meaning in each particular example of that text type [10]. Some of the text types for

which schemas or models have been developed with varying degree of details are: newspaper articles [10], arguments [11], editorials [12], and abstracts [13].

Our previous work focused on developing a news schema model [14]. We started from the journalistic, hierarchical newspaper text model proposed by van Dijk [10]. By using a sample of Wall Street Journal articles from 1987 to 1999, a revised news schema was developed. The revised schema retained segmentation of the overall structure into van Dijk's higher level categories, namely, summary, story, and comment, but added several components as warranted by the data. The components were: circumstances, consequence, credential, definition, error, evaluation, expectation, history, lead, main event, no comment, previous event, reference, and verbal reaction.

For the legal documents, we decided to develop a legal text schema by starting from four basic components [1]. The components are: 1) summary of the facts of the case; 2) identification of the issues of law raised in arguments by counsel for each of the parties; 3) pronouncement of the legal propositions supported by the controlling authorities; and 4) declaration of a decision that resolves the issues by applying the legal propositions to the facts of the case.

The appellate decisions, that we analyzed to develop a legal text schema, were taken from the LexisNexis Academic database. We used the top fifteen cases as the training set by searching the topic of 'patent infringement' while restricting our search to corporate law (federal and state cases) within the last six months. We also used the documents, which were ranked from 16th to 30th as the testing set.

Many cases included heading information such as 'disposition', 'counsel', 'judges', 'opinion by', 'facts', 'background', 'applicable law', 'standard', 'opinion', 'discussion', and 'conclusion'. Although these headings look useful at the beginning, there were no consistent headings across the fifteen training set. In fact one case did not have any heading at all. We decided to code each sentence according to the four basic components. We mainly found the legal propositions in the discussion section of the appellate decisions. The following shows three legal propositions cited in an appellate decision. Each proposition is followed by the citation information describing the source case.

Example 1. Personal jurisdiction in patent cases is decided under Federal Circuit law rather than the law of the regional circuit in which the case arose. Akro Corp. v. Luker, 45 F.3d 1541, 1543 (Fed Cir. 1995).

Example 2. The plaintiff bears the burden of establishing that the defendant had the requisite contacts with the forum. Inamed Corp. v. Kuzmak, 249 F,3d 1356 1360 (Fed Cir. 2001).

Example 3. When the district court does not hold an evidentiary hearing regarding personal jurisdiction, and the matter is submitted to the court based solely on affidavits and written materials, plaintiff must make only a prima facie showing. Deprenyl Animal Health, Inc. v. University of Toronto Innovations Foundation, 297 F.3d 1343, 1347 (Fed Cir. 2002) [15].

Conrad and Dabney [16] developed an elaborated judicial opinion structure model. However, we decided not to use an elaborated appellate decision text schema after reviewing the training set, because our primary goal was to identify the legal propositions and retrieve them for further processing. We limited our processing to classifying each sentence into one of the basic components outlined previously, namely, facts, issues, legal propositions, and decisions. Our 'legal propositions' component corresponds to 'binding authority' component in the Conrad and Dabney's model [16].

2.1 Extracting Text Classification Features

While coding the training data, we developed both defining features and properties for each component. The defining features convey the role and purpose of that component within the legal text schema while the properties provide suggestive clues for the recognition of that component in an appellate decision. The manual coding suggested to us that we were relying on five types of linguistic information during our coding. The data, which would provide these evidence sources, were then analyzed statistically and translated into computationally recognizable text characteristics. The five sources of evidences are described in the following.

Lexical Evidences: This source of evidence is a set of one and two word phrases for each component. The set of lexical evidences for each component was chosen based on observed frequencies and distributions. Not all lexical evidences were used to calculate the statistical likelihood of a sentence, which includes a particular lexical evidence, belonging to a particular component. Only the words or phrases with sufficient occurrences and statistically skewed observed frequency of occurrences in a particular component were used.

Before all one and two word phrases were extracted from the text, all words were converted into their root form. For example, plural noun forms were turned into the singular noun forms. Past tense and past participle forms of the verbs were also converted into the present forms. This morphological analysis process reduced the number of unique word/phrase forms significantly. Furthermore, all proper names were converted into their type. For example, there are three unique proper names in the following example sentence, which belong to the 'fact' component. Preprocessing refers to both the morphological analysis and the proper name interpretation.

Example – Before Preprocessing. 'On April 25, 2003, Monsanto filed suit against Roman seeking damages for infringement of its patented product and a permanent injunction enjoining Roman from future unauthorized use of its patented technology [17].'

We used an improved version of a proper name interpreter described in [18] to categorize 'April 25, 2003' as a date type proper name, 'Monsanto' as a company name, and finally 'Roman' as a person's name. The proper names are categorized according

to a five-level named entity ontology. Some of the top-level entries include, 'People', 'Thought', 'Communication & Communication Channels', 'Building & Structures', 'Substances', 'Materials', and 'Objects'. The proper names are categorized at the lowest possible level.

After the morphological analysis and the proper name interpretation are completed, the example sentence is transformed as shown in the following.

Example – After Preprocessing. 'on date , company file suit against person seek damage for infringement of it patent product and a permanent injunction enjoin person from future unauthorize use of it patent technology .'

After all coded training appellate decisions are processed by the lexical evidence extraction module, the frequency distribution of each piece of lexical evidence is further processed to generate the probability information. For example, **'company*** file' occurred five times in the sentence coded as 'fact', four times in the 'issue' sentences, two times in 'legal proposition', and once in 'decision' in the training data set. This tells us that the sentence should be coded as 'fact' five out of twelve times if the sentence includes **'company*** file'** as an example of two-word lexical evidences. In the same manner, **'company*** file'** is indicated four out of twelve times as 'issue', two out of twelve times as 'legal proposition', and one out of twelve times as 'decision'. After all one and two-word lexical evidence based on probability calculation is completed, a series of normalization processes is applied to weed out unreliable lexical evidences and also to remove other influences such as likelihood of component occurring. As each sentence is processed, the lexical evidences for the words and phrases in the sentence are combined using the Dempster-Shafer Theory of Evidence [19]. At the end of all processing, each sentence is assigned with four numbers ranging from zero to one. One number represents the sentence's probability of having been correctly coded as 'fact'. Other numbers are the probability figures for 'issue', 'legal proposition', and 'decision' components.

Syntactic Evidences: We utilize two types of syntactic evidences: 1) typical sentence length as measured in the average number of words per sentence for each component and 2) individual part-of-speech distribution based on the output of the part-of-speech tagging of each appellate decision using Machinese Syntax [20]. This evidence helps to recognize those components, which tend to have a disproportionate number of their words be of a particular part of speech. For example, 'legal proposition' component sentences tend to have no proper names at all except for case citation and court names.

Tense Evidences: Some components tend to contain verbs of a particular tense more than verbs of other tenses. For example, 'fact' sentences are almost always in the past or present perfect tense. The tense evidence is a byproduct of part-of-speech tagging. We utilize the fine grained verb types of Machinese Syntax [20] processing output.

Document Structure Evidences: There are two types of document structure evidences. Firstly, some appellate decisions have explicit section headings, which are closely aligned with four basic components. For example, first level heading of a few appellate decisions included 'Facts', 'Applicable Laws', 'Discussion' and 'Conclusion'. Not all sentences under the 'Facts' heading were 'facts'. Similarly, not all sentences under the 'Conclusion' heading were 'decisions'. However, there were significant correlation between the first level headings such as these and the basic components. Thus, we utilized the heading information as another evidence source. Secondly, we included the relative position of each sentence with respect to the source paragraph as another evidence source.

Order of Component Evidences: This source of evidence replies on the tendency of components to occur in a particular, relative order. We calculated the frequency with which each component followed every other component and the frequency with which each component preceded every other component. The results are stored in two four-by-four matrices. This evidence source is not used initially. At first, the text classifier uses other evidence sources to assign one or more component tags to each sentence in an appellate decision. If there is a sentence, which did not receive any component assignment by the text classifier then this order of component evidence is used to determine the most appropriate component to assign.

2.2 Text Classification

To assign a basic component label to each sentence, each sentence in the training data set is categorized according to the predetermined legal text schema. The first text classification task involves manually coding all sentences in a set of training documents in preparation for feeding the automatic system. Each sentence is classified as "in" or "out" of the individual component classes as outlined by the class definitions. The next step is to take these manually classified sentences and process them through the trainable text classification system. During this process, it builds a vector of lexical evidences, syntactic evidences, tense evidences, and document structure evidences. Multi-level Natural Language Processing outputs are the basis for these textual data feature representations.

This collection of automatically generated features is then used to determine inclusion of a sentence within a particular component class. The system determines the "certainty of membership" for each of the sentences compared to each of the classes. If we consider a range of one to zero where one refers to a sentence that is definitely a member of a certain component class, and zero means a sentence is definitely a non-member of a certain class, we can say that values of zero and one both have a 'certainty of membership' value of one. For either of these cases, we can confidently conclude that the sentence either 'does' or 'does not' belong within a given class. If we look at values close to .5 on the above scale, we have a 'certainty of membership' value close to zero. For these cases, we cannot automatically determine whether or not a given sentence should be assigned to a given class. These sentences are consid-

ered valuable in refining the classification system. By manually classifying these sentences, and then feeding them back into the automatic system, we train it to recognize the subtle differences that distinguish how these sentences should be classified.

3 Populating Appellate Decision Knowledge Base

We describe an information extraction system that allows users to visually browse summaries of legal cases. The system is designed to be domain-independent, and automatically builds its own subject knowledge base. It can be applied to any new corpus of text with quick results, and without lengthy manual input. For this reason, it is also a dynamic system that can acquire new knowledge and add to the knowledge base immediately by automatically identifying new names, events, or concepts. In short, a set of legal propositions is subjected to operations that extract concept-relation-concept triples (CRCs), which are stored in a data organization (such as a database) for browsing purposes. The CRCs are converted to a knowledge representation (KR) prior to indexing and storage.

3.1 Extracting Concept-Relation Triples

The system extracts information from text about any concept and its relations to any other concepts within that text. Using information-rich linguistic constructions in close proximity to a concept, it extracts information from a database and organizes that information. If desired, the information can be organized chronologically, independent of the artificial divisions of separate legal propositions, to create a merged chronological profile. The system recognizes concepts automatically, and it identifies the sources of its information so that separate facts can be traced to their origins. Furthermore, it records and reports relations of any concept in a database with other concepts. The system is modular, so that it can be adapted to various subject domains, knowledge representation schemes, or text types. While this specific implementation describes its use with legal sources, other sources such as medical literature or newspaper articles are possible applications.

Relations define how concepts are connected to each other. They define what role a concept plays in a proposition or sentence.

Example. The Federal Circuit applies a three-part test to make the due process determination …

In the example, ‘agent’ relation exist between ‘Federal Circuit’, a government organization type proper named concept, and ‘applies’, a verbal concept’.

Most of the relations are dyadic relations. That is, they connect two concepts to form a CRC triple. A relatively small number of the relations such as ‘necessary’, ‘negative’, ‘past’, and ‘possibility, are monadic relations. In other words, they are associated with a single concept to form a relation-concept (RC) pair. The relations

are mostly adopted from the case relations, which were described as a part of Conceptual Dependency Theory [21], and the set of relations, which were described in the Conceptual Graph Theory [22]. Our system extracts thirty-seven relations. Some of them are: 'accompany', 'affiliation', 'agent', 'argument', 'association', and 'cause'.

Some of the algorithms used to extract CRC triples are explained in the following.

Co-referential concept extraction algorithm is a method for identifying relations between subject concepts and constituents in appositional phrases. The same approach is applicable to extracting information in copula sentences. If an apposition belongs to the apposition proper category, then there is at least one noun phrase in the apposition that refers to the same entity to which the subject concept, which precedes or follows the apposition. The "ISA" or "class" relation is assigned between the co-referential noun phrase in the apposition and the subject concept.

Relation Revealing Formula is a method for identifying relations between the main concept and the constituents of the relative clause. It is a sublanguage approach to analyzing texts [23]. Sublanguage theory suggests that any type of text that is used for a common purpose within a group of individuals will develop characteristic syntax and semantics. A set of relative clauses, which modify one type of proper name, is assumed to constitute one sublanguage. Thus, for each sublanguage, it is possible to construct specific relation extraction rules based on: 1) typical attributes of a particular proper name category and 2) case frames, which are usually associated with the matrix verbs of sentences.

Special semantic relation algorithm is used to extract semantic relations, looking for specialized types of concepts and linguistic clues, including some prepositions, punctuation, or specialized phrases. The development of this algorithm was motivated by failure analysis of an earlier implementation of CRC triple extraction [6]. There were two types of problems. One was inaccuracy of the extraction, which affected precision, and the other was low recall due to limited coverage of the extraction rules. This algorithm helped to remedy the low recall problem by focusing on the identification of various linguistic patterns, customarily used to express particular semantic relations such as location.

Syntactic relation to semantic relation mapping algorithm maps syntactic relations such as "subject of the transitive verb" to their semantic functional equivalents so that a subject of a verb might be described as "agent of the action" of a verb. The notion of mapping syntactic relations to semantic relations falls on more traditional ways to analyze texts for information extraction. This kind of approach has been mainly used by early information extraction systems, which were based on traditional full-scale syntactic parsers [24].

3.2 Visual Browsing

The information visualization system is designed to provide browsable information about the legal propositions in a database. Because browsing is characterized by searching without specifying, and because the systems may require precise definitions or terminology, traditional computerized systems are not totally compatible with

browsing techniques. Additionally, computer interactions often provide insufficient dialog for a casual browser. Visual browsing does not require a set of legal propositions retrieved with a specific query. In addition, our system allows the semantic content of the legal propositions to be extracted and visualized for browsing while the traditional visualization systems mainly rely on simple word co-occurrence information. Since one of the most common premises of browsing is an ill-defined search objective, visualization of the whole database based on semantic content can be considered a "better fit" with the original spirit of browsing compared to traditional approaches. The browsing interface permits exploration of the contents of the system through a dynamic graphical hyperlinked display of a requested term, plus all the related concepts that are linked to it. We have implemented the visual browsing system by utilizing Graphviz, an open source graph drawing program [25].

4 Implementation and Evaluation

The computational modeling of instantiating a discourse-level model of the legal texts and extracting information from the legal propositions to enable a visually browsable knowledge base are ongoing efforts. We developed a prototype system by manually analyzing fifteen sample appellate decisions and tested our system using fifteen unseen appellate decisions. The first run and evaluation of the correctly categorizing four basic components resulted in 70% of the sentences being correctly identified.

There is no directly comparable legal text classification system. However, a news text classification system, which assigned sentences into one out of fourteen categories, performed at 72% correct rate in the fully automatic mode and 80% correct rate with various manual heuristic adjustments [14]. It should be noted that our text classifier did not utilize the second iteration of incorporating the sentences, with certainty membership value close to zero, as a part of new training data set. We believe the addition of this process will improve the correctness of our system.

Two methods of measuring effectiveness that are widely used in the information extraction research community have been selected to evaluate the information extraction performance [26]. The methods are:

Precision: the percentage of actual answers given that are correct.
Recall: the percentage of possible answers that are correctly extracted

Automatically extracted information was evaluated with the following criteria: 1) If the automatically extracted information and the answer key, which is generated manually, are deemed to be equivalent, then the automatic extraction output is considered as "correct" and 2) If the automatically extracted information and the answer key do not match then it is considered as "incorrect."

We evaluated information extraction effectiveness based on forty-two sentences from five test appellate decisions, which were correctly classified as legal propositions by the text classifier. Not all test decisions were used to evaluate the information extraction performance as not all answer keys were manually generated. The preci-

sion of the CRC extraction is 72% by dividing the number of correctly extracted CRCs (291) by the total number of all extracted CRCs (403). The recall value is 60% calculated by dividing the number of correctly extracted CRCs (291) by the total # of all possible CRCs (486).

Bruninghaus and Ashley [2] reported a legal information extraction system, which extracted trade secret-related noun phrases and product names with the precision and recall figures at 66% and 65% respectively. Our system performed better in terms of the precision but was less successful with the recall. One of the possible reasons for the lower recall value of our system is due to the significantly more complex nature of the target information to extract.

5 Summary

Although we are clearly in the early stages of developing a legal text discourse modeling system and a legal information extraction system, we find these results quite promising and eager to share our premature but empirical results and experiences in creating an operational system with other researchers.

We expect the browsable knowledge base to aid both counsels and judges finding supporting legal propositions for their arguments during the early stage of preparing court briefs or the appellate decisions. We also believe that the CRC extraction from the case neutral legal propositions could lead to discoveries of essential concepts and their relations amongst each other. The extraction process should assist in building a generally applicable legal ontology.

There are many tasks that we have yet to finish. The major missing task is the evaluation of the visually browsable legal proposition knowledge base. We also need to increase the size of test data set to improve the reliability of the evaluation results.

We have applied the legal text discourse model to appellate decisions by coding a sample of decisions. This effort in conjunction with our previous work with the newspaper texts shows that we can extract a particular section of the texts by utilizing a text type specific discourse model.

References

1. Branting, K., Lester, J.C., and Callaway, C.B.: Automating Judicial Document Drafting: A Discourse-Based Approach. Artificial Intelligence Law 6(2-4): 111-149 (1998)
2. Bruninghaus, S., Ashley, K.D.: Improving the Representation of Legal Case Texts with Information Extraction Methods. In Henry Prakken and Ron Loui (Editors) Proceedings of Seventh International Conference on Artificial Intelligence and Law (ICAIL-2001). St. Louis, MO. ACM Press: New York, NY (2001)
3. Aleven, V. Teaching Case-Based Argumentation through a Model and Examples. Ph.D Dissertation, University of Pittsburgh (1997)

4. Jackson, P., Al-Kofahi, K., Kreilick, C. & Grom, B.: Information extraction from case law and retrieval of prior cases by partial parsing and query generation. Proceedings of Seventh International Conference on Information & Knowledge Management (CIKM-98). New York, ACM Press: 60-67 (1998)
5. Fukusima, T. and Okumura, M.: Text Summarization Challenge: Text Summarization Evaluation at NTCIR Workshop2. Proceedings of the Second NTCIR Workshop on Research in Chinese & Japanese Text Retrieval and Text Summarization. Tokyo, Japan, National Institute of Informatics (2001)
6. Paik, W.: CHronological information Extraction SyStem (CHESS). PhD Dissertation, Syracuse University, Syracuse, NY. (2000)
7. Jones, L.B.: Pragmatic aspects of English text structure. Summer Institute of Linguistics, Arlington, TX (1983).
8. Rumelhart, D.: Understanding and summarizing brief stories. In D. LaBerge and S.J. Samuels (Editors) Basic processes in reading: Perception and comprehension. Hillsdale, NJ: Lawrence Earlbaum Associates: 265-303. (1977)
9. Rumelhart, D.: Schemata: the building blocks of cognition. In R. Spiro, B. Bruce, & W. Brewer (Editors) Theoretical issues in reading comprehension: Perspectives from cognitive psychology, linguistics, artificial intelligence and education. Hillsdale, NJ: Lawrence Earlbaum Associates: 33-8. (1980)
10. van Dijk, T.A.: News analysis: Case studies of international and national news in the press. Hillsdale, NJ: Lawrence Earlbaum Associates. (1988)
11. Cohen, R.: Analyzing the structure of argumentative discourse. Computational Linguistics, 13. 11-24. (1987)
12. Alvarado, S.J.: Understanding editorial text: A computer model of argument comprehension. Boston, MA: Kluwer Academic Publishers. (1990)
13. Liddy, E.D.: The discourse-level structure of empirical abstracts: An exploratory study. Information Processing and Management 27.1, Tarry Town, NY, Pergamon Press: 55-81. (1991)
14. Liddy, E.D., McVearry, K.A., Paik, W., Yu, E., and McKenna, M.: Development, Implementation and Testing of a Discourse Model for Newspaper Texts. Proceedings of a Human Language Technology Workshop. Plainsboro, NJ, Morgan Kaufmann Publishers: 159-164. (1993)
15. Jamison v. Olin Corp.-Winchester Division; U.S. Repeating Arms Co., Inc.; Browning; Browning Arms Co.; and G.I. Joe's Inc., Case No. 03-1036-KI, U.S. District Court for the District of Oregon. (2004)
16. Conrad, J.G. and Dabney, D.P.: A cognitive approach to judicial opinion structure: applying domain expertise to component analysis. The 8th International Conference on Artificial Intelligence and Law. Proceedings of the Conference. New York, NY, ACM: 1-11. (2001)
17. Monsanto Company Plaintiff, v. Mike Roman, a/k/a M. Roman, Defendant, Civil Action No. 1:03-CV-068-C, U.S. District Court for the Northern District of Texas, Abilene Division (2004).
18. Paik, W., Liddy, E.D., Yu, E., and McKenna, M.: Interpretation of Proper Nouns For Information Retrieval. Proceedings of a Human Language Technology Workshop. Plainsboro, NJ, Morgan Kaufmann Publishers: 309-313. (1993)
19. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference, Morgan Kaufmann Publishers, San Francisco, CA. (1988)
20. "Machinese Syntax." Connexor Oy. 1 July, 2004 http://www.connexor.com/m_syntax.html

21. Schank, R.: Conceptual Dependency: A Theory of Natural Language Understanding, Cognitive Psychology, 3 (4), (1972)
22. Sowa, J.: Conceptual Structures: Information Processing in Mind and Machine, Reading, MA: Addison-Wesley. (1984)
23. Sager, N., Friedman, C., & Lyman, M.S.: Medical Language Processing: Computer Management of Narrative Data, Reading, MA: Addison-Wesley. (1987)
24. Proceedings of the Third Message Understanding Conference (MUC-3), San Diego, CA, Morgan Kaufmann. (1991)
25. "Graphviz." AT&T Labs Research. 5 July, 2004 http://www.research.att.com/sw/tools/graphviz/
26. Chincor, N.: MUC-4 Evaluation Metrics, Proceedings of the Fourth Message Understanding Conference (MUC-4), McLean, VA. (1992)

Cases and Dialectical Arguments – An Approach to Case-Based Reasoning

Bram Roth[1] and Bart Verheij[2]

[1] Universiteit Maastricht,
BramRoth@cs.com
[2]Rijksuniversiteit Groningen

Abstract. Case-based reasoning in the law is a reasoning strategy in which legal conclusions are supported by decisions made by judges. If the case at hand is analogous to a settled case, then by judicial authority one can argue that the settled case should be followed. Case-based reasoning is a topic where ontology meets logic since one's conception of cases determines one's conception of reasoning with cases. In the paper, it is shown how reasoning with cases can be modelled by comparing the corresponding dialectical arguments. A unique characteristic thereby is the explicit recognition that it is in principle contingent which case features are relevant for case comparison. This contigency gives rise to some typical reasoning patterns. The present work is compared to other existing approaches to reasoning by case comparison, and some work on legal ontologies is briefly discussed regarding the role attributed to cases.

1 Introduction

Case-based reasoning is a common type of argumentation in the law, in which legal conclusions are supported with decided cases. If some decided case is sufficiently similar to the case at hand, then under the doctrine of *stare decisis* one should not depart from that decision, and the same conclusion should hold.

The principle of stare decisis holds that in settling new cases judges tend to adhere to prior decisions, in the sense that once a question of law is decided, the decision is normally not departed from afterwards. There are a number of reasons to adhere to earlier decisions. First, courts are often formally bound by decisions made by courts higher up in the hierarchy and by their own decisions (cf. Cross 1977, pp. 103f.). Second, a departure from settled cases would make judicial decisions uncertain and unpredictable, so that one could never be sure about the status of one's legal claims in court. Third, it is fair and just to treat all individuals equally under the law, and to make no difference between people in similar cases. Fourth, adhering to decisions turns out to be an economic and efficient practice (Bankowski 1997, p. 490).

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 634–651, 2004.
© Springer-Verlag Berlin Heidelberg 2004

Different methods to adhere to decisions have been described in the literature (Llewellyn 1960, pp. 77-84), and two of them will be discussed next: *rule extraction* and *case comparison.*

In the first method the reasons underlying the conclusion are isolated, the reasons and conclusion are generalised (or 'universalised', cf. Hage 1997, p. 47; MacCormick 1987, pp. 162f.) into a rule that can explain the outcome. This mechanism is called the 'rule extraction method' for short.

The second method of adhering to a decision is by assigning the corresponding legal conclusion to the new case as an authoritative example (cf. Ashley 1990, pp. 11-12; Aleven 1997, pp. 58; Oliphant 1928, pp. 144f.). If the decided case is 'sufficiently similar' to the case at hand, then one can argue by analogy and judicial authority that the decided case should be followed and that the same decision should be taken once more.

The methods of rule extraction and case comparison can be presented schematically as a list of reasoning steps, as in Fig. 1. The figure illustrates that there are strong relations between both methods, in the sense that they involve comparable reasoning steps.

Rule extraction method		**Case comparison method**	
(1) Extracting rules from decided cases		(1) Selecting relevant case facts	
(2) Showing that rule conditions are satisfied		(2) Establishing an analogy between cases	
(3a) Applying extracted rules to the case at hand	(3b) Pointing out exceptions to extracted rules	(3a) Following decided cases in the case at hand	(3b) Distinguishing decided cases from the case at hand

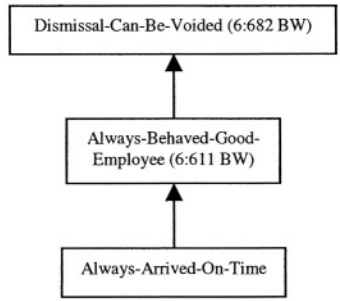
Fig. 1. The methods of rule extraction and case comparison

Our discussion of the two methods to adhere to decisions show that case-based reasoning is a topic where ontology meets logic. The two methods correspond to different ontological conceptions of cases that determine different logical analyses of reasoning. When cases are considered as authoritative sources of rules (as in the rule extraction method), reasoning with cases only differs from rule-based reasoning in the phase of rule extraction. After that phase, the extracted rules are applied, just like other rules. From an ontological point of view, the rule extraction method treats cases basically as sets of rules. In the method of case comparison, cases are considered differently, namely as authoritative sources of arguments and decisions. Cases are treated as wholes during all phases of reasoning. Ontologically, the case comparison method views cases basically as sets of arguments and decisions.

In the present paper, we have chosen the case comparison method as the basis for modelling case-based reasoning. The model is informally presented. Roth (2003) gives further details and a formal elaboration.

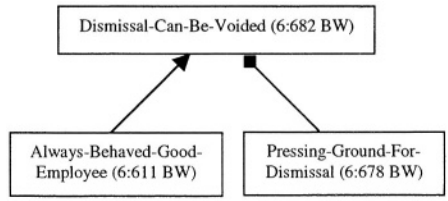
2 Dialectical Arguments

To arrive at a systematic analysis of cases, it is convenient to have a graphical representation of the argumentation in the cases. To this end tree-like structures are introduced, called *dialectical arguments;* cf. Verheij's (2000; 2001; 2003) naive dialectical arguments and Loui's (1997) and Loui and Norman's (1995, p. 164) records of disputation. Dialectical arguments consist of statements that support or attack other statements, the support and attack relations being represented by arrows. Here is a simplified example for the domain of Dutch dismissal law.



At the top of this figure one finds the legal conclusion that the dismissal can be voided, to which a judge can decide on the basis of article 6:682 of the Dutch Civil Code (art. 6:682 BW). The statement in the middle is that the dismissed person has always behaved like a good employee, a general obligation for employees that is codified in article 6:611 of the Dutch Civil Code (art. 6:611 BW). An arrow upwards indicates that the conclusion that the dismissal can be voided is supported by the statement that the dismissed person has always behaved like a good employee. This statement is in turn supported by the statement that the employee always arrived on time for work.

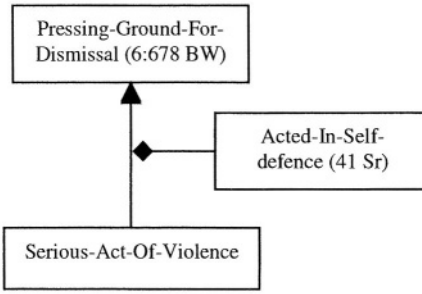
Conclusions cannot only be supported, but they can be attacked as well. Attacks are represented by arrows ending in a solid square. In the following figure one finds an example of this.



Here the conclusion that the dismissal can be voided is attacked by the statement that there is a pressing ground for dismissal according to article 6:678 of the Dutch Civil Code (art. 6:678 BW).

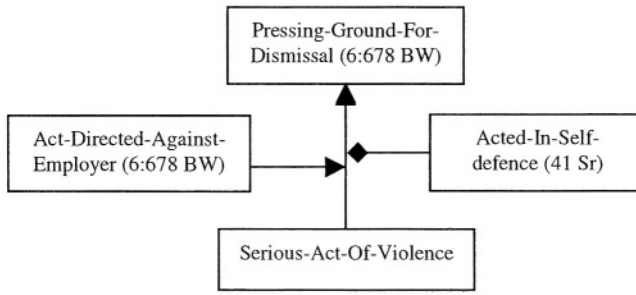
It is a key feature of the present approach that it can also be supported and attacked that a statement supports a conclusion (cf. Toulmin's warrants, 1958, pp. 98f.), or attacks it (Hage 1997, p. 166; Verheij 1996, pp. 200-201; Pollock 1995, pp. 41 and 86; Pollock 1987, p. 485).

A step forward to deal with this is to treat it as a statement itself that the conclusion is supported or attacked (cf. Verheij's DEFLOG, 2000, pp. 5f.l; 2003; see also Verheij 1999, pp. 45f.). Accordingly, one can represent by an arrow pointing at another arrow that it is supported or attacked that a statement supports or attacks a conclusion. This gives rise to a kind of entanglement of dialectical arguments (Roth 2001, pp. 31-33). An example is in the following figure.



At the top of this figure one finds the conclusion that there is a pressing ground for dismissal according to article 6:678 of the Dutch Civil Code. The conclusion is supported by the statement that the employee committed a serious act of violence. However, it is attacked that having committed a serious act of violence supports that there is a pressing ground for dismissal. The attacking statement is that the employee acted in self-defence, which is a general ground of justification according to article 41 of the Dutch Penal Code (art. 41 Sr). As a result, on the sole ground that the employee committed a serious act of violence, the conclusion does not follow that there is a pressing ground for dismissal.

It can also be supported that a statement supports or attacks a conclusion. An example is in the following figure.



Here the statement is made that the violent act was directed against the employer. This statement supports that having committed a serious act of violence supports that there is a pressing ground for dismissal, in accordance with article 6:678 of the Dutch Civil Code (art. 6:678 BW).

Dialectical arguments will be used as a convenient graphical representation of the argumentation in cases, but the question which conclusions follow is not answered by interpreting dialectical arguments regarding the status of the statements involved (cf. Verheij's dialectical interpretations, 2000, 2003).

3 Modelling Case-Based Reasoning by Case Comparison

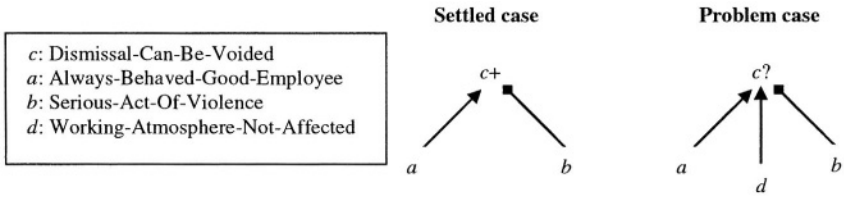
In this section reasoning by case comparison is modelled as a variant of reasoning a fortiori that involves dialectical arguments. The relevant case features appearing in these dialectical arguments are given by what is called a comparison basis. It is explained under which conditions a – possibly contradictory – set of settled cases can help decide a problem case.

3.1 Case Comparison

It is the purpose of case comparison to determine whether a settled case can be followed in a problem case. Intuitively one can certainly follow a settled case where, if there is at least as much support for the conclusion in the problem case. Then, by a kind of reasoning a fortiori, the conclusion should hold again.

The support for a conclusion is determined by the dialectical argument for it. In this connection we will use the term *dialectical support* for a conclusion. Case comparison will come down to comparing dialectical arguments regarding the dialectical support for their conclusion.

In the following a number of examples of increasing complexity are given.

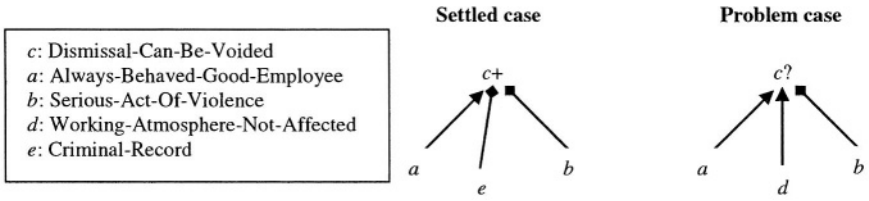


In this figure there is a settled case on the left where the conclusion (*c*) was drawn that a person's dismissal could be voided, as indicated by the plus sign. On the right there is a problem case where this conclusion is an issue, as indicated by the question mark.

In both cases the conclusion (*c*) that the dismissal can be voided is supported by the statement (*a*) that the person has always behaved like a good employee. The conclusion *c* is attacked by the statement *(b)* that the employee committed a serious

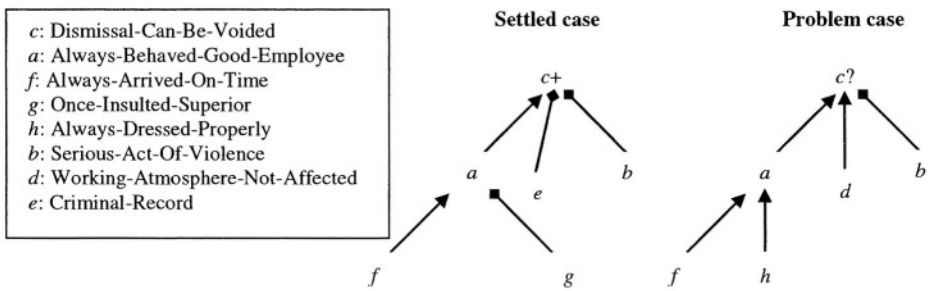
clusion *c* is attacked by the statement *(b)* that the employee committed a serious act of violence. In the problem case the conclusion *c* is also supported by the statement *(d)* that the working atmosphere has not been affected by the dismissal. As a result, there is more dialectical support for *c* in the problem case, so that it should follow there as well.

Another example is in the following figure.



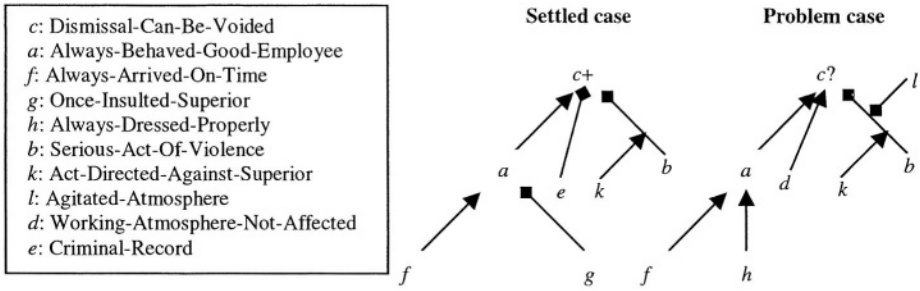
In the settled case the conclusion *c* is attacked by the statement *(e)* that the employee has a criminal record. Together with the difference *d* already discussed, this means that there is more dialectical support for *c* in the problem case. As a result, the conclusion (*c*) that the dismissal can be voided holds again.

Dialectical arguments can have a more complex structure. An example is in the following figure.



In this situation there is more dialectical support for the statement *a* in the problem case. In accordance with this, there is more dialectical support for conclusion *c* in the problem case than in the settled case, so that the conclusion can follow in the problem case as well. Note that for concluding to the outcome that there is more dialectical support for *c* in the problem case, it does not matter how the conflict with regard to the intermediate *a* is to be resolved.

It can itself be supported or attacked that one statement supports or attacks another. An example is in the following figure.



In this situation there is less dialectical support for the statement that *b* attacks *c* in the problem case, due to the attack by the statement (*l*) that the violent act took place in an agitated atmosphere. As a consequence, the problem case provides more dialectical support for conclusion *c* than the settled case, so that the conclusion can follow in the problem case as well. Note that for concluding to this result it is not necessary to resolve the conflict with regard to the attack by *b*.

3.2 The Comparison Basis

When comparing dialectical arguments one finds different types of statement. The conclusion at the top of these dialectical arguments tends to be the statement of an abstract legal state of affairs. At the bottom of the arguments one normally finds statements of non-legal and concrete facts.

Not all these states of affairs are relevant for the purpose of case comparison. The mere fact that John smashed a window, for example, can be irrelevant for the purpose of comparing his case with another case where liability for damage is an issue. At this point a distinction is therefore postulated between relevant and irrelevant states of affairs. From this point onwards all statements corresponding to relevant states of affairs will be called *factors,* to stress their role in case comparison.[1]

In general it is disputable which statements are factors and which are not. In other words, in the law it depends on a contingent choice which factors are taken into account when comparing cases. Among other things this choice will depend on the legal domain under consideration, such as dismissal or trade secret law.

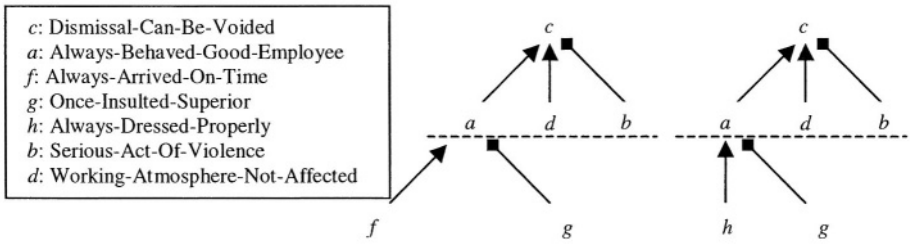
Some factors have the intuitive property that what supports them can be ignored for the purpose of case comparison, and that for this purpose all that matters is whether or not they can be derived. By definition these factors form a set called the *comparison basis,* and the factors in this set are called *basic.*[2]

In the figure below the comparison basis is visualised by drawing a line of division through the dialectical arguments. Statements above the line correspond to relevant states of affairs that have to be considered for the purpose of case comparison, while states of affairs below the line can be ignored. If the comparison basis is chosen as in

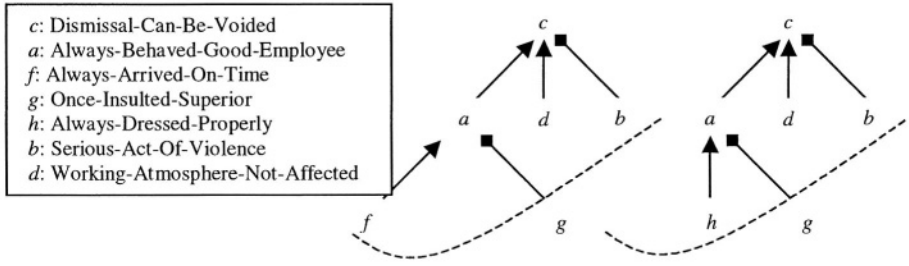
[1] Cf. the 'factors' in HYPO and CATO (cf. Ashley 1990, pp. 37-38).

[2] Cf. the 'base-level factors' in CATO (Aleven 1997 pp. 23 and 47).

the figure, then the two cases are comparable regarding their dialectical support for the conclusion *(c)* that the dismissal can be voided.



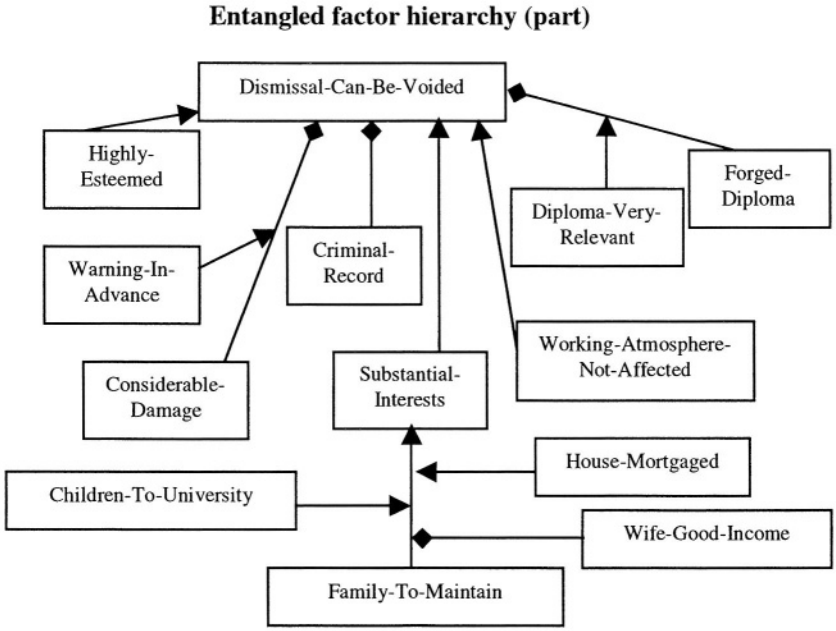
The following figure shows a different choice for the comparison basis in the same situation, according to which the supporting factors *f* and *h* are relevant as well. In other words, it makes a difference now whether the employee was always dressed properly, or always arrived on time. As a result, the two cases are now not comparable regarding the dialectical support for the conclusion.



These figures show that the comparison outcomes depend on the particular division made between factors and non-factors. They also illustrate the point that it is disputable which statements will count as basic factors.

3.3 The Entangled Factor Hierarchy

Let us consider all factors that follow from the comparison. Together these factors make up a representation of the argumentation pertaining to some domain of law: an ontology for the domain under consideration. An example of such a representation is depicted in the figure below.

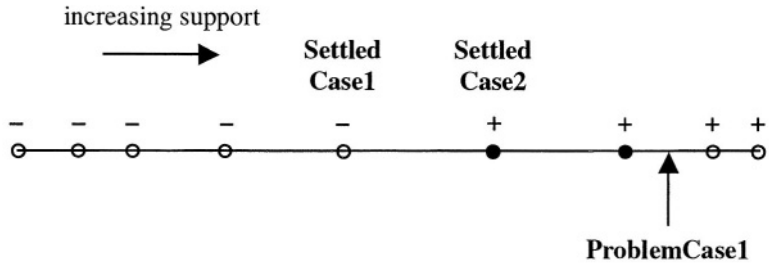


The representation thus obtained is similar to the so-called 'Factor Hierarchy' employed by CATO (Aleven 1997, pp. 44-45) as a case-independent knowledge structure. However, in contrast with CATO the present approach also allows for supporting and attacking statements of support or attack, which is why the present structure is called an *entangled factor hierarchy* (Roth 2001, pp. 31-33)

3.4 Which Settled Cases Are Relevant?

It is shown next how one can compare a number of cases regarding their dialectical support for some conclusion, and use the comparison to select the settled cases that are relevant to resolve a given problem case.

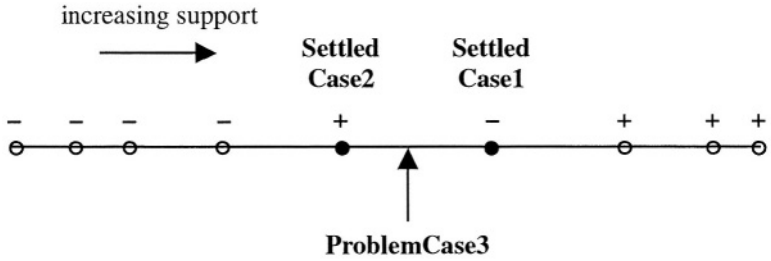
The settled cases are ordered regarding their dialectical support, which is represented graphically by placing the cases as dots on a line.



In this figure the small circles represent settled cases. The conclusion of each settled case is indicated by a plus or a minus sign, standing for a conclusion (say c) or its opposite conclusion ($\neg c$), respectively. From left to right there is increasing dialectical support for the conclusion (c).

ProblemCase1 is a problem case which has two positively decided cases on its left, represented by solid small circles. These are precedents for their conclusion (c), because **ProblemCase1** provides more dialectical support for the conclusion. Since there are no precedents for the opposite conclusion, it is clear that **ProblemCase1** should be decided positively as well.

In exceptional situations a judge may decide a case without taking recourse to the stare decisis principle. A reason for this could be, for instance, that a judge wants to take into account a changing view on the law. Whatever the reasons to depart from stare decisis, if it is done the set of precedents can become contradictory, and then it can become problematic to draw conclusions in problem cases. An example is in the following figure.



Obviously **SettledCase1** is now to the right of **SettledCase2.** As a result, the two decisions are contradictory, in the sense that the case with more support was decided negatively, while in the case with less support a positive conclusion was drawn. In other words, for whatever reason, at least one of these cases must have been decided against the principle of stare decisis.

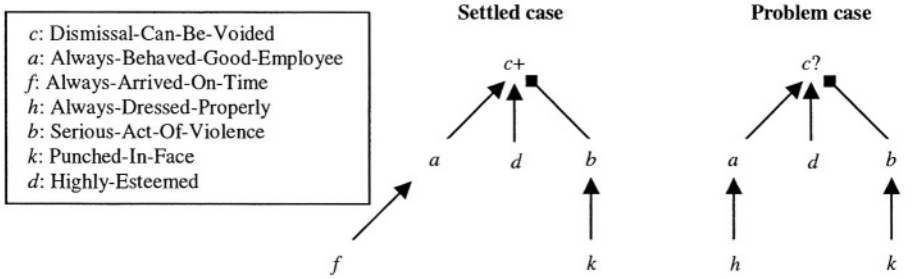
Now consider **ProblemCase3,** which is positioned between these two settled cases. Obviously, **SettledCase1** is a precedent for the opposite conclusion ($\neg c$) with respect to **ProblemCase3,** while **SettledCase2** is a precedent for the conclusion itself (c). As a result, the precedents cannot both be followed here.

This example shows that if precedents have been decided in a contradictory way and against the principle of stare decisis, then sometimes no conclusion follows from them in a problem case.

4 Applications

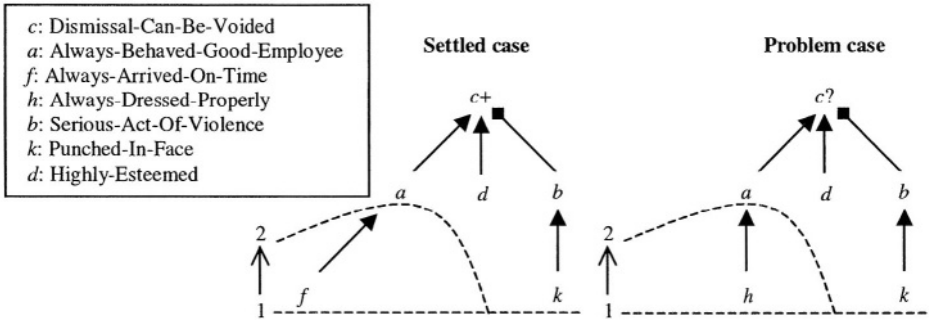
The ideas presented in the foregoing can be applied to arrive at an account of interesting reasoning patterns in case comparison. It is possible to accommodate the well-known analogising and distinguishing moves (Ashley 1990, pp. 25f.; Aleven 1997,

pp. 58f.; Roth 2003, pp. 86). A number of arguments on the importance of distinctions and similarities can also be captured, viz. downplaying and emphasising (cf. Aleven 1997, pp. 62f.). As an example of the latter kind, consider the situation of the following figure. Here a settled case is cited for the conclusion (*c*) that the dismissal can be voided.



Suppose that the settled case is distinguished from the problem case by pointing out the significant distinction (*f*) that the employee always arrived on time.

Downplaying with an Abstract Interpretation That Is a Significant Similarity
This way of downplaying is done by replacing a distinction with an abstract interpretation of it. Then the cases are compared in terms of this abstract interpretation, which then is a significant similarity. See the following figure.



In this figure there are two comparison bases. One is indicated with a straight dashed line at level 1, a second is represented by a curved dashed line at level 2.

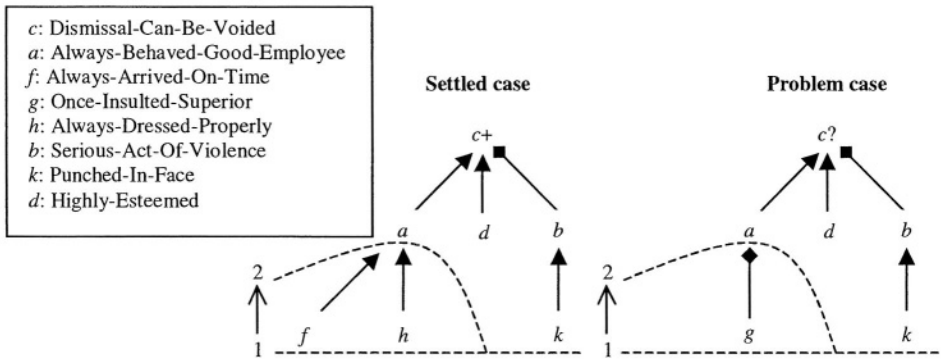
The figure shows that if the comparison basis at level 1 is chosen, then the factor (*f*) that the dismissed employee always arrived on time is a significant distinction. The distinction applies to the settled case and not to the problem case, and it supports the more abstract factor (*a*) that the person always behaved like a good employee.

The comparison can also be done in terms of this more abstract factor, which comes down to choosing another comparison basis. This second comparison basis differs from the first in that the original factor *f* is replaced with its abstract interpreta-

tion *a*. In the figure this second comparison basis is indicated by the curved dashed line at level 2.

Relative to this second comparison basis the factor *f* is no significant distinction because it is not a factor. Its abstract interpretation *a* is a basic factor that applies to both cases. As a result, relative to the second comparison basis the factor *a* is a significant similarity between the cases.

Emphasising with an Abstract Interpretation That Is a Significant Distinction
Emphasising a distinction also involves an abstract interpretation of the distinction. The difference is, though, that here the abstract interpretation is a significant distinction and not a significant similarity. See the following figure.



Again the focus is on the distinction *f*. The original distinction *f* is replaced with its abstract interpretation *a*, and again this yields another comparison basis at level 2.

This abstract interpretation *a* is a basic factor that applies to the settled case but not to the problem case. As a result, relative to the second comparison basis the factor *a* is a significant distinction between the cases.

These examples show how distinctions can be downplayed or emphasised. In a similar way one can downplay or emphasise significant similarities (Roth 2003, pp. 95f.).

5 Related Research

In Section 5.1 we discuss research that specifically focuses on case-based reasoning in the law. Section 5.2 addresses research on legal ontologies.

5.1 Case-Based Reasoning

The HYPO system represents one of the most important contributions to case-based reasoning in the field of Artificial Intelligence and Law. HYPO is an implemented model of case-based reasoning that can generate realistic arguments with cases on the basis of expert background knowledge on supporting and attacking factors (p. 26). The background knowledge used for generating arguments is represented by tagging factors with a plus or a minus sign if they support a conclusion or its opposite, respectively.

In HYPO it is not disputable whether or not a factor supports some conclusion or its opposite, however. Recall that in approach adopted in this paper, in contrast, it can be supported or attacked that one statement supports or attacks another (entanglement).

The CATO program (Aleven 1997; Aleven 1996; Aleven and Ashley 1997) uses a model of case-based argumentation in an instructional setting, to teach students basic skills of arguing with cases.

The CATO model of case-based reasoning relies on a Factor Hierarchy, a body of case-independent background knowledge about how the relevant factors in some domain relate to each other. Among other things the Factor Hierarchy is used to reason about the significance of distinctions between cases (downplaying and emphasizing). To produce such arguments CATO can infer abstract interpretations of with the help of the links encoded in its Factor Hierarchy, thereby guided by special strategic heuristics. As shown previously, such downplaying and emphasising arguments can also be accommodated in the present approach. There presently are no heuristics to guide a strategic choice among several possible ways of downplaying or emphasising, however.

Another difference with the present approach is that in CATO it cannot be supported or attacked that a factor supports a conclusion or its opposite.

Prakken and Sartor (1998) deal with case-based reasoning as a kind of dialectical argumentation. Cases are represented by collections of rules from which arguments can be constructed (p. 256). The conflict resolving potential of settled cases is exploited by interpreting them as sources of priority information on conflicting rules (p. 256).

In Prakken and Sartor's system the focus is on analogizing and distinguishing as ways of reasoning with cases. These two ways of reasoning are treated as a kind of premise introduction (p. 261) or theory construction (cf. Prakken 2000, pp. 51f.). They involve the introduction of new rules on the basis of rules of settled cases.

This way of reasoning with cases differs from the present approach in that a suitable comparison outcome must be established before a settled case can be followed.

There must be at least as much dialectical support for a conclusion in the problem case, and only then can one draw the conclusion.

Prakken and Sartor want to deal with case comparison in terms of HYPO's criterion of on pointness (p. 267). Due to problems in connection with derivable factors, however, they choose not to define comparison outcomes in terms of on pointness (p. 270). Instead they simply assume that some criterion for on pointness has already been agreed upon, as part of the premises from which the debate starts (pp. 270-271).

This is in contrast with the present approach, where the outcomes of case comparison are defined in terms of the level of dialectical support for a conclusion.

A final difference with the present approach is that the set of factors relevant for case comparison is not introduced explicitly to acknowledge the contingency of this set.

Bench-Capon and Sartor (2001) have proposed an approach in which reasoning with cases is treated as the construction and use of theories.

In their model, theories are intended to explain decided cases in terms of the values promoted by the legal system as a whole, and to help predict the outcomes of new cases. Briefly, decisions are interpreted as evidence for rule priorities, which are in turn interpreted in terms of priorities between the values upheld by the rules. These value priorities can then be used to derive new rule priorities that can help decide problem cases.

Bench-Capon and Sartor's model does not recognise explicitly that in the law it depends on a contingent choice which factors are relevant for case comparison.

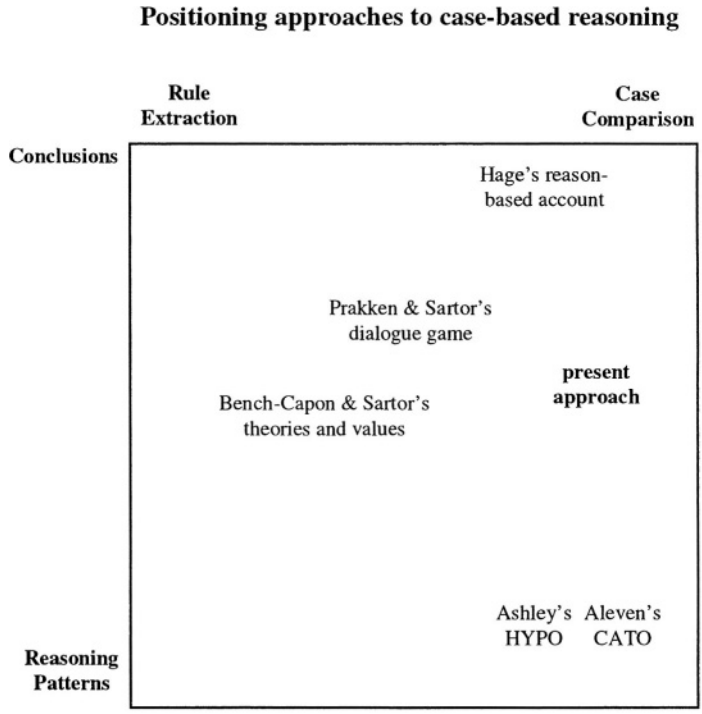
Hage gives an account of case-based reasoning in Reason-Based Logic (Hage 1997, Chapters IV and V; Verheij 1996, Chapters 2 and 3). Reason-Based Logic is a logical system which involves the use of rules and principles in legal reasoning, and which allows for exceptions to rules or principles, to the effect that conclusions cease to hold (Hage 1997, pp. 137-138, 141-143 and 150-151).

In Hage's reason-based account of case-based reasoning, case comparison is done in terms of the reasons for or against the disputed conclusion. In other words, Hage's account only captures one-step arguments.

Another important difference between Hage's account and the present one is that while entanglement of factors can in principle be captured in Hage's Reason-Based Logic, it does not play a role in his account of case comparison.

Two ways of categorising approaches to case-based reasoning are illustrated next. First, one can distinguish the approaches by the method of employing cases that they focus on: the rule extraction or case comparison method. Second, one can categorise approaches by their relative emphasis on the legal conclusions that follow by comparison with settled cases, or the reasoning patterns along which they follow.

An example of a categorisation along these distinctions is in the following figure.



5.2 Cases in Legal Ontologies

There exists a lively research community dealing with the topic of legal ontologies from an artificial intelligence perspective (cf., e.g., Visser & Winkels 1997, Winkels, Van Engers & Bench-Capon 2001).[3] We briefly discuss work on legal ontologies and address the question how cases are treated. It is also indicated whether the work is closer to the rule extraction method or to the case comparison method.

Valente (1995) has presented a functional ontology of the law. Valente's functional perspective on the law is reflected in the categories of legal knowledge he distinguishes: normative knowledge, world knowledge, responsibility knowledge, reactive knowledge, meta-legal knowledge and creative knowledge. Valente does not directly address case-based reasoning (although he gives a critical appraisal of different approaches in the field of artificial intelligence and law, including the case-based approach; p.12f.). He explicitly moves away from rule-based, case-based and logical approaches to artificial intelligence and law (p. 7f.), and uses a modelling approach

[3] A useful web site for pointers to research concerning legal ontologies is the home page of Radboud Winkels (http://www.lri.jur.uva.nl/~winkels/).

instead. This makes it hard to determine whether modelling case-based reasoning using his approach would be closer to rule extraction or to case comparison.

Valente defines cases (as sets of propositions, p. 91, and as descriptions of states of affairs in the world, p. 125), but his cases represent individual behaviour to which legal norms are applied. They do not represent the legal decisions that are central in case-based reasoning. Knowledge about legal decisions would in his ontology fall under different categories of knowledge. Decided cases are for instance sources for legal rules, classificatory rules and conflict resolution. Knowledge about the legal rules extracted from cases would fall under normative knowledge, classificatory knowledge is world knowledge and conflict resolution knowledge is meta-legal knowledge.

Van Kralingen (1995) and Visser (1995) have developed a frame-based conceptual model of the law. They distinguish three types of frames: norm frames, act frames and concept frames. Their approach is explicitly statute-oriented. For instance, their prime example is a model of the Dutch Unemployment Benefits Act. As a result, the role of decided cases is hardly addressed. In their ontology, case-based reasoning would have to be dealt with using the rule-extraction method by modelling the rules extracted from cases in the form of norm frames.

Hage and Verheij (1999) have described the law as a dynamic interconnected system of states of affairs. They use three primitives: states of affairs, events and rules. The primitives are used to analyse legal topics such as classification, rights, proof and validity. Their perspective on the law is rule-based, although Hage and Verheij also discuss principles and goals. In their approach, case-based reasoning would have to be addressed using the rule-extraction method.

Mommers (2002) has built a knowledge-based ontology of the legal domain. He distinguishes entities (e.g., legal norms and decisions), ontological status layers (e.g., existence, validity and recognition), epistemic roles (e.g., reasons and defeaters), relations (e.g., causation, counting as), acts (e.g., applying rules and making decisions) and facts (e.g., brute facts and institutional facts). Mommers recognizes the place of legal decisions (as legal entities) and of making decisions (as acts) in his ontology. He also specifies types of beliefs, with respect to their content (e.g., beliefs based on case law) and origination (e.g., beliefs resulting from interpretation). Mommers does not explicitly address case-based reasoning. As a result, it is indeterminate whether applying his ontology would lead to a rule extraction or to a case comparison method towards case-based reasoning.

From the sample of artificial intelligence research on legal ontologies discussed above, it appears that ontological issues concerning cases and case-based reasoning have as yet not been fully addressed in the artificial intelligence community.

6 Conclusion

The present paper has shown how case comparison can be used as the basis for modelling case-based reasoning. In the model proposed here (and further elaborated by Roth, 2003), cases are compared in terms of the dialectical arguments expressed in them. The distinction between rule-extraction and case comparison in case-based reasoning leads to different conceptions of cases and thus to different ways of modelling reasoning with them. The distinction has been used to discuss related research on case-based reasoning. A discussion of research on legal ontologies has shown that cases and case-based reasoning have received relatively little attention in such research.

References

1. Aleven, V. & Ashley, K.D. (1997). Evaluating a Learning Environment for Case-Based Argumentation Skills. *Proceedings of the Sixth International Conference on Artificial Intelligence and Law,* pp. 170-179. New York: ACM Press.
2. Aleven, V. (1996). How Different Is Different? Arguing about the Significance of Similarities and Differences. *Proceedings of the Third European Workshop: EWCBR-96,* pp. 1-15.
3. Aleven, V. (1997). *Teaching Case-Based Reasoning Through a Model and Examples.* Ph.D. dissertation University of Pittsburgh.
4. Ashley, K.D. (1989). Toward a Computational Theory of Arguing with Precedents: Accomodating Multiple Interpretations of Cases. *Proceedings of the Second International Conference on Artificial Intelligence and Law,* pp. 93-102. New York: ACM Press.
5. Ashley, K.D. (1990). *Modeling Legal Argument: Reasoning with Cases and Hypotheticals.* Cambridge: MIT Press/Bradford Books.
6. Ashley, K.D. (1991). Reasoning with cases and hypotheticals in HYPO. *International Journal ofMan-Machine Studies,* Volume 34, pp. 753-796.
7. Bankowski, Z. (1997). Rationales for Precedent. *Interpreting Precedents. A comparative study* (eds. MacCormick, D.N. and Summers, R.S.). Aldershot: Dartmouth Publishing Company Limited.
8. Bench-Capon, T.J.M. & Sartor, G. (2001). Theory Based Explanation of Case Law Domains. *Proceedings of the Eighth International Conference on Artificial Intelligence and Law,* pp. 12-21. New York: ACM Press.
9. Bench-Capon, T.J.M. (1999). Some Observations on Modelling Case Based Reasoning With Formal Argument Models. *Proceedings of the Seventh International Conference on Artificial Intelligence and Law,* pp. 36-42. New York: ACM Press.
10. Cross, R. (1977). *Precedent in English Law.* Third edition. Oxford University Press.
11. Hage, J.C. (1997). *Reasoning with Rules. An Essay on Legal Reasoning and Its Underlying Logic.* Dordrecht: Kluwer Law and Philosophy Library.
12. Llewellyn, K.N. (1960). *The Common Law Tradition.* Boston: Little, Brown and Company.
13. Loui, R.P. & Norman, J. (1995). Rationales and Argument Moves. *Artificial Intelligence and Law,* Volume 3, pp. 159-189.

14. Loui, R.P. (1997). Eliding the Arguments of Cases. *Draft.* Downloaded from http://www.cs.wustl.edu/~loui/ratjur.txt.
15. MacCormick, D.N. (1987). Why Cases have *Rationes* and what these are. In: *Precedent in Law* (ed. Goldstein, L.), pp. 155-182. Oxford: Clarendon Press.
16. Mommers, L. (2002). *Applied legal epistemology. Building a knowledge-based ontology of the legal domain.* Dissertation Leiden University.
17. Oliphant, H. (1928). A return to stare decisis. *American Bar Association Journal,* Volume 14. Reprint in: *American Legal Theory* (ed. Summers, R.S. 1992), pp. 143-153. Aldershot: Dartmouth Publishing Company Limited.
18. Pollock, J.L. (1987). Defeasible Reasoning. *Cognitive Science,* Volume 11, pp. 481-518.
19. Pollock, J.L. (1995). *Cognitive Carpentry : A Blueprint for How to Build a Person.* Cambridge: MIT Press.
20. Prakken, H. & Sartor, G. (1998). Modelling Reasoning with Precedents in a Formal Dialogue Game. *Artificial Intelligence and Law,* Volume 6, pp. 231-287.
21. Prakken, H. (2000). An exercise in formalising teleological case-based reasoning. *Legal Knowledge and Information Systems: Jurix 2000* (eds. Breuker, J., Leenes, R. and Winkels, R.), pp. 49-57. Amsterdam: IOS Press.
22. Roth, A.C. (2001). The Dialectics of Case Comparison: a Formal Reconstruction. *Proceedings of the Fourteenth Annual Jurix Conference* (eds. Verheij, B., Lodder, A.R., Loui, R.P. and Muntjewerff, A.J.), 29-40.
23. Roth, A.C. (2003). *Case-based reasoning in the law. A formal theory of reasoning by case comparison.* Dissertation Universiteit Maastricht.
24. Toulmin, S.E. (1958). *The Uses of Argument.* London: Cambridge University Press.
25. Valente, A. (1995). *Legal Knowledge Engineering. A Modelling Approach.* IOS Press, Amsterdam. Dissertation.
26. van Kralingen, R.W. (1995). *Frame-based Conceptual Models of Statute Law.* Dissertation Leiden University.
27. Verheij, B. (1996). *Rules, Reasons, Arguments. Formal studies of argumentation and defeat.* Dissertation Universiteit Maastricht.
28. Verheij, B. (1999). Automated Argument Assistance for Lawyers. *Proceedings of the Seventh International Conference on Artificial Intelligence and Law,* pp. 43-52. New York: ACM Press.
29. Verheij, B. (2000). DEFLOG - a logic of dialectical justification and defeat. *Draft.* Downloaded from http://www.metajur.unimaas.nl/~bart/publications.htm.
30. Verheij, B. (2003). DefLog: on the Logical Interpretation of Prima Facie Justified Assumptions. *Journal of Logic and Computation,* Volume 13, no. 3, pp. 319-346.
31. Visser, P.R.S. & Bench-Capon, T.J.M. (1998). A Comparison of Four Ontologies for the Design of Legal Knowledge Systems. *Artificial Intelligence and Law,* Vol. 6, No. 1, pp. 27-57.
32. Visser, P.R.S. & Winkels, R. (eds.) (1997). *Proceedings of the First International Workshop on Legal Ontologies. LEGONT '97.* Melbourne: University of Melbourne.
33. Visser, P.R.S. (1995). *Knowledge Specification for Multiple Legal Tasks. A Case Study of the Interaction Problem in the Legal Domain.* Dissertation Leiden University.
34. Winkels, R., Van Engers, T. & Bench-Capon, T.J.M. (eds.) (2001). *Proceedings of the Second International Workshop on Legal Ontologies.* Amsterdam: Universiteit van Amsterdam.

Modeling the Tax Code

Eric Melz and Andre Valente

USC Information Sciences Institute (ISI)
4676 Admiralty Way, Marina del Rey, CA 90292, U.S.A.
{melz, valente}@isi.edu

Abstract. In this paper, we argue for the necessity of constructing an ontology which represents the U.S. Internal Revenue Code (IRC). Doing so would enable the construction of a broad range of intelligent applications, including automatic auditing software, robust on-line help systems, and tax question-answering systems. In this paper, we propose the construction of a rich ontology which models the tax code. We examine some of the unique challenges presented by a tax ontology and provide examples of the types of knowledge necessary for such an ontology.

1 Introduction

The U.S. tax code is embodied in the Internal Revenue Code (IRC), a legal document containing some 50,000 definitions and rules. Ontologizing the IRC would allow a broad range of tax-related applications to be built, including help systems, auditor agents and question answering systems. In this paper we propose the construction of a rich ontology which models the IRC. We discuss general requirements of legal ontologies, and the unique challenges presented by the tax domain. We present an example to illustrate the types of knowledge necessary for a tax ontology.

This paper is organized as follows. Section 2 presents a brief overview the economic and social costs of the present tax system, and discuss the potential benefits of Knowledge Representation and Reasoning technology for the tax system. Section 3 identifies five main uses or roles for ontologies. Section 4 shows the types of knowledge that will be present in our Tax Ontology. Section 5 considers some potential methodologies for acquiring tax knowledge. Section 6 examines some styles of reasoning that are necessary in the tax domain. Finally, Section 7 wraps up with some conclusions.

2 The Tax Code

A famous saying states that "There are only two things in life that are certain, death and taxes". One difference between death and taxes is that death only occurs once in a lifetime. In contrast, every adult in the U.S. (and possibly the world) has to face their tax situation at least once a year and understanding the tax rules is no minor task. The U.S. tax code is so complex that no single person can possibly master the entire

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 652–661, 2004.
© Springer-Verlag Berlin Heidelberg 2004

document. The complete Internal Revenue Code is more than 21 megabytes in length, and contains more than 2.8 million words; printed 60 lines to the page, it would fill almost 6000 letter-size pages [1].

This fact of life has enormous social and economic costs. Each year, the government spends $3.5 Billion on post-filing compliance services such as audits [2]. Individuals needing non-trivial tax advice must hire expensive tax advisors, which may only be available to relatively wealthy people. In addition, there are costs which are more difficult to quantify, such as the frustration experienced and time wasted by individuals due to insufficient knowledge about the tax code.

Tax software such as TurboTax [3] has slightly eased the annual burden of tax preparation. However, these applications do not possess deep knowledge of the tax code. Typically, tax preparation software leads the user through a series of interviews to obtain financial information. At the end of the interview, the user is informed how much they owe or are owed by the government. These programs typically contain help systems which purport to answer questions about taxes, however they amount to little more than Google-style document retrieval systems. They are capable of retrieving tax publications, but the user is responsible for reading and understanding the knowledge contained within the retrieved documents.

In this paper, we argue for the necessity of constructing an ontology which represents the IRC. Doing so would enable the construction of a broad range of intelligent applications, including automatic auditing software, robust on-line help systems, and tax question-answering systems. In particular, we are interested in the development of question answering systems that can integrate Natural Language Processing and Knowledge Representation Techniques to help users find solutions to tax questions.

3 Roles of Ontologies in AI and Law

Before building an ontology, it is useful to understand why it is needed and what role it plays. This awareness can help us better make design decisions about how to build the ontology and what trade-offs are involved.

Valente [15] identified five main uses or roles for ontologies: (a) organize and structure information; (b) reasoning and problem solving; (c) semantic indexing and search; (d) semantics integration and interoperation; and (e) understanding the domain. Specifically in AI & Law, these uses have manifested as follows.

3.1 Organize and Structure Information

The basic role of ontologies in this case is to organize and structure information in the domain. Ontologies here are tools to describe things or phenomena in the domain of interest. The ontology thus plays the role of *vocabulary,* answering two main questions: (a) which terms can be used? (i.e., ontology as a *lexicon*); and (b) which (valid) sentences can be expressed (i.e. ontology as a *grammar*) say?

In AI & Law, this role is shown in the use of ontologies to define legal vocabularies. These are typically used to define the terms used in regulations. In this way, the

ontologies are not so much legal ontologies but representations of the world or domain the law is working on, e.g. taxes, crime, traffic, immigration, etc.

3.2 Reasoning and Problem Solving

The basic role of ontologies in this case is to represent the knowledge of the domain so that an automated reasoner can represent problems and generate solutions for these problems. The ontology here works as the *structure of the knowledge base.* This use is found in the many expert systems (problem solvers) and decision making systems developed in AI & Law. Notice that other uses of ontologies may require reasoning facilities; the difference here is that this is the emphasis of problem-solving

In using ontologies for this role, secondary goals are to create knowledge bases that are reusable, efficient, explainable, modular, etc. Indeed, one can argue that the use of ontologies in AI comes from research in the late eighties and nineties that aimed at improving knowledge engineering by attacking these roles by creating "well-structured" knowledge bases that would not only solve the problem at hand but be more maintainable, easier to extend, etc. In this sense, ontologies in this use are very much an engineering tool.

This role of ontologies implies the use of an *inference engine* that is used to conclude specific goals. An interesting problem that arises is the introduction of an *inference bias.* Valente et.al. [11] argued that ontological choices are strongly influenced by the purpose of the ontology. That is, the same knowledge will be structured or formalized differently depending of how it will be used by the reasoner in reaching the desired conclusions in a specific context. This indicates that reusability is a good idea, but it can never be accomplished completely. Indeed, we believe the inference bias is both *inevitable* and *positive.* It is positive because tailoring the ontology to its use privileges use before reuse. It is inevitable be cause no formulation will be completely neutral. Indeed, [11] showed that correct logical definitions may not be computationally useful depending on the inference engine. Therefore, we are better off embracing rather than avoiding the inference bias, making it explicit as a deign criteria in formulating our ontologies.

3.3 Semantic Indexing and Search

The basic role of ontologies in this case is to represent the contents of documents or other "soft" knowledge sources (picture, movies, etc.). The ontology here works as the a *semantic index of information,* that enables semantic search for content.

Law and legal practice produce vast amounts of knowledge in the forms of documents, charts, schemas, etc. There is a key need to organize and be able to find these documents. Ontologies can be used to represent and search semantically the content of documents – to go beyond word or keywords.

The traditional example that shows the need for this use of ontologies is the existence of multiple meaning of words – e.g., "sun" as the computer company or the star. Ontologies can be used to disambiguate these natural language meanings. Ontologies

can also be used in a more intentional way, as a mechanism for creating annotations – i.e., allowing a person to semantically mark content so it can be found later.

An example of this use is the work of [16], who created an application to retrieve FAQ questions about legal procedures that included an ontology-based interface for query and retrieval.

3.4 Semantic Integration/Interoperation

The basic role of ontologies in this case is to support applications to exchange information electronically. The ontology here works as an *interlingua* that defines a (narrow) vocabulary to be used to interchange information.

This use is less common in the legal domain, but there is potential for use in law enforcement, e.g., organizations exchanging information about criminals. There is also a lot of use in quasi-legal situations such as in complex systems in large bureaucracies that need to interoperate (e.g., the European Union). Since they can be seen as a semantic information schema, these ontologies may reuse parts of ontologies created for other uses.

3.5 Understanding a Domain

The basic role of ontologies in this case is to provide a view of what a domain is about – to try to make sense of the variety of knowledge in that domain. The ontology here works as a *map* that specifies what kinds of knowledge can be identified in the domain.

This type of ontology can be used as a basis for designing specialized representations.. Because it tries to get close to the nature of the domain, it frequently connects and draws from theories of that domain (e.g., theories of law). These types of ontologies have been called *core ontologies* [13]

An example of this type of ontology is the Functional Ontology of Law created by Valente and Breuker [12]. It defines a number of functional roles legal knowledge may play, namely (a) world knowledge, (b) normative knowledge, (c) responsibility knowledge, (d) reactive knowledge, (e) creative knowledge and (f) meta-legal knowledge. These functional roles were used to define specific formalisms and specialized reasoners in the ON-LINE architecture [12].

Other notable examples of core ontologies of law are Mommers' epistemology (ontology) of law [10] and McCarty's Language of Legal Discourse [14].

4 A Tax Ontology

Tax Ontologies are closely related to legal ontologies such as those discussed in Visser et al [17], but have several distinguishing features. For example, legal ontologies presented in the literature are primarily concerned with modeling the legal system,

and focus on modeling abstract legal entities such as normative knowledge (knowledge about what the world should be like) and responsibility knowledge (knowledge about the relationship between causality and legal responsibility) (cf Valente [12]). This type of knowledge can be useful in the tax domain, particularly for systems that answer questions about tax consequences given hypothetical situations. However, applications which make use of tax ontologies will require additional types of knowledge, such as mathematical and financial knowledge. The inclusion of such knowledge will allow applications to leverage tax ontologies for problem solving as well as question answering. For example, mathematical models of various depreciation methods will allow users to perform "what-if" scenarios given their particular tax situation.

The main intended application of our tax ontology is to support the development of question answering systems for tax questions. Our ontology will play two of the five roles above: organize and structure information, and semantic indexing and search. These uses guide the types of knowledge we need to model. In order to support both these roles, our ontology will need to include 'world knowledge" and "definitional knowledge" [12]. World knowledge is background knowledge not directly present in the IRC such as knowledge of economics, time, or accounting (e.g., concepts such as depreciation, basis, and income). Definitional knowledge consists of definitions of the terms used in the law; this requires encoding the rules and definitions embodied in the IRC. It turns out that the same kinds of knowledge are able to support semantic indexing and search.

As an example, consider IRC Title 26, Subtitle A, Chapter 1, Subchapter B, Part II, Sec. 109:

> Gross income does not include income (other than rent) derived by a lessor of real property on the termination of a lease, representing the value of such property attributable to buildings erected or other improvements made by the lessee.

Our interpretation of this rule is as follows: If a lessee (i.e., tenant) built something on the property that they rent, after they terminate their lease the lessor (i.e., landlord) can't count as gross income money derived from the tenant's building, unless that money is produced by renting the building. For example, suppose some tenants built a semiconductor factory on the landlord's property, and laterthe lease between the tenants and the landlord expired. At this point and afterward, the landlord would not be allowed to include the money from the sale of microchips into his/her gross income. However, if the landlord rented out the factory, the landlord would be able to include the rent into his/her gross income.

We would like to represent the IRC rules with enough fidelity such that a question-answering system would be capable of using the knowledge base to answer complex, scenario-based questions like: "If a tenant build a guest house on my property, should I include the rent in my Gross Income?"

The world knowledge required by this rule includes notions of lessors, lessees, leases, rent, buildings, knowledge about land and improvements to land, gross in-

come, value, property, and real property. The domain-specific knowledge is the representation of the IRC rule itself. Below, we consider each knowledge item, and show how the knowledge might be represented in KIF, a LISP-like language for first-order predicate calculus [5].

First, we consider world Knowledge about lessors, lessees, leases, and rent. We state that Lessors and Lessess are people, and define Real Property.

```
(defconcept |Person| (?x))

(defconcept |Lessor| (?x |Person|))

(defconcept |Lessee| (?x |Person|))

(defconcept |RealProperty| (?x))
```

Next, we need knowledge about leases and contracts. In the general case, we might want to have a detailed representation of contracts, and model facts like a contract is a written document binding two parties to an agreement, and that a violation of the contract causes certain things to happen. However, for the purposes of this example, we can simply model that a lease is a contract between a lessor and a lessee involving Real Property..

```
(defrelation |contract| ((?partyA |Person|) (?partyB |Person|))

(defrelation |lease-contract| ((?x |Lessor|) (?y |Lessee|)
                               (?z |RealProperty|))
     => (|contract| ?x ?y))
```

We also need knowledge about buildings and improvements, specifically that buildings are improvements and that real property can contain buildings and improvements. We also model the fact that if someone creates an improvement for a piece of Real Property, that property contains the improvement.

```
(defconcept |Improvement| (?x))

(defconcept |Building (?x |Improvement|))

(defconcept |ImprovementSet| (?x))

(forall (?i)
    (=> (member-of ?i |ImprovementSet|)
            (|Improvement| ?i)))

(defrelation |improvements| ((?x |RealProperty|)
                             (?y |ImprovementSet|)))

(defrelation |create-improvement-for| ((?person |Person|)
                        (?property |RealProperty|)
                        (?improvement |Improvement|)))

(forall (?person ?property ?improvement)
              (=> (|create-improvement-for| ?person ?property
                   ?improvement)
                  (exists (?is)
                     (and (|ImprovementSet| ?is)
                          (|improvements| ?property ?is)
                          (member-of ?improvement ?is)))))
```

We must also model the fact that money can be derived from different sources. Here, we create a taxonomy of money, specifying that Rent is a kind of Money, and that money derived from rent is disjoint from other types of money:

```
(defconcept |Money| (?x))

(defconcept |Improvement-Money| (?x |Money|))

(defconcept |Rent-Money| (?x |Improvement-Money|))

(defconcept |NonRent-Money| (?x |Improvement-Money|))

(forall (?x)
    (=> (|Rent-Money| ?x) (not (|NonRent-Money| ?x)))))

(forall (?x)
    (=> (|NonRent-Money| ?x) (not (|Rent-Money| ?x)))))
```

We also model the notion that money can be derived from improvements:

```
(defrelation |generates-money| ((?x |Improvement|)
                                (?y |Improvement-Money|)))
```

We model the fact that Gross Income can have many constituent sources, i.e., different types of Money:

```
(defconcept |GrossIncome| (?x))

(forall (?i)
    (=> (member-of ?i |GrossIncome|)
        (|Money| ?i)))

(deffunction |has-income| ((?x |Person|) (?y |GrossIncome|)))

(forall (?x)
    (=> (|Person| ?x)
        (exists (?income)
          (|has-income| ?x ?income))))
```

Finally, we model the domain-specific IRC rule, namely that if an improvement was created by a lessee, the "other" part of the money generated by the building/improvement will not be included in the Lessor's Gross Income, and the "rent" part will be included in the Lessor's Gross Income:

```
(forall (?lessor ?gross-income ?lessee ?property ?improvement)
          (=> (and (|create-improvement-for| ?lessee ?property
                    ?improvement)
                   (|Lessor| ?lessor)
                   (|Lessee| ?lessee)
                   (|Improvement| ?improvement)
                   (|GrossIncome| ?gross-income)
                   (|has-income| ?lessor ?gross-income)
                   (|generates-money| ?improvement
                                      ?improvement-money)
                   (|lease-contract| ?lessor ?lessee ?property))
              (and
                (or (and (|Rent-Money| ?improvement-money)
                         (member-of ?improvement-money
                                    ?gross-income))
                    (and (|NonRent-Money| ?improvement-money)
                         (not (member-of ?improvement-money
                                         ?gross-income)))))))))
```

To ask the question we posited above ("If a tenant built a guest house on my property, should I include the rent in my Gross Income?"), we assert the relevant facts and perform a backward-chaining query:

```
(assert (|lease-contract| |me| |tenant| |my-property|))
(assert (|create-improvement-for| |tenant| |myprop| |guest-house|))
(assert (|generates-money| |guest-house| |guest-house-rent|))
(assert (|Rent-Money| |guest-house-rent|))
```

Assuming all facts and rules have been modeled correctly, a Knowledge Representation and Reasoning system such as PowerLoom [5] should return TRUE to yje query:

```
(ask (member-of |guest-house-rent| (|has-income| |me|)))
```

5 Tax Knowledge Engineering

Clearly, modeling the IRC would be a massive effort. Not only would it be necessary to model each of the roughly 50,000 rules in the IRC, but it would be necessary to model a huge amount of world knowledge. While it might be the case that some world knowledge can be adapted from general-purpose ontologies such as FrameNet [6] and Cyc [7], a high-quality rendering of the IRC would inevitably require a massive amount of human engineering. In this section, we consider several approaches to the problem of acquiring tax knowledge.

Cyc Model - The Cyc knowledge base [7] was created by an organization consisting of dozens of people over a span of approximately two decades. A tax ontology could be constructed in a similar fashion- many people could be hired for the specific purpose of modeling the IRC. While such an endeavor would be costly, the cost could be justified given the potential savings to the U.S. government. For example, if it could be shown that a tax ontology would result in a 1% savings for the government, this would free $35 million per year for ontology development.

OpenMind Model - The OpenMind project [8] is a sort of open-source knowledge base, where members of the public collaboratively contribute knowledge to the system. This development model could be applied to tax ontology. In this model, a reward system could be developed whereby individuals who donate high-quality contributions to the ontology could gain kudos points or other tangible benefits, such as free time using tax software.

NLP Bootstrapping Model - If a sufficient amount of world knowledge was modeled, it might be possible to use Natural Language Understanding technology to create semantic representations of IRC rules. The system would have to have high-quality lexical and grammatical knowledge in order to process the rules, and post-editing of the rules by humans would likely be unavoidable.

Top-Down Model - In the top down model, knowledge would be entered on an "as needed" basis. For example, an analysis of an application domain might dictate the most important types of knowledge to add to the ontology. An example of such an analysis is in the domain of Tax Question Answering. Here, lists of Frequently Asked

Questions could be analyzed to determine which questions are frequently asked. This would in turn dictate the requisite ontological knowledge.

6 Reasoning with Tax Knowledge

As we explained in Section 1, one of our main interests is to use the ontology underlined above to develop question answering systems that integrate Natural Language Processing and Knowledge Representation Techniques to help users find solutions to tax questions. In this section, we examine some typical tax questions, and discuss the styles of reasoning necessary to support answering such questions. We categorize tax questions according to their difficulty: factoid, yes/no, simple scenario, and complex scenario. Examples of each question type are discussed below.

Factoid - These are the most basic type of questions. An example factoid question is, "How long does it take after you've filed to receive a refund?". Such a question can be answered by a simple fact lookup in the knowledge base.

Yes/No - These types of questions require a yes/no (i.e., TRUE/FALSE) type answer. These types of questions can be answered by trying to prove an assertion using backward chaining. An example of a yes/no question is "Is tuition reimbursement for school a form of taxable income?". Many times it is desirable to provide a more elaborate answer in addition to a simple Boolean value. In this case, the system might add: If you receive educational assistance from your employer under an educational assistance program, you can exclude up to $5,250 of those benefits each year." This knowledge could be obtained from the proof tree of the query.

Simple Scenario - In these types of questions, a user asserts some facts and performs a query which uses the facts as a premise. The facts which follow from the premise can be obtained by a simple knowledge-base lookup. An example is, "I am single and I own a house worth $200,000. What is my standard deduction?". Here, the system would assert the facts about being single and owning a house, and then query the system for the standard deduction.

Complex Scenario - These types of questions are similar to Simple Scenario questions in that the system asserts some premise facts before executing a query. The major difference is that the system will be required to perform some complex reasoning such as elaboration or backward chaining. An example of this type of question would be "I am single and I own a house worth $200,000. Should I take the standard deduction or itemize my deductions?".. This type of question would require simulating two scenarios, and comparing the results of each simulation.

Many systems which make use of a tax ontology will have a disproportionate burden of correctness due to the legal and financial consequences of errors. Therefore, it is absolutely critical that any system which reasons with tax knowledge support auditing capabilities. The Powerloom knowledge representation and reasoning system [5] is an example of one such system, since it has the ability to provide query justifications and the ability to debug failed queries [9].

7 Conclusions

In this paper, we have argued for the necessity of modeling the tax code. Doing so would enable a broad class of intelligent applications, including online help systems, tax auditing agents, and tax question answering systems. We proposed the construction of a rich tax ontology, and we provided a detailed example of how an IRC rule might be modeled. Several challenges were discussed, including the need for modeling massive amounts of world knowledge. Provided that these challenges can be met, the construction of a large-scale tax ontology would provide enormous benefits to society.

References

1. The Internal Revenue Code (2004). http://www.fourmilab.ch/ustax/ustax.html
2. GAO-02-711R IRS Budget Justification Options (2002). General Accounting Office Washington, DC. www.unclefed.com/GAOReports/d02711r.pdf
3. TurboTax. www.intuit.com
4. Genesereth, M. (1995). Knowledge Interchange Format specification. URL http://logic.stanford.edu/kif/specification.html
5. PowerLoom Knowledge Representation System (2004). http://www.isi.edu/isd/LOOM/PowerLoom/
6. FrameNet (2004). http://www.icsi.berkeley.edu/~framenet/
7. Cyc (2004). http://www.cyc.com
8. OpenMind (2004). http://www.openmind.org/
9. Chalupsky, H. (2002). WhyNot: debugging failed queries in large knowledge bases. In Proceedings of the Eighteenth national conference on Artificial Intelligence.
10. Mommers, L. (2002), Applied legal epistemology: building a knowledge-based ontology of the legal domain (Ph.D. thesis).
11. Valente, A., Russ, T., MacGregor, R. and Swartout, W. "Building, Using and Reusing an Ontology of Air Campaign Planning." IEEE Intelligent Systems, 14(1), pages 27—36, January/February 1999.
12. A. Valente and J. Breuker. "Legal Modelling and Automated Reasoning with ON-LINE." International Journal of Human-Computer Studies, 51(6), pages 1079-1125, December 1999.
13. A. Valente and J. Breuker. "Towards Principled Core Ontologies." In: B. Gaines and M. Musen, editors, Proceedings of the Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'96), pages 33/1—33/20, Banff, 1996.
14. McCarty, T. A language for legal discourse I. basic structures. In Proc. of the Second International Conference on AI and Law, pp. 180-189, Vancouver. ACM, 1989.
15. Valente, A. "Roles and Uses of Ontologies in AI and Law." In Law and the Semantic Web, Springer, 2004. In press.
16. Benjamins, V.R., Contreras, J., Casanovas, P. and Ayuso, M.. "Ontologies of Professional Legal Knowledge as the Basis for Intelligent IT Support for Judges". In: Proceedings of the ICAIL 2003 Workshop on Legal Ontologies & Web based legal information management, 2003.
17. Visser, P.S., Bench-Capon T.M.. "A Comparison of Four Ontologies for the Design of Legal Knowledge Systems". Artificial Intelligence and Law 6: 25-57, 1998.

Modelling Changes in Ontologies

Johann Eder and Christian Koncilia

University of Klagenfurt
Dep. of Informatics-Systems
{eder, koncilia}@isys.uni-klu.ac.at

Abstract. Ontologies are shared conceptualizations of certain domains. Especially in legal and regulatory ontologies modifications like the passing of a new law, decisions by high courts, new insights by scholars, etc. have to be considered. Otherwise, we would not be able to identify which knowledge (which ontology) was valid at an arbitrary timepoint in the past. And without this knowledge we would for instance not be able to identify why a user came to a specific decision.
In this paper we will show how a simple ontology description formalism, namely a directed graph, has to be extended to represent changing knowledge. Furthermore, we will present the operations that are necessary to manipulate such an ontology. Finally, we will discuss different implementation approaches.

1 Introduction

Ontologies are generally seen as a promising approach for adding semantics to data processing. Semantic Web, semantic web services, enterprise integration are but a few research avenues where ontologies are researched with high expectations.

An ontology is a shared conceptualization of a certain domain [Gru03]. It describes the concepts of a domain, their properties and their relationships. Much work has already been done to analyse multiple heterogeneous ontologies, their integration and their coexistence.

Surprisingly little attention was drawn to the fact that the reality an ontology describes and/or the view of the observers sharing the conceptualization on the reality may change. In legal and regulatory ontologies the passing of a new law, decisions by high courts, new insights by scholars, new entities in the real world which require a new or changing interpretation of legal concepts are changes which have to be considered.

There are three different basic approaches of how to deal with such changing knowledge. First of all, we could simply ignore modifications, and describe the world in a completely static way. Obviously, this approach is of limited suitability for real world applications. The second approach is a little bit more sophisticated: by adopting our knowledge description we always represent and store the most recent version of knowledge. This is the most frequent approach nowadays. It has the disadvantage that we lose knowledge about the past. If for example we

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 662–673, 2004.
© Springer-Verlag Berlin Heidelberg 2004

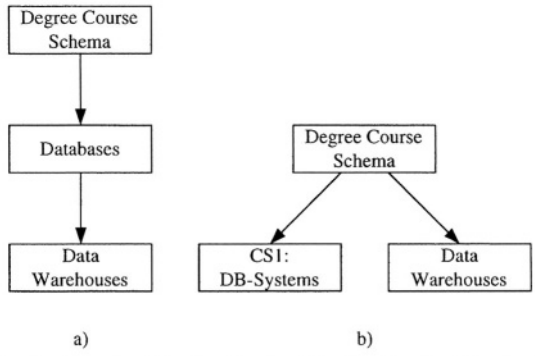


Fig. 1. Our running example represented as ontology graph

study an old court decision and we do not know the state of the law at *decision time* and at *case time,* we probably will not be able to understand the verdict. So only working with the most recent ontology is definitively not satisfactory. The third approach takes into account that the knowledge about modifications is again knowledge that may be important. In this approach we would have to describe different versions of knowledge and the relations between these versions. The last two approaches are well known in the temporal database community. The first one is called (Schema) Evolution, the latter one (Schema) Versioning [JD98].

During the last years several specification languages for ontologies have been developed. These includes DAML+OIL which is now going to become the Web Ontology Language OWL (both languages stem from the area of description logics), Loom, Open Knowledge Base Connectivity (OKBC), or Common Logics (CL).

In this paper we will discuss how a simple ontology description formalism, namely a directed graph, has to be extended to represent changing knowledge, and which extensions to such a "specification language" would be meaningful.

In order to achieve this goal the following extensions to an ontology description are necessary:

- **Temporal extension:** ontology data (classes and relations between classes) has to be time stamped in order to represent their valid time (the time in which a fact is true in the real world [JD98]).
- **Ontology versions:** by providing time stamps for ontology data we have to cope with different versions.
- **Ontology selection:** Our system has to support functions to select a specific version of an ontology.

The remaining parts of the paper are organized as follows: In section 2 we introduce a running example. In section 3 we discuss related work. After that, we will introduce our model of a versioning graph in section 4. Three different

implementation approaches will be briefly discussed in section 5. Finally, we will conclude in section 6.

2 Running Example

Through the rest of this paper we will use the following example to show that ontology versioning is vital for the correctness and expressiveness of ontologies and ontology queries: consider that we built an ontology that covers different degree course schemes at a university.

In 1990 the first version of a degree course schema for the Computer science study has been adopted. In this version, the database course was called "Databases". Furthermore, it was necessary to pass the exam of this course in order to take the "Data Warehouses" lecture. This first version is depicted in Fig. 1a).

Now consider that in 2000 a new version of this regulation became effective. In this new version the "Databases" has been renamed to "CS1: DB-Systems". Moreover, it is no longer required to pass this lecture in order to take the "Data Warehouses" lecture. This recent version is depicted in Fig. 1b).

If we would use a simple ontology evolution approach, i.e., if we would represent and store only the most recent version of our ontology, we could get problems with applications or other ontologies that use our ontology. In the best case, these ontologies / applications would recognize that the ontology changed and report an error. In the worst case, the new version doesn't produce syntax errors in the application / ontology but leads to wrong results.

Moreover, there are many cases in which the knowledge that an ontology changed and how it changed is not enough. In fact, we often need to know during which time interval a specific ontology version was valid! Just imagine that a student took "Databases" after he took "Data Warehouses". In order to answer questions like "was this correct according to the regulations valid in 1995?" we would have to timestamp each ontology version.

3 Related Work

Our approach builds on the techniques developed for temporal databases, schema evolution and schema versioning of databases [FGM00, JD98]. However, as shown and discussed in [NK03b] these approaches designed for database systems cannot be directly applied to ontologies. The authors of [NK03b] argue, that the main reasons for this are: (1) ontologies are both at the same time, schema and instances. (2) In contrast to (relational) database systems ontologies themselves incorporate semantics. (3) Ontologies are more often reused. (4) Ontologies lack of centralized control mechanisms as they are de-centralized by nature. (5) The data model of an ontology is usually richer. (6) Finally, they argue that in ontologies there is no clear distinction between classes and instances.

In [NK03a] the same authors present their framework for ontology evolution. In this paper the authors focus mainly on ontology evolution and not on ontology

versioning. As a consequence they do not provide any information about the valid time of a specific ontology. They discuss different change mechanisms and present an ontology of change operations.

In [KOF02] an ontology versioning methodology is presented. The authors present a tool (called OntoView) that helps the user to compare different versions of an ontology. Therefore, the systems analyzes changes between two version on a structural level. After that, the user may specify relations between the ontology versions, i.e., he may specify whether two concepts are "identical" or if a "conceptual change[1]" took place. The main focus of this paper is how to detect what changed from one version of an ontology to another one. Again, information about the valid time has not been considered.

4 Ontology Versioning Graph

A good definition of what an ontology is has been given in [Gru03]. In this paper, the author defines an ontology as an explicit specification of a conceptualization of a domain. This explicit specification may be specified by using one of the languages mentioned above, e.g., DAML+OIL, OWL or CL. Another possibility to specify such a conceptualization is to use a graph where nodes represent concepts, and edges represent the relations between two concepts [MWK00]. Figures 1 a) and 1 b) are examples for ontology graphs.

Until now we described intuitively how an ontology may be represented as a graph. We will now extend this description to define a temporal extension that supports valid time. Therefore, we have to introduce:

- **Time model:** Our model uses a linear and discrete model of time. This means that each point in time (each instant[2]) can be mapped to an integer, i.e., the model is isomorphic to the natural numbers, and each time point has a single successor. Furthermore, A chronon Q is defined as "a non-decomposable time interval of some fixed, minimal duration" [JD98]. In our model, a chronon is the smallest unit of time. Or, in other words, the time axis is a series of chronons.
- **Valid Time:** The term *Valid Time* is well-known in the temporal database community. It defines the time, in which a fact (in our model a fact may be both, a class or the relation between two classes respectively) is true in the real world. A fact may have more than one time points or time intervals during which it is true in the modelled world. [JD98]
- **Time Intervals:** In order to introduce valid time in an ontology all classes and all relations between these classes may exist in several versions. Each version must have a time interval $[T_s, T_e[$ representing the valid time where

[1] According to the authors a conceptual change is "a change in the way a domain is interpreted..."

[2] We use the term Instant as defined in [JD98]: "An instant is a time point on an underlying time axis."

T_s is the beginning of the valid time, T_e is the end of the valid time. We represent that a fact is valid until now by $T_e = \infty$.
Please note that we use the syntax $[A, B[$ to represent a half-closed interval. In this half-closed interval the instant A is included, whereas the instant B is excluded.

Until now we intuitively defined our model of an ontology versioning graph. We will now give a formal definition of such an ontology versioning graph.

Definition 1 (Ontology Versioning Graph Definition): *A ontology versioning graph G is defined as a tupple* $G = (\mathbb{N}, \mathbb{E})$ *where* $\mathbb{N}$ *is a set of classes C and* $\mathbb{E}$ *is a set of relations R between these classes.*

Definition 2 (Class Definition): *A class C is defined as a triple* $C = (label, VT, S)$ *where label is the label of the class (usually a noun representing a concept), VT is a tuple* $[T_s, T_e[$ *representing the valid time of the class, and S is a set of slots (attributes and properties assigned to a class).*

Definition 3 (Relation Definition): *A relation R is defined as a n-tuple* $R = (C_f, C_t, type, VT)$ *with* $C_f \in \mathbb{N} \wedge C_t \in \mathbb{N}$ *where* C_f *represents the class from which the relation leads to the class* C_t*, type represents the the type of the relation (i.e., InstanceOf, SubclassOf, ...) and VT is a tuple* $[T_s, T_e[$ *representing the valid time interval of the relation.*

Figure 2 a) shows an example of such a ontology versioning graph, where nodes represent the classes *Databases, Data Warehouses, Degree Course Schema* and *CS1: DB-Systems* and edges represent relations between these classes. As can be seen, each node and each edge has a time stamp representing its valid time. For sake of readability, we did not depict different types of relations and slots in this example.

4.1 Versioning Operations

Some basic operations have to be defined in order to manipulate a ontology versioning graph *G*. These operations enable us to insert, update and delete classes, or nodes respectively and to insert, update and delete relations, or edges respectively. They are defined as follows:

1. *Insert Class:* inserts a new class C' which is valid at a given time interval $I = [T_s, T_e[$ into a given graph $G = (\mathbb{N}, \mathbb{E})$. This operation results in a graph $G' = (\mathbb{N}', \mathbb{E})$ where $\mathbb{N}' = \mathbb{N} \cup C$.
2. *Update Class:* updates a class C which is an element of $\mathbb{N}$ of a given graph $G = (\mathbb{N}, \mathbb{E})$ to C' at an instant T. This operation results in a graph $G' = (\mathbb{N}', \mathbb{E})$where $\mathbb{N}' = \mathbb{N} \cup C'$. The valid time of C' is $[T, \infty[$ and the valid time of C is $[T_s^C, T[$ where T_s^C is the start of the valid time of C as it was before the update operation.
   This operation should not be confused with the `UPDATE` operation known from relational database systems. In fact updating a class in a temporal sense means to create a new version of the class concerned, e.g., because an attribute of that class changed, or an attribute of that class was deleted, etc.

3. *Delete Class:* "deletes" a class C (whose ending valid time equals ∞, i.e., has not been set yet) which is an element of $\mathbb{N}$ of a given graph $G = (\mathbb{N}, \mathbb{E})$ and sets its ending valid time to T. This operation results in a graph $G' = (\mathbb{N}', \mathbb{E}')$ where $\mathbb{N}' = \mathbb{N} \setminus C \cup C'$ and $C' = C$ except that the end of the valid time of C' is set to T. Formally $C' = (C.label, [C.T_s, T[, C.S)$.
   Furthermore, we have to set the end of the valid time of all relations that lead to or from class C. Hence, if $\mathbb{E}_c$ is the set of all relations that lead to or from class C ($\mathbb{E}_c = \{x | x \in \mathbb{E} \bullet x.C_f = C \vee x.C_t = C\}$), and $\mathbb{E}'_c$ is the set where the end of the valid time of all relations from or to C has been set to T then $\mathbb{E}' = \mathbb{E} | \mathbb{E}_c \cup \mathbb{E}'_c$.
   Again, this operation should not be confused with the `DELETE` operation known from traditional relational database systems. In fact, this operation does not delete anything - it sets the end time of the valid time of a class.
4. *Insert Relation:* inserts a new relation R which is valid at a given time interval $I = [T_s, T_e[$ into a given graph $G = (\mathbb{N}, \mathbb{E})$. This operation results in a graph $G' = (\mathbb{N}, \mathbb{E}')$ where $\mathbb{E}' = \mathbb{E} \cup R$.
5. *Update Relation:* updates a relation R which is an element of $\mathbb{E}$ of a given graph $G = (\mathbb{N}, \mathbb{E})$ to R' at an instant T. This operation results in a graph $G' = (\mathbb{N}, \mathbb{E}')$where $\mathbb{E}' = \mathbb{E} \cup R'$. The valid time of R' is $[T, \infty[$ and the valid time of R is $[T_s^C, T[$ where T_s^C is the start of the valid time of R as it was before the update operation.
   This update operation can be used to change the type of a relation, e.g., to change a relation of type *PartOf* to a *SubclassOf* relation.
6. *Delete Relation:* "deletes" a relation R which is an element of a given graph $G = (\mathbb{N}, \mathbb{E})$ and sets its ending valid time to T. The ending valid time of R has to be equal to ∞. This operation results in a graph $G' = (\mathbb{N}, \mathbb{E}')$ where $\mathbb{E}' = \mathbb{E} \setminus R \cup R'$ and $R' = R$ except that the end of the valid time of R' is set to T. Formally $R' = (R.label, [R.T_s, T[, R.S)$.

4.2 Integrity Constraints for a Ontology Versioning Graph

To bring in the concept of temporality into an ontology has several consequences - one is, that some constraints have to be fulfilled in order to guaranty the integrity of our model. In this paper, we will not fucus on a detailed description and definition of all integrity constraints. However, we have to define some constraints that we will need later on in this paper.

A basic constraint is that for all time stamps $[T_s, T_e[$ in our model $(T_s \leq T_e) \vee (T_e = \infty)$ has to be true, i.e., a class may not end before it starts.

Furthermore, the valid time of a relation $r = (C_f, C_t, type, VT)$ of a graph $G = (\mathbb{N}, \mathbb{E})$ between two classes C_f and C_t has to be within (to exist in) the valid time of both classes.

$$\forall r \in \mathbb{E} : exists_in(r, C_f) \wedge exists_in(r, C_t) \tag{1}$$

where $exists_in(X_i, X_j)$ describes that the time interval representing the valid time of temporal component X_i is a subset of the time interval representing the

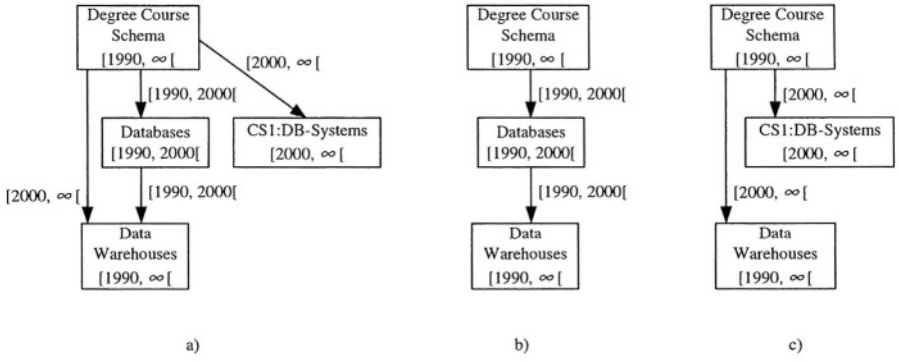


Fig. 2. a) A ontology versioning graph and its deduced versions b) and c)

valid time of temporal component X_j. A *temporal component* is a node or an edge in a ontology versioning graph $G = (\mathbb{N}, \mathbb{E})$.

Formally, we can define $exists_in(X_i, X_j)$ as follows:

$$exists_in(X_i, X_j) = \begin{cases} True & \text{if } \forall t \in [T_s^{X_i}, T_e^{X_i}[\bullet t \in [T_s^{X_j}, T_e^{X_j}[\\ False & \text{otherwise} \end{cases} \quad (2)$$

4.3 Selecting a Specific Ontology Version

The graph defined in section 4 consists of all possible ontology versions. Or, in other words, we do not define several ontologies where each ontology represents a version of an ontology. In fact, we define a single ontology which is composed of all ontology versions.

Figure 2 a) shows the ontology versioning graph for our running example. As can be seen, this ontology versioning graph consists of two versions b) and c). In this example, version b) is valid during the interval [1990, 2000[and version c) with a valid time $[2000, \infty[$. In this example the chronon (as defined in section 4) is a year. Hence, [1990, 2000[represents that this version is valid from 1990 until 1999 (2000 is not included as we use half-closed intervals).

Intuitively we can say that if we represent all timestamps $[T_s, T_e[$ of all temporal components within our ontology versioning graph on a linear time axis, the interval between two consecutive timestamps on this axis represents the valid time of an ontology version.

Formally, we can define such an ontology version $G(T)$ (an ontology versioning graph valid at instant T) over a graph $G = (\mathbb{N}, \mathbb{E})$ as follows:

$$G(T) = (\mathbb{N}_\mathbb{T}, \mathbb{E}_\mathbb{T}) \quad (3)$$

where

$$\mathbb{N}_{\mathbb{T}} = \{x | x \in \mathbb{N} \wedge x.T_s \leq T \leq x.T_e\} \quad (4)$$

and

$$\mathbb{E}_{\mathbb{T}} = \{x | x \in \mathbb{E} \wedge x.T_s \leq T \leq x.T_e\} \quad (5)$$

Intrinsically, we should define that all nodes (classes) referenced by edges (relations) have to be valid in the selected ontology version $G(T)$. However, there is no need to check whether both classes C_f and C_t of all edges $R = (C_f, C_t, type, VT)$ are valid within the selected ontology version. In fact, this constraint can be deduced from the constraint (1) as defined in section 4.2. We will give a proof of this in appendix A.

This leads us to the definition of a stable interval. Intuitively, we can say that such a stable interval is a view defined on an ontology versioning graph that is valid for a given time interval $[T_s, T_e[$. All classes and relations within this ontology versioning graph are also valid for the given time interval. In other words, within such a stable interval there cannot exist different versions of classes or relations. In the example shown in Fig. 2 we have two stable intervals: the first is valid during the interval [1990, 2000[, the second one during the interval [2000, ∞[.

Formally, we define a stable interval as follows: Let $\mathbb{T}$ be the set of all instants of changes, i.e., $\mathbb{T} = \{x.T_s | x \in \mathbb{N} \vee x \in \mathbb{R}\} \cup \{x.T_e | x \in \mathbb{N} \vee x \in \mathbb{R}\}$. For the example shown in Fig. 2 a) $\mathbb{T}$ would be $\mathbb{T} = \{1990, 2000, \infty\}$. Then a stable interval is a interval $[T_s, T_e[$ between to contiguous instants T_s and T_e of the set $\mathbb{T}$. Two instants T_1 and T_2 (with $T_1 \leq T_2$) are contiguous if there exists no instant T such that $T_1 < T < T_2$.

5 Implementation Approaches

In general, our approach for ontology versioning can be implemented in three different ways:

- **Meta-Ontology:** The basic idea of the first approach is depicted in Fig. 3. In this approach a meta-ontology is used to store the valid time of different ontology versions and to deal with inter-structure relationships between these ontology versions. The main advantage of this approach is that any ontology description language (for instance, OWL) can be used to define the different ontology versions.
- **Standard Extension:** In this approach a temporal extension for a standard representation language for ontologies has to be developed. The main advantage of this approach is that a uniform standard produces uniform solutions. Hence, such a standard could be defined in a way in which for instance temporal reasoning is supported.
- **Using Standard:** The last approach uses the techniques that a standard like OWL, or DAML+OIL provides. In contrast to an extension of the standard this approach will result in different solutions. Hence, it cannot be guaranteed that for instance reasoning algorithms take into account the temporal aspects of an ontology in a correct way.

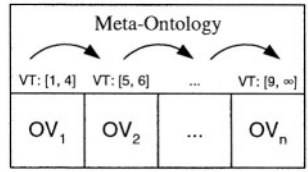


Fig. 3. Using a Meta-Ontology over ontology versions $OV_1, ..., OV_n$

We will now briefly discuss how OWL may be used to implement an ontology as presented in section 4. We would like to emphasize that the implementation discussed here is only one possible (straight forward) implementation. Other approaches are of course feasible. Moreover, we will see that using OWL to represent valid time has its drawbacks.

OWL is a ontology representation language defined by the W3C Web Ontology Working Group. In contrast to other languages like for instance RDF (Resource Description Framework) OWL provides more expressiveness [AvH04].

However, the support for ontology versioning is very limited in OWL. A tag `versionInfo` may be used to describe the current version of the ontology, for instance by using RCS/CVS keywords. Nevertheless, this tag doesn't contribute to the logical meaning of the ontology at all. The following example how this `versionInfo` tag may be used has been taken from [Wor02]:

```
<Ontology rdf:about="">
  <versionInfo>$Id: Overview.html,v 1.4 2002/07/31
               19:44:09 henri Exp $</versionInfo>
  <rdfs:comment>An example ontology</rdfs:comment>
  <imports rdf:resource="http://www.w3.org/2002/07/owl"/>
</Ontology>
```

Furthermore, two tags may be used to indicate whether or not two ontologies are compatible: `owl:backwardCompatibleWith` and `owl:incompatibleWith`.

The tags `owl:DeprecatedClass` and `owl:DeprecatedProperty` may be used to indicate that a class or property is likely to change.

We will now discuss how OWL may be used to incorporate time stamps, i.e., versioning information, into an ontology. In order to achieve this we have to time stamp the nodes and edges of our graph model, i.e., the `Class` and `ObjectProperty` (a binary relation between two classes) elements in OWL.

First we have to define a `DatatypeProperty` (a binary relation between classes and datatypes) that represents the beginning of the valid time interval, and one to represent the end of the valid time interval. The following statement defines such a `DatatypeProperty` called `vtStart` whose range is a date:

```
<owl:DatatypeProperty rdf:ID="vtStart">
 ...
 <rdfs:rangerdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
</owl:DatatypeProperty>
```

Please note that "..." indicates that there is additional text. The full example can be downloaded at `http://www.ifi.uni-klu.ac.at/Publications`.

Now, we can use this property to define the valid time interval of classes. This can be done by using `owl:hasValue` to "restrict" the class to a specific value that the property `vtStart` must have. For instance, we can use the following to specify the beginning of the valid time interval (January, 1st 1999) for the class `DegreeCourseSchema`:

```
<owl:Class rdf:ID="DegreeCourseSchema">
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#vtStart"/>
         <owl:hasValue rdf:datatype="&xsd;date">
         1990-01-01
         </owl:hasValue>
      </owl:Restriction>
      ...
```

Something similar can be used to describe the valid time of a relation between two classes, i.e., an `ObjectProperty` in OWL.

According to the integrity constraint defined in section 4.2 the valid time of a relation between two classes has to be within the valid time of both classes. Usually, the starting time of the valid time interval of the relation will be equal to the starting time of one of the classes. Accordingly the ending time of the relation is usually equal to the ending time of one of the classes. If so, we could use the `isDefinedBy` tag to define this relation between valid times:

```
 ...
<rdfs:isDefinedBy>
   <Databases rdf:ID="620.202">
     <ValidTime rdf:datatype="&xsd;date"
     >2000-01-01</ValidTime>
   </Databases>
 </rdfs:isDefinedBy>
```

One drawback of the example given above is that the valid time of the relation is in fact bound to the valid time of an instance of the class and not the class itself.

Another approach would be to use the `versionInfo` tag mentioned before:

```
<owl:ObjectProperty rdf:ID="attendanceOrder">
  <rdfs:domain rdf:resource="Databases"/>
  <rdfs:range rdf:resource="DataWarehouses"/>
  <owl:versionInfo rdf:datatype="&xsd;date"
  >vtStart: 1990-01-01</owl:versionInfo>
  ...
```

All these techniques to incorporate valid time into OWL ontologies have their drawbacks. The first drawback is that such an ontology would not support reasoning as long as the time dimension is not considered in a correct way. For instance, such a temporal reasoning algorithm should consider that if $A \rightarrow B$ (where $A \rightarrow B$ means "A semantically implies B") and $B \rightarrow C$ then $A \rightarrow C$ is only true if there exists at least one timepoint which is within the valid time of both, A and C.

Another drawback is, that OWL would not guarantee that all integrity constraints are considered in a correct way. For example, the valid time of a relation between two classes has to be within the valid time of both classes.

Taking all together we conclude that all these techniques are "extralogical". This means that they don't incorporate any additional (temporal) semantics that maybe used by a reasoning algorithm or any other application.

6 Conclusions

We have shown a novel approach to represent changes in ontologies. By introducing information about the valid time of concepts represented in ontologies, we are able to identify which knowledge (which ontology) was valid at an arbitrary timepoint in the past. This enables a user of an ontology to understand a verdict if, for instance, he studies an old court decision and doesn't know the state of the law at *decision time* and at *case time*.

Incorporation of other time dimensions will be one of the main research avenues that we will investigate in the near future. This includes for instance *transaction time* (the time when a fact is current in the database [JD98], i.e., in the system). For instance, this time dimension is important when we would like to deal with regulations and laws that become effective retroactively.

References

[AvH04] G. Antoniou and F. van Harmelen. Web Ontology Language: OWL. In *Handbook on Ontologies,* pages 67–92. Springer-Verlag, 2004. In [SS04].

[EJS98] O. Etzion, S. Jajodia, and S. Sripada, editors. *Temporal Databases: Research and Practise.* Springer-Verlag (LNCS 1399), 1998.

[FGM00] E. Franconi, F. Grandi, and F. Mandreoli. Schema Evolution and Versioning: a Logical and Computational Characterisation. In *Workshop on Foundations of Models and Languages for Data and Objects,* 2000.

[Gru03] T. Gruber. A Translation Approach to Portable Ontology Specification. In *Knowledge Acquisition 5(2): 199-220.* World Wide Web Consortium (W3C), 2003.

[JD98] C. S. Jensen and C. E. Dyreson, editors. *A consensus Glossary of Temporal Database Concepts - Feb. 1998 Version,* pages 367–405. Springer-Verlag, 1998. In [EJS98].

[KOF02] M. Klein, A. Kiryakov D. Ognyanov, and D. Fensel. Ontology Versioning and Change Detection on the Web. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02),* 2002.

[MWK00] P. Mitra, G. Wiederhold, and M. Kersten. A Graph-Oriented Model for Articulation of Ontology Interdependencies. In *Proceedings Conference on Extending Database Technology 2000 (EDBT'2000), Konstanz, Germany, 2000,* volume LNCS: 1777, pages 86+, 2000.

[NK03a] N. Noy and M. Klein. A component-based framework for ontology evolution. In *Proceedings of the Workshop on Ontologies and Distributed Systems (IJCAI'03),* 2003.

[NK03b] N. Noy and M. Klein. Ontology Evolution: Not the Same as Schema Evolution. In *Knowledge and Information Systems,* volume 5, 2003.

[SS04] S. Staab and R. Studer, editors. *Handbook on Ontologies.* Springer-Verlag (ISBN 3-540-40834-7), 2004.

[Wor02] World Wide Web Consortium. *OWL Web Ontology Language 1.0 Reference, W3C Working Draft 29 July 2002.* World Wide Web Consortium, 2002. URL: http://www.w3.org/TR/2002/WD-owl-ref-20020729/.

A Proof of Theorem

Let $G(T) = (\mathbb{N}_\mathbb{T}, \mathbb{E}_\mathbb{T})$ be an ontology version (an ontology versioning graph valid at time point T) over a graph $G = (\mathbb{N}, \mathbb{E})$ such that $\mathbb{N}_\mathbb{T} = \{x | x \in \mathbb{N} \wedge x.T_s \leq T \leq x.T_e\}$ and $\mathbb{E}_\mathbb{T} = \{x | x \in \mathbb{E} \wedge x.T_s \leq T \leq x.T_e\}$. Q is the defined chronon.

If $\forall r \in \mathbb{E} : exists_in(r, C_f) \wedge exists_in(r, C_t)$ (as defined in section 4.2) is true, than from this it follows that $\forall T \in \{r.T_s, \ldots, r.T_e - Q\} \bullet T \in \{C_f.T_s, \ldots, C_f.T_e - Q\} \wedge T \in \{C_t.T_s, \ldots, C_t.T_e - Q\}$ □

An Ontology-Based Model for Representing "Bundle-of-Rights"

Maria-Teresa Sagri[1], Daniela Tiscornia[1], and Aldo Gangemi[2]

[1]ITTIG (Institute for Theory and Techniques for Legal Information)
Italian Research Council, Florence, Italy
{Sagri, Tiscornia}@Ittig.cnr.it
[2]ISTC (Institute of Cognitive Science and Technologies)
Italian Research Council, Rome, Italy
a.gangemi@istc.cnr.it

Abstract. This paper describes an application of the lexical resource JurWordNet and of Core Legal Ontology as a descriptive vocabulary for modeling legal domains. The two resources can be viewed as a repository of structured knowledge aimed at supporting the modeling of normative domains. In our ontological approach legal entities are represented as conceptual units and their interrelations are described inside the same language. Besides practical applications, these new forms of representation allow exercises of computational jurisprudence, in that they consent to formally analyze phenomena and aspects of juridical factuality for which the general theory of law has already created models and theories. This contribution aims to present a preliminary analysis of the relation among regulative units and to investigate how further aspects linked to the description of parts of norms and complex concepts can be represented in an ontology-based framework. As a case study, in the field of the Intellectual Property Right (IPR) management, the representation of click-on licences for re-using Public Sector Information is presented.

1 Introduction

The use of ontology-based methodologies has greatly expanded in recent years, and, as a consequence, the term 'ontology' has taken on a wide range of meanings. One of the distinctions that is most commonly accepted is that between *Semantic lexicons* (also called lightweight ontologies, Hirst G. 2003) and *Formal ontologies*.

We have developed a legal semantic lexicon, JurWordNet [Miller 1995, Sagri 2003], as a support in legal information searching. Leaf concepts are structured according to taxonomic and semantic relationships based on linguistic rules, and high level concepts have been framed and organized via a Core Legal Ontology (CLO) in order to remove terminological and conceptual ambiguities [Gangemi, Sagri and Tiscornia 2003]. CLO requires that cognitive assumptions underlying the meaning of concepts are made explicit and formally defined in a wider dimension, thus CLO is a Law-oriented specialization of DOLCE, a formal foundational ontology that provides a powerful and logically-sound reusable component for ontology design. The lexical

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 674–688, 2004.
© Springer-Verlag Berlin Heidelberg 2004

repository and the core ontology constitute a descriptive vocabulary for modeling specific legal domains and provide a new approach to the formal representation of legal knowledge.

In formal ontologies both concepts and relations of the domain of discourse are defined through universal properties and metaproperties that are explicitly represented and axiomatized.

Ontology-based methodologies focus their attention on an objective way of describing entities that populate the world of Law. Unlike traditional neopositivistic approaches [Comanducci 1999], DOLCE formal ontology shares the assumption [cf. Searle 1995] that non physical (social) entities - such as organizations, roles, and regulations - can be described in the same terms as physical objects, considering them as pertaining to the same universe of discourse.

The paper is structured as follows:

Sect. 1 introduces the basic theoretical framework underlying the ontology-based methodology and the components of this model. Some of the main structural aspects of the methodology will be described:

- the Legal World interpretation according to the basic assumptions of the DOLCE foundational ontology and its extensions
- the main classes of concepts in the Core Legal Ontology
- the lexical extension through the synsets in JurWordNet.

Sect. 2 describes some cases of relations among normative statements: in common use, one tends to consider a regulatory text (a law or an agreement) as a collection of atomic norms, and rights, powers, and obligations as simple, independent and indivisible, while in fact norms are complex objects. The licences regulating copyright are a clear example, as they collect interrelated permits, obligations, and restrictions. As a case study, we refer to the *click-on* licenses enacted by the UK government for Public Sector Information (PSI) handling.

2 An Ontology-Based Characterization of Law

The Core Legal Ontology (CLO) is a joint project carried together with the ISTC-CNR. CLO is a specialization of DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) foundational Ontology [Masolo et al., 2003].

The four basic categories of DOLCE are *endurant* (including object- or substance-like entities, either physical or non-physical), *perdurant* (including state- or process-like entities), *quality* and *region* (including dimensional spaces of attributes such as time, geographical space, color, etc.). DOLCE includes several primitive relations, such as *part, connection, constituency, inherence* of qualities in entities, *participation* of endurants in perdurants, etc. We refer to DOLCE documentation for a full description of DOLCE top categories.

In DOLCE extended with the theory of Descriptions and Situations (D&S), a new top-category *situation* has been added. D&S is an extension of DOLCE aimed at providing a theory that supports a first order manipulation of contexts, theories and models. The basic assumption in D&S is that cognitive processes create descriptive

entities that allow cognitive agents to build a rational account of the external world, and to behave accordingly when interacting with the environment and with other agents. Social objects like *roles, organizations, rules, plans,* etc. are typical descriptive entities.

In the formal framework of D&S [Gangemi and Mika, 2003, Masolo et al., 2004, Gangemi et al., 2004], the following predicates are introduced (only an excerpt of their axioms is included here):

(i) a **Description** is a social object, which represents a conceptualization, hence it is generically dependent (GD) on some agent, and is communicable [Masolo et al., 2004]. Like physical objects, non-physical ones have a lifecycle, can have parts, etc. Differently from physical objects, non-physical ones are dependent on some agentive physical object that is able to *conceive* them. Descriptions have typical components, called *concepts.* Example of descriptions are regulations, beliefs, desires, plans, laws, diagnoses, projects, plots, techniques, system specifications, ecosystem rules, product designs, etc. [for an explanation of such a variety, and of respective differences, cf. Gangemi et al., 2004].

(ii) a **Situation** is an entity that appears in the domain of an ontology only because there is a description whose components can "carve up" a view *(setting)* on that domain. A situation aims at representing the referent of a "cognitive disposition" towards a world, i.e. the willingness, expectation, desire, belief, etc. to carve up that world in a certain way. Consequently, a situation has to *satisfy* a description. Examples of situations, related to the examples of descriptions above, are: facts, desired states, plan executions, legal cases, diagnostic cases, attempted projects, performances, technical actions, system functioning, ecosystems, finished working products, etc.

(iii) a **Concept** is also a social object, which is **defined by** a description. Once defined, a concept can be **used in** other descriptions. The **classifies** relation relates concepts and entities (then possibly even concepts). There are several kinds of concepts reified in D&S, the primary ones (**role, course,** and **parameter**) being distinguished by the categories of entities they classify in DOLCE:

- $\text{Concept}(x) \rightarrow \exists y.\ \text{Defines}(y,x) \wedge \text{Description}(y)$
- $\text{Role}(x) =_{df} \text{Concept}(x) \wedge \forall y.\ \text{Classifies}(x,y) \rightarrow \text{Endurant}(y)$
- $\text{Course}(x) =_{df} \text{Concept}(x) \wedge \forall y.\ \text{Classifies}(x,y) \rightarrow \text{Perdurant}(y)$
- $\text{Parameter}(x) =_{df} \text{Concept}(x) \wedge \exists y.\text{Classifies}(x,y) \wedge \forall y.\text{Classifies}(x,y) \rightarrow \text{Region}(y)$

Examples of roles are: *manager, student, assistant, actuator, toxic agent,* etc. Examples of courses are *routes, pathways, tasks,* etc. Examples of parameters are: *speed limits, allowed colors* (e.g. for a certain book cover), *temporal constraints,* etc.

(iv) **Figures,** or **social individuals** (either agentive or not) are also social objects, defined by descriptions, but differently from concepts, they do not classify entities:

- $\text{Figure}(x) \rightarrow \text{SocialObject}(x)$
- $\text{Figure}(x) \rightarrow \exists y.\ \text{Description}(y) \wedge \text{Defines}(y,x)$

Examples of figures are organisations, political geographic objects, sacred symbols, etc.

(v) **Agentive figures** are those which are assigned (agentive) roles from a society or community; hence, they can *act* like a physical agent:

- AgentiveFigure(x) → Figure(x) ∧ ∃y,z,w. Description(y) ∧ Role(z) ∧ Description(w) ∧ y≠w ∧ Defines(y,z) ∧ Defines(w,x) ∧ Classifies(z,x)

Typical agentive figures are societies, organizations, and in general all socially constructed persons.

(vi) Figures are not dependent on roles defined or used in the same descriptions they are defined or used, but they can act because they **depute** some powers to some of those roles. In other words, a figure selected by some agentive role can play that role because there are other roles in the descriptions that define or use the figure. Those roles select endurants that result to **enact** the figure:

- DeputedBy(r,f) → Role(r) ∧ Figure(f) ∧ ∃d. Description(d) ∧ Uses(d,r) ∧ Uses(d,f)
- DeputedBy(r,f) → ∃r_1. Role(r_1) ∧ Classifies(r_1,f)
- Enacts(e,f) → ∃r. Role(r) ∧ DeputedBy(r,f) ∧ Classifies(r,e)

In complex figures, like organizations or societies, a *total enactment* is possible when an endurant plays a *delegate,* or *representative* role of the figure.

(vii) Since descriptions and concepts are (non-physical) objects, they can also be selected by a role in another description. This recursivity allows to manage meta-level descriptions in D&S (e.g. a *norm* for enforcing norms will define a role that can select the *enforced norm*).

(viii) The *classifies* relation is specialized by three subrelations: **played by, sequences,** and **valued by,** for three different categories in DOLCE (Endurant, Perdurant, and Region)[1]:

- PlayedBy(x,y) =$_{df}$ Role(x) ∧ Endurant(y) ∧ Classifies(x,y)
- Sequences(x,y) =$_{df}$ Course(x) ∧ Perdurant(y) ∧ Classifies(x,y)
- ValuedBy(x,y) =$_{df}$ Parameter(x) ∧ Region(y) ∧ Classifies(x,y)

(ix) Roles or figures, and courses are related by relations expressing the **attitudes** that roles or figures can have **towards** a course:

- AttitudeTowards(x,y) → (Role(x) ∨ Figure(x)) ∧ Course(y)

Attitude towards is the descriptive counterpart of the 'participant-in' relation used in the ground ontology, i.e. attitudes are *participation modes.*

In other words, the AttitudeTowards relation can be used to state *beliefs, attitudes, attention* or *subjection* that an object can have wrt an action or process.

For example, a person is usually *obliged* to drive in a way that prevents hurting otherpersons. Or a person can have the *right* to express her ideas. Another, more complex example: a BDI application to a certain ordered set of tasks including initial conditions (beliefs), final conditions (desires), and ways to reach goals (intentions). In other words, moving from beliefs to goals is a way of bounding one or more agent(s) to a sequence of actions. In the plan ontology this intuition is deepened considerably.

[1] Only three categories from DOLCE have been assigned a concept type at the descriptive layer, because the resulting design pattern is simpler, and no relevant knowledge seems to be lost, at least in applications developed until now.

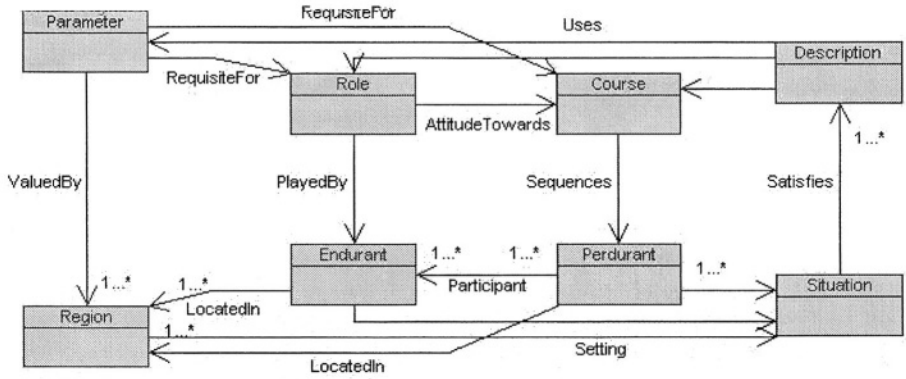


Fig. 1. The D&S Design Pattern as a UML class diagram. The lower part of the pattern is called the *ground ontology,* the higher is called the *descriptive ontology;* a situation satisfies a description if the two parts match according to specified rules. Part of the structure matching expected by situations satisfying descriptions appears symmetrically from the overall shape.

(x) Parameters and roles, figures, or courses are related by a **requisite for** relation, expressing the kind of requisites entities that are selected by roles or courses should have:

- $\text{RequisiteFor}(x,y) \rightarrow \text{Parameter}(x) \wedge (\text{Role}(x) \vee \text{Figure}(x) \vee \text{Course}(y))$

Requisites are constraints over the attributes of entities.

When a situation satisfies a description that uses parameters, endurants and perdurants that constitute the situation must have attributes that range within the boundaries stated by parameters (in DOLCE terms, entities must have qualities that are mapped to certain value ranges of regions).

For example, a *speed limit of 60kmph* can be a requisite for a *driving* task; the satisfying situation will have to constrain any *speed* of e.g. *travelling within Rome by car* to be less or equal to *60kmph.*

According to this approach, the legal world is conceived as a representation, or a description of reality, an ideal view of the behavior of a social group, according to a system of rules that is generally accepted and acknowledged.

Following DOLCE+D&S, a **norm** is a *legal description* composed of *legal roles, legal courses* of events, and *legal parameters* on entities that result to be bound to *the setting* created by *a legal case.* Between *legal descriptions* and *cases* a *satisfaction* relation holds, and through it, state of affairs relevant for law are selected: a case is such because there is a law that envisages it, even though more than one way of describing it is possible. Judgments describe in general terms instances of state of affairs that, as a result, are situations. Norms can be either *regulative* or *constitutive* [Searle, 1995, Gangemi et al., 2003].

Legal Roles are a sub-set of social roles. They are typically defined by a *constitutive norm.* Legal roles are *played* by legal subjects (either natural or legally-constructed persons), and *bound* to *legally-constructed institutions.* The role of legal agent depends on physical person and implies the legal subject, but not the contrary, as Institutional Agents act through their physical representatives.

Institutional Agents, or Legally-constructed institutions are social Individuals (e.g. organizations, public bodies). Like Social roles, they are defined by constitutive norms, but, unlike legal roles, they do not classify legal subjects. In many cases the same description defines both an institutional agent (e.g. Ministry) and a legal role (Minister) as a *representative* for the individual that can classify legal subjects. In other words, the identity of legally-constructed individuals is provided by themselves, while the identity of a legal role requires a legal subject that is classified by it at a certain time.

Modal Descriptions are *proper parts* of regulative norms that contain some *attitude towards* relation between legal roles (legal agents involved in the norm) and legal courses of events (descriptions of actions to be executed according to the norm). The classification of Modal Descriptions is based on Hofheld's *Theory of basic conception* and on the *Theory ofnormative positions* [Lindhal, 1977][Kanger, 1972].

Legal Facts *(cases)* are situations satisfying norms (only facts relevant for legal systems are legal facts). Legal facts usually contain as elements of their setting:

- *Natural facts* (e.g. phenomena like *death*), which are independent of human actions
- *Human facts,* depending on consciousness of the act, called *activities.* Among these:
  - *Legal acts* (in a strict sense), depending additionally on *will*
  - *Legal transactions,* depending on *intentionality*
  - *Institutional facts,* the legal counterpart of phenomena, legally enacted by *(satisfying)* constitutive rules.

Qualities *inherent in* Legal transactions are: *temporal qualities* such as duration (quality region are: deferment, expiration, term, etc) and the *validity-assessment quality* (valid, void, voidable).

3 JurWordNet

CLO concepts are specialised by the JurWordNet *synsets* (expressing lexical-oriented concepts). JurWordNet is an extension to the legal area of the Italian part of the *EuroWordNet* initiative: consistent with WordNet projects, the developing methodology favours the use and harmonization of already existing lexical resources: relevant concepts have been spotted *bottom-up* from the queries of legal information systems. From legal phrases, a taxonomy has been automatically created by using NLP techniques over the dictionary glossaries. A consolidated corpus of about 2000 synsets will be automatically increased through the link with thesauri and key words for legal databases until 3000 synsets coverage is reached.

Through the project *LOIS,* funded by the European e-Content program, the Italian legal network will be extended to five European languages (English, German, Portuguese, Czech, and Italian, linked by English). The cross-lingual linkage will be improved by one of the most interesting functions of the wordnet methodology, that is the distinction of meanings in polysemic terms. This aspect is of crucial importance in mapping terms between different languages; shifting emphasis from the linguistic expression to content allows comparing concepts through properties and meta-

properties, and to assess not only whether the concept itself occurs in different contexts, but also how the concept is instantiated in different regulatory structures.

During this first phase, the localisation methodology is based on the automatic mapping between already existing lexicons, considering that semantic connections between the concepts of a language can be mapped through the relationship between equivalent concepts in another language. This tests the lexicon coverage with respect to the domain and provides an initial base of conceptual equivalents. The preliminary results from a study on the lexicon of EU laws (via the Eurodicautom[2] database) out of the 2000 synsets of the Italian law lexicon 800 could be found in the German one, 470 in the Dutch, 490 in the Portuguese and 580 in the English one. The intersection with the Princeton WordNet showed 600 JurWordNet synsets in the 1800 English legal terms. A set of 350 synsets is shared by all lexica and can be considered as a preliminary *core* of the multilingual lexicon, whose building up is in progress. The CLO classes will act as an interlingua, enabling the core meaning of concepts to be interpreted and the consistency of the cross-lingual relations to be checked.

4 Norms About Norms, Parts of Norms, and Norm Components

In a social dimension, *Law* includes social and ethical rules, practices, and conventions. It is a complex, autonomous entity, which includes its self-recognition, constitution, and evaluation rules, [Hart, 1961] therefore it cannot be considered as a mere sum of norms.

In a strict (positivistic) sense, a legal norm is a (conceptualized) norm, *expressed by* a Normative Text that is *physically represented by* a Document.

On the other hand, a legal norm is defined in legal theory as the 'meaning of a prescriptive proposition'. Since we consider a legal text as a collection of prescriptive propositions and not as a sequence of linguistic expressions, we do not claim for a strict correspondence between a norm as conceptualized in our model and the segment of a legal text, as partitioned by the hierarchical structure of legislative acts (paragraphs, sub-paragraph, etc). In other words, we assume an anisomorphism between the *conceptual* and the *informational* structures of Law.

In a wider meaning, all normative propositions deductively inferred from a norm [Alchourr–n & Buligyn, 1971], including norms instantiated by a legal decision or stated by contract, can be considered norms. In our taxonomy the class *regulation* encompasses all entities which can be considered norms according this wide interpretation, suitable to be modeled in the DOLCE+D&S framework and CLO.

In D&S, norms as involved in some juridical context, may even be *satisfied by* purely juridical situations, as for norms that describe others norms: (e.g. amendments, law making norms, validity norms). This enables us to use this distinction to represent meta-juridical conceptualizations (meta-norms) as well, and to provide a unity criterion. According to the class they pertain to, norms may have *parts* (other descriptions) and *components* (defined or reused concepts and figures) such as:

---

[2] Eurodicautom is an aid for translators created by the European Commission http://europa.eu.int/eurodicautom/Controller

- Legal roles (defined by constitutive norms)
- Institutional agents (defined by constitutive norms)
- Institutional powers (defined by power-conferring norms)
- Behaviors (defined by regulative norms)
- Incrimination acts (situations satisfying incriminating norms)
- Cognitive states (presumptions: preconditions for other norms being satisfiable)

The two main classes of norms are *constitutive* and *regulative.*

Constitutive norms define Legally-constructed Institutions (e.g. Organization, Public Body, Society), Legal Roles (e.g. President, Judge, Defendant), and Legal Powers.

Among Legal Powers, Institutional powers enact Institutional Facts (e.g. Marriage, Agreement), where a physical or non-physical entity acquires an institutional status if one (or a system of multiple) constitutive rule justifies the *status function* (X counts as Y). *Definition* and *power-conferring* rules are sub-classes of constitutive norms.

Constitutive norms create all Law's entities, including laws themselves. Therefore, a legal norm *functionally depends on* a Constitutive Norm and on Collective Acceptance [Searle 1995].

Regulative norms define *behaviors* (as courses of events), and have at least one modalized description as a *proper part.*

In representing regulatory knowledge, we should also consider that norms pertain to a legal system, and are far from being *independent.* This context-dependency can also be found in norms pertaining agreements that result from private power and entail juridical effects among the parties: they are not autonomous, since they must comply to civil law norms.

Norms describe *complex objects.* It is hard to decide when a norm is *atomic,* and cannot be broken down into simpler ones. Some theorists consider definitions as parts of norms, other as norms themselves. Others yet deny procedural norms their status of norm, and so on. Many basic legal concepts, e.g. ownership, cannot be described as a single power of a person on an asset, but according to the modern view of jurisprudence, as a 'bundle of rights' [McCarthy, 2002].

5 The Regulatory Domain: European Norms About the Re-use and Commercial Exploitation of Public Sector Information

Our case study addresses a specific topic in the domain of Copyright Management, dealing with the representation of click-on licences for re-using Public Sector Information (PSI). The reason of this choice is that conceptualizations of Intellectual Property Right (IPR) and DRM (Digital Right Management) are a matter of great interest both from the ontological and the technical perspective [Dulong de Rosnay, 2003, Delgado et alii, 2003]. Many international projects aim to define common models for the management of digital rights that involve both the substantial definition of normally accepted rules (I-Commons[3]) as well as the definition of

[3] http://creativecommons.org/projects/international/

languages and models for the digitalization of the rules governing the use of digital information and web resources a (XrML[4], ODRL-Open Digital Rights Language[5]).

Moreover, the subject is strictly linked to the commercial exploitation of public sector information, which has been recently regulated by European norms.

On the 24 October 2003 the European Community formally adopted the Directive on the re-use and commercial exploitation of PSI[6]. The goal of the EU is to provide enterprises with new job opportunities, by taking into account the considerable economic potential of PSI, as the new Information Society Technology makes it possible to combine data taken from different sources and create a vast range of added value products and services. The Directive aims at ensuring that in relation to the re-use of PSI the same basic conditions apply to all players in the European information market, that more transparency is achieved on the conditions for reuse and that unjustified market distortions are removed.

According to the Directive[7], *"Public sector bodies should be encouraged to make available for re-use any documents held by them... The Directive should apply to documents that are made accessible for re-use when public sector bodies license, sell, disseminate, exchange or give out information. That is produced and charged for exclusively on a commercial basis and in competition with others in the marker"*[8].

Art. 1, sub-sects 2,3,4 constrain the exercise of reuse, stating that it shall not affect the access right of citizen[9], the personal data protection[10] and the IPR[11].

The Directive sets some rules for the re-use of PSI: "In some cases the re-use of documents will take place without a license being agreed. In other cases a license will be issued imposing conditions on the re-use by the licensee dealing with issues such as liability, the proper use of documents, guaranteeing non-alteration and the acknowledgement of source. If public sector bodies license documents for re-use, the license conditions should be fair and transparent. Standard licenses that are available

---

4 http://www.xrml.org/

5 http://odrl.net/

6 See Directive 2003/98/EC of the European parliament and of the concil of 17 November 2003
http://europa.eu.int/information_society/topics/multi/psi/library/index_en.htm#1.%20Directive%20on%20the%20re-use%20of%20PSI

7 Art. 3, General prevision of the Directive 2003/98/EC: "Member States shall ensure that, where the re-use of documents held by public sector bodies is allowed, these documents shall be re-usable for commercial or non-commercial purposes in accordance with the condition set out in Chapters III and IV".

8 Premises of the Directive.

9 Art.1, sub-sect.3. "This Directive builds on and is without prejudice to the existing access regimes in the Member States...".

10 Art., sub-sect.4.: "This Directive leaves intact and in no way affects the level of protection of individuals with regard to the processing of personal data)".

11 Art.1, sub-sect.2.: "This Directive does not apply to documents covered by industrial property rights, such as patents, registered designs and trademarks. The Directive does not affect the existence or ownership of intellectual property rights of public sector bodies, nor does it limit the exercise of these rights in any way beyond the boundaries set by this Directive. The obligations imposed by this Directive should apply only insofar as they are compatible with the provisions of international agreements on the protection of intellectual property rights".

online may also play an important role in this respect"[12]. As a condition for licensing, Public sector bodies should respect competition rules when establishing the principles for re-use of documents avoiding exclusive agreements as far as possible.

5.1 Modeling a License

UK Government has already started the process of implementing the EU rules. During 2003 a Consultation on a Partial Regulatory Impact Assessment (RIA[13]) on the Proposal for a Directive was held, considering the ways to ensure that the public sector complies with the measures set out in the Directive. Moreover, Her Majesty's Stationery Office (HMSO[14]), who is responsible of managing and licensing the re-use of **Crown copyright** material, launched online *Click-Use Licence.* There are currently licences, which allow unrestricted use of *core* government information under licence. A new phase will extend the Click-Use approach to *value added licences* where fees are charged and collected online.

We choose UK licences as a subject for testing our modelling framework[15], which is presented here informally.

A **licence** is a kind of *authorization.* It is a *description* including other typical descriptions as parts: *rights* that set actions that an actor can (i.e. has explicit permission to) exercise on a resource; *constraints* (limitations); *conditions* (exceptions that make permissions expire) and *requirements* (obligations that must be met before permissions can be exercised).

According to D&S, a licence is therefore a *bundle of descriptions* stating permissions about use and re-use; *constraints, conditions* and *requirements* are expressed in term of descriptions (specially *modal* ones) that are parts of that bundle.

License descriptions typically define or involve (the *defines* relation holds between descriptions and concepts, while *involves* holds between classified entities and descriptions that define the classifying concepts):

Perdurants: activities involved in the norms, such as *access, copy, redistribute, republish.* In our case study, such actions affect a non-physical entity (information) and are "sequenced by" a legal *course* of events (e.g. what *re-use* means in terms of permissions and obligations).

Endurants: agents and resources participating in activities, such as *individuals,* groups, organizations, or other *agentive figures.*

Agents can play **roles** such as *author, rights holder, user,* etc. *Rights holder* (who grants permissions) is played by Public Bodies (that also play the *producer* role), by agents that also play the role of *custodian* (who has the managing power over the resource, e.g. assignment and maintenance of access control markings), or by agents playing the *representative* role. The *user* role (who accesses electronic or digital versions of the Products) is "specialized by" *End-user* and *Reuser. Rights holder*

---

[12] Art. 8: Licences sub-sect. 1 "Public sector bodies may allow for re-use of documents without conditions or may impose conditions, where appropriate through a licence, dealing with relevant issues".

[13] http://www.oqps.gov.uk/copyright/ria_consultation_03_archive.htm

[14] http://www.hmso.gov.uk/copyright/licences/click-use-home.htm

[15] Entity descriptions are based on the UK Metada Standard Framework.

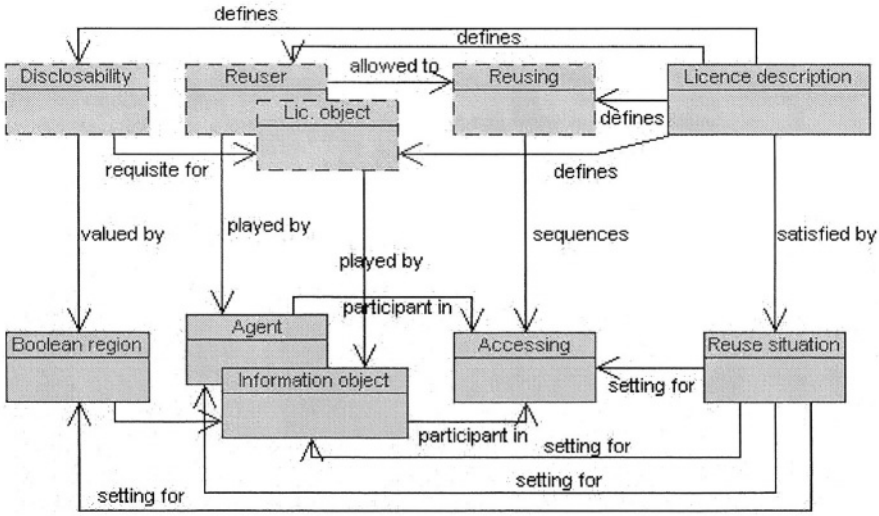


Fig. 2. Partial represention of the concepts involved in a licence description, and of the classes of entities that are classified by those concepts. The dashed rectangles represent individuals.

players are asserted in descriptions stating the existence of the right *(Copyright Statement).*

Legal parameters are *requisites for roles* and *courses* (e.g. certain *types* and *formats of data* can be requisites for their delivery). Specific parameters for DRM are *accessibility* (whether particular users *are able* to access or use the resource) and *disclosability*: as a general rule, user is *allowed* to access *disclosable* only data. The material that is covered by security classification, legal or policy restrictions, is excluded. More specifically, *disclosability* is concurrently constrained by*:* Data Protection Act, Freedom of Information Act, Environmental Information Regulations. Declaration of disclosability is subject to review. Therefore, *disclosable* requires a temporal location of the disclosability review.

Regions in a case setting must be values for some *legal parameter* (e.g. quantity of data required, the date of the formal decision regarding the disclosability review)

Legal roles have some *attitude towards* a *course* of events (e.g. *citizens* are allowed to access legal information and are forbidden to access *undisclosable* data).

The general rule states that reuse of disclosable information is permitted. The rule is expressed in our model by a modal description, where agents *play* the *reuser* role, *reusing PSI* is a course modalized as *permitted* (reusers are *allowed to* access disclosable data), and e.g. *disclosable* is a (boolean) value for the parameter *disclosability,* which is a *requisite for licensed object,* which is a role played by data (information objects). Figure 2 summarizes some of the elements introduced in order to formalize a licence.

5.2 Bundles of Norms

As specific cases, three licence types can be singled out:

CASE 1) A **Free-use licence** covers material where Crown copyright is asserted[16], but waived. Waiver material can be re-used free of charge without requiring a formal licence, provided that the material is:

- acknowledged
- reproduced accurately and kept up to date
- not used in a misleading way

In a description called *Free-use,* the role *reuser has-duty-towards* the following courses of action: *acknowledge, reproduce accurately, update PSI. Using PSI in a misleading way* can be represented as well, but this kind of courses includes usually "open-textured" notions in the legal system, and should be left to legal interpretation. The negation actually means a prohibition modality on that course. The different duties and prohibitions constitute a bundle of norms (modal descriptions).

CASE 2) A **Core licence** covers core material, which is likely to satisfy the following conditions (represented as either roles or inherent properties):

- it is essential to the business of government
- it explains government policy
- it is the only source of the information[17]

Among obligations, in core licences the role *reuser* must (has an *obligation-towards*):

- reproduce data accurately from the current Official Source except where it is made clear that there is a more up to date version available
- identify the source of the data and feature the copyright statement when publishing
- not use the data for the principal purpose of advertising or promoting a particular product or service, etc.

Requisites are expressed by *parameters* which affect the quality inherent to public information, for instance the quality *exclusiveness* must have a positive value.

Conditions are expressed by means of other modal descriptions.

CASE 3) A **Value-added licence** usually satisfy the following conditions:

- data bring together information from a variety of sources, Value will often be added by means of commentary, analysis, indexing, search facilities or text retrieval software. There may be similar competing commercial services and products in the marketplace
- data creation is not vital to the workings of government. There will often be alternative suppliers of such information, etc.

[16] Public Sector Information are subject in UK to Crown or Parliament Copyright.

[17] Only part of the actual requirements and conditions has been listed.

Among obligations, the *reuser* role ought to:

- identify the source of the information objects
- specifically to *Royalty* licences, to keep full and accurate records of the sales of a product

Here a specific condition: *fee or royalty payment* is represented as a course, which is a precondition for the reuse-activity. The main requisite is a parameter valued by kinds and quantities of sales.

5.3 Managing Bundles of Rights

From the previous analysis, it appears that the merging of the description of the general licence regulation with further descriptions is able to catch the representation of the various types of licenses. This conforms to the observation that many regulations are spread among different norms.

Each complex of descriptions describes the 'bundle of right' linked to the license: the permitted agent is *liable toward* the *copyright holder* role (or in the case of Public Bodies, their representative) to fulfill conditions, the *custodian* role has the *power* and *duty* to check that *requirements* are met, the *reuser,* as allowed agent, has the *power* (in case of value-added licenses only) to create a new IPR on the transformed product.

This allows considering the complex of legal effects as a primitive object that in turn may be embedded in legal descriptions and hence sequenced by legal courses, appling a method proposed by Hart as "contextual definition" of complex legal entities.

For instance, an allowed agent could assign its rights since she is potentially *empowered* to transfer the right. In fact, the prohibition to re-assign rights is the main constraint, as expressly provided for in the management of copyrights[18]; and Public bodies are obliged to respect non-exclusive rules in licensing.

6 Conclusions and Future Work

This is a preliminary result of a project to develop a domain ontology of IPR using entities defined in our Core Legal Ontology. The test is meant to assess the validity and coverage of the ontology, the expressivity of the representation framework and the reliability of the model. We consider the work still in progress, since further aspects need to be refined, e.g.:

1) the distinction of different kinds of knowledge in term of the role they play within a system of norms. Ontological features of constitutive and regulatory norms are clearly expressed in the norm structure, but the epistemologically different roles (the descriptive role of constitutive definitions and the prescriptive role of the norms imposing conditions and constraints are not trivially entrenched) need further

[18] The licence granted in Section 3 above is expressly made subject to and limited by the following restrictions you may not sublicence the Work, cf. http://creativecommons.org.

investigation. A similar hardness lays among properties that are *parameters* for juridical effects, and those that are conditions upon which effects are legally valid: the former are meant as *practical* requisites for an action (without which the action could not be carried out), while the latter as *juridical* requisites (without which the action is not validly carried out).

2) linking the domain ontology to the lexical level of JurWordNet is not straightforward. There is a two-way relation between a semantic lexicon and an ontology: *"it is possible that a lexicon with a semantic hierarchy might serve as the basis for a useful ontology, and an ontology may serve as a grounding for a lexicon. This may be so in particular domains, in which vocabulary and ontology are more closely tied than in more general domains"* [Hirst2004]. Up to now, we have tested the latter, while the former approach seems to offer interesting results in the field of cognitive interfaces [Borges et alii 2003].

References

1. Alchourrón C. E., Buligyn E. Normative System, Vienna: Springer Verlag, 1971.
2. Borges F., Bourcier D., Andreewsky E. and Borges R. Conception of cognitive Interfaces for legal knowledge, in Proceedings of ICAIL 2001, ACM Press, 2001.
3. Breuker J, and Winkels R. Use and reuse of legal ontologies in knowledge engineering and information management. ICAIL03 Wks on Legal Ontologies and Web-based Information, Management, Edinburgh, http://lri.jur.uva.nl/~winkels/legontICAIL2003.html, 2003.
4. Breuker J. e den Haan N. Separating world and regulation knowledge: where is the logic?, in Proceedings of the Third International Conference on Artificial Intelligence and Law, ACM, N.Y., 1991.
5. Comanducci P. Kelsen vs. Searle: A Tale of Two Constructivists. Analisi e diritto, 1999.
6. Delgado J., Gallego I., Llorente S., García R. IPROnto: An Ontology for Digital Rights Management, in Proceedings of Jurix 2003, IOS Press, 2003.
7. Dulong de Rosnay M., Cognitive Interface for Legal Expressions Description - Application to Copyrighted Works Online Sharing and Transactions, in Proceedings of Jurix 2003, IOS Press.
8. Gangemi A. Prisco A, Sagri M-T., Steve G., Tiscornia D.,Some ontological tools to support legal regulatory compliance, with a case study, Proceedings of the second International Workshop on Regulatory Ontologies WORM 2004 -Part of the International Federated Conferences (OTM '04), Springer LNCS, 2004.
9. Gangemi A., Mika P. 2003, Understanding the Semantic Web through Descriptions and Situation, Meersman R, et al. (eds.), Proceedings of ODBASE03, Springer, Berlin, 2003.
10. Gangemi A, Prisco A, Sagri MT, Steve G, Tiscornia D, "Some ontological tools to support legal regulatory compliance, with a case study", in Jarrar M, et al. (eds.), Proceedings of the WORM03 Workshop at ODBASE03 Conference, Springer Verlag, Berlin, 2003.
11. Gangemi A, Catenacci C, Lehmann J, Borgo S. Task taxonomies for knowledge content, EU 6FP METOKIS D07, http://metokis.salzburgresearch.at, 2004.
12. Hart H.L.A. The Concept of Law. Clarendon Press, Oxford, 1961.
13. Hirst G. Ontology and the lexicon. In Staab, Steffen and Studer, Rudi (editors) Handbook on Ontologies in Information Systems, Berlin: Springer, 2004, p. 14.
14. Jones A. A logical Framework, in J.Pitt (ed.), Open Agents Societies: Normative Specifications, in *Multi-Agent Systems,* Wiley and S., 2003.
15. Jones A and. Sergot M. A formal characterisation of institutional power, Journal of IGPL, 4 (3), 429-445, 1996.

16. Kanger S. Law and Logic. Theoria, 38, 105-132, 1972.
17. Lindhal L. Position and Change. A study in Law and Logic, Reidel, 1977.
18. McCarthy L.T., Onwnership:A Case Study in the representation of Legal Concepts, in aI &Law Journal, vol.10, 2002.
19. Masolo C., Vieu L., Bottazzi E., Catenacci C., Ferrario R., Gangemi A., Guarino N. Social Roles and their Descriptions, in Proceedings of KR 2004.
20. Masolo C., Borgo S., Gangemi A, Guarino N, Oltramari A. The WonderWeb Library of Foundational Ontologies", IST 2001-33052 Wonder Web. http://wonderweb.semanticweb.org/deliverables/documents/D18.pdf, 2003.
21. Miller G. WordNet: A lexical database for English, in Communications of the ACM 38(11), 1995.
22. Pörn, I. Action Theory and Social Science, Some Formal Models, Reidel, 1977.
23. Sagri M.T. Progetto per lo sviluppo di una rete lessicale giuridica on line, in Informatica e Diritto, ESI, Napoli, 2003.
24. Searle J. R., The Construction of Social Reality. Free Press: N.Y., 1995.
25. Valente A. Legal Knowledge Engineering, A modelling Approach. Amsterdam: IOS Press, 1995.

Workshop on Ontologies, Semantics, and E-learning (WOSE) PC Co-chairs' Message

Ontologies define precisely and formally the semantics of objects and their inter-relationships for a specific application domain. As ontologies enable semantic interoperability between information systems, they are a central notion for the Semantic Web. The application domain for this workshop is (e-)learning. Indeed, many kinds of software and content components, platforms and legacy systems for e-learning exist, commercial and freeware. The challenge is to make this large variety of components interoperable, so that an open, global infrastructure for learning can be tailored to the needs of learners, trainers and teachers. Already, some learning technology standards have XML and/or RDF bindings, which makes them suitable for processing with semantic web technology.

As program chairs, we are happy to introduce to the readers of this volume a number of subsequent papers aiming at relating insights from the field of the Semantic Web with the E-learning domain. All the submitters have done their best to report on their latest scientific progress and wanted to share their insights and results. The members of the program committee had a tough job to operate a just selection. They have carefully judged the papers and upheld the quality of this workshop, which involved in a some cases extra discussion via email. We therefore very warmly thank the members of our program committee.

August 2004

Erik Duval, Katholieke Universiteit Leuven (Belgium)
Peter Spyns, Vrije Universiteit Brussel (Belgium)
(WOSE 2004 PC Co-chairs)

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, p. 689, 2004.
© Springer-Verlag Berlin Heidelberg 2004

Improving LOM-Based Interoperability of Learning Repositories

Germán M. Rivera[1], Bernd Simon[2], Juan Quemada[1], and Joaquín Salvachúa[1]

[1] Universidad Politécnica de Madrid, ETSI Telecomunicación
Ciudad Universitaria s/n, 28040 Madrid, Spain
{rivera, jquemada, jsr}@dit.upm.es
[2] Wirtschaftsuniversität Wien, Information Systems Department
Augasse 2-6, 1090 Vienna, Austria
bernd.simon@wu-wien.ac.at

Abstract. This paper analyses the use of LOM Application Profiles for learning object repository interoperability. Based on an exemplifying use case the paper presents a case study, which aims at developing a LOM Application Profile to realized Smart Spaces for Learning. Finally, a schema is designed which selects the necessary LOM elements and makes a LOM-conformant extension to represent usage conditions and learning activities. This work is part of the Elena project, which focuses on integrating advanced educational mediators in a network of federated services.

1 Introduction

Interoperability between learning object repositories – also referred to as repositories for learning, learning repositories, or knowledge pools – still remains a pending issue in information systems research. Repositories for learning hold information on learning objects and make them available according to pre-defined processes. Taking advantage of proprietary user interfaces, users are able to access learning objects in order to perform learning, organize courses, or develop content. Since many of these repositories lack interoperability with other systems, the learning objects accessible are restricted to those stored in the repository. This paper aims to contribute to interoperability research by providing an illustrative case study on an interoperable network of learning object repositories that has been set up on the basis of a Learning Object Metadata (LOM) Application Profile.

The paper first defines the application profile notion and relates it to the ontology concept. In this section also guidelines for designing application profiles are discussed. In Section 3 an introduction to the ELENA project, which aims to build Smart Spaces for Learning is given. From this vision requirements for a LOM application profile are derived. The application profile and its design process are presented in Section 4. The application profile is based on a classification of learning objects that distinguishes between learning material and learning activities and customizes the LOM conceptual model according to this differentiation. The paper concludes in Section 5 where the limitations of the current approach are discussed.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 690–699, 2004.
© Springer-Verlag Berlin Heidelberg 2004

2 LOM Application Profiles

2.1 Application Profile Definition and Ontologies

Application profiles are defined as an assemblage of metadata elements selected from one or more metadata schemas and combined in a compound schema [1]. They provide the means to express principles of modularity and extensibility. The purpose of an application profile is to adapt or combine existing schemas into a package that is tailored to the functional requirements of a particular application, while retaining interoperability with the original base schemas.

This paper deals with LOM Application Profiles. The LOM standard [2] is an accredited IEEE standard and was first released in 2002. Since then its significance has constantly increased. By being referenced also by other specifications such as IMS Metadata and SCORM, it has adopted a key role in the context of learning metadata. Its conceptual model is well known and has been deeply discussed in the e-learning community. So, in order to achieve interoperability between heterogeneous learning repositories, LOM is a good starting point to integrate the information of diverse metadata profiles. Additionally, from a pure business point of view, a (meta)data model that is based on an existing standard such as LOM is attractive for implementers, since it reduces the risk of sunk costs.

With the dawn of the Semantic Web the notion of ontology has increasingly become a topic of interest in computer science. In this context, an ontology can be defined as an engineering artefact, constituted by a specific vocabulary used to describe a certain reality, plus a set of explicit assumptions regarding the intended meaning of the vocabulary [3]. According to this definition an application profile can be viewed as a kind of ontology, although application profiles like the one described herein are not represented in an ontology language [4]. Following the classification of ontologies proposed by Guarino [5], an application profile can be considered as an 'application ontology' taking advantage of concepts defined in higher level ontologies such as the LOM Standard or Dublin Core.

A LOM Application Profile adapts the LOM conceptual model in order to be able to capture all the specific information that an application needs while keeping it as simple as possible. The application 'profile design process requires the following measures:

1. Refinement of the learning object notion
2. Selection of LOM elements
3. Semantics refinement
4. Specification of multiplicity constraints
5. Specification of value restrictions
6. Introduction of required extensions

2.2 What Makes a LOM Application Profile?

The IEEE LOM Standard depicts a rich conceptual model intended for describing heterogenous learning objects. However, for a specific application scenario it may be convenient to restrict the number of elements used in order to reduce the complexity and scattering of the descriptions. This is crucial in case this profile must serve as a

common core model for the learning object descriptions of existing heterogeneous repositories.

The profile may also describe how each LOM element is interpreted, but always consistently with the original definitions. Additionally, multiplicity including if the element is optional or mandatory should be specified. However, only a minimum set of elements indispensable for the service should be required otherwise the barrier of providing learning objects might be too high. In order to maintain LOM compatibility, originally non-multiple elements must not be made multiple.

In some cases it might turn out to be useful to restrict the value spaces of elements (These value spaces are also referred to as 'permissible values'). For example, in the presented case study vCard properties also need to be restricted (vCard elements are permissible element values for the LOM Element 2.3.2 *Entity*). In addition, LOM vocabularies can be restricted selecting the words permitted for each element with the same purpose as of element selection.

It is possible that the LOM conceptual model does not cover all aspects of all the types of learning objects managed in an application. In such a case, extensions are needed. The extension mechanism described in the IEEE LOM Standard can be used. LOM can be extended in two ways, elements and vocabulary. New elements semantics must not intersect with the original LOM semantic model. Words added to a vocabulary must be associated with their own source and, of course, must not interfere with the original vocabulary. This action reduces considerably the semantic interoperability of the element with LOM, so implementers must make sure that it is really worthwhile for the conceptual model of the profile.

It is recommendable to take non-LOM elements from other standards. Also, mappings of the rest of the elements of the profile to these standards may be specified. This facilitates the conversion of learning object descriptions between standards and so increases the utility of the profile.

2.3 Requirements for LOM Backwards Compatibility

It can be useful for an application to be able to process not only descriptions based on its own application profile, but also pure LOM descriptions. This can be defined as "LOM backwards compatibility" of the application. Such an application and its application profile must fulfill the following conditions:

- No extended elements should be mandatory. Obviously, that would discard every pure LOM description. Otherwise, the application should be able to conclude the value of such element when it receives an original LOM description.
- LOM elements not present in the application profile must be ignored by the application. That means that every parsing system must overpass those elements and no application must modify nor delete them.
- The application must extract as much information as possible from LOM elements. If the complete value is not interpretable, then the element must be ignored. In LOM, there exist some vocabulary elements (semantics-adjusting elements) that alter the meaning of the value of other related elements. For example, Element 2.3.1. *Role* modifies the semantics of the Element 2.3.2 *Entity*. In such a case, the application should discard all related elements if it does not recognize the value of the vocabulary element. An example of a partially interpretable value could be Ele-

ment 2.3.2 *Entity* again, whose value (vCard) is supposed to contain only a subset of the properties specified in the vCard 3.0 Standard.

3 The ELENA Smart Space for Learning

3.1 The Need for Interoperability in Smart Spaces for Learning

The LOM Application Profile presented in this paper has been initiated by research carried out in the context of the IST Project ELENA and is further extended by IST-funded Network of Excellence ProLearn. The objective of the ELENA Project is to design, implement, and test the applicability of Smart Spaces for Learning [6]. A Smart Space for Learning is defined as a network (space) of learning repositories, which supports the personalized (smart) mediation of heterogeneous learning objects.

ELENA is primarily targeting corporate learners. Today's corporate learners are served by Internet access through their desktop and mobile phone, business-unit specific knowledge repositories and course(ware) databases available through intranets. Leading business organisations are offering its workforce a heterogeneous set of learning objects ranging from traditional seminars over knowledge management activities such as sharing of best practices to e-learning content. While such a sophisticated learning environment creates competitive advantage by intellectually empowering a company's workforce, some shortcomings limit the benefits, mainly from the perspectives of decision effectiveness, process administration, and IT infrastructure management.

The lack of interoperability of learning object repositories, for instance, does not allow for one, unique view on the learning objects offered. As a result, with each repository introduced to the environment, a user's search costs increase (multiple searches have to be submitted) and the transparency of learning objects offered suffers (some potential target repositories are not considered in a search). This need for interoperability is further supported by a series of qualitative interviews of personnel managers who also would like to provide corporate learners an integrated view on the learning objects available in a corporation [7]. For example, whenever a learning need has been identified, learning objects going beyond traditional courses shall be accessible to the learner. In such a scenario, even company-external repositories shall be considered in a query for learning objects (e.g. an online bookshop, a course broker). Interoperability achieved through a common (meta)data model manifested in a LOM Application Profile is the enabler for an integrated view on heterogeneous learning object repositories.

3.2 Requirement: Semantic Search

Beyond achieving interoperability, a LOM Application Profile also lays down the foundations for a new type of search across multiple targets. Current document-based search engines, such as the ones typically available on the Web, are mainly supporting navigational search [8], hereby answering queries such as: "Provide me with the URL

of the web site mentioning concept X". However, a network of repositories based on a common LOM Application Profile, can also support other kinds of searches, such as:

- Duration-restricted searches, e.g. find me all courses offered between today and the 20th of December,
- Location-restricted searches, e.g. find me all courses offered in Madrid and Vienna,
- Cost-restricted searches, e.g. find me all learning objects with a net price below 100 EUR.

Only such kind of semantic searches allows a system to locate services and activities - opposed to documents and materials - in a user-friendly and reliable manner. A semantic search in this context refers to a search, which is based on some kind of ontology or common data model that connects several systems.

3.3 Differentiating Learning Material from Learning Activity

In Smart Spaces for Learning the distinction between two different types of learning objects, namely Learning Materials (LM) and Learning Activities (LA), is essential and also drives the requirements for the application profile.

Learning materials are available asynchronously and can be consumed independent of time and location. An LM is manifested in a physical or digital good. Learning activities, on the other hand, are live events that are delivered according to a schedule at physical or virtual location. An LA is manifested in a synchronous service, which is frequently supported by information technology nowadays. A course constitutes an example of a learning activity, a book an example of a learning material.

The distinction between learning activities and learning material is not new [9]. The IMS Learning Design Specification [10], for example, provides means to express LAs as components of an educational experience (called "learning design") that are based on a pedagogical strategy. The creation of the Learning Activity Management System (LAMS), a tool for creating sequences of learning activities, has even been inspired by this specification [11]. Similarly, in the SCORM Sequencing and Navigation Book [12] LAs are considered as elements of a learning flow that must be delivered to the learner in a pre-defined way. In order to achieve this, the SCORM Sequencing and Navigation Book defines a proper representation model for activities delivered via Learning Management Systems. However, both standardization initiatives present the learning activity notion in a particular context and do not treat an LA as an autonomous entity, whose description is intended to be exchanged between repositories.

In order to support searching for LAs and LMs, different concepts are needed for specifying the search as well as the learning object descriptions the search aims to match. Obviously, the time when a LA is offered is an important selection criterion. Beyond schedule information, the location where the activity is offered is considered to be important. In case of an electronic learning material, users also might want to specify the media format. Language and price are common attributes of learning activity and learning material.

A Smart Space for Learning, which is based on an application profile aligned to requirements mention above, is able to provide and can be tested against the following search capabilities:

- Only learning objects meeting a user's language preferences are selected
- Only learning objects within a specific price range are selected
- Only learning material meeting user's media format preferences are selected
- Only learning activities meeting a user's location preferences are selected
- Only learning activities that are available in the preferred time period

The application profile is not only determined by the concepts used for specifying the query, but also by the concepts used when presenting search results. In a Smart Space for Learning the following concepts are presented to the end user:

- Title,
- Provider,
- Description,
- Contributors (e.g. Author, Instructor, etc.),
- Price Yes/No,
- Restrictions Apply Yes/No,
- URI to full description
- URI to access.

A result list including the data like the one above meets following requirements:

- Users can determine from the results list if a learning objects is for free,
- Users can determine from the results list if usage restrictions apply
- Users can see "who is behind a learning object".

4 A LOM Application Profile for a Smart Space for Learning

This paper will not describe the LOM Application Profile in detail, but will document the most important decisions concerning its design. The latest version of the application profile specification is available at the Learning Object Repository Interoperability Site [13].

4.1 Application Profile Design

Refinement of the Learning Object Notion. The application profile introduces two major aspects in to the learning object notion that will guide the design of the semantic model of the profile. On the one hand, it distinguishes between two fundamentally different types of learning objects: learning activities and learning material as defined in Section 3.3. On the other, the brokerage aspect is considered to be predominant. Hence, the application profile provided shall be expressive enough to provide all data that a potential consumer needs to know from a contractual point of view.

Driven by these requirements a learning object in a Smart Space for Learning can be defined as any kind of material or activity that is made available by a learning object provider under specified conditions for the purpose of facilitating learning.

Figure 1 shows the classes of the semantic model and its relations. LM information fits properly in LOM conceptual model. That is not the case of LAs and usage conditions, which are partially described in LOM, but inevitably involve some extensions. Contributor roles differ between LA and LM (e.g. author for LM, instructor for LA, etc.). A contributor can either be a person or an organization.

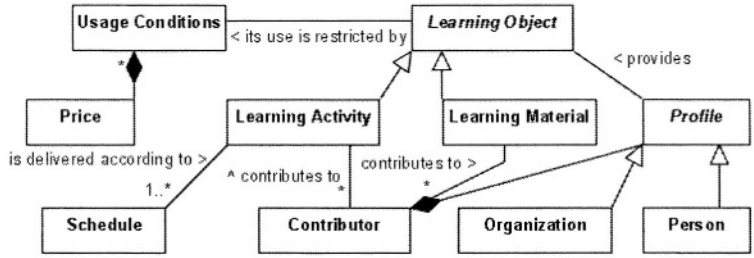


Fig. 1. Simple UML class diagram of the semantic model

Selection of LOM Elements. The elements of the application profile shall be sufficient to allow users to make a selection decision in favor or against a particular learning object. LM and LAs share a set of characteristics that are described by a common group of elements. Most of the LM-specific elements were selected from LOM, although in some cases it has been necessary to refine the value that LOM specifies by inserting a new element. Additionally, LOM covers also some aspects of LAs and usage conditions, what increases the number of LOM elements included in the schema. Based on the requirements specified in Section 3 LOM elements such as title, language, description, cost, copyright and other restrictions were selected.

Semantics Refinement. The Application Profile provides a description of how the elements will be interpreted in the context of a Smart Space for Learning. For instance, it specifies whether a LOM element applies to LM, LAs or both. Additionally, LOM elements such as learning resource type is driven by the semantic requirements of the Smart Space for Learning use case, which results in a narrower definition as set forth by LOM itself.

Specification of Multiplicity Constraints. While in LOM the occurrence of every element is optional, the application profile differentiates between mandatory elements, essential for efficient resource management, and conditional elements, needed for semantic coherence. In the case of vocabulary elements, multiplicity needs to be specified for each value. For instance, if the application profile element educational objective can only be specified once, LOM Element 9.1 *Purpose* can only adopt the value "educational objective" once.

Specification of Value Restrictions. In the proposed application profile restrictions are applied to two types of values: LOM Value Spaces and vCard Elements.

In the case of "non semantics-adjusting" vocabularies (see Section 2.2) values are restricted in order to limit the value range of a concept, for example in case of LOM Element 5.2 *Learning Resource Type.* Restrictions of "semantics-adjusting" vocabularies, in the case of LOM Element 2.3.1 *Role* for example, restrict the possible meanings of the value of its related element, so it produces a similar effect to element selection as it reduces the number of concepts of the schema.

vCard Elements considerably increase the complexity of the parser that analyses the LOM (or LOM Application Profile) description due to its particular syntax and

rich set of properties. In order to maintain the application profile close to the LOM conceptual model, the syntax is not changed, but vCard admissible properties are restricted to a subset that allows describing essential identification and contact information for both people and organizations.

Introduction of Extensions. The application profile has extended LOM both in vocabulary and elements. Apart from LOM Element 5.2 *Learning Resource Type,* the application profile extends the vocabulary of 2.3.1 *Role,* which enumerates the possible roles of LM contributors. Extended elements have been included into the respective LOM category. Usage conditions extended elements all correspond to *rights* category, following the LOM criterion. No extensions have been included exclusively for LMs, but *educational* category has been extended to describe shared information between activities and materials (apart from usage conditions). On the other hand, LAs are described using extended elements of categories *educational* and *lifecycle,* like *delivery,* described next.

4.2 Overview of Selected Elements and Mappings

LOM distinguishes between nine element categories. Here are presented the top elements (not their selected or extended sub-elements) in each category with a short description of some of them. The schema specification indicates whether each (sub-)element or vocabulary value applies to LA, LM or any learning object (both LA and LM).

The *general* category comprises *identifier, title, language* and *description,* used for basic identification of a learning object. *Lifecycle* comprises *contribute,* extended to capture all the contributors to LA and LM; *version* for LM, and *delivery,* a new element that captures the schedule and location of a LA whose datatype is iCalendar (Dawson & Stenerson, 1998). The third category, *metametadata,* includes *language* and *contribute,* which captures the creation date of the metadata of a learning object.

The technical format and requirements of a LM are expressed with the elements *format, requirement* and *other platform requirements* of the *technical* category.

In the *educational* category, *learning resource type* value space has been widely extended to designate all kinds of LA and LM. Also, five non-LOM elements have been included: the essential *learning resource class,* whose possible values are learning activity and learning material, *minimum* and *maximum participants* for LA, and *target learner* and *target learner profile* which, with the LOM element *intended end user role,* designates the target group of a learning object.

All the LOM elements from *rights* category have been selected and they have been complemented with *price, VAT, valid thru* and *special conditions* new elements to be able to describe in detail the cost of the use of the learning object and its restrictions. The application profile will also take advantage of the versatility of the classification category and will use restrictively its four LOM elements to express the objective of a LA and the subject and prerequisites of any learning object.

On the contrary, since each object is considered isolated from others and will be provided individually, the relation category is not used. Two elements from educational category are selected. Finally, annotation category is worthless for the purposes of the schema.

The mappings between LOM and DC specified in the IEEE Draft Standard for Learning Object Metadata are also applicable to the LOM elements included in the application profile with one exception: LOM Element 5.2 *Learning Resource Type,* which IEEE maps to *DC.Type.* Since LOM *Learning Resource Type* refers to LM type in the Common Schema, *DC.Type* must map to the highest-level classification element of type of the object in the schema: *Learning Resource Class.* Twelve of the fifteen DC elements can be mapped to Common Schema.

Since many of the extended elements have been directly extracted from Open-Qcat [14] standard for education and training services, mappings for those extended elements and LOM included elements (where possible) have been also specified.

5 Conclusion

This paper illustrated the design of a LOM Application Profile for the purpose of building Smart Spaces for Learning. A generic design process for building LOM Application Profiles is presented and applied. The application profile documented in this paper is driven by a conceptual model, which differentiates learning objects in learning materials and learning activities. Learning objects are described for brokerage proposes, which requires a significant extension of LOM with usage restriction aspects.

Learning object repository interoperability is discussed only from a pure data model perspective. Questions related to the design of application program interfaces, such as the Simple Query Interface [15], are not covered by the paper. The model presented herein does not cover how single learning object descriptions can be aggregated using for example IMS Content Packaging specification or the SCORM Content Aggregation Model. Issues related to the actual bindings of the model in XML or RDF constitute additional avenues for further research.

Acknowledgements. This work was supported by the IST project ELENA (http://www.elena-project.org/) and the IST Network of Excellence PROLEARN (http://www.prolearn-project.org/), which are partly sponsored by the European Commission. The authors would like to express their gratitude to the following consortium members who actively participated in the model discussion Barbara Kieslinger, David Massart, Dirk Heckler, Erik Duval, Frans van Assche, Nikos Papazis, Peter Dolog, Muth, Sam Guinea, Sandra Aguirre, Stefaan Ternier, Stefan Brantner, Tomaz Klobucar, Wolfgang Nejdl and Zoltán Miklós.

References

[1] E. Duval, W. Hodgins, S. A. Sutton, and S. L. Weibel, "Metadata Principles and Practicalities," *D-Lib Magazine,* vol. 8, http://www.dlib.org/dlib/april02/weibel/04weibel.html, 2002.

[2] W. Hodgins and E. Duval, *IEEE 1484.12.1 - 2002 Learning Object Metadata (LOM) Standard,* 2002.

[3] A. Maedche, *Ontology Learning for the Semantic Web.* Karlsruhe, Germany: Kluwer Academic Publishers, 2002.

[4] D. Fensel, *Ontologies: A Silver Bullet for Knowledge Management.* Berlin: Springer, 2004.
[5] N. Guarino, "Formal Ontology and Information Systems," in *Proceedings of FOIS'98 - Formal Ontology and Information Systems,* N. Guarino, Ed. Trento, Italy: IOS Press, 1998, pp. 3-15.
[6] B. Simon, P. Dolog, Z. Miklós, D. Olmedilla, and M. Sintek, "Conceptualizing Smart Spaces for Learning," *Journal of Interactive Media in Education,* http://www-jime.open.ac.Uk/2004/9/simon-2004-9.pdf, 2004.
[7] S. Gunnarsdottir, B. Kieslinger, T. Küchler, and B. Simon, "From e-Learning to Learning Management: Results of an International Survey," in *Proceedings of 4th International Conference on Knowledge Management.* Graz, Austria, 2004.
[8] A. Broder, "A taxonomy of web search," *ACM SIGIR Forum,* vol. 36, pp. 3-10, 2002.
[9] J. Quemada and B. Simon, "A Use-Case Based Model for Learning Resources in Educational Mediators," *Educational Technology & Society (ISSN 1436-4522),* vol. 6, pp. 149-163, 2003.
[10] IMS, *IMS Learning Design Specification:* http://www.imsglobal.org/learningdesign/index.cfm, 2003.
[11] J. Dalziel, *Implementing Learning Design: The Learning Activity Management System (LAMS):* http://www.lamsinternational.com/documents/ASCILITE2003.Dalziel.Final.pdf, 2004.
[12] S. E. Thropp, *SCORM Sequencing and Navigation. Version 1.3.* ADL: http://www.adlnet.org/screens/shares/dsp_displayfile.cfm?fileid=998, 2004.
[13] LORI, *Learning Object Repository Interoperability (LORI) Web Site.* http://www.prolearn-project.org/lori/, 2004.
[14] S. Rex and A. Hettrich, *Spezifikation Open-Qcat. Version 0.9.5 - Public Draft.* http://www.open-q.de/, 2002.
[15] B. Simon, E. Duval, and D. Massart, *Simple Query Interface Specification:* http://nm.wu-wien.ac.at/e-learning/inter/sqi/sqi.pdf, 2004.

Domain Ontology as a Resource Providing Adaptivity in eLearning

Galia Angelova, Ognian Kalaydjiev, and Albena Strupchanska

Linguistic Modelling Department, Institute for Parallel Information Processing,
Bulgarian Academy of Sciences
25A Acad. G. Bonchev Str., 1113 Sofia, Bulgaria
{galia, ogi, albena}@lml.bas.bg

Abstract. This paper presents a knowledge-based approach to eLearning, where the domain ontology plays central role as a resource structuring the learning content and supporting flexible adaptive strategies for navigation through it. The content is oriented to computer aided language learning of English financial terminology. Domain knowledge is acquired to cover the conceptualisation of basic notions about financial markets. The learning objects are annotated according to the type labels of a precisely elaborated type hierarchy and associated first-order logic propositions. The learner model and adaptivity decisions exploit the hierarchy of concepts. We claim that the well-organised domain knowledge might be a prerequisite which provides interoperability of learning objects, at least in a significant number of learning applications. Unfortunately, exchange between ontologies is difficult, as there is no much knowledge available (in the Semantic Web). But our approach shows clear paths to the development of shareable learning objects, assuming that some new ontologies will appear with the evolution of semantic-based web content.

1 Introduction

The present expectations are that the emerging Semantic Web will develop a solid basis for future progress in eLearning. As [1] points out, "none of these technologies has reached full maturity as yet or been deployed widely so it is hard to gauge their success or failure". Nevertheless the potential importance and impact of semantic-based technologies in eLearning are easy to estimate. The inspiring idea to develop reusable atomic learning components and to capture their characteristics in widely-accepted, formal metadata descriptions will most probably attract learning object providers to annotate their products with the accepted standards. The architecture of the learning objects seems to be generic enough to make use of any properly annotated www-content. The standardised metadata descriptions will enable a more unified approach to adaptivity. Initial developments of this kind already appear and illustrate the use of ontologies for educational services [2].

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 700–712, 2004.
© Springer-Verlag Berlin Heidelberg 2004

This paper reports some results of the Lasrflats project[1] and their ongoing elaboration. The project was presented in several papers ([3,4,5,6]) which consider in detail the integration of advanced language technologies in computer-aided language learning (CALL). In contrast to these publications, here we focus on the domain ontology that is the backbone supporting the content structuring and an unified approach to adaptivity. Section 2 discusses the Larflast project and its approach to assign a central role to the domain ontology. Section 3 sketches ongoing work. Section 4 contains possible directions for future work and the conclusion.

2 An Approach to Knowledge-Based CALL

There are intelligent tutoring systems which contain no explicit domain model and no explicit type hierarchy. For instance, table 1 in [7] lists and classifies forty CALL systems. Many of them are considered "intelligent" because they integrate knowledge about typical language mistakes, learning strategies, questions and answers, or specific linguistic knowledge about e.g. correct spelling of verb forms. Such CALL systems are most often domain independent as they operate with the vocabulary included in the system lexicons. Similarly, adaptivity might be implemented without explicit encoding of domain knowledge as well. For instance, Knowledge Sea [8] applies a neural network-based mechanism to process pages from different Web-based tutorials along with a set of closed corpus documents (e.g. lecture notes) and groups them by similarity; as a result, a list of (hyperlinks to) relevant readings is offered to the user. In general, domain knowledge is not a typical resource in running eLearning applications as it is an expensive and sophisticated artifact, which requires much effort for acquisition, integration and tuning. On the other hand, almost all CALL systems, which check the semantics of free learner's utterances, need domain/world knowledge to perform formal analysis and interpretations of the learner's answers. These CALL prototypes contain "knowledge base" from some sort. For instance, knowledge can be encoded as "lexical conceptual structures" (BRIDGE/MILT [9]) or as "facts and propositions in a knowledge base" (as chosen in one of the recent CALL systems Why2-Atlas that checks the correctness of students essays in physics [10]). Our understanding, however, is that the "facts and propositions" in general and the type hierarchy in particular might be a resource supporting almost all system activities and functions. Perhaps this is the most innovative aspect of the project reported below, as it applies the same knowledge base for semantic analysis of free natural language utterances, for annotation of the learning objects and for support of adaptive navigation facilities. In this section we present and discuss our results.

[1] Learning Foreign Language Scientific Terminology, a Copernicus'98 Joint Research Project, funded by the European Commission in 1998-2001, with partners CBLU Leeds, UK; UMIST Manchester, UK; LIRM Montpellier, France; Academy of Sciences, Romania; Simferopol University, Ukrain; Sofia University and Virtech Ltd., Bulgaria.)

2.1 General Description

The project Larflast developed an adaptive Web-based terminology learning environment with focus on the harmonic integration of advanced natural language processing (NLP) techniques and innovative learning solutions for system-student communication [3]. The users are students with background in economics, business or management who study English as a foreign language. From pedagogical perspective, the main objective was to elaborate and to test with actual users a distributed Web-based learning environment, which enhances learner performance, increases his/her autonomy and provides motivational and informational stimuli over and above what a traditional paper-based approach to learning terminology normally offers. The project aims to find some balance between innovative research and practical needs. Initially a user study was performed, which focused on paper-based teaching of foreign language terminology. Typical errors were analysed together with the user preferences regarding the desired feedback and diagnostics. It was also important to chose the proper subdomain for teaching representative core of English terminology. Some 150 terms in financial markets were initially selected for the prototypical implementation. More terms were added at later knowledge acquisition phases, to cover features of the basic notions and facts encoding their important interconnections. The ontology in OWL format can be seen at the project site[2].

Fig. 1 sketches the relevant aspects of the system architecture and shows resources and modules which deal with the domain knowledge. The ontology is the central static resource. The granularity of the concepts reflects the granularity of terms as the main application task is to provide self-tuition in terminology learning. The ontology contains:

- *a type hierarchy* - about 300 concepts in the area of finances, labeled by the actual English terms whenever possible, and
- *associated propositions* - about 150 statements in first-order logic (FOL), which encode facts that are relevant to the terminology learning task.

The ontology labels as well as the identifiers of the FOL propositions are inserted in the annotation of the pedagogical resources. There are two kinds of learning objects (LO):

- *static exercises* specially designed by a teaching expert to check user's comprehension of the domain. Some exercises require short answers in free English, the latter are checked by an original prover (details about the prover are given in [3,6]);
- *relevant readings* dynamically collected from Internet and filtered according to their relevance to terms that might be "unknown" at particular learning situations [3].

The student reactions are reflected in the Learner Model (LM) and special cases of misconceptions are stored for later consideration. Elements of the LO

[2] http://www.larflast.bas.bg

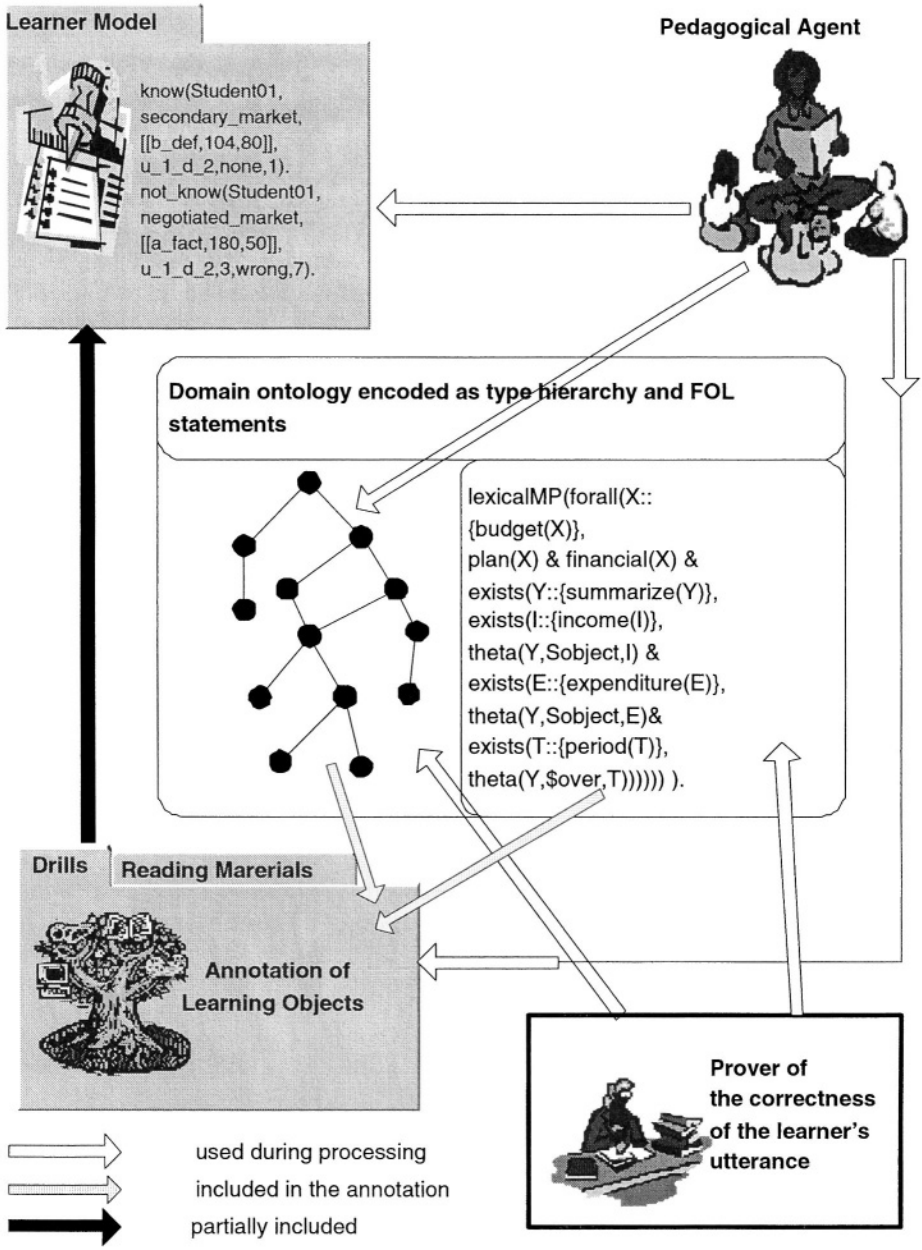


Fig. 1. Domain ontology and its usage in system resources and processing modules

annotation are repeated in the learner model, to facilitate the Pedagogical Agent (PA) and other modules at remote servers to track the learner's performance. According to the learner model content, the pedagogical agent plans the next learner's step and shows some drill or relevant reading. A sophisticated prover checks the correctness and appropriateness of the learner's utterances [3]. Please note that the pedagogical agent treats the FOL statements as "Identifiers" as long as they are included in the LO annotation, while the prover really employs them in the reasoning process.

2.2 Tagging the Learning Objects

The learning object is an atomic pedagogical content. Its annotation contains:

- `course identifier`,
- `topic identifier`,
- `unit identifier`,
- `Kind of LO` ("exercise" or "reading"),
- `List_of_TestedAspects` - empty for "reading" LO and list of triples for an "exercise"LO: `[Aspect, Proposition_Id, Weight]`, where
  - `Aspect` represents major types of aspects: *(i)* `b_def:` basic definition, *(ii)* `a_fact:` additional fact, `(iii) rel:` encodes possible relations i.e. object, agent, attribute, characteristic, instrument and etc.
  - `PropositionId` is a Id-number of an ontology statement and
  - `Weight` is a predefined number between 1 and 100, which reflects the importance of the tested information regarding domain understanding while teaching foreign language terminology.

  Sometimes there are more than one aspects of a concept that can be tested for an exercise entry which are encoded in separate triples.
- `ConceptLabels` - one concept (one term) for "exercise entry" and a list of pairs *[concept, relevance score]* for "reading" (because one text can be relevant to several terms),
- `Autorship`,
- `CreationDate`.

Annotations of the exercises are manually encoded when the object is created. They link the LO to the ontology items and indicate which domain concepts and facts are described or tested by every learning object. The FOL prepositions are used as references and enable system's reactions by pre-defined feedback. The concept weights for exercises are assigned by the teaching expert. In contrast to exercises, the readings are automatically annotated by concept labels that are English terms. The phrasal terms are assigned to readings with corresponding relevance score which is automatically calculated by a supervised information retrieval technique called Latent Semantic Analysis. It would be good to have a deeper annotation at the level of the FOL statements, i.e. to know the facts discussed in each text, but unfortunately the methods for "similarity" calculation cannot distinguish finer facts in the same domain (this is how the information retrieval works today especially with the most popular bag-of-words models).

2.3 Learner Model

The learner model keeps clauses to describe learners' familiarity with the terminology which is closely related to domain knowledge:

- **know** - the learner knows a term; the clause is inserted for correct answers;
- **not_know** - the learner doesn't know a term; the clause is inserted for wrong answers;
- **self_not_know** - inserted when the learner chooses the "don't know" answer to certain LO, if such answer exists; and
- **know_wrongly** - the learner's knowledge is considered wrong (eventually, might need corrections); the clause is inserted for partially correct answer to a certain LO.

Each of the learner model clauses has seven arguments:

- *user name,* logging identifier of a user;
- *ontology concept,* the tag `ConceptLabels` from the LO annotation;
- *tested facts,* the tag `List_of_TestedAspects` from the LO annotation;
- *exercise identifier* (an unique Id corresponding to the tags cource `identifier`, `topic identifier` and `unit identifier` from the LO annotation) ;
- *counter* how many times the user passes trough the tested learning object;
- *indication of linguistic and conceptual mistakes* (more details in [3]) and
- *unique index* for tracking the whole dialog history.

There are two sample LM clauses at Fig.1: that 'student01' knows the basic definition of secondary market (at the 1st pass through the exercise) and does not know an additional definition of negotiated market (at the 3rd pass through the same testing object). It is worth to emphasize that references to ontology enter the learner model and the pedagogical agent via the LO annotation and thus both the learner model and the pedagogical agent are domain independent. Manual efforts need to be invested only for the ontology acquisition and the LO annotation; afterwards the labels of ontology items are propagated through all other resources and components at no cost. The learner model is also used by system modules which are not presented here (see [3] for discussion of the dynamically generated explanatory www-pages which contain term definitions and concordancers of term contextual usages for wrongly-known terms).

2.4 Pedagogical Agent

The learning environment presented here is a self-tuition framework, which compliments the classroom activities. The prototype supports the student to accomplish drills and to read suggested training materials on relevant subjects. There is a default sequencing of performing drills (passive sequencing) but the pedagogical agent offers an alternative presentation which is tailored to the learner's

competence as it is recorded in the learner model (active sequencing). The dynamic archive of updated readings enables choices within a growing corpus of relevant texts that may be suggested to the student.

The pedagogical agent has two main strategies for active sequencing: local and global. The local strategy plans the movement between drills testing different characteristics of one concept. Its main goal is to create a complete view about learner's knowledge concerning this concept. This strategy chooses exercises with increasing complexity when the learner answers correctly and it gives again previously completed drills if the learner has performed poorly. For instance, if a student does not know some fact related to the tested concept (term) which is encoded in the exercise annotation with low weight, the pedagogical agent will suggest a reading. The global strategy plans the movement between exercises testing different concepts, according to their place in the financial ontology. For instance, if the student does not know the basic definition of a concept and its major additional facts, the pedagogical agent will choose to test first whether the student knows at all the super-concepts and only afterwards to suggest basic readings for the unknown concept.

As shown at Figure 1, the pedagogical agent chooses the next learner's movement depending on:

- the annotations of available learning objects,
- the position of the tested concept in the type hierarchy, and
- the current LM user's status: history and quality of learner's performance.

2.5 Discussion

Today there is no cognitive theory of human learning which is elaborated enough to support the development of standardised models of the eLearning activities. Rather, there are partial solutions to separate tasks in particular implementations. We claim that without explicit domain knowledge, the semantics of the learning objects can be described in general terms only. In our view the domain ontology is an inevitable part of advanced learning solutions as it:

- structures in a natural way the learning content and provides a backbone for unification of the granularity of all kinds of learning objects;
- enables knowledge-based solutions to complex tasks (e.g. checking the correctness of learner's utterances in natural language);
- allows for clearer diagnostics of the student misconceptions and supports a consistent, domain independent strategy for planning the adaptive behavior of the system;
- provides annotation markers that might facilitate the interoperability and exchange of learning resources.

To support these claims, let us have a closer look to the Larflast ontology and the gains of its central place in the prototype. Acquiring the ontology and integrating all the processing components was an effort-consuming task which

took almost three person years. The main difficulty was to harmonise the exercises allowing answers in free English and the mechanisms to generate consistent feedback and proper error diagnostics. In other words, the user expects some consistency in the interface appearance and the system reactions, so much efforts were invested to assure "fluent" diagnostic dialogs after semantic errors are encountered and to design the consequent moves. So proving the correctness of the learner's utterance is a complex tasks and its integration in CALL turned to be complicated as well. Fortunately we discovered simpler solutions which are also very beneficial for the user and facilitate his/her training performance.

At first we noticed that the acquisition of the type hierarchy itself is relatively easier, moreover updates and changes do not affect much the already elaborated components and resources. The type hierarchy in Larflast encodes partitioning perspectives which might look relatively complex to non-specialists but it is never shown completely to the learner, which would be the case of manual navigation through the conceptual content. The pedagogical agent, despite its simplicity, takes care to guide the student through the hierarchy. The sequence of moves was evaluated positively by the learners in the final user study. In addition, while planning further "readings", the pedagogical agent tries to offers materials containing *known terms* which is possible because of the unified backbone for annotation of all learning objects [11]. This fact pleased the students as well as the teaching experts who evaluated the prototype.

Another interesting resource is the dynamic archive of readings which is automatically annotated by the same ontological labels. Obviously, manual search in Internet (e.g. via google with the same phrases) would deliver the same documents but ordered in a different ranking. Moreover, the system deletes the documents that seem to contain much non-textual information. The supervised methods for calculating relevance are better than simple search by key words and allow for boolean requests that avoid the usage of unknown terms. In this way annotating the readings by the labels of the same type hierarchy enables personalised information retrieval. Integration of heterogeneous components via one simpler conceptual resource ensures a well-balanced functionality of the whole environment.

3 Towards Interoperability and Visualisation

Recently we align the Larflast prototype to the advanced research on Semantic Web and eLearning. The first step was to look more deeply to ontology interoperability from the Semantic Web perspective, moreover several components in our learning environment are independent from the particular ontology we use. However, there are serious obstacles to make further progress towards interoperability of learning objects, due to the lack of aligned ontologies in the domain of finances. Another recent development concerns the visualisation of domain knowledge to the learners. In this section we sketch the ongoing work in these directions.

3.1 Interoperability via Ontology Alignment

The LO content should be exchangeable between different providers and the exchange is to be ensured by the metadata annotations, which obligatory include descriptions of the LO semantics. For instance, the LOM standard [12] includes field for semantic description. The annotation presented in section 2.2 contains reference to semantic descriptions too (but is not encoded according to some of the existing standards, as the prototype was implemented before the appearance of the LO standards). So as a first guess, interoperability between learning objects might be achieved at the level of their semantic content in case that we find efficient methods to "compare" and align it.

We studied the publicly available ontologies, to find out some resource similar to the one we developed in finances, in order to make experiments with the content of our LO. We considered in depth the public financial "branch" of the MILO ontology [13] which contains all concepts mentioned at least three times in the Brown corpus and additionally, is relatively easy for people to understand and practical for computational use. MILO is integrated under SUMO [14] and its financial part covers 246 basic entities (terms). In Teknowledge's ontologies, terms belong to one of five basic types: class, relation, function, attribute, or individual. We manually mapped MILO to the financial ontology we developed for educational purposes in the Larflast project. Three quarters of the terms coincide, however the design decisions are quite different. For instance, in MILO *dividend* is a relation while for us it is a concept. It looks difficult even for knowledge engineers to formulate precise rules how to align the semantics of concepts to relations. Mapping of too different ontologies will require complex semantic resources, which define better the meanings, and in-depth inference. Additional study shows that the ontological design and choices differ drastically even for partitioning in simple taxonomies. The overview of existing public constructions displays primarily the multiple perspectives to classification of reality objects, relations and their attributes and features. The reader may consider as an example three taxonomies in computer science: *(i)* the ACM taxonomy of keywords, designed by the IEEE Computer Society [15], *(ii)* a taxonomy in Computer Graphics Interfaces, developed within a NSF-funded project as a searchable database index for a large part of the ASEC Digital Library [16] and *(iii)* the SeSDL Taxonomy of Educational Technology, which gives links to the British Educational Thesaurus BET [17]. These three hierarchies are built for different purposes and a careful study of their concepts shows that less than 5% of the nodes are common. As a rule, even for these "intersecting" concepts the classification into subconcepts is different. Similar variety exists concerning the emerging ontologies.

In this way, semantic interoperability of LO is not a trivial task and it will requite much efforts for the elaboration of agreements between the providers of semantic content.

3.2 Visualisation Scenario

Another promising direction while learning foreign-language terminology is the visualisation of domain knowledge. This could be useful for second language learners or domain novices who read a document and do not grasp the meaning of unknown or specific terms.

In our case, at the presentation interface to the learners, the 'reading' LOs are displayed as html-pages which are semantically indexed by the terms - labels of the type hierarchy sketched in Fig. 1. Recently we implemented a tool supporting the hierarchy visualisation [4]. The hierarchy itself appears in a graphical window with the classification perspective displayed in different colors and associated comments. The logical statements attached to a chosen concept node can be seen by clicking the right mouse button. At present we have implemented the following visualisation functionality: Suppose a student is reading a page concerning financial instruments, but some of the domain specific terms are unknown to him/her and then s/he could 'load' the underlying ontology. At this stage the semantic index is visualised in the text and a special window appears, which contains the list of all terms anchored to ontology labels. By just clicking on a term from the list, the term is highlighted in the hierarchy and by right-clicking on the same list term, the user can view definition, properties, attributes and relations associated to this term. Although the FOL-format may be really discouraging for end users, it is beneficial to see the conceptual environments in the hierarchy - sister concepts, super- and sub-concepts, etc. Initial user evaluation suggests that simple visual hints please both teachers and users, who dislike complex drawings that focus their attention to activities which are not directly related to language learning and language comprehension. All in all by selecting an item from the list the user can see all the appearances of this term highlighted in the www-page (i.e the language context of its usage) and also its position in the hierarchy of ontology terms (i.e its ontological environment).

4 Conclusion

The Larflast project was long enough (3 years) to allow for two-three development iterations: knowledge acquisition and components design, implementation, and evaluation. Starting with the knowledged based approach from the very beginning, we gathered much experience how to acquire and tune domain ontologies to eLearning applications. As we already said above, the taxonomy as such is a very good conceptual resource, especially when it incorporates concept definitions for notions with different granularity and multiple perspectives [4]. Many components in our learning environment work with the taxonomy only because it turned out that this relatively "flat" conceptual resource can be successfully used as a unifying backbone across all the system resources and components. We believe that our approach to structuring and processing taxonomies opens the door to domain independent architectural solutions in eLearning. In this way the acquisition of sophisticated knowledge chunks (which is always domain-dependent) is not necessary for initial prototypical developments and it makes

sense to start with simpler tasks and to carry out initial user studies with simpler prototypes. We encountered as well that the reasoning is not an obligatory instrument; sensible functionality can be achieved by simpler operations like type expansion, type contraction and projection without making proper inferences. We also learned that the whole is more important than the separate components and that both the students and the teaching experts need to see a broader operational context in order to evaluate relevantly the system functionality.

Regarding the interoperability of learning objects, it seems that the progress in this field depends on the progress in ontology development and alignment, which is one of the main challenges in the semantic web. Current educational standards look rather human-oriented to offer effective solutions for formal encoding of semantic content, e.g. LOM is originally designed as a platform for publishing and exchange of teaching programs of higher education institutions between human experts. New views to LO semantic interoperability will probably evolve, inspired by the expected evolution of ontology merging and alignment algorithms. On the other hand, the partial solutions are feasible as well, when sub-groups of LO content providers may agree to apply coherent conceptualisations. In our view the taxonomies might be important knowledge structures in this process as their relative simplicity and readability allow for easier negotiation. (In fact the taxonomies correspond to the thesauri in the paper-based world; it is much easier to agree on a thesaurus than on a holistic textbook content). Our future plans in this respect include more experiments with different ontologies and their taxonomies, to better estimate the hierarchies as a resource providing interoperability of LO as well as domain independence of eLearning implementations.

Regarding the visualisation, we assume that graphical representations are a necessary component of the advanced multimodal interfaces. The user-friendly and intuitive visualisation of conceptual contexts is very important because it facilitates domain comprehension not only in language learning but in any kind of getting acquainted with a new domain. At present we adopt an earlier tool for acquisition of knowledge bases [18] to the needs of visualisation in language comprehension and learning, as it was sketched in section 3.2. We plan to continue this activity together with the necessary user evaluation.

Current eLearning systems need increased abilities in quality, quantity, and interaction. We believe that the solid grounds of the knowledge based approaches will facilitate further implementations. Some very serious questions remain open, especially the discouraging mismatch between ontologies acquired for the same domain. Perhaps the application of simpler knowledge structures is a feasible alternative that is worth to be investigated, until we find some balance between the desired usefulness of our systems and the human abilities to define knowledge formally.

References

1. Stutt A., Motta E.: Semantic Learning Webs. In Journal of Interactive Media in Education, May 2004: JIME Special Issue on the Educational Semantic Web, available from http://www-jime.open.ac.uk/.
2. Simon, B., Miklos, Z., Nejdl, W., Sintek, M. and Salvachua, J.: Smart Space for Learning: A Mediation Infrastructure for Learning Services. Proceedings of the 12th World Wide Web Conference, May, 2003.
3. Angelova G., Boytcheva, Sv., Kalaydjiev, O. Trausan-Matu, St., Nakov, P. and A. Strupchanska.: Adaptivity in Web-based CALL. In Proc. of ECAI-02, the 15th European Conference on AI, IOS Press (2002) 445-449.
4. Angelova G., Strupchanska, A., Kalaydjiev, O., Boytcheva, Sv. and I. Vitanova.: Terminological Grid and Free Text Repositories in Computer-Aided Teaching of Foreign Language Terminology. In Vertan, Monachesi, Jekat, v.Hahn (Eds.) Proc. "Language Resources: Integration & Development in elearning & in Teaching Computational Linguistics", Workshop at LREC 2004, May 2004, Lisbon, 35-40.
5. Boytcheva Sv., Vitanova, I., Strupchanska, A., Yankova, M. and G. Angelova.: Towards the assessment of free learner's utterances in CALL. Proc. "NLP and Speech Technologies in Advanced Language Learning Systems", InSTIL/ICALL Symposium, Venice, 17-19 June 2004, 187-190.
6. Angelova, G., Strupchanska, A., Kalaydjiev, O., Yankova, M., Boytcheva, Sv., Vitanova, I., and P. Nakov.: Towards Deeper Understanding and Personalisation in CALL. In Proc. Workshop 6 "eLearning for Computational Linguistics and Computational Linguistics for eLearning", COLING-04, Geneva, August 2004 (to appear).
7. Gamper, J. and J. Knapp.: Review of intelligent CALL systems. Computer Assisted Language Learning **15/4** (2002) 329-342.
8. Brusilovsky, P.: Adaptive navigation support in educational hypermedia: The role of student knowledge level and the case for meta-adaptation. British Journal of Educational Technology, **34 (4)** (2003) 487-497.
9. Dorr, B., Hendler, J., Blanksteen, S. and B. Migdaloff.: On Beyond Syntax: Use of Lexical Conceptual Structure for Intelligent Tutoring. In Holland, M., Kaplan, J. and R. Sams (eds.) Intelligent Language Tutors: Theory Shaping Technology. Lawrence Erlbaum Associates, Inc. (1995) 289-311.
10. VanLehn, K., Jordan, P., Rose, C., Bhembe, D., Boettner, M., Gaydos, A., Makatchev, M., Pappuswamy, U., Rindenberg, M., Roque, A., Siler, A. and Srivastava, R.: The Architecture of Why2-Atlas: A Coach for Qualitative Physics Essay Writing. In Proc. of the Int. Conf. on Intelligent Tutoring Systems, Springer, Lecture Notes in CS 2363 (2002) 158-162.
11. Kalaydjiev, O.: An Intelligent Pedagogical Agent in CALL. In Proc. of the Int. Conf. on User Modelling 2001, Springer, Lecture Notes in AI 2109 (2001) 283-285.
12. LOM - Learning Object Metadata, 2002, http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf
13. Nichols D. and Terry, A.: User's Guide to Teknowledge Ontologies. Teknowledge Corp., December 2003, ontology.teknowledge.com/Ontology_User_Guide.doc
14. Pease A., Niles I., and Li, J.: The Suggested Upper Merged Ontology: A Large Ontology for the Semantic Web and its Applications. In Working Notes of the AAAI-2002 Workshop on Ontologies and the Semantic Web, Edmonton, Canada, July 28-August 1 (2002) http://projects.teknowledge.com/AAAI-2002/Pease.ps

15. ACM Computing Classification System, www.computer.org/mc/keywords/keywords.htm
16. CGI Taxonomy, http://www.siggraph.org/education/curriculum/projects/Taxonomy2001.htm
17. SeSDLTaxonomy,www.sesdl.scotcit.ac.uk/taxonomy/ed_tech.html
18. Dobrev, P., Strupchanska, A. and Toutanova, K.: CGWorld-2001: New Features and New Directions. In Proc. CGTools Workshop in connection with ICCS-2001, Stanford, CA, 2001. Online access at http://www.cs.nmsu.edu/~hdp/CGTools/proceedings/papers/CGWorld.pdf

Towards a Global Component Architecture for Learning Objects: An Ontology Based Approach

Katrien Verbert, Joris Klerkx, Michael Meire, Jehad Najjar, and Erik Duval

Dept. Computerwetenschappen, Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
{katrien, jorisk, michael, najjar, erikd}@cs.kuleuven.ac.be

Abstract. This paper investigates basic research issues that need to be addressed in order to reuse learning objects in a flexible way. We propose an ontology based approach. Our ontology for learning objects defines content structures and relationships between their components. A conceptual framework for structuring learning objects and their components is introduced. Architectures like Horn's Information Blocks and the Darwin Information Typing Architecture are investigated as an approach to define learning object component types.

1 Introduction

Learning objects are often regarded as traditional documents. The interest is to re-use and re-purpose the learning object in its entirety. In many cases, a paragraph, a sentence or an illustration of a document is re-used by copy and paste in new and different documents. However, it is possible to reuse learning objects in a much more sophisticated way if we can access the components of a learning object and re-purpose them on-the-fly. This requires a more innovative and flexible underlying model of learning object components [DH03]. In earlier work we developed an abstract model that defines a framework for learning objects and their components [VD04]. Much more detail is required in order to develop a flexible architecture that enables on-the-fly composition of learning objects. It is necessary to develop an architecture that enables:

- Structuring of learning objects and their components
- Interactions between learning objects and their components

Separating structure from content is an important step towards reusability. In this paper we will focus on structural aspects of learning objects and their components. In the next section, we briefly outline our abstract learning object content model. In section 3, we further detail this content model and introduce a conceptual framework for an ontological approach for structural aspects of learning objects. Information architectures, like Horn's Information Blocks and the Darwin Information Typing Architecture, are introduced in order to define learning objects and their components in a precise way. Conclusions and remarks on future work conclude this paper.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 713–722, 2004.
© Springer-Verlag Berlin Heidelberg 2004

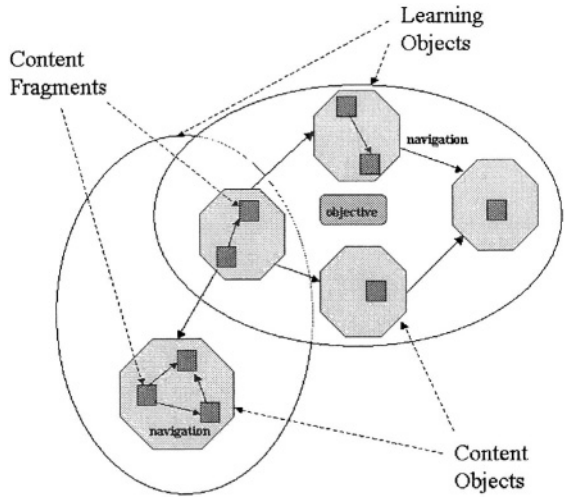


Fig. 1. Abstract Learning Object Content Model

2 Abstract Learning Object Content Model

We developed an abstract model that roughly outlines learning objects and their components [VD04]. Figure 1 illustrates the model. We distinguish between:

- content fragments
- content objects
- learning objects

Content fragments are learning content elements in their most basic form, like text, audio and video. They represent individual resources in isolation. Content objects are sets of content fragments. They aggregate content fragments and add navigation. A content object can include other content objects. At the next level, learning objects aggregate content objects and add a learning objective. Various learning object content models, like the SCORM Content Aggregation Model[1] and the Learnativity Content Model[2] are more or less a specific profile of our abstract model. We need this general model to develop an architecture that allows interoperability of learning objects and their components developed within a different content model. It would be nice if a SCORM learning object can reuse learning object components of a CISCO[3] learning object and vice versa. In the next section, we will further detail our model.

[1] http://www.adlnet.org/
[2] http://www.learnativity.com/
[3] http://www.cisco.com/

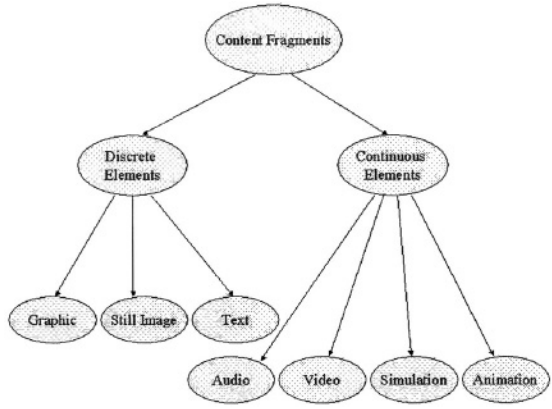


Fig. 2. Content Fragment Classification

3 Learning Object Ontologies

Separating structure from content is an important step towards reusability. Learning Object Content Models provide a common representation of learning objects and their components. Fine-grained descriptions of these learning objects and components are needed to allow for interoperability and on-the-fly composition. For example, if a learner needs a graph, it should be possible to search at component level rather than having to guess what type of learning object might contain a graph.

We will focus on structural aspects of learning objects and introduce a conceptual framework for an ontological approach. In this way, structured content with proper and consistent semantic labels can be created so that applications can understand the structure. The benefits of an ontological presentation lie in the capabilities of explicitly defining concepts, their attributes and relationships [QH04]. Thus, an ontology based approach can help us to more strictly define the learning object components we outlined in section 2.

3.1 Content Fragments

We can take our general learning object content model as a starting point to develop a learning object ontology. An ontology used to describe content fragments can make an early distinction between classes that describe continuous elements and classes that describe discrete elements. Subclasses of continuous elements can be audio, video, animation, etc. Discrete elements can be subdivided in still images, graphics and text. Figure 2 illustrates this classification.

3.2 Content Objects

Content fragments can be aggregated into content objects. Defining an ontology for content objects includes the definition of content structures and relation-

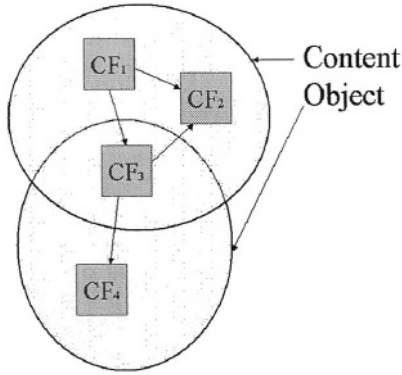


Fig. 3. Content Objects

ships between their components. Figure 3 represents two content objects, their components and relationships between them. In the next section, we introduce approaches to define these content structures and relationships in a precise way.

Information blocks. An ontology for content objects can be based on 'information blocks' developed by Horn [Hor98]. Robert Horn is often referred to as "the man who kicked the paragraph out of writing" [Nam00b]. He replaced the paragraph by a block, a chunk of information that is organized around a single subject, containing one clear purpose. A block can contain text, a list, a table, a graphic, an audio sequence, a video sequence etc. Horn has defined about 200 types of information blocks, including *analogy, checklist, classification list, comment, definition, objective, synonym* and *theorem.* He developed these categories based on seven types of information:

1. procedure
   A task or number of steps leading to a result
2. process
   Explanations, describing why a task or process is done
3. structure
   Descriptions, describing the structure of a physical, material object
4. concept
   Definitions and examples
5. principle
   A policy, rule telling what is allowed and what is not
6. fact
   Physical characteristics, proposition without proof or argumentation
7. classification
   Types of categories, sorting units into classes

The Darwin Information Typing Architecture (DITA). Content object types can also be based on the Darwin Information Typing Architecture (DITA).

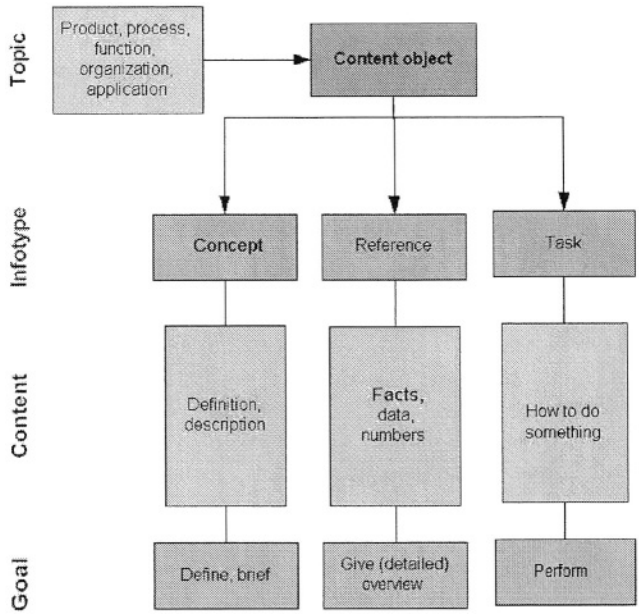


Fig. 4. DITA [Nam02]

DITA building blocks are, like Horn's information blocks, chunks of information organized around a single subject. Each building block has its own information type. DITA has four primary information types, a generic *topic* and three core information types [Pri01]:

- Concept
  An extended definition of a process, a function, an organization, an application, a theme, ... Typically a concept contains a title, some text, and maybe an example or a graphic.
- Task
  A number of steps, describing how to do something.
- Reference
  An overview of the constituent parts, a reference contains mostly data-oriented information that is often stored in a database (numbers, addresses, codes...).

DITA has defined a set of inheritance rules, that define how new information types can be created: e.g. each specialized information type must map to an existing one, must have characteristics of its parents information type, and must be more restrictive in the content that it allows.

DITA versus Information Blocks. DITA uses information blocks as an inspiration, since it uses principles of information blocks as a basic unit of information

Table 1. Horn and DITA information types

DITA Information Types	*IMAP Information Types*
Concept	Concept, Process, Structure
Task	Procedure
Reference	Fact
?	Classification
?	Principle

and information types. Whereas information blocks have a fixed set of information types, DITA information types are more flexible. Specialized information types can be defined, using the existing ones as a starting point. Horn's model refers to text and illustrations (as it is based on pioneering work in the mid 1960's), DITA deals with a much broader scope of issues [Nam00b].

The DITA architecture defines three information types, Horn defines seven information types. Horn's *procedure* type can be mapped on a DITA *task* type, and Horn's *fact* can be mapped on DITA's *reference* (see also table 1). The *concept, process* and *structure* type are regrouped in DITA's more general *concept* type. Horn defines a concept as definitions and examples, a structure describes the structure of an object and a process describes why a task is done. DITA's concept type is an extended definition of these major abstractions. It is not clear where we can situate Horn's *classification* and *principle.* Horn has defined information blocks like classification lists and a classification type. In general, DITA specifies less information types and information blocks, but provides the mechanism to extend these information types and blocks.

A DITA Example. Figure 5 illustrates content object types in DITA. These are well defined aggregations of several content fragments and other content objects.

- A concept content object contains a title block, a description (optional), a concept body, related links (optional) and 0 or more other content objects (topics). The concept body contains among other 0 or more definitions, sections, images and examples.
- A reference content object contains a title, a description (optional), a reference body, related links (optional) and 0 or more other topics. The reference body contains 0 or more properties, sections and examples. A property contains three optional elements: a type, a value and a description.
- A task content object contains also a title, a description (optional), a task body, related links (optional) and 0 or more other topics. The task body contains the following optional components: prerequisites, steps, a result, examples and postrequisites.

All these elements are predefined. We can also introduce other content objects, for example a *slide* content object. We define this content object as a subclass of the DITA *topic* class. A slide contains a title component, a description

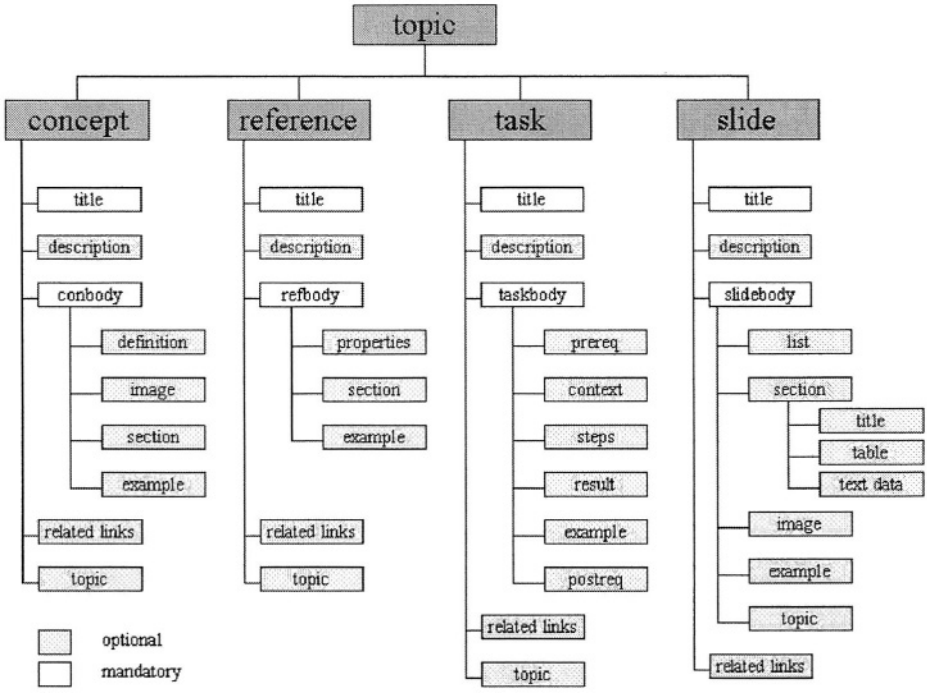


Fig. 5. Concept, task, reference and slide content objects

(optional), a slide body component and optional related links. We can further define these components. A title can contain content fragments like text, terms, booleans, states or images. The slide body can contain the following components: a list, a section, an image, an example and 0 or more other topics. A section is again predefined in the DITA architecture as containing a title, table, text data and other content objects or content fragments.

DITA defines aggregational relationships (*isPartOf, hasPart*). Content fragments and content objects should also be connected by navigational relationships in order to create (other) composite content objects. Figure 3 represents a very simple example of two content objects. The upper content object contains three content fragments (CF1, CF2 and CF3). The second content object reuses the third content fragment (CF3) and adds another content fragment (CF4). The third content fragment (CF3) *isPartOf* both content objects. Navigational rules are identified. The first content fragment comes before (*Prev*) content fragment two and content fragment three. The second content fragment comes after *(Next)* content fragment three. The result contains structural relationships like:

```
CF1 Next CF2
CF3 Next CF2
CF1 IsPartOf C01
CF3 IsPartOf C02
```

The aggregational relationships (*isPartOf, hasPart*) are defined within DITA. Navigational relationships *(Next, Prev)* can be expressed in the DITA architecture by using *map* elements. A map describes relationships among a set of DITA topics. The following examples of relationships can be described in a DITA map [lan03]:

- Hierarchical (Parent/Child): Nested topics create a hierarchical relationship. The topic that does the nesting is the parent, and the topics that are nested are the children.
- Ordered: Topics can be labeled as having an ordered relationship, which means they are referenced in a definite sequence.
- Family: Topics can be labeled as having a family relationship, which means they all refer to each other.

3.3 Learning Objects

We detailed content fragments and content objects. Content objects are aggregated into learning objects around a learning objective. We need to define learning object structures and relationships with their components. A presentation in the form of slides, a scientific report and an assessment presentation in the form of a test are examples of learning object types [Som04]. We can model a presentation as an aggregation of slides. Analogously, we can define questions and answers in the DITA architecture and model a test as an aggregation of these components. We are currently investigating some practical experiments to model these learning object types in the DITA architecture and using DITA against actual content.

3.4 Use Case

The following example represents an instance of the slide content object we defined in section 3.2. The slide contains a title and an image component. With this common specification for learning object components, we can make useful aggregations and disaggregations.

```
<slide id="a1">
   <title>An example of a slide component</title>
   <slidebody>
      <image href="example.gif">
   </slidebody>
</slide>
```

Suppose a learner wants to reuse an image of a slide. He can manually copy and paste the image into the new learning object he is creating. Our common representation for components however, allows to automatically assemble the image into new contexts. Applications can be developed that understand this structure and can access the components of the slide. The alternative, manually

cutting and pasting the image into the new document you are creating, is more time-intensive and error-prone.

A more extensive scenario involves different types of learning objects. A slide presentation and a scientific report both contain components like images, examples and related links. A common representation for these components allows us to create interoperable learning objects. An example used in a scientific report can be dynamically assembled into a presentation slide and vice versa. Suppose a learner wants to repurpose an example of a report and an image of a slide presentation. He has to open an appropriate application for the report, search the component he wants to reuse and copy it. The same thing for the image in the presentation. If we convert the report and presentation to our common representation, one application can be used to list all components and a user can select the components he needs. Making new learning objects according to an XML specification is easy, the main challenge will be converting all existing documents (sometimes with terrible formatting).

4 Conclusions and Future Work

We detailed our content model to some extent, identifying structured components and relationships. We investigated the Darwin Information Typing architecture and Horn's Information Blocks as an approach to define learning object component types. The advantage of using DITA is that the architecture is very flexible, any type of content structure can be defined. The major drawback is that there is still a lot to be build, DITA is not yet supported by a set of tools. Horn's information blocks are much further evolved, supported by a set of tools and well documented. The weakness of this architecture is that it is mainly an authoring environment (focusing on Word templates and writing principles), rather than an overall information architecture like DITA (dealing with linking, transformations, etc.) [Nam00a].

The content structures used in above mentioned examples can be easily defined within the DITA architecture. This kind of flexibility needs to be investigated for the architecture of Horn. We are currently investigating some practical experiments, using these architectures against actual content. The long term objective of this work is converting any type of document (MS Word Document, PDF document, etc.) to a common representation, allowing to create real interoperable learning objects.

Acknowledgements. We gratefully acknowledge the financial support of the K.U.Leuven Research Fund, in the context of the BALO project on "BAsic research on Learning Objects".

References

[DH03] E. Duval and W. Hodgins. A LOM research agenda. In *Proceedings of the twelfth international conference on World Wide Web,* pages 1–9. ACM Press, 2003. http://www2003.org/cdrom/papers/alternate/P659/p659-duval.html.html.

[Hor98] R. E. Horn. Structured writing as a paradigm. In *Instructional Development: the State of the Art. Englewood Cliffs, N.J.,* 1998.

[lan03] *DITA Language Reference. Release 1.2. First Edition,* May 2003. http://xml.coverpages.org/DITALangugeReference20030606.pdf.

[Nam00a] Namahn. Darwin information typing architecture (DITA): a research note by Namahn. http://www.namahn.com/resources/documents/note-DITA.pdf, 2000.

[Nam00b] Namahn. Information mapping: a research note by Namahn. http://www.namahn.com/resources/documents/note-IM.pdf, 2000.

[Nam02] Namahn. Content mapping: a research note by Namahn - Version 3.0. http://www.namahn.com/resources/documents/note-CMap.pdf, 2002.

[Pri01] M. Priestley. DITA XML: a reuse by reference architecture for technical documentation. In *Proceedings of the 19th annual international conference on Computer documentation,* pages 152–156. ACM, ACM Press, 2001. http://xml.coverpages.org/DITA-ReuseByReference.pdf.

[QH04] J. Qin and N. Hernandez. Ontological representation of learning objects: Building interoperable vocabulary and structure. In *Proceedings of the thirteenth international conference on World Wide Web,* pages 348–349. ACM Press, 2004.

[Som04] L. Sommaruga. An approach to content re-usability and adaptibility in e-learning. In *Proceedings of the ED-MEDIA 2004 World Conference on Educational Multimedia, Hypermedia and Telecommunications,* pages 2098–2104. AACE, AACE, 2004.

[VD04] K. Verbert and E. Duval. Towards a global architecture for learning objects: a comparative analysis of learning object content models. In *Proceedings of the ED-MEDIA 2004 World Conference on Educational Multimedia, Hypermedia and Telecommunications,* pages 202–209. AACE, AACE, 2004. http://www.cs.kuleuven.ac.be/~hmdb/publications/publicationDetails.php?id=41315.

Human-Generated Learning Object Metadata

Andrew Brasher and Patrick McAndrew

UserLab, Institute of Educational Technology, Open University, UK.
{a.j.brasher, p.mcandrew}@open.ac.uk

Abstract. The paper examines how to address the need for a production process for e-learning resources to include human generated metadata, and considers how users will exploit this metadata. It identifies situations in which human production of metadata is unavoidable, and examines fundamental problems concerned with human metadata generation such as motivation and shared understanding. It proposes and discusses some methods to exploit endemic motivational factors within communities in an attempt to ensure good quality human generated metadata, and identifies how ontological constructs can support the exploitation of such metadata. The relevance of these methods to the semantic web in general is discussed.

1 Introduction

In common with metadata schemas from many other domains, learning object metadata proposals such as the IEEE standard for learning object metadata [1] are specified terms of descriptors related to particular aspects of the resource being described. In any situation in which metadata is to be generated, every such descriptor can be considered to be from one of two distinct categories of sources:

1. *Intrinsic sources* - those sources that are contained within the resource itself, which are a necessary part of the resource itself. Examples of *intrinsic sources* include format of a resource, and title of a resource;
2. *Extrinsic sources* – those sources that are not contained within the resource itself. Examples of *extrinsic sources* include personal or organisational views about the expected use of the resource (e.g. the IEEE LOM elements *educational level, difficulty,* [1]).

No matter which metadata schema one applies, and no matter which domain of knowledge one considers, these two categories of sources exist. The relative usefulness of each category will depend on the particular application under consideration.

Software applications exist which can take some forms of *intrinsic sources* as inputs and from these sources automatically determine relevant metadata descriptors. See for example the software tools listed by UKOLN (http://www.ukoln.ac.uk/metadata/software-tools/). Sources such as file size, file format, and location (e.g. the URL) of a resource fall into this automatically determinable category. In addition, resources for which the semantic content is

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 723–730, 2004.
© Springer-Verlag Berlin Heidelberg 2004

primarily textual contain many *intrinsic sources* from which descriptors related to the content can be generated automatically through the use of suitable software performing e.g. linguistic analysis on this *intrinsic source* material [2], [3], [4] and through techniques such as latent semantic indexing which make use of the properties of collections of material [5]. Many web pages and word processor documents belong to this primarily textual group. There is also technology available to extract metadata from printed textual material [6].

However, for other sorts of resources i.e. those for which the semantic content is not primarily textual (e.g. sound files, movie files, and multimedia resources) it is more difficult to extract descriptors automatically [7] or to obviate the need for descriptors by using other methods (see e.g. papers on visual query of image and video content in [8]). Furthermore, no matter which modality the semantic content of e-learning resources is encoded in (be it text, sound, video) there are many useful characteristics of e-learning resources which are not automatically determinable, but require human intervention because they reside in *extrinsic sources* and are characteristics of the *expected use* or *actual use* of a resource [9].

There are thus two types of sources which require human intervention to create metadata descriptors:

i. *extrinsic sources*
   and
ii. *intrinsic sources* within non-textual resources.

2 Relevance to Learning Object Metadata

Of the 11 descriptors in the IEEE LOM's educational category, 6 are defined as characteristics of expected use, i.e. must be generated by humans ('Intended User Role', 'Context', 'Typical Age Range', 'Difficulty', 'Typical Learning Time', 'Language') and 5 will require human intervention if the resource is non-textual ('Interactivity Type', 'Learning Resource Type', 'Interactivity Level', 'Semantic Density', 'Description').

The purpose of metadata such as this is to facilitate discovery and (re)use of resources by enabling computer systems to support the identification by users of e-learning resources relevant to the users' needs. In general, a system that exploits metadata to this end will involve people, computer systems, the resources that the metadata describes, and the metadata itself; hereafter we will refer to such a system as a 'Metadata Exploitation System'. The computer systems within such a Metadata Exploitation System will invariably include search and retrieval algorithms, and the complexity of design and performance of these algorithms will be influenced by the expected quality of the metadata. One crucial aspect of quality is accuracy, and the accuracy of any metadata descriptor will depend on the system which creates it. This means that to optimise the performance of the Metadata Exploitation System one must optimise the performance of the system which creates the metadata i.e. it is necessary to consider two systems

1. A system for exploiting the metadata: the Metadata Exploitation System
2. A system for creating the metadata: the Metadata Creation System

and how these systems will interoperate. Both the exploitation and creation systems will involve people, computer systems, the resources that the metadata describes, and the metadata itself.

Experience at the Open University and elsewhere [10, 11] has shown that human produced metadata is often neither complete nor consistent. If the quality of metadata is assumed to be related to its utility in retrieval systems, then factors which influence the quality of human produced metadata can be considered to fall into three categories:

- **motivation of the producers** – how is the metadata to be useful to those who are producing it?
- **accuracy** – does the metadata describe the resource fully and as intended by the designers of the system(s) which will exploit the metadata?
- **consistency** – can the metadata produced by different people be interpreted in the same way by the systems which will use it?

For the first of these categories, motivation, it is clear that expectations of increases in efficiency of production of resources (e.g. via reuse), or value of the produced resources (e.g. via personalisation), is providing an incentive for *organisations* to include production of metadata within their workflow for production of e-learning resources. For some organisations the existence of standards (e.g. [1]), systems for handling stores of resources (e.g. [12]) is providing an additional incentive. However, for *individuals* within the production system the motivational factors can be more difficult to determine: if benefits are only in reuse or the subsequent finding of objects by a third party the value perceived by the producer will be low at the time the object is produced. Solutions to accuracy and consistency within an organisation can take a central approach, for example by focusing the work of adding metadata on a few individuals such as information science specialists who bring cataloguing experience. In such systems, where metadata is seen as an additional aspect often added at the end of a workflow there is a risk of adopting simplified systems and missing opportunities to exploit the metadata descriptions in the initial production of the materials (i.e. before the materials are made available to students or teachers). The characterising of educational material also requires specialist knowledge to determine 'expected use'. The experts in 'expected use' are the users (e.g. teachers, tutors) and authors of the material itself, thus this community of practice needs to be engaged to both properly create and exploit these descriptors. Individual motivation needs to be encouraged through identifying advantages during creation, community motivation empowered through developing effective sharing of ontologies and vocabularies used inside the community.

3 Mechanisms to Exploit Motivational Factors

In this section we outline how an approach based on structured vocabularies may be used in the design of Metadata Creation Systems, and then describe an approach to considering motivational factors. Finally we examine how a method building on a formal description of two communities shared understanding may overcome limitations which can limit the previous approach to single community.

A community of practice (e.g. staff working in a particular faculty at a particular University) can use the educational category of the IEEE LOM to accurately and consistently describe educational resources they produce, given suitable tools (e.g. as described in [13]). Structured vocabularies have been shown to improve consistency amongst metadata creators, in particular Kabel et al. report the positive effect of vocabulary structure on consistency [14]. It is clear that a vocabulary structured using (for example) the IMS Vocabulary Definition Exchange (VDEX) specification [15] can provide extra semantic information than a list of terms thus facilitating accuracy. Furthermore, we propose an approach to rapid prototyping of Metadata Creation Systems can be based on the creation and iterative development of structured vocabularies. For example, vocabularies structured according to e.g. the IMS VDEX specification can be exploited in the prototyping process via transformations which facilitate evaluation and iterative development of

1. vocabulary terms and vocabulary structure
2. 'Help' information for metadata creators.

The sort of transformations we are referring to are typically XSL [16] transformations which will operate on a structured vocabulary to produced e.g. XHTML user interface components which enable the vocabulary to evaluated by the relevant user group.

However, we argue that the motivation of individuals who are called on to create metadata must also be considered, and that better results should be produced by such communities if the descriptors individuals within the community in question are asked to produce are also of use to the individuals themselves. This implies the necessity for analysis of the tasks normally carried out by individuals e.g. using methods such as the socio-cognitive engineering method described by Sharples et al. [17], to identify how individuals themselves could exploit descriptors intended for end users. Thus vocabularies and structures which are developed (or identified) for use within a Metadata Exploitation System should form the basis from which the development of vocabularies for the Metadata Creation System begins, and proceeds via the prototyping process described above. However, such a task analysis may yield alternate generation mechanisms.

To demonstrate this we consider an example based on existing learning material: a collection of learning objects created for the Uk-eu masters-level course "Learning in the connected economy". This course is part of a Postgraduate Certificate in Online and Distance Education which was offered by the UKeU (see [18] for an overview of the course, and [19] for a description of the learning object structure of the course).

In the case of the learning objects created for the course "Learning in the connected economy", the course team that created the material had a primary typical target audience in mind (note: we use the term 'course team' to refer to the team of authors and editors responsible for creating material for a particular course), and characteristics of this audience were described on the course web site. The details of this description need not concern us, but they included such requirements as "a first degree" and "proficiency in English". In developing the learning objects for this course, one of the pedagogic requirements that the course team for this course had to comply with was that students should be given an indication of the "estimated study time" that each learning object should take to complete. It is apparent then that this "estimated study time" figure was generated by the content authors in accordance

with the guidelines suggested in the IEEE standard for the "Typical Learning Time" metadata element:

"Approximate or typical time it takes to work with or through this learning object for the typical intended target audience." [1].

Thus the "estimated study time" information required could be automatically generated from learning object metadata, if this metadata included the correct figure within the "Typical Learning Time" element. This then provides the motivation for authors to enter a reasonable figure for "Estimated study time".

With respect to the "Difficulty" element, the explanatory note in the IEEE standard states:

"How hard it is to work with or through this learning object for the typical intended target audience." [1], and that with respect to both the "Typical Learning Time" and "Difficulty" elements the standard states

"NOTE—The "typical target audience" can be characterized by data elements
5.6:Educational.Context and
5.7:Educational.TypicalAgeRange." [1].

Next, we assume that within this context (i.e. the educational context characterized by the "typical target audience" referred to previously) that "Typical learning time" will be related to "difficulty" for this typical target audience. The exact nature of the relationship is not important – we merely assume that as "Typical learning time" increases, so does "difficulty". Now, assume that the authors of the course think that the range of levels of "difficulty" for this course is "very easy" to "difficult", i.e. they agree, as a community, that this range of descriptors of "difficulty" are apt. Then by analysing the "Typical learning time" information given to students (and entered into the metadata) for the complete course, this community of authors proposes a categorisation based on this "Typical learning time". This categorisation enables "Difficulty" metadata to be automatically generated for this particular educational context, and hence inserted into the relevant metadata record.

This metadata is sufficient to be useful within this particular educational context. It can enable learning objects to be sorted and compared in terms of "difficulty" (or "Typical learning time") for this context; knowledge of this context is implicit in the community of authors decision about the range of applicable descriptors of "difficulty", and their description of the "difficulty" of individual learning objects.

However, difficulty metadata that is generated by the mechanism described hereto is only valid for one educational context, and can only be correctly interpreted by people and systems that are 'aware' of this fact. With the resources described so far (i.e. the vocabulary and metadata resources) it is not possible for an algorithm to make useful comparisons between the difficulty of a learning object on a particular topic from this post graduate course with a learning object on the same topic from an undergraduate course in another subject area. To make such comparisons possible, extensions to the metadata and the creation and exploitation systems are necessary. For example, the IEEE LOM permits any number of Educational elements, hence a Metadata Creation System could implement a difficulty element for every educational context perceived to be of interest, and a Metadata Exploitation System could include an algorithm to perform the necessary comparisons. However, there are problems with this approach, not least the burden of creating this additional metadata (note that although the difficulty metadata can be automatically generated, the "Typical learning

time" metadata is still a prerequisite, i.e. for every context someone will have to ascribe a value for this, hence allowing the "difficulty" value to be generated.

We now propose a method to reduce the burden placed upon communities of authors, yet still enable the difficulty of learning objects to be compared across contexts.

Assume the course team responsible for creating material for context1 is team1, and that responsible for creating material for context2 is team2. These course teams can meet, discuss and agree how the difficulty of learning objects they have created for their own context will be perceived by students in the other context. For example, the teams may agree that learning objects created for context1 (e.g. postgraduate level) will be perceived as "more difficult" by students in context2 (e.g. undergraduate level) than the difficulty descriptors that have been applied by team1 for context1 appear to indicate. This agreement establishes a relationship between descriptions of difficulty in one context and perceptions of difficulty in the other. We propose that in general (i.e. for any pair of contexts and no matter what the exact nature of the contexts in question is), that the gamut of useful descriptors of this contextual relationship is:

- Assignments of difficulty made in this context **context1** are perceived as "more difficult" than assignments of difficulty made in this context **context2**
- Assignments of difficulty made in this context **context1** are perceived as "less difficult" than assignments of difficulty made in this context **context2**
- Assignments of difficulty made in this context **context1** are perceived as "as difficult" as assignments of difficulty made in this context **context2**

(note that in this case 'assignment' should be interpreted as 'the action of assigning', not as ' a task allocated to somebody as part of a course of study').

Providing there is metadata available which describes the difficulty of learning objects for these two educational contexts context1 and context2 (perhaps automatically generated by the mechanism described previously) then what is required is a mechanism for encoding the statements describing the relationships presented above so that they can be interpreted by machines and people.

We have developed a prototype ontology in OWL [20] which enables machine interpretation of such contextual relationships. The classes in this ontology are described in table 1.

All the individuals necessary for such an ontology to be exploited may be automatically generated from metadata records created for a particular context, except the ContextRelationship individuals. It is envisaged that these could be created e.g. by discussions and agreements between individual managers representing the course teams of the contexts in question.

4 Conclusions

Systems which seek to exploit metadata descriptions (e.g. a search and retrieval system) which have been created with more than one 'expected use' in mind must have an understanding of the semantics of the descriptions of 'expected use', and of

Table 1. Description of classes of concepts in the ontology

Difficulty class
Individuals of class "Difficulty" represent a level of difficulty that can be assigned to individuals of class "LO". A particular individual (i.e. level of difficulty) can be related to other individuals via the slots 'less_difficult_than' and 'more_difficult_than'.
Assignment class
The Assignment class describes assignments of individuals of "Difficulty" to individuals of "LO" by a particular actor in a particular "Context". For example, the individual of Assignment identified as Henry Hall should be interpreted as: "Henry Hall says that learning object individual lo2 is 'difficult' for students in the 'Physics' context.".
Context class
Individuals of the class "Context" represent the context (e.g. educational level and study skills), that individuals of class "Assignment" maybe assigned to.
ContextRelationship class
The "ContextRelationship" class describes relationships between assignments of difficulty in different contexts. An example of an individual is e.g. R1: "For students in the Maths context, the assignments of difficulty made for students in the Physics context are less difficult."
LO class
Individuals of the class LO represent learning objects, i.e. they are references to metadata records which describe the relevant learning object.

the semantic relationships between these values of the descriptors. In most current systems this understanding exists outside of the computer system, in the shared knowledge and understanding of the communities that both create and exploit the values via the computer system. For computer systems to do this without human intervention, there must be a machine interpretable description of the relationships between the contexts of 'expected use'. Formal languages which enable such machine interpretable descriptions to be realized can help address this e.g. by implementing representations of communities shared understanding of relevant aspects using the Web Ontology Language, OWL [20]. The use of OWL was not necessitated by the requirements of the problem we have addressed herein; other languages could have been used. However, we realise that organisational motivation is needed to create and/or utilise an ontology such as the one described. One factor in the choice of the OWL language to implement the relatively simple (compared with what is possible with OWL) constraints and relationships necessary for this ontology is the expectation of availability of tools that can reason about them [21], which should enable widespread exploitation of context relationship instances implemented in OWL. Finally we remark that the situations which necessitate human-generated metadata for educational resources (i.e. extrinsic sources and intrinsic sources for non-textual resources) are exactly similar for resources from all other domains. Thus any semantic web application which has a reliance on metadata created from these sources could make use of the methods described in this paper i.e. to consider and exploit the motivational and community aspects of human metadata generation.

References

1. IEEE, *IEEE Standard for Learning Object Metadata,* in *IEEE Std 1484.12.1-2002*. 2002, The Institute of Electrical and Electronics Engineers, Inc. p. i-32.
2. Sebastiani, F., *Machine Learning in Automated Text Categorization.* ACM Computing Surveys, 2002. **34**(1).
3. Lam, W., M. Ruiz, and P. Srinivasan, *Automatic Text Categorization and Its Application to Text Retrieval.* IEEE Transactions on Knowledge and Data Engineering, 1999. **11**(6): p. 865 -879.
4. Liddy, E.D., et al. *Automatic metadata generation & evaluation.* in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval.* 2002. Tampere, Finland: ACM Press.
5. Deerwester, S., et al., *Indexing by latent semantic analysis.* Journal of the Society for Information Science, 1990. **41**(6): p. 391 - 407.
6. *METAe - Meta Data Engine.* Vol. 2003. 2002: The METAe project group. http://meta-e.aib.uni-linz.ac.at/.
7. Brunelli, R., O. Mich, and C.M. Modena, *A Survey on the Automatic Indexing of Video Data,* Journal of Visual Communication and Image Representation, 1999. **10**(2): p. 78-112.
8. Ngan, K.N., et al., eds. *Special issue on segmentation, description and retrieval of video content.* 1998.
9. Marshall, C. *Making Metadata: a study of metadata creation for a mixed physical-digital collection.* in *ACM Digital Libraries '98 Conference.* 1998. Pittsburgh, PA.
10. Currier, S., et al., *Quality Assurance for Digital Learning Object Repositories: Issues for the Metadata Creation Process.* ALT-J, research in Learning Technology, 2004. **12**(1): p. 6-20.
11. Chan, L.M., *Inter-Indexer Consistency in Subject Cataloging.* Information Technology and Libraries, 1989. **8**(4): p. 349-358.
12. GEM, *The Gateway to Educational Materials.* 2003. http://www.thegateway.org/.
13. Brasher, A. and P. McAndrew, *Metadata vocabularies for describing learning objects: implementation and exploitation issues.* Learning Technology, 2003. **5**(1). http://lttf.ieee.org/learn_tech/issues/january2003/.
14. Kabel, S., R.d. Hoog, and B.J. Wielinga. *Consistency in Indexing Learning Objects: an Empirical Investigation.* in *ED-Media 2003.* 2003. Honolulu, Hawaii, USA.
15. IMS, *IMS Vocabulary Definition Exchange*. 2004. http://www.imsglobal.org/vdex/.
16. Clark, J., *XSL Transformations (XSLT) Version 1.0.* 1999, W3C. http://www.w3.org/TR/xslt.
17. Sharples, M., et al., *Socio-Cognitive Engineering: A Methodology for the Design of Human-Centred Technology.* European Journal of Operational Research, 2002. **136**(2): p. 310-323.
18. UKeU, *Postgraduate Certificate in Online and Distance Education.* 2003/4. http://www.ukeu.com/courses/connectedeconomy/courses_connectedeconomy_programme.php?site=.
19. Weller, M., C. Pegler, and R. Mason, *Working with learning objects - some pedagogical suggestions.*2003.http://iet.open.ac.uk/pp/m.j.weller/pub/altc.doc.
20. McGuinness, D.L. and F.v. Harmelen, *OWL Web Ontology Language Overview.* 2003, W3C.http://www.w3.org/TR/owl-features/.
21. Abecker, A. and R. Tellmann, *Analysis of Interaction between Semantic Web Languages, P2P architectures, and Agents.* 2003, SWWS – Semantic Web Enabled Web Services project IST-2002-37134. http://swws.semanticweb.org/public_doc/D1.3.pdf.

Extracting Sub-ontology from Multiple Ontologies

Dazhou Kang[1], Baowen Xu[1,2], Jianjiang Lu[1,2,3], Peng Wang[1], and Yanhui Li[1]

[1] Department of Computer Science and Engineering, Southeast University, Nanjing 210096, China
[2] Jiangsu Institute of Software Quality, Nanjing 210096, China
[3] PLA University of Science and Technology, Nanjing, 210007, China
bwxu@seu.edu.cn

Abstract. Web ontologies are often too large to be used in a single application. The solution is to extract sub-ontology from large-scale ontology. An application may require several large-scale ontologies in different domains. Current approaches mainly focus on extraction in single ontology; it needs to integrate the ontologies and then extract sub-ontology from the integrated one. However, it is often difficult or even impossible to integrate large ontologies. Approach of extracting sub-ontology from multiple ontologies is required. This paper proposes a framework of extracting sub-ontologies from multiple ontologies, and suggests some first step notions. In our framework, we first translate different Ontology language code into a unified visualized model representing ontologies, and then divide the user demand to extract sub-ontology from each single ontology. We carry out the sub-ontology extraction based on the unified model. Finally, we integrate these smaller sub-ontologies into the ontology of the user demand. The result can be translated back to the Ontology language the user uses.

1 Introduction

The Semantic Web [1] is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. It enables intelligent information services, personalized web sites, and semantically empowered search engines.

The key technology in the Semantic Web is Ontology [2]. Ontology is shared conceptualization of some domain that formally defines concepts and relations among the concepts and has a set of axioms in order to inference. It is the basic of sharing and reusing knowledge on the Web.

Ontology creators attempt to model certain domains accurately and completely; however, this often leads to large-scale ontologies. For example, the Unified Medical Language System [3] ontology has more than 800,000 concepts and 9,000,000 relationships. Large-scale ontologies are hard to maintain and use.

An application may only need a small part of a large-scale ontology [4]. Using the whole ontology will greatly increase complexity and redundancy, and reduce efficiency. Extracted form the original ontology, a smaller and simpler sub-ontology

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 731–740, 2004.
© Springer-Verlag Berlin Heidelberg 2004

can make the application more efficient [5]. The sub-ontology only contains the particular parts of the whole ontology required by the application; the application do not benefit from other outlying information.

Early researches study generating application-focused databases from large ontologies [4]. Sub-ontology extraction is a new research area. Some researchers point out sequential extraction process called Materialized Ontology View Extraction [5], [6]. Under application requirements, MOVE support optimization schemes to guarantee sub-ontology with high quality. However, it proves to be computationally expensive. *Subsequent researchers* proposed a distributed approach [7] to the sub-ontology extraction process to decrease cost of sub-ontology extraction from large complex ontology. They also analyze the semantic completeness issue [8] in their method. GoPubmed system [9] presents the relevant sub-ontology for browsing GeneOntology; it shows the extraction of sub-ontology is profitable.

Current methods only focus on extracting sub-ontology from single ontology. However, in many practical cases, especially in the web environments, we often face multiple ontologies in one application [10]. Some of the related ontologies or the integration of them may be too large. It faces the same problem of complexity and efficiency. Using sub-ontology can solve the problem. Nevertheless, there are no existing methods to extracting sub-ontology from multiple ontologies. The methods for single ontology are not able to solve many new problems with multiple ontologies.

This paper proposes a framework of extracting sub-ontologies from multiple ontologies, and suggests some first step notions. Section 2 discusses several critical problems in extracting sub-ontology from multiple ontologies. Section 3 introduces our framework. Section 4 discusses the extracting process in detail. Section 5 gives the conclusion.

2 Problems in Extracting Sub-ontology from Multiple Ontology

How to extracting sub-ontolgoy from multiple ontologies is actually an open question. When facing multiple ontologies, several problems rise.

2.1 Need for Unified Ontology Model

There are many different Ontology languages on the Web, such as OWL [13], DAML-OIL and Ontolingua. They are different in syntax and structure, and based on different logic foundations. Therefore, the analyzing, extracting and integrating methods of them are different. Translating ontologies into a unified internal representation, i.e. a unified Ontology model, is necessary.

In our idea, we first translate different Ontology language code into a unified model, and then do the extraction and integration based on this model. Therefore, both the extraction and integration process can ignore the different language issues. The result ontology should be able to translate into the language as user demands.

The unified model must involve most of the features in typical Ontology languages in order to represent ontologies as exactly as possible. It also has to be simple because we need to do many extraction and integration work on it. Especially to make the extraction easier, the elements in Ontology need to be organized in a modularized model. The model should also be visualized for browsing ontologies. Users can browse through an ontology in order to understand its scope, structure and content, and rapidly search it for terms of interest and related terms.

2.2 Extraction and Integration

There are mainly two approaches to extract sub-ontology from multiple ontologies:

1. Integrate all the original ontologies, and then extract the demanded sub-ontology from the integrated one;
2. Extract sub-ontology form each ontology separately, and then integrate them into the demanded one.

The former can use current methods in single ontology. However, there are two disadvantages: the difficulty in ontology integration and the complexity in ontology extraction. The ontology integration problem [11] is one of the most difficult problems to be solved on the Semantic Web, especially integrating large ontologies. Current methods of extracting sub-ontology proves to be computationally expensive [7]. Even dealing with a single large ontology, many optimization schemes have to be applied.

The latter wipes off the outlying information first, so the sub-ontologies to be integrated are much smaller than original ones. It is better than the former approach at least in aspect of efficiency. We adopt this approach in our framework. However, there are also two problems in this approach: how to divide requirements in order to guide extraction in every ontology; and how to get the “outlying” information required in integration process. We will talk about them in the next two subsections.

2.3 Requirements

When extracting sub-ontology from single ontology, user requirements can directly conduct the extraction. Nevertheless, it is hard to decide requirements to extracting sub-ontology from a certain ontology in multiple ontologies. The key problem is how to divide the requirements, or even whether to divide them.

The first idea is not to divide the requirements, just use them to extract sub-ontology from each ontology. The sub-ontologies should be as complete as possible. The redundant information in sub-ontologies may help the integration process.

An opposite idea is to make sub-ontologies as small as possible. Requirements are divided into many small parts; each part is to guide extracting a sub-ontology from a certain ontology. Notice that it may extract several sub-ontolgies from one ontology.

The sub-ontologies are independent of each other; there is no information helped to integrate them.

However, we choose the second method in our framework. Comparing to the benefit of redundant information, the disadvantages are more remarkable. It does many redundant works in extraction, and needs to remove all the redundant information in integration. As both extraction and integration are difficult and computationally expensive, the costs are often unacceptable. Moreover, the redundant information rarely contains all the information needed in integration. A better way is to separate it from sub-ontologies. We use bridge ontology to maintain the information used in integration more conveniently and completely.

2.4 Bridge Ontology

The Ontology integration is a laborious work, and may have not effective solutions, as ontologies created by integrating are weak during ontology evolution.

Distributed Description Logic [12] is one of method for tackling this problem, and has a simple bridge rule to express the concept subsumption relations between ontologies. Bridge ontology [10] can describe more refined relations. The bridge ontology is a peculiar ontology, and has the ability of expressing the complex relations between multiple ontologies. It can be created and maintained conveniently, and is effective in the applications based on multiple ontologies. It has the advantages of low-cost, scalable, robust in the web circumstance, avoiding the unnecessary ontology extending and integration, and promoting ontology reuse. The bridge information can provide information needed in the integration process.

3 Proposed Framework

Our framework contains four processes to extract sub-ontology form multiple ontologies:

1. Convert all the ontologies into a unified ontology model;
2. Divide the requirements into sub-requirements;
3. Extract sub-ontologies based on the sub-requirements respectively;
4. Integrate the sub-ontologies.

An example of apply the framework is showed in Fig.l. The squares are processes; the real line arrows are inputs and outputs of the processes, and the dashed arrows shows the requirements guiding the processes.

The ontologies are all in the unified Ontology model. The bridge ontology contains information about the relations between different ontologies to help the integration process. It is optional and can be replaced by other mechanisms such as distributed description logic.

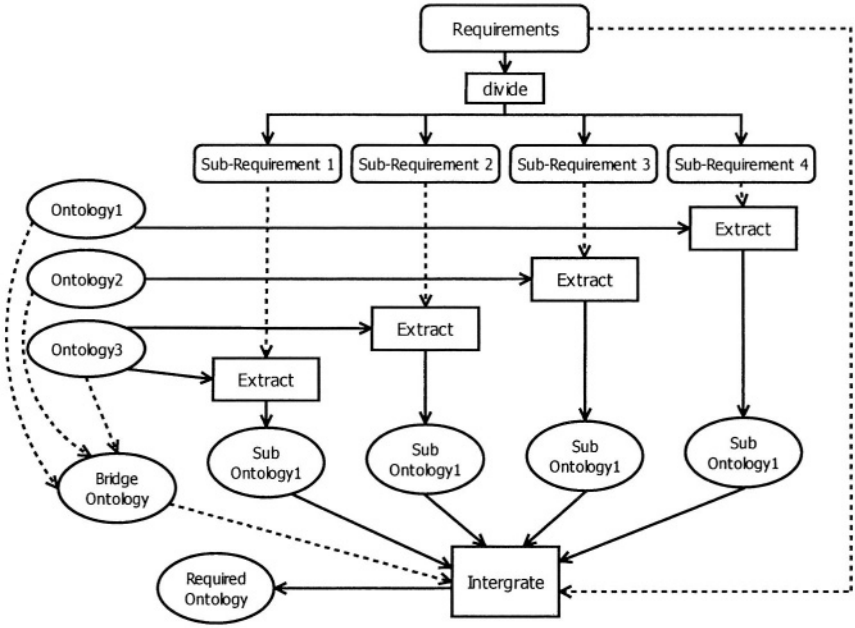


Fig. 1. The flow chart of the framework

3.1 Unified Ontology Model

Generally, an ontology can be seen as a quadruple $O = (C, R, X, \Delta)$, where Δ is the set of individuals; C is the set of concepts, which are subsets of Δ; R is the set of relations, which are subsets of $\Delta \times \Delta$; X is the set of axioms. This definition is very broad and general. It is hard to extract sub-ontology in such representation.

In our framework, the information about individuals is not concerned. We focus on the concepts and relations. They are both fundamental elements in most Ontology languages. It makes the extraction difficult. We consider concepts to be the only fundamental element, and organize them into a hierarchy. Relations are divided into attributes (datatype properties in OWL) and other relations (object properties in OWL). Attributes are special relations that between individuals and literals. Both attributes and relations are depending on the concepts that they linked; they are not the fundamental element in Ontology. (The meaning of these elements in Ontology can be found in OWL standards of W3C [13])

Definition 1. We consider an ontology as a eight-tuple

$$O = (C, A^C, R, A^R, S(C), E(C), H, X) \tag{1}$$

where C is the set of concepts; A^c is the set of attributes about concept $c \in C$; R is the set of relations; A^r is the set of attributes about relation $r \in R$; $S(c)$ and $E(c)$ are the sets of relations that can start and end with concept $c \in C$; H represents the concept hierarchy; and X is the set of axioms.

The concept hierarchy is the set of two-tuples of concepts that have subsumption relations. It organizes all the concepts into a well-formed hierarchy.

Relations need to depend on certain concepts. However, each relation associate to pairs of concepts; the number of pairs that is the square number of concepts may be too large to handle. An alternate plan is to describe the starting and ending of relations respectively. $S(c)$ is the set of relations $\{r \mid r \in R \wedge \exists a,b(a \in c \wedge (a,b) \in r)\}$; $E(c)$ is the set of relations $\{r \mid r \in R \wedge \exists a,b(b \in c \wedge (a,b) \in r)\}$, where a, b are individuals. An obvious fact is that $c_1 \subseteq c_2 \rightarrow S(c_1) \subseteq S(c_2) \wedge E(c_1) \subseteq E(c_2)$; the redundancy can be cleared based on this during implementing.

Attributes are specific relations depending on only one certain concept or relation; for example, person name is a string attribute of person. Axioms are restrictions about the concepts, relations and attributes. Each axiom is of course depending on the elements which it put the restrictions.

This model of Ontology is expressive enough to represent ontologies in most Ontology languages. It is possible to translate ontologies in other languages into this model and vice versa.

The extraction of sub-ontology is actually a selection of elements in the original ontology. In our model, we can select concepts first. The selection of relations is depending on the selected concepts; the selection of attributes is depending on the selected concepts and relations; and the selection of axioms is depending on them all. It modularizes the extraction process and makes it easier.

Visualization of this model is realizable. It can be viewed as a tree-like concept hierarchy with concepts as the nodes and relations between the nodes; attributes are contained in certain concepts and relations.

3.2 Requirements Division

User requirements are often in inexplicit natural language. The users may not know much about Ontology, and they often do not know what are they really want. Requirement analyzers have to understand the user requirements. They must be wise to existing ontologies about the related domains as well. They convert the informal requirements from users to formal ones.

The formal requirement of sub-ontology is expected to be a selection of concepts, relations and attributes. It announces which of them are needed, and which of them are not needed. However, there are multiple ontologies with different terminologies. It is often impossible to get such requirements.

Functional requirements are more befitting here. It captured the intended behavior of the required ontology. Use Cases in UML and many other tools can represent the functional requirements. Requirement analyzers can convert the user requirements into any of them.

Then analyzers decide what domains are related to the functional requirements, and select the existing ontologies of the domains. For each function, they analyze what concepts or relations are needed in one or more ontologies, and get several sub-

requirement for each ontology. Each sub-requirement describes the concepts and relations needed or not needed in a certain ontology.

Definition 2. A sub-requirements is a seven-tuple

$$Re = (SC, SR, SA, UC, UR, UA, EX) \quad (2)$$

where they are sets of concepts, relations and attributes in the select ontology, and are short for Select Concepts, Select Relations, Select Attributes, Unselect Concepts, Unselect Relations, Unselect Attributes, and *EX* is the set of extra requirements.

Different functional requirements can share sub-requirements, and sub-requirements in the same ontology can merge into one if they have something in common. This can reduce the number of sub-ontologies. However, if sub-requirements in the same ontology have nothing to do with each other, it is better to treat them separately in order to make the extraction easier (the complexity of extraction and integration is superlinear in the size of sub-ontology).

Sub-requirements are not stable. They can be changed in extraction and integration processes. The sub-requirements can be showed in visualized ontology model marking selected and unselected elements differently. Analyzers can browse the visual model, and change the requirements.

3.3 Sub-ontology Integration

After the extraction process, it has to integrate the sub-ontologies into one. We will not propose any new integration methods here. The Ontology integration is a difficult task. However, some existing methods are able to integrate simple ontologies [11]. Sub-ontologies are often small and simple; these methods can integrate them with acceptable results. Nevertheless, there are two additional points to be noticed: the requirements and the bridge ontology.

The integration process has to make sure that the integrated ontology is meeting all the requirements. In addition, another important issue is the performance of function. Different ontology structures lead to different performance in certain business. It is also based on the requirements. How to use requirements to conduct the integration process is an open question.

We consider using bridge ontology [10] to keep information about the relations between different original ontologies. The information can be helpful in integration process. The bridge ontology can be reused when extracting another sub-ontology from these ontologies.

4 Extraction

Extracting sub-ontology from original ontology is the key process of the framework. The input of this process is a ontology and a sub-requirement, the output is the sub-ontology extracted from the original ontology according to the sub-requirement.

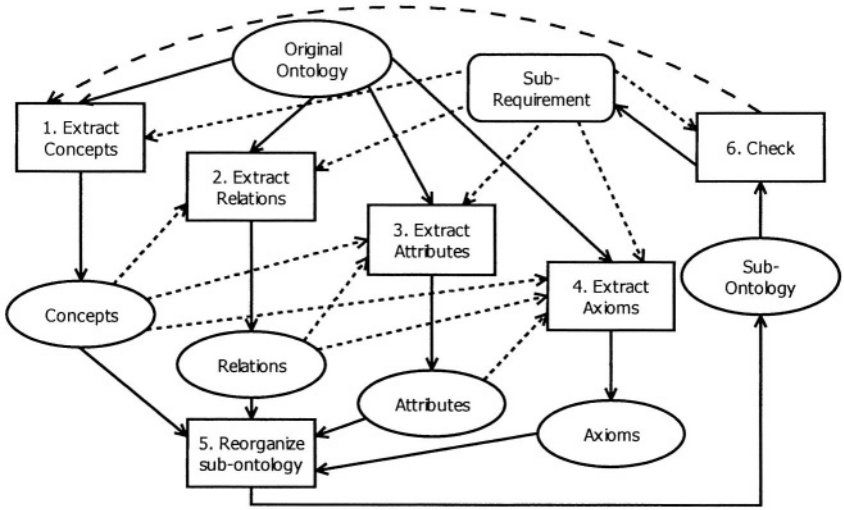


Fig. 2. The steps in extraction process

The extraction contains mainly six steps: extract concepts; extract relations; extract attributes; extract axioms; reorganize sub-ontology and check it. Each step uses the results of previous steps as input. The six steps are iterative; the process will continue until the sub-ontology satisfies the sub-requirement. The steps are show in Fig.2.

4.1 Extracting Concepts

Concepts are the fundamental elements in our ontology model. The extraction of concepts is the base of all following works.

A sophisticated method is to include all the concepts on the path from the selected concepts to the root of the concept hierarchy except the unselected ones. Formally, the concept hierarchy is a set of two-tuples $\{(c_1,c_2) \mid c_1 \subseteq c_2 \wedge c_1,c_2 \in C\}$; the extracted concepts are $\{c \mid \exists c_1(c_1 \subseteq c \wedge c_1 \in SC) \wedge c \notin UC\}$. This method is simple but quite efficient in many existing systems [9].

Another method is to extract concepts that are the smallest parent of any pair of the selected concepts: $\{c \mid c \in SC \vee c \notin UC \wedge \exists(c_1,c_2 \in SC)(c_1 \neq c_2 \wedge c_1 \subseteq c \wedge c_2 \subseteq c \wedge \neg\exists c_3(c_3 \subset c \wedge c_1 \subseteq c_3 \wedge c_2 \subseteq c_3 \wedge c_3 \notin UC))\}$. It greatly reduced the number of concepts extracted. However, it may lose many related concepts.

The above methods focus the structure of concept hierarchy. The semantic of concepts can also be used in extraction. For example, if one selected concept is defined in terms of another concept, the latter should be extracted [8]. However, it is more difficult to be implemented.

Besides the automatical concept extraction methods, it is helpful to have a visual tool for user to browse the ontology and select or unselect any concepts at will.

4.2 Extracting Relations, Attributes, and Axioms

Relations are depending on the concepts they linked. Let the extracted concepts in the previous step be *EC*. We can consider extracting relations based on both *SR* and *EC*: $\{r \mid r \in SR \wedge \exists(c_1, c_2 \in EC)(r \in S(c_1) \wedge r \in E(c_2))\}$ and specific relations between the extracted concepts: $\{r \mid r \notin UR \wedge \exists(c_1, c_2 \in EC)(r \in S(c_1) \wedge \forall c(c \subset c_1 \rightarrow r \notin S(c)) \wedge r \in E(c_2) \wedge \forall c(c \subset c_2 \rightarrow r \notin E(c)))\}$.

Attributes are depending on certain concepts and relations. Let the extracted relations be *ER*. The extraction of attributes is based on *SA*, *EC* and *ER*: $\{a \mid a \in SA \wedge (\exists c(c \in EC \wedge a \in A^c) \vee \exists r(r \in ER \wedge a \in A^r))\}$. Let it be *EA*.

Axioms are not in sub-requirements except extra requirements. The extraction of axioms is based on *EC, ER* and *EA*. If all the elements that an axiom describes are in them, the axiom is extracted.

These steps are all based on the results of the previous steps, and they extract different kinds of elements in ontology separately. Moreover, user can select and unselect all these elements manually when browsing the ontology.

4.3 Reorganizing and Checking the Sub-ontology

Reorganizing sub-ontology is a simple work, for no new information is introduced and all the extracted elements are in the original ontology.

It is important to check the sub-ontology. The checking step adjusts sub-ontology and the sub-requirements, and decides whether to end the extraction or repeat it.

1. Check for the consistency of the sub-requirements, for it is hard to avoid contradictions in requirements and user may change his idea at any moment.
2. Check for the semantic completeness of the sub-ontology.
3. Check whether the sub-ontology meets the sub-requirements.
4. Check whether the sub-ontology is the smallest possible solution.

Materialized ontology view extraction [5, 6] does extraction process that involves various optimization schemes that handle similar issues. However, it extracts all the elements in each scheme. That makes the process computationally expensive.

Our extraction process is iterative and incremental. If the checking fails, it repeats the extracting process but the previous results can be reused. Moreover, it corrects the sub-requirements to make the next result more appropriate. If the dependent concept of a selected attribute is not extracted, i.e. $\{c \mid c \notin EC \wedge \exists a(a \in SA \wedge a \in A^c)\}$, the concept should be added to sub-requirements. If a selected relation is not extracted because it starts or ends with no extracted concepts, certain concepts need to be selected; but if it starts and ends both with no extracted concepts, the relation may not be selected. If the dependent relation of a selected attribute is not extracted nor selected, the relation should be selected.

The generated ontology is in our Ontology model. However, user may require a ontology in a certain Ontology language. The translation is possible; but how to ensure that the translation loses information as little as possible is an open question.

5 Conclusion and Future Work

In many practical cases, especially in the web environments, we often face multiple ontologies in one application; and it is required to extracting sub-ontologies from multiple ontologies.

We propose a framework of extracting sub-ontologies from multiple ontologies, and suggest some first step notions. We first translate different Ontology language code into a unified visualized model representing ontologies, and then divide the user demand to extract sub-ontology from each single ontology. We carry out the sub-ontology extraction based on the unified model in an iterative and incremental way. Finally, we integrate these smaller sub-ontologies into the ontology of the user demand. The result can be translated back to the Ontology language the user uses.

The high efficient methods of extraction and integration aim at the framework and the visualized tools implementing the framework are the future work.

References

1. Berners-Lee, T., Hendler, J., and Lassila, O.: The Semantic Web. Scientific American, 284(5) (2001) 34-43
2. Stuckenschmidt, H.: Ontology-Based Information Sharing in Weakly Structured Environments. PhD thesis, AI Department, Vrije Universiteit Amsterdam (2002)
3. Unified medical language system. http://www.nlm.nih.gov/research/umls/
4. Peterson, B.J., Andersen, W.A., and Engel, J.: Knowledge Bus: Generating Application-Focused Databases from Large Ontologies. Proceedings of the 5th Knowledge Representation Meets Databases (KRDB) Workshop, Seattle, WA (1998) 2.1-2.10
5. Wouters, C., Dillon, T., Rahayu, W., and Chang, E.: A practical walkthrough of the ontology derivation rules. In Proceedings of DEXA 2002, Aix-en (2002) 259-268
6. Wouters, C., Dillon, T., Rahayu, W., Chang, E., and Meersman, R.: A practical approach to the derivation of materialized ontology view. Web Information Systems, Idea Group Publishing (2004) 191-226
7. Bhatt, M., Flahive, A., Wouters, C., Rahayu, W., and Taniar, D.: A distributed approach to sub-ontology extraction. Proceedings of the 18th International Conference on Advanced Information Networking and Application (2004) 636-641
8. Bhatt, M., Wouters, C., Flahive, A., Rahayu, W., and Taniar, D.: Semantic Completeness in Sub-ontology Extraction Using Distributed Methods. ICCSA (3) (2004) 508-517
9. Delfs, R., Doms, A., Kozlenkov, A., and Schroeder, M.: GoPubMed: ontology-based literature search applied to Gene Ontology and PubMed. German Conference on Bioinformatics (2004) 1-9
10. Peng, W., Baowen, X., Jianjiang, L., Dazhou, K., and Yanhui, L.: A Novel Approach to Semantic Annotation Based on Multi-ontologies. Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai (2004)
11. Keet, C.M.: Aspects of Ontology Integration. Internal report to the PhD proposal (2004)
12. Borgida, A., and Serafini, L.: Distributed Description Logics: Directed Domain correspondences in Federated Information Sources. LNCS, Vol. 2519, Springer Verlag, Heidelberg (2002) 36-53
13. W3C Web-Ontology (WebOnt) Working Group. http://www.w3.org/2001/sw/WebOnt/

MAZETTE: Multi Agent MUSETTE for Sharing and Reusing Ontologies

Jesus Arana, Salima Hassas, and Yannick Prié

Equipe Cognition, Expérience et Agents Situés
LIRIS FRE 2672 CNRS
Université Claude Bernard Lyon 1
Bâtiment Nautibus – 8 Bd Niels Bohr
69622 Villeurbanne
http://experience.univ-lyon1.fr/
{jarana, hassas, yprie}@liris.cnrs.fr

Abstract. During the realization of a document-mediated collective task the participants act and interact by creating documents and ontologies, by modifying them, annotating them and exchanging them. This article presents the general principles of a model based on a multi-agent architecture, aimed at facilitating the co-construction of common ontologies. The model is built on the MUSETTE (Modelling USEs and Tasks for Tracing Experience) approach, which was designed to provide users with assistance based on the recording and reusing of their system use traces. Our model, called MAZETTE (Multi-Agent MUSETTE), defines a framework for considering sharing and reusing of collective traces and experience, amongst which we focus on ontology co-construction and reuse.

1 Introduction

Nowadays, the use of computers is getting not only useful but also essential, as a lot of works are becoming computer-mediated tasks. So from the mixed of the increasing number of persons using computers and the emergence of hundreds or thousand of new softwares and applications, has create the need to develop software agents that act as assistants to aid the users to realise their new tasks.

With the improvement of the information systems, the Internet, and the number of computers growing, the augmentation of web pages submitted by the users has grow in an exponential way. That's one of the main reasons of the existence of the semantic web. One of the purposes of the semantic web , is that computers will be able to use the data on the web for automation, integration and reuse of data across various applications[1] not just for display purposes. To accomplish this, it requires at least two

[1] http://www.w3.org/2001/sw/

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 741–752, 2004.
© Springer-Verlag Berlin Heidelberg 2004

things, first, data should be structured, this permits computers to distinguish and identify pieces of data; the second reason is that these pieces of data must be described in such a way, that the meaning it can be exploited by machines. This is obtained when the relevant information is represented in a declarative and semantic precise way, and when it is thus understandable for the computer. And this is very important to consider, cause other way, it will not be possible for the computer to assist the user.

Considering all these factors, we are interested by the manipulations of a collective group, when it is interacting in order to realise a computer collective-mediated task. To achieve our analyse we are going to model the users' traces considering a shared knowledge space made up of documents, ontologies and as well as annotations on these documents. In other words, the knowledge space that is available to a user to carry out his tasks. However, this modelling is kind of complex due all the different factors that must be considered, and for that reason, we are going to propose a model based in a multi agent architecture [1], since an agent is an informatics system located in an environment that he can perceive, and that he can act over in an autonomous manner. One of the main objectives of this research is to facilitate the co-construction of ontologies, during the users' interaction.

1.1 Issues in Ontology Sharing and Reuse

It exist several definitions of ontology, the most cited is *"an ontology is an explicit specification of a conceptualisation"* [2], for our context we are going to chose this one: "an ontology provides the common vocabulary of a specific domain and defines, more or less formally, terms meaning and some of their relationships" [3].

So ontologies are extremely important or even essentials to represent the knowledge, because different systems may use different names, even for the same kind of entities; or even worst, they may use the same names for different entities. That's why, the importance of constructing ontologies, to communicate and share knowledge between different users.

In this paper, we propose a model based on a multi-agent architecture, which facilitates the co-construction (by emergence) of common ontologies. This model is based on the MUSETTE approach (Modelling USEs and Tasks for Tracing Experience) [4], a general framework for representing concrete experience in relation with its context of use. So the MAZETTE approach (Multi agent MUSETTE), propose a model based in a multi-agent system to share and reuse collective experience. The approach models the traces of different users in an informatics system. At first, it has a use model that describes all the interest objects such as entities, transitions and relations. After that, the users themselves can track their own use traces, which can be stored in databases. So the purpose of these data is that they can be shared and reused by different users to realise collective tasks.

In the next section, we are going to present the existing related work. In section 3 we present the MUSETTE approach followed by the MAZETTE approach in section 4, where we present the use model, as well as a scenario and we describe the Mazette interface. Finally in section 5 we present our conclusions followed by the references.

2 Related Works

The goal of the semantic web is a web where resources are *machine understandable,* and where the information can be exchanged and processed in an automatic way by persons and agents. So a very important feature of the web ontology language, is that it must have a well defined semantics. OWL (Web Ontology Language) [5], can be used to represent the meaning of terms in vocabularies and their relationships. (This representation of terms and their relationships is an ontology). This language has more capabilities to express meaning and semantic for not only XML, RDF, RDF-Schema since OWL has the ability to represent machine interpretable content on the web.

Related works to achieve the semantic web is through the semantic annotation, and it exists where the content of annotation consists of some more rich semantic information. This idea of Semantic Annotation has been pursued in Ontobroker [6], SHOE [7] and the COHSE project [8]. The objective of Ontobroker is the creation and exploitation of rich semantic structures for machine-supported access to and automated processing of distributed, explicit and implicit knowledge. SHOE allows web page authors to annotate their web documents with machine-readable knowledge. SHOE makes real intelligent agent software on the web possible. In this approach ontologies are mark up in an extension to HTML, and are stored and referred to using an URL. The COHSE project has like objective to improve significantly the quality, consistency and breadth of linking of web documents at retrieval time (browsing documents) and authoring time (as authors creating documents).

With respect to related works in the creation of ontology-based metadata by semantic annotation, we found that these frameworks are some of the most important: CREAM [9, 10, 11], ANNOTEA [12, 13, 14, 15], WebKB [16], and the MnM annotation tool [17]. The CREAM project provides a framework for relational annotation metadata, as well as to annotate HTML pages in order to make their content available to software agents with inference capabilities. ANNOTEA is a tool that enhances collaboration through shared metadata based web annotations. Allows the annotation of web resources with comments. It relies on RDF-Schema, and is a tool that shares the idea of creating a kind of user comments about web pages. WebKB is a Web knowledge-based server based on conceptual graphs to represent the semantic concept of web documents. It embeds conceptual graphs statements into HTML pages. The MnM Annotation tool allows automated and semi-automatic support for annotating web pages with semantic contents. MnM integrates a web browser with an ontology editor and provides open APIs to link to ontology servers and for integrating information extraction tools.

With respect to multi-agents systems research in [18] is proposed a distributed ontology development environment in a multi-agent environment, and a common ontology was build between different users, this approach is convenient when the ontologies of the different parties are stable; but when they are in continuous evolution, is not very good. In the InfoSleuth Project [19], their agents utilise multiple ontologies in order to increase the chances of finding a semantic match of concepts, but not to discover relationships between concepts in the different ontologies. By the other hand

the DOGGIE project [20, 21] (Distributed Ontology Gathering Group Integration Environment) proposed how agents with several ontologies can locate and translate semantic concepts distributed among them, in order to share knowledge by automated methods and agent communication strategies. In this approach, they do not assume that ontologies shared commonly labelled concepts, but rather a distributed collective memory of objects that can be categorized into the agent's ontology.

In our model we propose a different approach that take the principles of MUSETTE to track and model the users' manipulations with the objective of reuse the experience. The model is based on a multi agent system and considers users that share a documentary space with the objective of the achievement of a collective task. Our idea is to model each user's traces as a sub-graph. So the total experience is going to be represented like a total graph, where we are going to extract the interesting things, like the relations amongst different users' experience. (Co-construction of emergent ontologies).

3 MUSETTE

The MUSETTE approach is born from the idea of modelling a users' traces with the focus in order to assist him to carry out his tasks (Modelling USEs and Tasks for Tracing Experience) [4] defined at CEXAS *(Cognition Experience et Agents Situés)* at the laboratory LIRIS. The general principles of MUSETTE are described in figure 1. In this approach, there is a user who interacts with a system to modify his documents in his workspace (limited by its files, its software and all that the user can handle). In this system, there is an observant agent that is guided by an observation model that generates primitive traces by respecting a use model starting from these interactions that he observes. Then a trace's generic analyser extracts (starting from

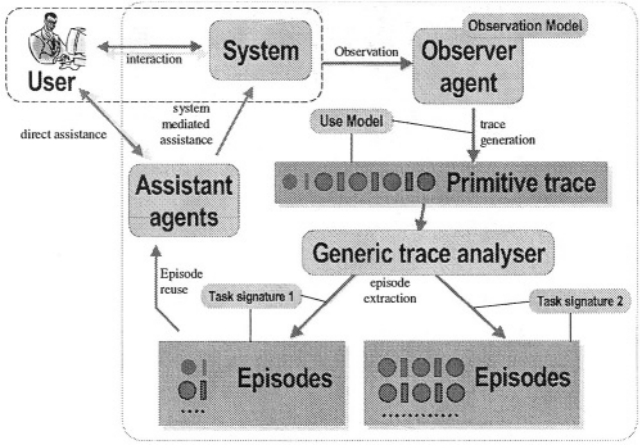


Fig. 1. MUSETTE Approach

the primitive trace) the significant episodes in agreement with signatures from beforehand-defined explained tasks. These episodes will be able to be used by assistant agents, which will be able to assist the user either in a direct way or in an indirect way (instrumentation of the system).

The observation model defines the vocabulary and the necessary rules to determine the relevant data in order to allow the effective construction of the trace, while the use model, describes the objects of interest to be observed which will be registered in the trace, such as the entities (the objects present for the user in its interaction with the system), the events (the objects which occurs during the interaction) and their relations (the binary relations between the entities and the events, entity-entity, entity-event or event-event).

In [4] is proposed a simple example of one use model of a Web navigator (figure 2), in whom the entities correspond to the Web pages (Page), with the hyperlinks (Link), the images (Img) and the user preferences (Cust). The events of this use model are the clicks of the user (Click), the pages that's saves (Sav), the favorite pages which marks (Bm), and the changes of language which it carries out (Lang).

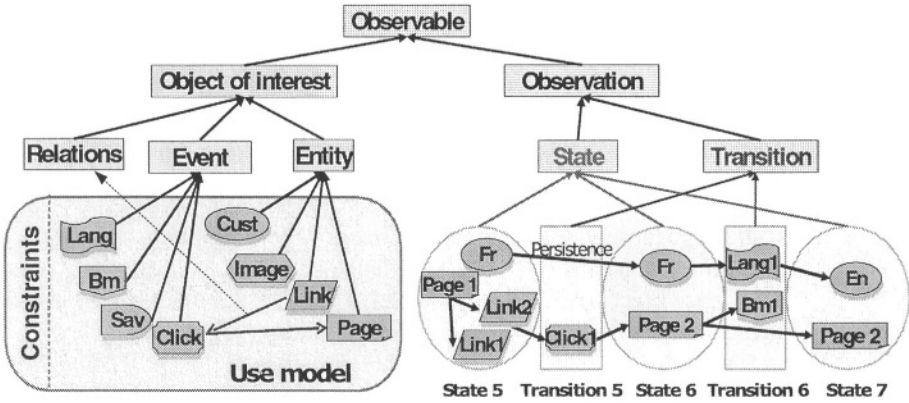


Fig. 2. A simple use model and a web navigation trace

The trace is a representation of the interaction of the user with the system, and it has two structures: states (made up of entities, which represents the state of the system at a given *time*) and transitions (made up of events produced between two states). A trace will be an alternation of states and transitions. One can note in the example, which a change of state is registered in the trace each time that a page is reloaded. An episode is a segment of the trace that represents an experience in the realization of a observed particular task. An episode is determined by a explained task signature (EXTASI) who allows us to find similar experiences, i.e. observations identified for a particular task. The episodes are not cases strictly in the sense of CBR [23], but they are regarded as forming part of bases of potential cases dependent to the EXTASI making it possible to extract them, and as such reusable. In the generation of tasks

signatures, there are two characteristics which are opposed: versatility and system effectiveness; in a very general model, there will be different tasks which will not be differentiated the ones from the others, and the episodes will be very vague for beings re-used in an effective way. On the other hand, if the system is very specific, therefore some tasks outside its goal will become undetectable.

4 MAZETTE Approach

4.1 MAZETTE in General

Our approach consists to generalize the MUSETTE approach to assist the user in the process of the sharing and the reuse of the experience during the realization of a collective task. The Mazette approach principles are illustrated in figure 3. First of all, one considers that each user handles a *knowledge space* made up of documents, ontologies, as well as annotations on these documents, with concepts of the ontologies. Knowledge spaces are at disposal of the user within the framework of the realization of its task.

For each user, one considers a software agent *alter ego* which is its representative in the system and which has as task, to provide him its personal ontologies and to assist it according to the experience. This agent alter ego accumulates experience starting from the action of the user in the knowledge space.

To model this experience, we base ourselves on a use model, which we built in accordance with the definitions of the Musette model.

The agent alter ego observes and generates traces according to this use model, those correspond to a representation of the knowledge space and its evolution. These traces will be generated (according to a use model) and stocked in databases, as well as the EXTASIS. This way the alter ego might assist the user by making queries according to the EXTASIS. The alter ego will be able to assist the users starting from episodes segments in the trace according to the explained tasks signatures (EXTASI). Besides this *traditional* assistance (MUSETTE model applied to an agent alter ego), one considers in MAZETTE (Multi Agent MUSETTE) an assistance in which the alter ego will use the collective experience of the user of more or less shared documentary spaces. It is then a question of considering the total trace left in the general knowledge space, composed of the traces generated by each alter ego. The experience of each user will be modelled as a sub-graph, so the global experience of all users will represent a global graph.

The context that we are considering for our application is for people who will make distance teaching and which will handles documents with many applications. It would be interesting to track the way, in which they work, to design agents that will provide assistance to new users when they are creating a new course, or to improve the way of doing it.

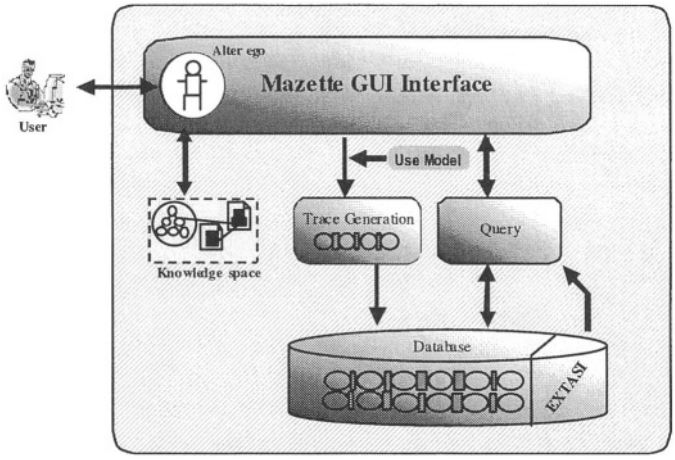


Fig. 3. Mazette Approach

4.2 Use Model

The use model describes the whole of the objects of interest (OIs) that the modeller of the experience wants to analyse, all that he considers important to represent in the trace; i.e. the whole of the entities, the events and the relations. The choice of all these objects of interest is strongly related to the assistance that he wants to provide. Therefore, if one wants to bring an assistance of low level, OIs will be simple and close to the system observed, by the other hand if the desired assistance is of high level, OIs to be tracked will be complex and abstract.

- ***Entities***

We said that the complexity or the simplicity of OIs to be modelled corresponds to the degree of complexity of the assistance to be provided. In our case, where we are in the course of construction of the use model, OIs are limited enough. Initially we have the entity {User} and the entities corresponding to each application in which one will track the actions which occur on these entities: {Word}, {Excel}, {Navigator}. Another level we have the {Documents}, the {Annotations}, the {Ontologies}, etc.

- ***Events***

The events explain us what occurs in the workspace, for example the various actions in which the user utilise a functionality of the system, such as {Create new Doc.}, {Cut}, {Copy}, {Paste}, {Save}, {Annotate}, {Delete}, {Create new concept} in the case of an ontology.

- ***Relations***

The relations are binary and oriented, and they can exist as much as for the entities that for the events, i.e. relations between entity-entity, event-event or entity-event, the number of the relations will be according to the level of the desired assistance. For

example, a relation of very low level would be when a piece of the text is selected, and will not be relevant only when the user selects it to be copied and pasted it in another document.

4.3 Scenario

To visualise the things that we can do, with our model, let's imagine a scenario where three users making some manipulations, the systems is going to observe and this way to stock the traces in order to save these manipulations.

User 1. The user execute the Internet application querying the web to search web pages for "Web Semantic" & "RDF". After he found the results, the user click over a link named "Semantic web", and finally he bookmark this site.
User 2. The user annotates a file named "The professions" with a domain ontology. After that he e-mail the file to a contact.
User 3. The user annotates a web page, in an annotation server. By the other hand, he annotates a web page of Industrial Engineering, with a domain Ontology.

In the figure 4 we show all these manipulations created with a tool to draw a user's trace with respect to a use model; this tool is a plug-in for Protégé 2000 realised at the laboratory LIRIS [24].

Once the users have use their systems, the model is going to be more intelligent, in others words, after we use the system, this one, it will learn, from the preferences of the user as well as the way it works.

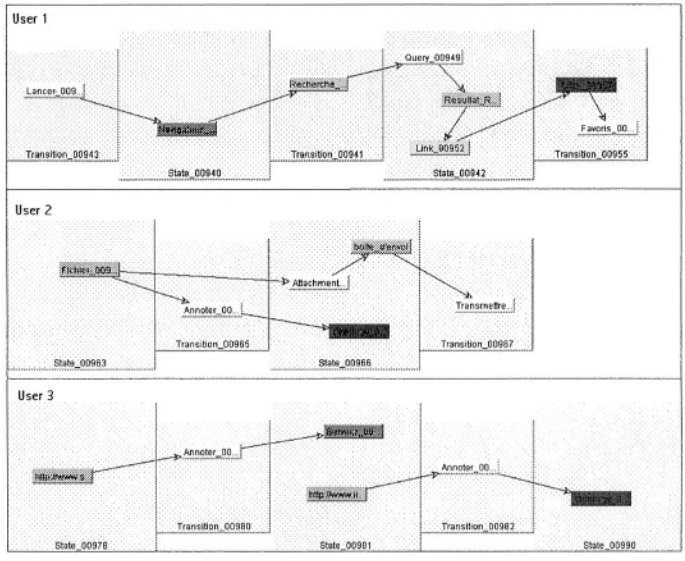


Fig. 4. Use scenarios.

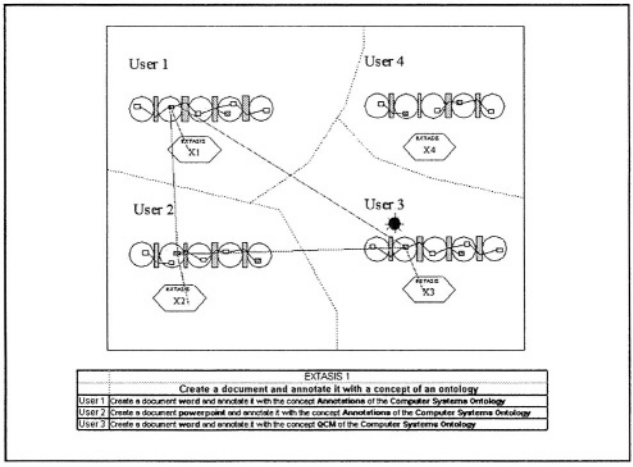


Fig. 5. Sub-graph.

The way where we will store the use's traces, the task signatures and the extasis will be through databases of the MySQL type. The tables will be all the various entities, the events and their relations. In this way, one will be able to store all handling made by the users and they can be retrieved in the same way.

Agents are going to be moving from the database of each and all the users that share the system, and their focus is to find interesting things. For example for the first user that has bookmarked one page, one possible assistance may be to find similar web sites, or find who else has bookmarked a specific site. For the second scenario, that the user has bookmarked a file with a domain ontology, one possible assistance may be, to find what else has been bookmarked with the same ontology. For the third scenario, that the user has annotate a web site in an annotation server, and a web site with an ontology, it will be interesting of read the annotations that other people has made from some web pages, when we are navigating those pages. An example of how the traces can be seen like a sub-graph is shown in figure 5.

4.4 Mazette GUI, the Application

Initially, being given there is not a tool to manage the annotations on all the types of documents that we would like to treat, we will create a graphic interface which will be used to seize the traces according to a Musette ontology by one side, and following personal ontologies to the user by the other side, which will be useful to manage the ontologies of the user, as well as his annotations. Currently, we are working developing this application in Java due its portability, amongst several advantages.

In this graphic interface the user:

- will build use traces of him-self (because for the moment we cannot develop a tool for automatic tracing); who will be stored directly into a MySQL database.
- will manage its different ontologies;

- will annotate the documents by using concepts of its ontologies.
- Will query the database.

From the trace will be extracted the episodes following the task signatures, which could be used to assist the user (in a more or less complex way).

Once the chance of building individual traces in the alter ego provided, we will model the total experience of the traces of several alter egos in the shape of a total graph. We will be able to determine in this way one of the interesting forms corresponding to collective tasks (generalization of task signatures), but to also make emerge, according to the interactions agents, new knowledge of assistance (for example interesting links). The total system will be implemented in a FIPA architecture by using the platform JADE (Java Agent DEvelopment Framework).

5 Conclusion

This project is in the convergence of the domains of SMA [1], the task modelling [25], and ontologies [2] (since an ontology is used to express a knowledge base in a formal language of representation, and such a way that a computer can use it).

In this paper, we presented the general principles of MAZETTE model. This project was born from the need of modelling the actions of a user while they are interacting with the objective of a collective computer-mediated task in order to build intelligent systems that are able to provide him assistance. The goal of the MAZETTE project is to concretise in one application that will be able to store and track the user's manipulations, will manage its different ontologies and will be able to annotate his documents through his ontologies. All this experience will be represented as a partial graph, and the graph global will be the addition of all users' experience, so they can share it. Future works includes the creation of a tool for automatic use tracing, and the final objective is to achieve into a web tool for sharing and reusing experience.

There are work which proceeds within the team in the same direction and are based on the Musette model; like, the modelling of the experience in a cartographic software directed towards the *veille informatique* in [26], and the development of a MemSim prototype to share the experience in a collaborative activity of integrated design [27].

References

[1] Sycara, K.P., 1998. Multi agents Systems. AI Magazine, Vol 10, No. 2, 1998, pp. 79-93.

[2] Gruber, T. R., 1993. A translation approach to portable ontology specifications. Knowledge Acquisition, 5(2), 1993.

[3] A. Gomez-Perez. Développements récents en matière de conception, de maintenance et d'utilisation des ontologies. In Proceedings of colloque Terminologie et intelligence artificielle de Nantes, 10-11 mai 1999, revue terminologies nouvelles, pp. 9-20.

[4] Champin P.-A., Prié Y., Mille A., 2003. MUSETTE: Modelling USEs and Tasks for Tracing Experience ,Proc. From structured cases to unstructured problem solving episodes - WS 5 of ICCBR'03, Trondheim (NO), NTNU, Trondheim (NO), pp279-286, June 2003. http://www710.univ-lyon1.fr/~champin/publis/iccbr2003b.pdf

[5] OWL. http://www.w3.org/TR/owl-features/

[6] Decker S., Erdmann M., Fensel D., Studel R., 1999. Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In R. Meersman, Z. Tari,, and S. Steevens, editors, Semantic Issues in Multimedia Systems. Proceedings of DS-8, pages 351-369. Kluwer Academic Publishers.

[7] Heflin Hendler J., Luke S., 1999. SHOE: A Knowledge Representation Language for Internet Applications. Technical Report CS-TR-4078 (UMIACS TR-99-71), Department of Computer Science, University of Maryland.

[8] Carr L., Bechhofer S., Goble C. A., Hall W., 2001. Conceptual Linking: Ontology-based Open Hypermedia. In Proceedings of WWW10, Tenth World Wide Web Conference, Hong Kong, May 2001.

[9] Handschuh S., Staab S., Maedche A., 2001. CREAM – Creating relational metadata with a component-based, ontology-driven annotation framework. In proceedings of K-CAP 2001, First International Conference on Knowledge Capture, Victoria, B.C. Canada, October 2001.

[10] Handschuh S., Staab S., Maedche A., 2002. CREAting relational Metadata (CREAM) – a framekork for semantic annotation. Frontiers in artificial intelligence and applications. IOS press, Amsterdan (NL).

[11] Handschuh S., Staab S., Maedche A., 2003. Annotation of the shallow and the deep Web. Annotation for the Semantic Web, eds. Siegfried Handschuh and Steffen Staab. IOS Press, Frontiers in Artifical Intelligence and Applications. Vol 96. 2003.

[12] Kalian J., Koivunen M.-R., Prud'Hommeaux E., Swick R., 2001. Annotea: An Open RDF Infrastructure for Shared Web Annotations. In Proceedings of the Tenth International World Wide Web Conference, Hong Kong , May 2001.

[13] World Wide Web Consortium. Annotea Project. http://www.w3.org/2001/Annotea.

[14] Koivunen M.-R., Brickley D., Kahan J., Prud'Hommeaux E., and Swick R., 2000. The W3C Collaborative Web Annotation Project ... or how to have fun while building an RDF infrastructure.
http://www.w3.org/2000/02/collaboration/annotation/papers/annotationinfrastructure, May 2000.

[15] Koivunen M.-R., Swick R., 2003. Collaboration through Annotations in the Semantic. Annotation for the Semantic Web, eds. Siegfried Handschuh and Steffen Staab. IOS Press, Frontiers in Artifical Intelligence and Applications. Vol 96. 2003.

[16] Martin P. and Eklund P., 1999. Embedding Knowledge in Web Documents. In Proceedings of the 8th Int. World Wide Web Conf. (WWW'8), Toronto, May 1999, pages 1403-1419. Elsevier Science B.V.

[17] Vargas-Vera M., Motta E., Domingue J., Lanzoni M., Stutt A., Ciravegna F., 2002. MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. In EKAW02, 13th International Conference on Knowledge Engineering and Knowledge Management, LNCS/LNAI 2473, pages 379-391. Siguenza, Spain, October 2002. Springer.

[18] Takaai, M., H. Takeda, and T. Nishida (1997). Distributed Ontology Development Environment for Multi-Agent Systems, Working Notes for AAAI-97 Spring Symp. Series on Ontological Engineering, pp. 149-153.

[19] Bayardo R., Bohrer W., Brice R., Cichocki A., Fowler J., Helal A., Kashyap V., Ksiezyk T., Martin G., Nodine M., Rashid M., Rusinkiewicz M., Shea R., Unnikrishnan C., Unruh A., Woelk D., 1998. InfoSleuth : Agent-Based Semantic Integration of Information in Open and Dynamic Environments, 205-216, Morgan Kaufmann, San Francisco, 1998.

[20] DOGGIE: A Multiagent System for Learning Diverse Web Ontologies, Andrew B. Williams, Costas Tsatsoulis; Seventeenth National Conference on Artificial Intelligence; Austin, TX, July 2000.

[21] Williams, B. A. and Z. Ren (2001). Agents Teaching Agents to Share Meaning, Proc. ICMAS2001, ACM Press, Montreal, Canada, 465-472.

[22] http://experience.univ-lyon1.fr/

[23] Fuchs B., Mille A., 1999. Une modélisation au niveau connaissances de la tâche d'adaptation en raisonnement à partir de cas. In Actes de RàPC'99, Palaiseau, Juin 1999 , pages 27-36., AFIA, 1999.

[24] Huvier E., 2003. Rapport de Travail de Fin d'Etude Atelier de maquettage pour la modélisation de l 'expérience d'utilisation d'une application informatique. LIRIS, Sep. 2003.

[25] Sohreiber G., Hakkermans A., Anjewierden R. de Hoog, Shadbolt N., Van de Welde W et Wielinga B, 1999. Knowledge Engineering and Management: The CommonKADS methodology. The MIT Press, 1999.

[26] Laflaquière J, 2003. L'expérience d'utilisation d'un espace documentaire pour améliorer l'assistance à l'exploitation de l'information dans le cadre de la veille. Mémoire de DEA informatique, Université Lyon 1.

[27] Stuber A., Hassas S., Mille A., 2003. Combining MAS and Experience Reuse for assisting collective task achievement, in proceedings of CBR-2003 Workshop « From structured cases to unstructured problem solving episodes for experience-based assistance », Trondheim, Norway, June 23 to 26, 2003, Lecture Notes in Computer Science-Springer Verlag.

Literate Ontology Development*

Achim Mahnke and Bernd Krieg-Brückner

FB3 Mathematik und Informatik, Universität Bremen

Abstract. To achieve sharing and reuse for *sustainable development,* documents are semantically interrelated by ontologies, specified formally using the specification language CASL. The specification of properties for relations in the system's ontology is used to check invariant properties during *change management.* Documents are structured for fine-grained management of *variants* for essentially the "same" concept in various representations. This technology is self-applied to the "literate" development of consistent (interrelated packages of) documented ontologies.

1 Introduction

Semantic Interrelation and Change Management. Sharing and reuse are the key to efficient *sustainable development,* i.e. continuous long-term usability of the contents of documents. They need to be supported by tools and methods taking into account the *semantic* structure of a document. In [8,9], we have addressed this problem by introducing a methodoloy to specify coherence and consistency of documents by interrelation of semantic terms and structural entities, supported by a tool for fine-grained version control and configuration management including change management. Semantic interrelation explicates the meaning behind the textual content, and relates the semantic concepts within and across documents by means of an *ontology.* To allow change management, each document is structured in-the-small. Each document corresponds to a package, and packages may be structured in-the-large using folders and import relations.

Variants for Literate Programming and Specification. A particular issue in reuse and change management is the use of variants to manage, for example, consistent English and German variants of a lecture in parallel. A similar situation arises when we want to manage, say, a program and its documentation together; the notion "literate programming" has been introduced by Knuth in his seminal work [6]. For a self-contained and complete program (module), its fragments are interspersed with documentation, such that, on the one hand, a complete documentation, and, on the other hand, the program alone can be derived as separate documents from the same source, to be submitted to a document formatting system (such as LaTeX) or a compiler, resp. Here we assume

* This paper has been supported by bmb+f in the programme *"New Media in Education",* project "Multi-Media instruction in Safe and Secure Systems", MMISS, and by Deutsche Forschungsgemeinschaft in the SFB/TR8 *"Spatial Cognition"*.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 753–757, 2004.
© Springer-Verlag Berlin Heidelberg 2004

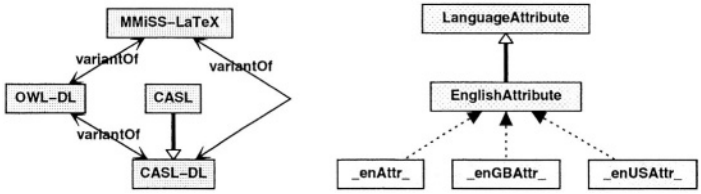


Fig. 1. (a) Ontology Formalisms; (b) Ontology of Language Attributes

that a complete and hopefully correct program is given in an appendix; we may then reuse fragments, by an include mechanism, in the main body of a document, for example in a lecture where such fragments are explained step by step. The situation is analogous for any other kind of formal document that is to be submitted to a tool for further processing. As a particular case, we apply the technology to the "literate" development of ontologies.

2 Various Ontology Specification Languages

Lightweight Ontologies in OWL and LaTeX. Ontologies may come in various specification formalisms and representation formats, cf. Fig. 1. For lightweight ontologies, i.e. just classes (`CASL-DL` is a subclass of `CASL`, denoted by a hollow arrow point), objects (`enAttr` is an object of `EnglishAttribute`, denoted by a dotted arrow) and relations (`MMiSS-LaTeX variantOf OWL-DL`), we use the special MMiSS-LaTeX format that can be translated to the OWL standard [3] and vice-versa. The Bremen ontology visualisation tool, based on the generic graph visualisation tool DaVinci [4], supports incremental presentation of and navigation in such graphs, as in the examples shown in this paper.

Description Logic in OWL and CASL. CASL-DL, a sublanguage of CASL, is restricted to the Description Logic underlying OWL DL, see [11]. It offers precise concepts and definitions together with a translation from and to OWL DL. Thus a variety of tools developed for OWL DL become available for CASL-DL, and vice versa. CASL, the Common Algebraic Specification Language approved by IFIP WG1.3 [1,12], covers full First Order Logic, plus predicates, partial and total functions, and subsorting; moreover, features for structuring, or "specification in the large", have been included. Tools in HETS, the Heterogeneous Tool Set, are available for strong (sub-)type checking, overloading resolution, interfaces to verification tools and re-use of proofs with the Development Graph Manager MAYA [2]. With CASL, full formalisation of ontologies becomes possible, while the expressiveness of CASL-DL (or OWL DL) is often insufficient. General axioms are often required for a precise specification of classes or, in particular, relations in ontologies, cf. e.g. the DOLCE framework [5,10], or [7].

MMiSS-LaTeX

```
\begin{Definition}[Title={Ontology of Language Attributes},
                   Label=LanguageAttributeExample, Language=enGB]
Consider an ontology fragment for \Ref{LanguageAttribute}s:
\begin{Declaration}[Label=LangAtts,Formalism=MMiSSLaTeX,Format=ASCII]
    \Class{LanguageAttribute}{Language\-Attribute}{VariantAttribute}
     \Class{EnglishAttribute}{English\-Attribute}{LanguageAttribute}
      \Object{enAttribute}{en}{EnglishAttribute}
      \Object{enGBAttribute}{enGB}{EnglishAttribute}
      \Object{enUSAttribute}{enUS}{EnglishAttribute}
\end{Declaration}
A \Def{LanguageAttribute} may have various subclasses, among them
\Def{EnglishAttribute}, having several objects, e.g. \Def{enAttribute}
for 'generic' English, or \Def{enUSAttribute} for British English.
\begin{Note}[Label=LanguageAttrsNote,Formalism=MMiSSLaTeX]
  In the \MMiSSLaTeX{} variant of the ontology, the second parameter of
  each command provides a textual phrase that is substituted for each
  reference to it, e.g. in \Source{LanguageAttribute}.
\end{Note}
\Relate{annotates}{LanguageAttrsNote}{LanguageAttrs}
\end{Definition}
```

MMiSS-LaTeX

```
\begin{Declaration}[Label=LangAtts,Formalism=CASL_DL,Format=ISOLatinOne]
  sorts VariantAttribute < Thing; LanguageAttribute < VariantAttribute;
        EnglishAttribute, GermanAttribute < LanguageAttribute;
  ops   enAttribute, enGBAttribute, enUSAttribute : EnglishAttribute
\end{Declaration}
\begin{Note}[Label=LanguageAttrsNote,Formalism=CASL_DL]
In the \CASLDL{} variant of the ontology, subclasses become subsorts.
\end{Note}
```

Fig. 2. MMiSS-LaTeX and CASL-DL Variants

3 Variants

The concept of variants adds a new dimension to hierarchically structured documents. The idea is to maintain and manage different variants of structural entities (document subtrees) which represent the same information in different ways — variants are meant to glue them together.

Managing different natural language variants in parallel is an obvious example. In practice, language versions are often kept in separate documents that tend to diverge as the material develops over time. The same problem arises for reuse by copy in different contexts where small adaptions are occasionally made. However, the relationship between these copies is lost when using old fashioned cut-and-paste.

In the MMiSS-LaTeX authoring format, variant information may be attached to (parts of) structured documents by variant attributes (see [9]). The `formalism` attribute covers the domain of formal languages such as programming languages,

specification languages, ontology languages, etc. Considering ontology development, we propose to use variants to maintain different formal representations for the same semantic concept together with its documentation.

Variants for Different Ontology Representations. Fig. 2 illustrates the use of variant attributes. It shows portions of a document that describes and defines an ontology for structured document entities, in this case attributes (cf. Fig. 1 b). The concept of a `LanguageAttribute` together with its subclasses is defined in the body of the `Definition` entity. The `Declaration` element contains the actual declarations in a specific formalism — in this case the formalism of MMiSS-LaTeX ontologies. The format attribute (another variant domain) holds information about the concrete format in which the content is prepared.

The example given in the upper part of Fig. 2 can be translated to OWL as another formalism variant (not shown here). Similarly, a variant for CASL-DL may be provided. From its ISOLatin1 source variant given in the lower part of Fig. 2, a "pretty" format variant is generated automatically by the CASL tools.

All these variants of the Declaration entity abstractly represent the same piece of information and are therefore given the same label. The MMiSS repository treats them as different variants of the same entity. When checking out a document, different document configurations may be compiled by specifying the desired variants. To keep the authoring format concise, attribute values are inherited along the hierarchical structure.

Change Management. [9] shows how the maintenance and preservation of consistency and completeness during document development can be supported by semantic interrelation, in particular via the system's ontology itself, of which Fig. 2 shows a tiny example. Dependencies between structural entities are formalised by relations. The *reliesOn* relation, for example, is defined to express strong dependencies, e.g. a proof that depends on the formula it proves, etc.

Variants give rise to another relation, *variantOf*, tying together semantically equivalent variants. This relation carries a completeness constraint that for each entity there must be a variant for all requested formalisms. Furthermore, it can be assumed that a change in one formalism variant is affecting the others, thus violating this constraint. The repository should then inform the author that s/he has to investigate the effects of his change on the other variants.

4 Conclusion

We have illustrated how the MMiSS approach to *sustainable development* of (educational) documents that are semantically interrelated by ontologies can be self-applied to the *literate development of consistent interrelated packages of documented ontologies.* Since ontologies can be specified formally using CASL, the specification of properties for relations in the system's ontology may be used to check invariant properties during *change management.*

Documents are structured for fine-grained management of *variants.* We presented various ontology representations as variants for essentially the "same" concepts. The user need not be aware of variant management in the Repository; variant selection will automatically extract the appropriate structural entities and check for consistency and completeness. The basic system is operational; tools for variant selection and change management are presently being implemented. Content and tools are freely available, see `www.mmiss.de`.

We are grateful to the members of the MMiSS project, see also [8], in particular Arne Lindow, and Klaus Lüttich for the CASL examples.

References

1. E. Astesiano, M. Bidoit, B. Krieg-Brückner, H. Kirchner, P. D. Mosses, D. Sannella, and A. Tarlecki. CASL – the common algebraic specification language. *Theoretical Computer Science,* 286:153–196, 2002. `www.cofi.info`.
2. S. Autexier, D. Hutter, T. Mossakowski, and A. Schairer. The development graph manager MAYA (system description). In H. Kirchner and C. Reingeissen, editors, *Algebraic Methodology and Software Technology, 2002,* volume 2422 of *LNCS,* pages 495–502. Springer, 2002.
3. S. Bechhofer, F. van Hermelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. W3C: OWL Web Ontology Language – Reference. W3C Recommendation `http://www.w3.org/TR/owl-ref/`, February 2004.
4. `http://www.informatik.uni-bremen.de/~davinci/`.
5. A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. Sweetening ontologies with DOLCE. In A. Gómez-Pérez and V. R. Benjamins, editors, *EKAW,* volume 2473 of *LNCS,* pages 166–181. Springer, 2002.
6. Donald E. Knuth. Literate programming. *The Computer Journal,* 27(2):97–111, May, 1984.
7. B. Krieg-Brückner, U. Frese, K. Lüttich, C. Mandel, T. Mossakowski, and R. Ross. Specification of an ontology for route graphs. In *Spatial Cognition IV,* Lecture Notes in Artificial Intelligence. Springer, 2004. submitted.
8. B. Krieg-Brückner, D. Hutter, A. Lindow, C. Lüth, A. Mahnke, E. Melis, P. Meier, A. Poetzsch-Heffter, M. Roggenbach, G. Russell, J.-G. Smaus, and M. Wirsing. Multimedia instruction in safe and secure systems. In *Recent Trends in Algebraic Development Techniques,* volume 2755 of *LNCS,* pages 82–117. Springer, 2003.
9. B. Krieg-Brückner, C. Lüth, A. Mahnke, and G. Russell. Semantic interrelation of documents via an ontology. In G. Engels and S. Seehusen, editors, *Proc. DELFI 2004, Paderborn,* Lecture Notes in Informatics. Springer, 2004. to appear.
10. K. Lüttich and T. Mossakowski. Specification of ontologies in CASL. In *International Conference on Formal Ontology in Information Systems.* IOS-Press, 2004. to appear.
11. K. Lüttich, T. Mossakowski, and B. Krieg-Brückner. Ontologies for the semantic web in CASL. *17th Int. Workshop on Algebraic Development Techniques,* 2004. Submitted for publication.
12. P. D. Mosses, editor. CASL *Reference Manual*, volume 2960 of *Lecture Notes in Computer Science.* Springer, 2004.

An E-learning Concept Mindmap and the Management of the Underlying Learning Objects

Jeanne Schreurs, Gerd Balette, and Rachel Moreau

Limburgs Universitair Centrum, Universitaire Campus,
Gebouw D, 3590 Diepenbeek, Belgium
jeanne.schreurs@luc.ac.be

Abstract. We opted to work with the E-mindmap-principle cause of the innumerable advantages. It allows that the content is provided in a graphical way. The e-learning process will be built as a sequence of MINDMAPS. To work with the E-mindmap-principle, the mindmaps and units have to be structured as independent LO's. A generic data structure for those LO's and atomic LO's is developed and a set of metadata has been defined following the SCORM standards.

1 E-MINDMAP

An e-MINDMAP being used in an e-learning course module and presenting the learning content in a graphical way, consists of a number of blocks representing the learning content elements. The learning process, or the learning path, has been defined as a sequence of a number of steps, corresponding to the blocks and is telling the story in a visual way.
The blocks are composed of some atomic learning-elements, being the short text or audio document, the full text, some additional text or graphical presentations, or pictures, some questions and answers, some tests, some mouse-over animations, ...
As a consequence the learner can complete the learning activity on a first level by reading only the short text components or can evolve to a deeper study of the topic by reading the full text components in the pre-defined sequence or learning path. In another way the learner can select his preferred learning topics and selects the corresponding blocks in the MINDMAP.

2 Independent Learning Objects Used in the E-MINDMAP Concept

2.1 Introduction

For the E-mindmap– principle, we use learning objects (LO). A learning object can be defined as any digital content resource that supports learning, that can be re-used

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 758–762, 2004.
© Springer-Verlag Berlin Heidelberg 2004

and that can be delivered on demand across the network, be it large or small. Examples of smaller reusable digital resources include short text documents, figures, digital images or photos, live data feeds, live or prerecorded video or audio snippets, animations, and smaller web-delivered applications. Examples of larger reusable digital resources include entire web pages that combine text, images and other media or applications to deliver course modules. In our paper the first ones are named the atomic LO's, the second ones are named the LO's.
A LO can function as an independent learning content module, which makes reusability of the learning content in different courses possible. Those components have to be managed as independent components in a warehouse.
In our MINDMAP concept a learning object is defined as a MINDMAP and on sublevel as a UNIT. A UNIT/MINDMAP includes some blocks linked together in one story of learning content.
The MINDMAPS and the UNITS can be structured as independent learning objects.
The blocks are structured as independent information components composed of a set of atomic learning content elements or atomic LO's, being the short text, an audio fragment, a full text document, an additional text /figure and others.

2.2 Generic Structure for UNIT/MINDMAP as LO's

A generic data structure for the UNIT/MINDMAP LO's is being developed. To guarantee a flexible composition of a course module, keeping in mind the different educational models and fulfilling the needs corresponding to different learning styles, the required characteristics of the objects have been defined.

a. The normal sequence of the course topics in the MINDMAP is structured as *a sequence of blocks* (sequence following numbers of blocks).
b. Second: we try to build in the opportunity for the learner *to learn in more phases:* First a fast draft reading through the content followed by a more deep learning phase. That is why we have the short text and the full text documents.
c. We try to deliver the opportunity of learning following *different learning styles.* OR the learner follows the sequential structure we built (see 1 and 2) , OR the learners selects those (main) blocks, corresponding the learner 's decision on what to learn and on which level to learn.
d. We *start* each MINDMAP delivering a block 0, being an introduction/overview of the MINDMAP. We *conclude* a MINDMAP delivering a block 99, being a repetition and overview of the most important content elements of the MINDMAP.

3 International Defined Standards and Metadata

3.1 SCORM Standard for Metadata of Content

By following the international defined standards on learning content, the inter operability of the objects in different learning systems is being guaranteed.

For learning objects to be used they must be found. The solution is to store not only learning objects but also descriptions of the learning objects, the metadata. Metadata potentially includes information about the title, author, version number, creation date, technical requirements and educational context and intent. Learning Object Metadata is compatible with the metadata used by the digital and online library community.
SCORM has a place for metadata in every SCO and in every content package.
We base the composition of the metadata on the basic book about "technology and standards used in the development of digital content" (2003) and on the work "using metadata in DU projects" (2004), both from the Digital University. (http://www.digiuni.nl). Theory is conform the IMS (Instructional Management Systems) standards and specifications and the SCORM (sharable content object reference model) reference model. (http://www.imsglobal.org/metadata/index.cfm). SCORM has been developed by ADL (Advanced Distributed Learning), ADL and IMS being the most important players in the standardisation of learning technology solutions.
SCORM differentiates between 3 levels of objects. Different requirements on point of metadata are set forward for the three levels.

3.2 Three Levels of Objects and the Requirements on Point of Metadata

3.2.1 Assets

An asset is a digital content element that cannot be split in smaller content elements. In our e-mindmap concept it corresponds to our atomic learning objects, being the short text and the full text documents, the illustrations, ...
The compulsory metadata consists of the title, the description and the copyright. For illustrations the description must be as clear as possible. A description must be a full description of the object.

3.2.2 Sharable Content Objects (SCO)

A SCO is a set of different assets. We can compare the SCO with our UNIT/MINDMAP in our e-MINDMAP concept.
The SCO's must be independent to become re-usable.
The compulsory metadata consists of the metadata already required for an asset, supplemented with two additional metadata items, the language and the elapsed time.

3.2.3 Content Aggregation Model (CAM)

A course module or a course will be built by packaging a set of selected SCO's, being stored as a Content Aggregation Model or a CAM.
A special characteristic of a CAM is the possibility of "nesting" different CAM's. A CAM can be compared with our chapter, being a learning path composed of several e-MINDMAPS, in our e-MINDMAP concept.
The compulsory metadata consists of the metadata already required in a SCO, supplemented with learning level of the content and the required foreknowledge.

3.3 Which Metadata Has to Be Delivered for the LO's in the E-MINDMAP Concept?

We can differentiate between two kinds of metadata, technical oriented and learning oriented metadata. The last one can again split into characteristics on point of learning objectives and learning styles from one side and learning content characteristics from the other side.
In our MINDMAP concept we differentiate between LO's and atomic LO's. The metadata for atomic LO's is limited to technical metadata, because they are linked into a block, not being an individual learning content component.

The metadata linked with our atomic objects and the Blocks consists of data of creation, format, title, description and copyright.
The learning oriented metadata of a UNIT consists of learning level, learning domain, title and description. The technical metadata of a UNIT consists of date of creation, last-adjustment date, elapsed time, language, copyright and format.
The e-MINDMAP inherits the metadata of the UNIT on point of learning oriented metadata. It will be supplemented with a title and a description of the e-MINDMAP. The technical metadata of the e-MINDMAP consists of the language, being inherited from the UNIT, the elapsed time, being the sum of the elapsed times of the UNITS, and supplementary the creation date, the last-adjustment date, the format and the copyright.

4 Conclusion

To guarantee the re-usability of the learning content and as we'll evolve to a new and advanced e-learning concept e-MINDMAP, we need to define some smaller content components, being the atomic learning objects. We must structure them in such a way, that those atomic components can be used to model the e-MINDMAPS. E-learning modules and e-learning courses will be built by packaging a selection of e-MINDMAPS.
We have done several experiments in using MINDMAPS in our traditional courses. The students were very enthusiast about it. The transformation of them to e-MINDMAP concept is very straightforward and also very promising. The first courses have been implemented now. The evaluation will be done in the near future. The learning objects and the atomic learning objects will be stored and managed in a LO-warehouse, part of a LOMS (learning object management system). The user can model the MINDMAP using the system e-SCHEMAS. Our LOMS is a complete learning support system including all functions concerning the input and the management of the learning materials and the modeling of the e-learning course modules (MINDMAPS).

References

F. Benneker, B. Delchot, R. Paankeet, J. Schoonenboom, A. Strijker : "Werken met metadata in DU projecten, Deell: Handleiding; Stichting Digitale Universiteit, 1/2004. (http://www.digiuni.nl/digiuni/download/7B8DE6C2-FAA2-4E6C-4B48947A2948CF25.pdf)

CISCO SYSTEMS, Reusable Learning Object Authoring guidelines: How to build modules, lessons and topics. 8/2003. (http://business.cisco.com/prod/tree.taf%3Fasset_id=104120&ID=44748&public_view=true&kbns=1.html)

A. Damstra, M. Van Geloven, F. Kresin: "Handbook technologie en standaarden, voor het ontwikkelen van digitale content." Stichting Digitale Universiteit Utrecht 5/2003. (http://www.digiuni.nl/digiuni//download/DI.PROC.014.Handboek%20techniek.v11.pdf)

The Herridge Group, Learning Objects and instructional design, online version. (http://www.herridgegroup.com/pdfs/learning%20objects%20&%20instructional%20design.pdf

H.W. Hodgins: The instructional use of LO's, Online version: "The future of Learning Objects" (http://www.reusability.org/read/)

M. Martinez, The instructional use of LO's, online version: "Designing Learning Objects to personalize Learning (http://www.reusability.org/read/)

Robson: 1999. Object Oriented instructional design and web-based Authoring. http://robby.orst.edu/papers/objectoriented.html .

J. Schreurs, R. Moreau, I. Picart: "e-learning process and content management", Proceedings Eurosys ISBN 90-77381-09-0.

J. Schreurs, R. Moreau, I. Picart: "Building an e-blended learning course using learning objects" IADIS 3-6 June 2003, Lissabon.

J. Schreurs, R. Moreau, I. Picart: The use of learner objects for building a flexible e-learning course, EDEN, 15-18 June Rhodes.

J. Schreurs, R. Moreau, I. Picart: "E-blended learning course building based on learner objects or learning process objects", EISTA 2003 Florida 31 July-2 August 2003.

Theory and Semi-automatic Generation of Bridge Ontology in Multi-ontologies Environment*

Peng Wang[1], Baowen Xu[1,2], Jianjiang Lu[1,2,3], Yanhui Li[1], and Dazhou Kang[1]

[1] Department of Computer Science and Engineering, Southeast University, Nanjing 210096, China
[2] Jiangsu Institute of Software Quality, Nanjing 210096, China
[3] PLA University of Science and Technology, Nanjing, 210007, China
Bwxu@seu.edu.cn

Abstract. Multi-ontologies environments are common in semantic web and knowledge representation fields. How to represent the complex relationships between ontologies is a key problem of the multi-ontologies-based applications. For the purpose of solving the problem, this paper proposed the theory of bridge ontology to represent twelve kinds of relationships between ontologies. Subsequently, detailed discussions focused on the semi-automatic generation of bridge ontology, and two semi-automatic methods based on the ontology structure and annotation instances were studied respectively.

1 Introduction

Ontology is a formal, explicit specification of a shared conceptualization [1], and is widespread in fields such as intelligent information integration and knowledge management. Traditional ontology researches mainly focused on single ontology, but multi-ontologies applications become more and more common in practical cases recently. In the previous semantic annotation work based on multi-ontologies [2], we think that discovering, organizing and representing inter-relationships between ontologies are very important issues. To tackle the problem, traditional approaches such as extending or integrating ontologies are laborious [3]. Distributed Description Logic is another method, which uses the predefined bridge rules to associate with different ontologies [4, 5], but the several rules are not enough for representing the complex relationships between ontologies too. Our primary contribution is analyzing and summarizing the various relationships between ontologies, proposing the theory of bridge ontology. Then we study the semi-automatic generation of bridge ontology based on ontology structures and annotation instances respectively.

* This work was supported in part by the Young Scientist's Fund of NSFC (60373066, 60303024), National Grand Fundamental Research 973 Program of China (2002CB312000), and National Research Foundation for the Doctoral Program of Higher Education of China (20020286004).
Correspondence to: Baowen Xu, Dept. of Computer Science and Engineering, Southeast University, Nanjing 210096, China. E-mail: bwxu@seu.edu.cn

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 763–767, 2004.
© Springer-Verlag Berlin Heidelberg 2004

2 Bridge Ontology

There are abundant inter-relationships between ontologies, and we call them bridge relations. Bridge relations can be further classified as two general classes: concept bridges and relation bridges. Concept bridges have nine subclasses: (1) *Synonym bridge* represents the identical or almost close concepts from different ontologies. (2) *Polysemous bridge* represents that the same concept in different ontologies has different meaning. (3) *Hypernym bridge* is also called 'is a' bridge, and represents the hierarchy relations of concepts from different ontologies. (4) *Hyponym bridge* is also called 'instance of bridge, and expresses the inverse hierarchy relations against the hypernym bridges. (5) *Overlap bridge* expresses the concepts in different ontologies have similar but not absolute identical meaning. (6) M*eronym/holonym bridge* represents part-whole relationships, and is also called 'has a' bridge. (7) C*over bridge* expresses that the disjunction of several concepts of different ontologies can cover another disjunction of several concepts. (8) *Opposed bridge* expresses that two concepts from different ontologies have opposed meaning. (9) C*onnectby bridge* represents the concepts of different ontologies can be associated by some terms. Relation bridges have three subclasses: (10) S*ubsume bridge* expresses the relationships from different ontologies may have subsumption relations. (11) *Inverse bridge* represents that relations from different ontologies may be inverse. (12) *Composed bridge* expresses two relationships of different ontologies may compose a new kind of relationship.

We propose the bridge ontology to integrate these bridge relations for describing the various relationships between ontologies.

Definition 1. *(Bridge Ontology)* A bridge ontology O_{Bridge} is a special ontology and has four-layer structure. Top layer is a universal concept $\top$; Second layer has two concepts: *ConceptBridge* and *RelationBridge;* Third layer consists of 12 kinds of subclass bridges; Fourth layer consists of a set of instances created dynamically.

The structure of bridge ontology is given in Fig. 1. The three upper layers are fixed and called static layers. The instances of bottom layer are created dynamically according to the given multi-ontologies, and called dynamic layer. Every bridge instance has its specific format, which can denote the bridge relation's information and the mapping type between ontologies. Ten kinds of bridges: *BCequal, BCdifferent, BCisa, BCinstanceof, BCoverlap, BCopposed, BRsubsume, BRinverse, BCconnect* and

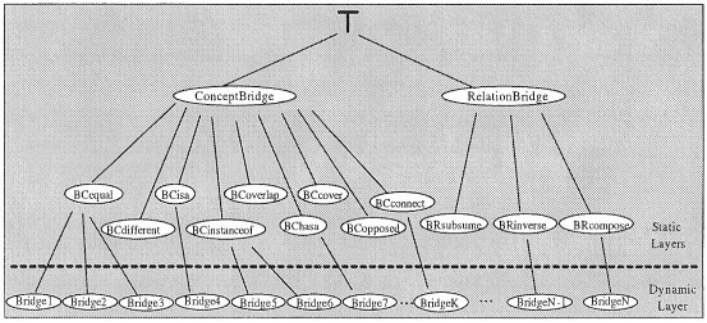


Fig. 1. The structure of bridge ontology

BRcompose express the relationship of 1-to-1, and besides *BCconnect* and *BRcompose* bridges, other eight kinds of bridges have $\mathcal{B}_k(O_i{:}C_1,O_j{:}C_2)$ or $\mathcal{B}_k(O_i{:}R_1,O_j{:}R_2)$ property formats that describe the bridge relations of two concepts or relations between different ontologies. The *BCconnect* bridge has $\mathcal{B}_k(O_i{:}C_1,O_j{:}C_2,$ *term*) property format, which use a term to link two concepts between ontologies. The *BRcompose* bridge has the $\mathcal{B}_k(O_i{:}R_1,O_j{:}R_2,$ *newrelation*) property format, which use *newrelation* to record the new composite relation. The property of *BCcover* expresses the N-to-M mapping type, and its format is $\mathcal{B}_k((O_1{:}C_1,\ldots,O_n{:}C_N),(O'_1{:}C'_1,\ldots,O'_m{:}C'_M))$. The property of *BChasa* expresses the 1-to-N mapping type with the format $\mathcal{B}_k(O_x{:}C_X,O_1{:}C_1,O_2{:}C_2,\ldots,O_n{:}C_N)$. For example, *Bridgek*(*Ocs: Chairman,* O_{swcom}: *Consultor*) is a subclass bridge of *BCisa,* which expresses the *'Chairman'* of CS ontology 'is a' '*Consultor*' of swcom ontology.

3 Generating Bridge Ontology Semi-automatically

Automatic or semi-automatic generation of bridge ontology is useful for the multi-ontologies applications in large scale or distributed web environments. Due to the top three layers of bridge ontology is fixed, the generation just focus on the dynamic layer.

3.1 Generating Bridges Based on Ontology Structures

Here we employ the concept similarity between different ontologies to study the automatic generation method of bridge ontology, and concept similarity includes three independent parts [6]: (1) *Similarity of synonym concept set:* the semantic identical or close concept set [7]. (2) *Similarity of concept features:* properties, relations, and the constraints on them comprise the concept features. The above two measure the internal similarity between concepts. (3) *Similarity of semantic neighborhood concept set* is based on the context of concept in the ontology. Semantic neighborhood concept set of concept C_0 can be denoted with $N(C_o,r)=\{C_i \mid \forall i, d(C_o,C_i)\le r\}$, where d is the distance between concepts, its value is the number of the nearest relations between two concepts; $d \le r$ denotes the current concept satisfies the given semantic distance.

Based on the above analysis, the concept similarity formula between ontologies is:

$$S(C_p,C_q)=W_w * S_w(C_p,C_q)+W_u * S_u(C_p,C_q)+W_n * S_n(C_p,C_q) \quad (1)$$

Where S_w, S_u, and S_n denote the concepts name, feature and context similarity respectively; W_w, W_u and W_n are the corresponding weights.

Tversky defined the asymmetrical similarity in the terminology match [8].

$$S(a,b)=\frac{|A\cap B|}{|A\cap B|+\alpha(a,b)|A/B|+(1-\alpha(a,b))|B/A|} \quad (2)$$

Where *a, b* are the concept to be measured; set A and B here may be the synonym concept set, features set or semantic neighborhood concept set; $\|$ is the cardinality of set; $\alpha(a,b)$ is decided by the class structure:

$$\alpha(a,b) = \begin{cases} \dfrac{depth(a)}{depth(a)+depth(b)} & depth(a) \le depth(b) \\ \dfrac{depth(b)}{depth(a)+depth(b)} & depth(a) \ge depth(b) \end{cases} \tag{3}$$

Where *depth()* is the depth of the current concept in the structure, and its value is the length of the shortest path from current concept to the top concept.

With formula (2) and (3), we can compute the similarity value of S_w, S_u, and S_n according to synonym concept set, features set and semantic neighborhood concept set respectively. Then the general similarity can be computed from formula (1).

The methods of computing the similarity of S_w, S_u, and S_n can aid the generation of bridge ontology. (1) If two concepts have synonym name, same feature and same context structures, namely, $S_w(a,b) = S_u(a,b) = S_n(a,b) = 1$, we derive *a BCequal b* . Frequently, if two concepts have high value in S_w, S_u, and S_n, or in the general similarity, we think they are equal. (2) Miller et al pointed out that the similarity from father concept to child concept is less than the one from child to father usually [7]. It can be extended to the range between ontologies to aid the discovery of relation. (3) The concept similarity measures are not applicable to the similarity between multi-concepts, so *BCcover* and *BChas* bridges cannot be generated by this method.

Therefore, the bridges can be generated by the concept similarity are two kinds: *BCequal* bridge and *BCisa* bridge. Follows is the extracting rules.

Definition 2. *(Extracting rules of BCequal bridge based on ontology structures)* If the mutual similarity of two concepts from two ontologies is large than a constant, the two concepts can be linked by a *BCequal* bridge. Namely, $\forall O_a{:}C_i, O_b{:}C_j, S(O_a{:}C_i, O_b{:}C_j)$ *and* $S(O_b : C_j, O_a : C_i) \ge \beta \Rightarrow BCequal(O_a{:}C_i, O_b{:}C_j)$.

Definition 3. *(Extracting rules of BCisa bridge based on ontology structures)* If the similarity from C_i to C_j is large than a constant, and the ratio of this similarity to the one from C_j to C_i is large than a constant meanwhile, two concepts can be linked by a *BCisa* bridge. Namely, $\forall O_a{:}C_i, O_b{:}C_j, S(O_a{:}C_i, O_b{:}C_j) \ge \beta$ and $S(O_a{:}C_i, O_b{:}C_j) / S(O_b{:}C_j, O_a{:}C_i) \ge \gamma \Rightarrow BCisa(O_a : C_i, O_b : C_j)$.

3.2 Generating Bridges Based on Annotation Instances

Another method of generating bridge ontology is based on the instances. In multi-ontologies-based applications, we select a set of document as the training set. Then the experts annotate these documents. The annotation instances provided by each expert is called conclusions, and all conclusions comprise the instances set based on the multi-ontologies. The bridges can be extracted from the instance set, and the bridge ontology can be created. Then, the bridge ontology is applied to the document set. This is an iterative process.

If current concept is $O_a{:}C_i$, $I(O_a{:}C_i)$ denotes its instance set. We provide the rules of extracting part of the bridges as Table 1 shown. With the instances, we can learn *BCconnect* bridges: if two concepts' instances always exist a particular verb, the verb can be considered as the term of the *BCconnect* bridge. Some kinds of bridges such as

Table 1. The extracting rules of concept bridges based on the instances

No.	Adding Bridges	Extracting rules
1	$BCequal(O_i : C_1, O_j : C_2)$	$I(O_i : C_1) = I(O_j : C_2)$
2	$BCdifferent(O_i : C_1, O_j : C_2)$	$I(O_i : C_1) \neq I(O_j : C_2)$
3	$BCisa(O_i : C_1, O_j : C_2)$	$I(O_i : C_1) \subset I(O_j : C_2)$
4	$BCoverlap(O_i : C_1, O_j : C_2)$	$I(O_i : C_1) \cap I(O_j : C_2) \neq \varnothing$
5	$BCcover((O_1 : C_1, ..., O_n : C_n), (O'_1 : C_1, ..., O'_m : C_m))$	$\bigcup_{i=1}^{n} I(O_i : C_i) \supset \bigcup_{i=1}^{m} I(O'_j : C_2)$
6	$BCopposed(O_i : C_1, O_j : C_2)$	$I(O_i : C_1) \cap I(O_j : C_2) = \varnothing$

BChasa bridge can not be learn from the instances easily. Therefore, the generation based on the instances just solves part automatic generation of bridge ontology.

4 Conclusions

This paper proposed the theory of bridge ontology to represent the complex relationships between ontologies, discussed the semi-automatic generation of the bridge ontology. We will try to use the bridge ontology to the areas based on multi-ontologies in the future work.

References

1. Gruber, T.R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, Vol. 5. (1993) 199-220
2. Wang, P., Xu, B.W., Lu, J.J., Kang, D.Z., Li, Y.H.: A Novel Approach to Semantic Annotation Based on Multi-ontologies. Proceedings of the Third International Conference on Machine learning and Cybernetics, Shanghai (2004)
3. Heflin, J.: Towards the Semantic Web: Knowledge Representation in a Dynamic, Distributed Environment. Ph.D. thesis, University of Maryland (2001)
4. Zuo, Z.H., Zhou, M.T.: A Distributed Description Logic Approach to Semantic Annotation. Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Application and Technologies, PDCAT'2003, Chengdu (2003) 219–221
5. Borgida, A., Serafini, L.: Distributed Description Logics: Directed Domain Correspondences in Federated Information Sources. Lecture Notes in Computer Science, Vol. 2519. Springer-Verlag, Heidelberg (2002) 36–53
6. Rodríguez, M.A., Egenhofer, M.J.: Determining Semantic Similarity Among Entity Classes from Different Ontologies. IEEE Transactions on Knowledge and Data Engineering, Vol. 15. (2003) 442-456
7. Miller, G.A., Bechwith, R., Fellbaum, C. et al: Introduction to WordNet: An On-Line Lexical Database. Int'l J.Lexicography, Vol. 3. (1990) 235-244
8. Tversky, A.: Features of Similarity. Psychological Rev., Vol. 84. (1977) 327-352

Integrating Ontologies into Learning Management Systems – A Case of Czech

Pavel Smrž

Faculty of Informatics, Masaryk University Brno
Botanická 68a, 602 00 Brno, Czech Republic
smrz@fi.muni.cz

Abstract. The paper deals with the role of ontologies in e-learning. We report our research on the application of ontologies in intelligent platforms for self-learning and computer-mediated education that are developed at the Faculty of Informatics, Masaryk University, Brno, Czech Republic. The use of ontologies in the organization of knowledge about the area of learning management systems is tackled first. The next part discusses a new direction of our research aiming at automatic generation of e-learning self-tests based on information extracted from ontologies. A very new topic deals with the preliminary experiments on an ontology-based automatic evaluation of student essays.

1 Introduction

The importance of ontologies is widely recognized, especially in relation to the Semantic Web vision. Ontologies provide common conceptualization, i.e. identify and properly define concepts and relations among them. Although the role of ontologies is often presented only in the context of machines that will understand information on the future web, they are obviously helpful in present-day systems to define shared understanding and unification of conceptual framework.

The role of ontologies in e-learning and particularly in today's learning management systems is often neglected. This paper reports our work in this area. The first part stresses the fact that ontologies can be extremely important not only for machine processing but also for organization of knowledge in e-learning, as means to facilitate communication and cooperation among people and to enable interoperability among learning management systems (LMS). Also system engineering benefits — reusability, reliability, and specification — are briefly discussed.

The formal specification of the concepts and relations between them takes usually advantage of the new XML-family standards, RDF and OWL. The latter serves as a base for our recent research on automatic generation of exercise questions asking for relations between concepts given by an ontology. As the number of available ontologies for various subject domains will surely increase this direction of research has significant potential.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 768–772, 2004.
© Springer-Verlag Berlin Heidelberg 2004

A completely new domain of ontology application discussed in the paper is their use for self-testing. As far as we know, this is the first application of ontologies for such a task. Relations among concepts extracted from a domain ontology are used in automatic generator of e-learning tests (and an automatic evaluation of students' answers). Even more visionary seem to be experiments with the inverse process. The content of students' essays is compared to the concepts and their relations extracted from an ontology, and the automatic evaluation of (information) quality of the text is performed. Although we are not able to present ultimate results on this topic, the preliminary experiments are very promising.

2 Shared Understanding as a Necessary Condition for Integration of Learning Management Systems

The range of LMS used or tested at Masaryk University, Brno, Czech Republic (MU) is rather broad. The most important ones are ILIAS (http://www.ilias.uni-koeln.de/) and MOODLE (http://moodle.org). Both systems are developed and distributed under the term of the GNU General Public License and provide open platform appropriate for the integration of NLP solutions. The actual project at MU aims at unification of the used e-learning platforms and their integration with the administrative information server IS MU [1]. Although separate systems would be more modular, easily maintainable and extendable, we opt for the integrated solution that will benefit from the permanent technical support and personal assistance of the administrative information server team.

The discussion of the needs of the unified e-learning platform at our university showed that participants not only differ in their opinions but also in the conceptualization of the subject area. Such a situation is well known in other areas and software engineering techniques exist that can facilitate the search for consensus. The necessary prerequisite of the consensus however is the sharing of concepts participants are speaking about. Ontologies are natural devices for the task.

An advisory group of current LMS users have been formed to be a partner for the developers of the IS MU. The members successively form the ontology of the area of LMS at MU. Of course, the core is based on the general conceptualization of e-learning that is connected to the international standards in the area. The support of open e-learning standards is rather crucial here as all new learning objects developed at MU should be developed in such a way to facilitate sharing and interchange. As the field of LMS is rather hot and many commercial as well as public-domain systems emerged recently, the newly produced learning object should also reflect the possibility of a switch between platforms without an information loss. The currently developed part of the ontology is compliant with LOM/SCORM standareds. It could help to put the platform independent standards into wide use.

What is specific for our understanding of LMS is the must to cover area of the education administration functions supported by IS MU. For example,

the ontology has to deal with the mapping the LMS data model to that of IS MU. Also, the concepts have to reflect the multilinguality of the systems at MU. Other important issue is the need for strong authentication. The Kerberos authentication protocol is commonly used in large systems with many users. Main advantages are security, reliability and transparency for the users.

The developed ontology provides also conceptualization for:

- Teacher-friendly editing and semi-automated publication of study materials prepared in DocBook and its derivatives (DocBook Slides).
- Integration with standalone system for semi-automated testing of students' programs in Java and other languages.
- NLP support for e-learning enabling to go beyond simple multiple-choice tests.
- Automatic linking of additional study materials available at MU based on intelligent text analysis.

3 Self-Test Generation and Evaluation Based on Ontologies

The promising results of automatic question answering (QA) reported recently [2] led us to the idea to automatically generate questions and perhaps whole tests on the base of e-learning courses. It is usually easy to extract "what is" questions or ask about a particular fact explicitly stated. In some cases, the structure of the documents itself helps to identify the important knowledge; sometimes, the keyword extraction algorithm can be employed.

The tricky part of the process is therefore to create a procedure that would be reliable enough for the automatic checking of answers. Again, the basic QA module is satisfactory for the questions that expect factoid as the answer. However, it is much more difficult to automatically evaluate more elaborate answers. Although we are not able to present final results for this part yet, the preliminary ones show that the application of the same method as for the comparison of student essays with the content could provide at least a good approximation of the assessment that would be given by human teachers.

The weak point of the described automatic generator of exercises is its focus on the factography and impossibility to verify whether students really understand the content, whether they got the heart of the matter and are able to apply the obtained knowledge. Obviously, it is still far from today, when computers will be able to substitute human teachers in this respect. An interesting step, that at least simulates such a function and that is investigated by our current experiments, is the employment of standard ontologies for this purpose.

Table 1 presents the relations from SUMO[1] that have been used in our experiments. A set of test-question templates is defined for each type of the relations and natural language questions (in Czech) are generated.

[1] http://www.ontologyportal.org/

Table 1. Relations from SUMO covered by semantic patterns

beforeOrEqual	geometricDistance
causes	geometricPart
causesSubclass	geopoliticalSubdivision
completelyFills	greaterThan
component	greaterThanOrEqualTo
conditionalProbability	hasPurpose
confersObligation	immediateInstance
confersRight	immediateSubclass
connected	increasesLikelihood
connectedEngineeringComponents	independentProbability
connects	lessThanOrEqualTo
connectsEngineeringComponents	material
containsInformation	meetsSpatially
contraryAttribute	meetsTemporally
decreasesLikelihood	member
disjointDecomposition	occupiesPosition
disjointRelation	origin
domainSubclass	overlapsPartially
employs	overlapsSpatially
experiencer	overlapsTemporally

The attempts to make the assessment of student knowledge as objective as possible and to reduce the teacher's work led to the spread of multi-choice tests in last decades. It is a well-known fact that this form of testing has many disadvantages too and that the focus on them can easily produce test-experts rather than people understanding of the subject. The current technology offers various means to implement intelligent tests and to escape the trap of multi-choice tests. Our research concentrates on the integration of ontologies into the evaluation module of LMS. The experience shows that it is relatively easy to provide such functionality for short answers in the form of phrases described by simple grammatical patterns. The results of the first limited experiment are very promising as the method reduced the number of answers that needed to be processed manually significantly (only 31 % of the cases remained for the manual check). However, there are also many open questions concerning scaling-up the method of the answer patterns. The longer the answers are, the less matches between the pre-defined applications are found. A more general solution of the analyzing mechanism in the form of a general grammar is obviously needed. However, the available grammar of Czech is developed for a robust analyzer, so the ambiguity of the analysis tends to be rather high. Also it is much more difficult for authors that do not specialize in the computational linguistics to define the acceptable forms of answers. The assessment of open questions is definitively one of the actual topics of our research.

Although the current state-of-the-art in NLP does not allow full computerization of the general test assessment, there are areas, where the technology can already take over the role traditionally falling upon the teacher. We have re-

cently implemented the automatic evaluation of programs in the Java programming course at FI MU and also the phrase-structure grammars developed by students of the "Introduction to the computational linguistics" course. Students are obliged to provide their assignments in the designated form. Pre-defined tests are applied to verify the correctness of the provided solutions. We are to provide a similar function for a grammatical pre-checking of students' essays. Even though only a limited number of error types can be identified automatically, the method can still reduce a significant portion of the tedious teachers' work.

4 Conclusions

We currently work on a module that will be able to summarize the content of related documents based on the ontological information. The immediate target of the tool is the summarization of messages in course discussion groups. Sometimes, students are very active (the discussion is occasionally off-topic) and even a simple browsing through the discussion threads from previous runs of a course could present a tedious work. The automatic extraction of the most important points can significantly help newcomers.

Ontologies can serve as a launching point for the LMS personalization. Some e-learning courses already include special mechanisms to offer a path through the teaching material adapted to the needs of a particular user. Such an enhancement can improve the acceptance of the content by users which often face learning materials that are too in-depth in some parts and too sketchy in others. Also, the speed of presentation of the teaching material can vary according to the user's needs. Besides the above-mentioned assignment of essays, the simple statistical technique has been adopted that automatically evaluates the match of students' answers and determines knowledge levels of the students compared to relations extracted from ontologies. The solution is now ready to be applied generally to all e-learning courses provided by MU. It can be generalized to allow semi-automatic generation of tests for each part of the subject matter that would "send" students (back) to the parts they should go through again.

Acknowledgements. This work was supported by the Academy of Sciences of the Czech Republic, "Information society" program, research grant 1ET208050401

References

1. Pavlovic, J., Pitner, T., Smrz, P., Verner, J.: Customization of ILIAS and its integration with the university information system. In: ILIAS 2003, Cologne, Germany (2003)
2. Smrz, P.: Integrating natural language processing into e-learning — a case of czech. In Lemnitzer, L., Meurers, D., Hinrichs, E., eds.: Proceedings of the Workshop on eLearning for Computational Linguistics and Computational Linguistics for eLearning, COLING 2004, Geneva, Switzerland (2004) 1–10

Towards an Ontological Support for eLearning Courses

Antonio De Nicola, Michele Missikoff, and Frederica Schiappelli

LEKS, IASI-CNR
Viale Manzoni 30, Rome,
00185, Italy
{denicola, missikoff, schiappelli}@iasi.cnr.it

Abstract. An ontological system integrated with an eLearning platform can support teachers in building courses and students in accessing courses content. To this end we propose an ontological approach to semantically enrich eLearning courses. In particular the association of semantics to the learning resources can greatly improve their organization and management, both for students and teachers. This paper proposes a method for creating eLearning courses based on repositories of semantically annotated learning objects. The proposed method supports teachers in flexibly building learning paths and students in customising and extending the courses according to their needs.

1 Introduction

eLearning is based on the availability of a system which allows online courses to be developed and deployed. It offers several advantages especially concerning the management of the courses content [1], allowing the learning paths to be adapted to the students needs. Furthermore, an eLearning course can be easily modified by teachers and students. Using an eLearning system, the access to the learning resources can be considered not linear and continuous. In fact, a student can decide to access the resources in a sequence different from the one defined by the teacher.

One of the key issues in developing an eLearning course is the organization of the learning resources. A good organization of such resources allows the users to easily discover and access new resources.

In this paper we propose the use of ontologies for eLearning courses in gathering and organization of teaching material, in course construction and in the learning phase. An ontology is a formal, explicit specification of a shared conceptualisation [2]. It gathers a set of concepts (concerning entities, attributes, processes, etc.) related to a given application domain, together with their definitions and inter-relationships. The precise, formalized and not ambiguous description of the course content can be achieved with explicit references to the ontology content. Such references, referred to as semantic annotations [3], are associated to the learning resources, supposedly represented as Reusable Learning Objects[1] (RLOs). Semantic annotation can be also used to support the semantic search and retrieval of RLOs. This functionality will be used in selecting the learning objects during the process of course development.

[1] An RLO is any entity, digital or non-digital, which can be used, re-used or referenced during technology supported learning. [http://ltsc.ieee.org/wg12/index.html]

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 773–777, 2004.
© Springer-Verlag Berlin Heidelberg 2004

The EDUTELLA Project [4] is one of the first proposals concerning the association of semantics to the learning resources. This open source project represents a peer-to-peer network for RDF[2]-based digital resources used for the exchange of learning objects and services [5]. One of the most important services offered by EDUTELLA is the *Query Service* for the search and the retrieval of the learning objects.

EducaNext [6] is a portal that allows sharing, retrieving and reusing learning resources to be achieved. It is based on the Universal Brokerage Platform [7] which, unlike our ontology-based approach, is conceived for keywords managing rather than concepts.

In the rest of the paper, in Section 2, we describe how an ontology can be used by teachers in the *course building phase* and by students in the *learning phase.* Finally, in Section 3, we present our conclusions and the possible future work.

2 Semantic Annotation for eLearning Courses

Ontologies can be an important support both for the *course building phase* and the *learning phase.* In the *course building phase* ontologies support teachers in the activities of analysis and semantic annotation of learning objects and the definition of a teaching course. A course, from a conceptual point of view, is seen as a path over the graph that models an ontology. In the *learning phase* ontologies support students in following a course from the eLearning platform. Students may follow the given learning path, or may dynamically modify it.

To model the structure of an ontology-based course, we propose a two level organization of information. An upper level, with the set of concepts related to the course topics, selected among the concepts defined in the domain ontology. A lower level, with the learning resources (books, web presentations, movies, test..) associated to the concepts (Fig. 1).

Following this approach, teaching courses can be created, at a conceptual level, browsing the ontology, i.e., following the semantic relations of the underlying semantic network. Since every concept is associated with one or more learning resources,

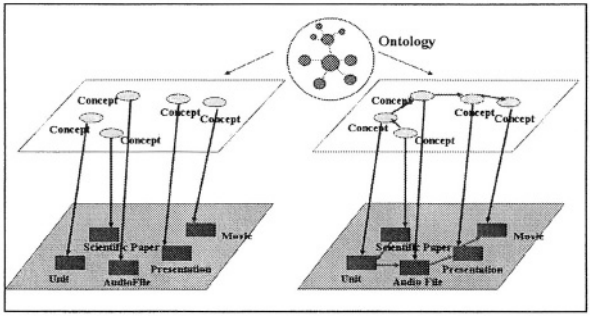


Fig. 1. Two level organization of information in an ontology-based course

[2] RDF: Resource Description Framework. [http://www.w3.org/RDF/]

every learning path defined on the semantic net generates a partially ordered sequence of learning objects. In essence, in the *course building phase* we have:

- a sequence of concepts selected during the ontology browsing. This sequence gives the order of deployment of the learning objects.
- a corresponding sequence of learning objects, associated to the ontology concepts, that compose the customised course, used by the student to learn, and by the teacher, to verify the students' learning.

This method can be open also to students allowing them to choose the course topics, the learning objects and the lessons' sequence. In this way an autodidact can adapt the course sequence topics, and learning objects, to his real learning needs, trying to flexibly achieve his learning goal.

We envisage two ways of building learning paths, along two different dimensions: the "Horizontal Dimension" and the "Vertical Dimension" (Fig. 2).

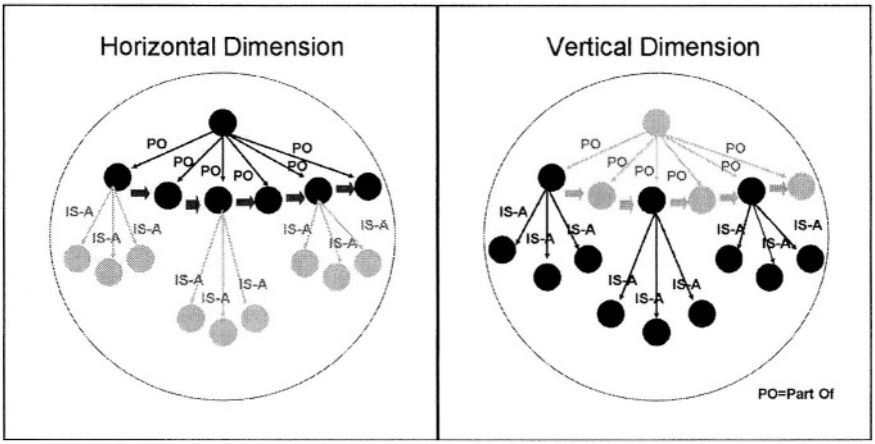


Fig. 2. Ways envisaged to build learning paths

Following the first dimension, a teacher (or a student) can build a learning sequence browsing the ontology through the decomposition relations: starting from a given concept (corresponding to the main topic of the course). Then, they can select from the "part-of concepts the ones that will participate in the course sequence. The actual course will consist of the learning objects associated to the selected concepts.

With the second approach, we consider that an ontology provides elements for a further elaboration of a given topic, reached by following the specialisation links. For instance, considering a course on ontological modelling, we have the concept *Ontology Representation Language* associated to the learning object *Survey on ontology definition methodologies* (Fig. 3). To better understand this topic we can consider the learning objects, associated to the specialized concepts: *Graphical Representation Language* and *Logic-based Representation Language*. Furthermore, following upward the specialisation links, a teacher can reach more general concepts, associated to learning objects allowing the student to wrap-up and make a synthesis of the studied material.

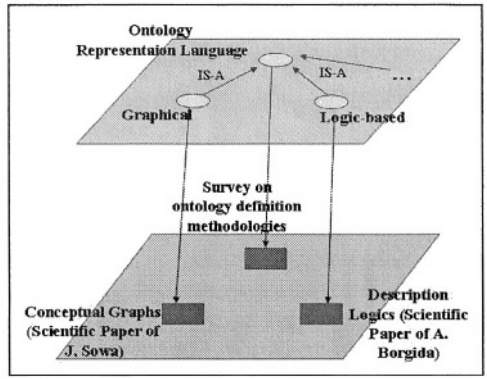


Fig. 3 Browsing through the IS-A hierarchies to analyse better the course's topics

3 Conclusions and Future Work

In this paper we have presented how ontologies can be used to organize and support the access to the learning objects of an eLearning course. We have seen that they can be useful both in the *course building phase* (both from the teacher and the student point of view), and in the *learning phase* (from the student point of view).

The work presented in this paper, carried out within the Italian Project *Web Learning,* is based on the ontology management system SymontoZ and the ontology representation method OPAL, developed by LEKS (IASI-CNR) [8].

As future work, we are going to address the techniques for semantic queries on an ontology, in order to improve the retrieval of the learning resources. Furthermore we are going to improve the *learning paths building* techniques. In particular we are elaborating further issues, such as the ones addressed in KBS Hyperbook, to adapt contents to learning goals and previous knowledge of students [9].

To validate our approach, we are developing a course on ontological modelling. Currently we have developed a first version of the reference ontology, called METAONTO, that holds 168 concepts [10] relevant in the Ontology Modelling domain. The next step of the course development will be the collection of learning objects and their semantic annotation.

References

1. P. Drucker: Need to Know: Integrating e-Learning with High Velocity Value Chains. A Delphi Group White Paper, 2000, [http://www.delphigroup.com/pubs/whitepapers/20001213-e-learning-wp.pdf]
2. T. R. Gruber: A translation approach to portable ontologies. Knowledge Acquisition. 5(2), (1993), pp. 199-220. [http://ksl-web.stanford.edu/knowledge-sharing/papers/#ontolingua].
3. M. Missikoff, F. Schiappelli, F. Taglino: A Controlled Language for Semantic Annotation and Interoperability in e-Business Applications. Proceedings of the 2nd International Semantic Web Conference (ISWC2003), Sanibel - Florida, Oct 2003.

4. W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, T. Rish: EDUTELLA: A P2P Networking Infrastructure Based on RDF. Proceedings of the 11th World Wide Web Conference (WWW2002), May 7-11, 2002, Honolulu, Hawaii, USA.
5. Jan Brase, Wolfgang Nejdl: Ontologies and Metadata for eLearning. - Handbook on Ontologies – Springer-Verlag (2004)
6. EducaNext: A Service for Knowledge Sharing-Technology. White Paper [http://www.educanext.org/ubp/PUSH/aboutTechWhitepaper]
7. G. Vrabic, B. Simon: Learning Resource Catalogue Design of the UNIVERSAL Brokerage Platform. Proceedings of ED-MEDIA 2001. Copyright by AACE. Reprinted from the ED-MEDIA 2001 Proceedings, June 2001 with permission of AACE, Tampere, Finland, June, 2001
8. M. Missikoff, F. Taglino: Symontox: A Web-Ontology Tool for eBusiness Domains. Proceedings of the 4th International Conference on Web Information Systems Engineering (WISE2003), Roma-Italy, Dec 2003
9. N. Henze: Open Adaptive Hypermedia: An approach to adaptive information presentation on the Web. First International Conference on Universal Access in Human-Computer Interaction (UAHCI 2001), held jointly with HCI International 2001, 5-10 August 2001, New Orleans, Louisiana, USA.
10. A. De Nicola, M. Missikoff, F. Schiappelli: Verso un supporto ontologico per la creazione di corsi eLearning. I Workshop annuale del "Progetto Web-Learning per la qualità del capitale umano", Trento (Italia), Settembre 2003, pp. 52-64.

PhD Symposium (PhDS)
PC Co-chairs' Message

We are proud to present the first Symposium for PhD students associated with the "On The Move confederated conferences". The word 'symposium' has two basic meanings. Firstly, it means 'the presentation and discussion of short essays on a common topic by different authors', and secondly 'a merry feast' or 'drinking together'. As this goes back to famous philosophers such as Socrates and Plato, who are we to deviate from this esteemed tradition of 'dulce et utile' (as elegantly summarised by Horace).

Of the 'dulce' (sweet or nice) aspects, we cannot include publishable material (or maybe rather should not?). Regarding the 'utile' (useful or profitable) aspect, we gratefully acknowledge the opportunity offered by OTM to junior researchers to publish their paper in the famous Springer LNCS series.

Four eminent professors help to recreate the Socratic dialogue setting of scientific interrogation, discussion, reflexion and stimulation between students and mentors. For taking up this challenging task, we would like to vividly thank our team of accompanying professors, namely:

- Sonia Bergamaschi, DataBase Group, Facoltà di Ingegneria - sede di Modena, Università di Modena e Reggio Emilia, Italy
- Jaime Delgado, Distributed Multimedia Applications Group (DMAG), Technology Department, Universitat Pompeu , Spain
- Jan Dietz, Department of Information Systems, Faculty of Technical Mathematics and Informatics, Technical University of Delft, The Netherlands
- Werner Nutt, School of Mathematical and Computer Sciences, Heriot-Watt University Edinburgh, U.K.

We invite the reader to subsequently discover the next generation of promising scientists in our research field and get acquainted with their research topics.

Best regards from the OTM 2004 PhD Symposium program chairs!

August 2004

Peter Spyns, Vrije Universiteit Brussel (Belgium)
Antonia Albani, University of Augsburg (Germany)
Alexios Palinginis, Swiss Federal Institute of Technology Zurich (Switzerland)
(PhD Symposium 2004 PC Co-chairs)

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, p. 778, 2004.
© Springer-Verlag Berlin Heidelberg 2004

Management of Dynamic and Imperfect User Context Information

Andreas Schmidt

FZI – Research Center for Information Technologies
Haid-und-Neu-Straße 10-14, 76131 Karlsruhe, GERMANY
Andreas.Schmidt@fzi.de

Abstract. In high-volume information systems, there is a trade-off between ease of querying and selectivity of the query (and thus relevance of the result set). This can be overcome by incorporating user context information into the interaction. But user context information is hard to acquire due to its dynamic nature and the approximative methods. This thesis work proposes a context model, which builds on annotations for qualifying and quantifying imperfection, and introduces the notion of sub contexts, which can be used to capture the dynamics of user context information. Its approach aligns with Semantic Web technologies.

1 Introduction

With the ever-increasing volume of accessible information and the advent of ubiquitous information access via mobile and wearable devices, the focus of information systems research has shifted to increasing the efficiency of information access for the user. This encompasses both simplifying the query formulation process and improving the relevance of query results. In general, there is a trade-off between preciseness (and thus selectivity) of queries and ease of query formulation. The most promising approach is to incorporate information about the user and his current situation (i.e. his "context" (Dey 2001)), which is the fundamental approach in context-aware and situation-aware systems.

Let us take an example from the e-learning domain. Current e-learning solutions offer the learner the possibility either to take pre-assigned learning units or to actively search (or browse) for available courses. Especially in a corporate setting, this leads to an unnecessary separation of work and learning. It would be desirable to offer the learner (or worker) during his work relevant learning objects (which should be rather small ones), based on his knowledge profile, his task-at-hand, his short-term or long-term goals, and other learning preferences. From this information plus some background knowledge about the knowledge requirements of context aspects, the system can automatically propose relevant learning material (see Schmidt & Winterhalter 2004).

Apart from determining how the context influences the information need of a user, the key problem in these systems is how to acquire user context information. In general, only indirect methods can be used and several sources of context information have to be considered. Some information can be derived from the user's interaction

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 779–786, 2004.
© Springer-Verlag Berlin Heidelberg 2004

with a specific applications, other pieces can be obtained from data stored in other systems, e.g. in a corporate environment from Human Resources data, Workflow Management Systems (van Elst et al. 2001), or Personal Information Managers (addresses, calendar etc.), still others can be retrieved on demand with specialized operations (e.g. localization in wireless networks). This complexity should be hidden from a context-aware application. Additionally, this effort for collecting user context information does not pay off if we use it only for a single application. So the basic idea is to establish a "user context management" middleware that provides applications with an up-to-date view of the user's current context, partially materializing it, partially retrieving it dynamically from other sources. This context management infrastructure can also provide facilities to bridge the abstraction gap between the information that can be collected and the need of the application by providing an aggregation and reasoning framework.

However, this idea faces several challenges, which cannot be adequately met by existing information management solutions:

- **Dynamics.** The context of a user continuously changes. Different features of the context change at different pace; e.g. name and occupation change less frequently than location or current task. Additionally, we have to distinguish between context evolution and context switching. In the latter case, part of the context changes, but it is quite likely that it later changes back again so that we should store the information for later use. Typical examples are private and professional information, project-specific context and role-specific context.
- **Imperfection.** User context information is typically collected via indirect methods that observe a user's behavior and infer facts from these observations. These methods do not yield certain results, in some case they are more, in other cases less probable (uncertainty). Additionally, some of the information cannot be determined exactly (imprecision). The most typical example here is location information. Depending on the method (GSM, GPS, etc.), the precision of the information can vary a lot, which is particularly important for the most prominent examples of context-aware services: location-based services. Additional aspects of imperfection in the area of user context information are incompleteness and (as we collect it from several sources and/or with a variety of methods) inconsistency.

These challenges have led to the primary idea of this Ph.D. thesis: Provide a user context management service that is aware of the imperfection and dynamics of the managed information and can still provide certain guarantees about the quality of the provided information.

2 Approach

2.1 Context Model

As a foundation for the proposed context management service, a context model has been developed on the basis of a subset of the RDFS data model, which was chosen

because (1) it provides sufficient schema flexibility in an open environment, (2) it allows for defining semantic links on both schema and instance level and (3) it is supported by a range of different products, which could serve as backend services.

The RDFS-based data model is extended by adding "imperfection metadata", which is based on probability theory and temporal elements. On instance level, property values can be annotated with distribution information and confidence levels for values. On schema level, property definitions are annotated by aging functions which specify how confidence (or certainty) decreases over time since the system received evidence about a property value. Additionally, the context model incorporates some elements of temporal data models by including transaction times (needed for aging) and validity intervals (which enables to reason about context history).

Apart from aging, the main contribution to manage dynamics of context information is the notion of sub contexts, groups of property values that typically change together. Sub contexts have the following characteristics:

- Sub contexts can be nested.
- Sub contexts conform to a schema that defines (1) available properties and (2) nesting relationships.
- For each sub context schema, there is at most one sub context active at a certain instant of time. The others are present as inactive, "parallel" sub contexts yielding the same aspects about the user.

An example for a sub context structure is given in figure 1. Here you have a user with location-dependent, role and project-dependent information. Currently the user "John Smith" is at his office and thus has broadband network access, but no loud-speakers (which could influence whether the system can deliver video or audio

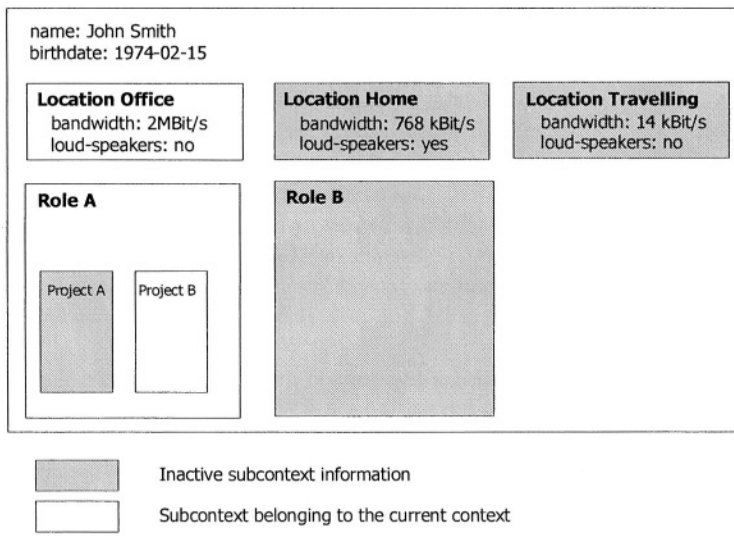


Fig. 1. A sample user context

material). The system also knows the characteristics of other locations, but those are represented as currently inactive sub contexts. There is also role-specific information and within each role project-specific information. In the case of corporate e-learning, this information could in its majority be provided manually by a learning coordinator or human resources department.

A formal semantics of the context model and its supported query capabilities and update operations is currently being developed based on probabilistic (see Parsons 1996) and temporal (see Artale 2000) extensions to Description Logics, with Description Logics being the foundation for OWL-DL and major parts of Semantic Web research. This approach allows for incorporating (computable) logic-based reasoning capabilities into the context model.

2.2 Architecture

The second element of work is a reference architecture for a context middleware supporting context-aware applications. The conceptual architecture for the user context management middleware has the following components:

- The core of the context information management system consists of a **Feature Value Store,** a **Schema Store** and a **Sub Context Manager.** While the first two are comparable to traditional databases, the Sub Context Manager encapsulates the context specific behavior. It has the responsibility of (1) storing sub contexts and sub context schemas, of (2) keeping the set of active sub contexts up to date and of (3) recognizing patterns in context changes that allow to build the sub context schema. For (2) and (3), the Sub Context Manager will consist of an extensible set of strategies.
- **Context Sources** are expected to provide metadata about confidence and precision. The transmission of user context information can be initiated by the context sources themselves (e.g. sensor-based sources, UI event driven sources or workflow engines) or by the management core (e.g. determine the location of a user via GSM, access to calendar or address book).
- Sensor-based data, but also low-level user interface events are often too fine-grained and rather ephemeral. It can be used as an input for short-term user modeling techniques based on Bayesian networks, Hidden Markov Models or similar techniques, which yield information about the user on a more abstract level. These techniques can be encapsulated in so-called **Aggregators.** An example would be an aggregator that collects user interface events inside an application in order to determine the task-at-hand or current process step of the user.
- **Context Reasoning Agents** allow to add capabilities to calculate additional information of the user from existing values. They can use the management core as a blackboard. All their updates to the user context management must be accompanied by dependency information and indications about the confidence.

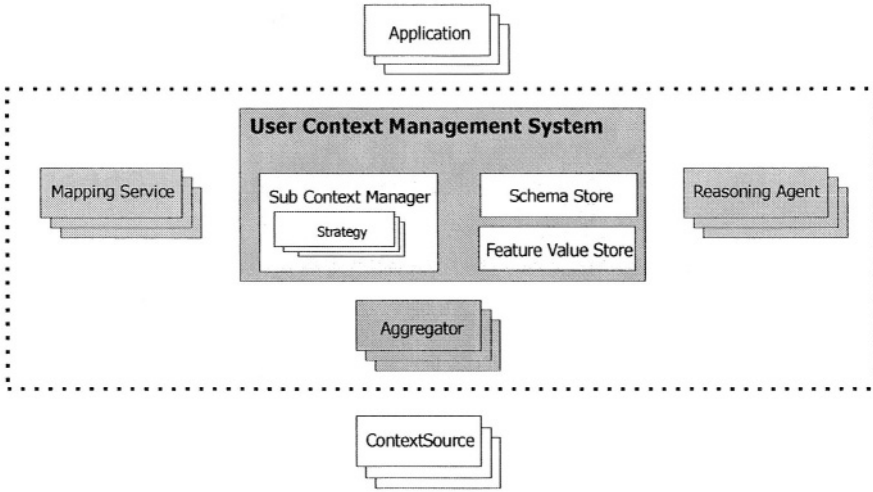


Fig. 2. Conceptual architecture of the user context management middleware

- **Mapping Services** allow to bridge schema heterogeneity between our management system and the "outside world". It cannot be expected that there will be a single user context ontology, so we have to provide integration services (similar to Kazakos et al. 2002).

2.3 Management of Dynamics

The introduction of sub contexts opens several research issues, which are represented in the architecture as a set of strategies of the sub context manager and are the primary focus of current research:

- **Detection of sub context changes.** The most crucial part of the research is how to detect context changes, or to be more precise: how to distinguish context evolution from context switching.
- **Automatic construction of sub contexts.** Although being quite practical for closed environment like intranet e-learning solutions, the manual specification of schemas for sub contexts limits the scope of a generic user context management service. Therefore automatic methods for sub context detection are investigated. Promising approaches here are Data Mining approaches, but they have to be adapted to (1) the scarce amount of available data and (2) uncertainty aspects.
- **Sharing of sub contexts across different users.** Context-aware services distinguish themselves from personalized services by the fact that they do not only consider direct attributes of the user, but also characteristics of the environment (e.g. location, company, community etc.), which are not user-specific. This can be easily represented with shared sub contexts and can drastically improve the quality of data delivered by the user context management service.

3 State of the Art

There has been a long research tradition in user modeling (especially in Information Retrieval, Belkin 1997), and user context-awareness can be seen as an extension of the user modeling approach by not only considering the user's direct interaction with the system, but also a larger set of relevant information about the user. Many of the user modeling techniques specialize on a certain application, but there also has been some significant research on "generic user modeling" (Kobsa 2001), which has yielded several successful commercial applications. These systems however focus more on aggregation and reasoning techniques (using the terms introduced in section 5) than on the dynamic nature of user context and its implication on the management infrastructure.

For most context-aware applications and research projects during the last years, location is the primary form of context. A pioneering project in that area was the GUIDE project, providing a location-aware tourist guide (Cheverst 2000). Context modeling was a rather simple task. ContextToolkit (Salber et al 1999, Dey et al. 2002) is a "context middleware" for sensor-originating context information that needs to be aggregated. Its data model is rather simple; its focus lies on the aggregation of sensor data. Due to this real-time nature of context information, the persistence and management of such data does not play a prominent role. This is different if we consider information not collected from sensors, but from users' interaction with the system.

Recent research in the area of user modeling (especially for e-learning) also follows ontological or Semantic Web approaches (Heckmann 2003a, Heckmann 2003b, Dolog & Nejdl 2003, Nébel et al. 2003). Especially the approach presented in (Heckmann 2003a, 2003b) goes in the same direction, as it also tries to represent user context data in RDF. It also allows for representing uncertainty. However, this approach does not address the dynamic nature of user context information. It neither supports context switching nor any schema-level metadata that help the user context management application to adapt to the specific dynamics of context features.

4 Research Methodology and Status

In order to gain insight into the nature of user context information, the research process was split into two phases. In the first phase, preliminary scenario studies for context-aware e-learning were conducted (e.g. in a sales department of a logistics company). This has led to a set of requirements for context-aware e-learning and to the first version of the context model and the architecture. Within the framework of the LIP project (Learning in Process, Schmidt 2004), the model was refined with additional case studies and scenario-based evaluation techniques (Cook et al. 2004) conducted by other partners within the project, and a basic version of the user context manager was implemented ontop of an ontology management platform (KAON, Mädche et al. 2003). As context sources, a browser plugin for Internet Explorer and a plugin for Microsoft Office were developed. The prototype for a context-aware e-learning platform is currently being evaluated with end users.

In a second phase, based on the results of the project, the strategies for context switching, sub context construction and sub context sharing will be investigated further in order to provide more sophisticated support for the dynamics of user context information. A formal model based on Description Logics currently being specified will allow for deducing properties of this context model. A set of strategies will be elaborated and compared in a simulation setting.

Also schema management issues will have to be considered, which allow to dynamically extend the schema of stored context information.

The results of this phase will eventually be evaluated.

5 Conclusions

The most fundamental contribution of this PhD work is the consideration of the dynamic and imperfect nature of user context information. This is achieved through two ideas: annotations with information about imperfection and parallel sub contexts that do not only capture the current context, but also past and potential future parts of the user's context, which can increase both quantity and quality of available information about a user.

The proposed user context management infrastructure can reduce the effort for implementing context-aware functionality in a new generation of applications, utilizing a well-founded query language and update operations.

References

Artale, A. and Franconi, E. 2000. A Survey of Temporal Extensions of Description Logics. *Annals of Mathematics and Artificial Intelligence,* vol. 30, pp. 171-210.

Belkin, N. 1997. User Modeling in Information Retrieval. Tutorial *at the 6th Conference on User Modeling (UM 97),* Chia Laguna

Cheverst, K., Davies, N., Mitchell, K. and Smith, P. 2000. Providing Tailored (Context-Aware) Information to City Visitors. *Conference on Adaptive Hypermedia and Adaptive Web-based Systems,* Trento

Cook, J., Bradley, C. and Franzolini, P. 2004. Gathering Users' Requirements Using the Artifact-Claims-Implication Technique. *Proceedings of Ed-Media 2004.* Lugano, Switzerland

Dey, A. Mankoff, J., Abowd, G., Carter, S. 2002. Distributed Mediation of Ambiguous Context in Aware Environments. *Proceedings of the 15th Annual Symposium on User Interface Software and Technology (UIST 2002),* Paris, France, October 28-30, 2002, pp. 121-130.

Dey, A. 2001. Understanding and Using Context. *In Personal and Ubiquitous Computing Journal,* Vol. 5, No. 1, pp 4-7.

Dolog, P., and Nejdl, W. 2003. Challenges and Benefits of the Semantic Web for User Modelling. *Proceedings ofAdaptive Hypermedia '03*

Heckmann, D. 2003b. Introducing "Situational Statements" as an integrating Data Structure for User Modeling, Context-Awareness and Resource-Adaptive Computing. *Proceedings of Workshop on Adaptivity and User Modeling in Interactive Software Systems,* Karlsruhe, October

Heckmann, D. 2003a. A Specialized Representation for Ubiquitous Computing and User Modeling. *Proceedings of the First Workshop on User Modelling for Ubiquitous Computing at UM 2003,* June 2003

Kazakos, W. et al. 2002. XI³ - Towards an Integration Web. *Proceedings of WITS'02,* Barcelona

Kobsa, A. 2001. Generic User Modeling. *In: User Modeling and User-Adapted Interaction,* Vol. 11 No. 49, pp. 49–63

Mädche, A., Motik, B., Stojanovic, L., Studer, R., and Volz, R. 2003. An Infrastructure for Searching, Reusing and Evolving Distributed Ontologies. *Proceedings of WWW 2003,* Budapest, Hungary

Nébel, I., Smith, B., Paschke, R. 2003. A User Profiling Component with the aid of User Ontologies. *Proceedings of Workshop Learning – Teaching – Knowledge – Adaptivity (LLWA 03),* Karlsruhe

Parsons, S. 1996. Current Approaches to Handling Imperfect Information in Data and Knowledge Bases. *IEEE Transactions on Knowledge and Data Engineering,* vol. 8, no. 3, pp.353-372

Salber, D., Dey, A.K., Abowd, G. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In: Proceedings of the Conference on Human Factors in Computing Systems (CHI 99), Pittsburgh, 1999, pp. 434-441

Schmidt, A. 2004. Context-Steered Learning: The Learning in Process Approach. *Proceedings of IEEE International Conference on Advanced Learning Technologies (ICALT '04),* Joensuu, Finland, 2004.

Schmidt, A., and Winterhalter, C. 2004. User Context Aware Delivery of E-Learning Material – Approach and Architecture. *Journal of Universal Computer Science,* Vol. 10, No. 1, pp. 28-36

Van Elst, L., Abecker, A., and Maus, H. 2001. Exploiting User and Process Context for Knowledge Management Systems. *Proceedings of UM 2001, Workshop on User Modelling for Context-Aware Applications*

Bridging Information Gaps in an OPM-Based System Development Lifecycle

Avi Soffer

Technion, Israel Institute of Technology
asoffer@tx.technion.ac.il

Abstract. Securing the consistency and continuity in the representation of various types of information along the system's development lifecycle is proposed as an approach to reducing the mismatch between the required and the implemented system. At the basis of this proposal is the model continuity principle, which stipulates that a system modeling and development methodology should facilitate seamless transition between the different stages in the development process, represented in a single unified model that supports the entire lifecycle of the system. Implementing this principal as part of OPM-based system development is expected to significantly reduce the inherent complexity of model-based system development process and close the undesired gap that normally exists between the required system and its implementation.

1 Introduction

The complex nature of software intensive information systems introduces an undesired situation: at the end of the process of converting a vision into a working system there is a mismatch between the implemented system and the envisioned one. Eliminating or at least reducing this mismatch is a significant challenge for the software engineering research. One approach to this problem may be an ongoing verification activity that parallels all stages and processes throughout the system's life cycle. This is an expensive approach. Since the goal of all product development cycles is to achieve economic efficiency, the effort to secure the system's integrity has to be balanced with the cost goals of the development process. Therefore a more efficient solution is desired.

A different approach is the use of advanced and powerful modeling methods. The growing attention and wide acceptance of this approach shows that the software engineering community places great hopes on software modeling notations and techniques that will support the cognitive effort required in creating a system. This approach introduces new difficulties that stem from the inherent complexity of the development process. Effective and efficient solutions within the field of system modeling will reduce the mismatch between the required and the implemented system.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 787–797, 2004.
© Springer-Verlag Berlin Heidelberg 2004

1.1 System Modeling Challenges

As part of the research and practice of the development of large, complex systems, the problem of maintaining a coherent, clear and correct representation of all stakeholders' views along the development life-cycle is recognized for a long time [9]. One fundamental requirement in system development is to accurately capture and represent the system's vision and needs in a complete and concise manner. A second challenge is to transform this representation – the system's model - into a working system that will realize that vision and respond to the stakeholders' needs. Achieving good results in the second challenge strongly depends on the level of success in meeting the first one.

Numerous system development methodologies support modeling of specific stages during the system's lifecycle. The variety of models may include conceptual models, specification-structural/behavioral, architecture/design, user and system interfaces, data storage, etc., thus differentially reflecting various aspects of the system. An array of models is required in order to support the entire life cycle development process. The approach that calls for model integration extends the system's representation by capturing the inter-view relationships and describing interactions among multiple views. However due to the difficulty of maintaining a coherent, clear and correct representation of all stakeholders' views along the system evolution cycle, they usually fall short of meeting the challenge of fully-integrated representation.

The failure of proper integration causes discontinuity of information flow from the complete life-cycle perspective, which contributes to the mismatches between the system requirements and the implemented system.

1.2 Object Process Methodology

Object-Process Methodology (OPM) is a comprehensive approach to systems modeling, engineering, and life-cycle support [1]. OPM integrates structure, function, and behavior of a system into a unified, holistic model. OPM also handles the inherent complexity of model-based system development by supporting gradual refinement/ abstraction of information and smooth transition across life-cycle phases. Furthermore, OPM facilitates consistent graphical formalism along with semantically equivalent text representation in natural English. This unified, bimodal representation of a system increases the human processing capability potential, which is further enhanced through automation CASE tool support (OPCAT). System developers who follow the OPM-based development process model [2] use the same methodology-supported language, which could be universal or domain-specific, to produce the artifacts that are pertinent to each stage of the system's life cycle.

OPM advocates the integration of the various life-cycle system aspects into a single, unified model. The OPM philosophy is that (a) the same system model that was started in the requirements stage is continuously developed and refined all the way to implementation, and (b) the transitions between the system's lifecycle phases are gradual and smooth rather than abrupt.

1.3 Research Motivation and Goals

Securing the continuity and consistency in the representation of various types of information that pertain to the different stages of the system's development process is proposed as an approach to narrowing or even closing the information gaps along the system's lifecycle. At the basis of this research is the ***model continuity principle,*** which stipulates that a system modeling and evolution methodology should facilitate seamless transition between the different stages in the lifecycle in a single unified model that supports the entire lifecycle of the system.

According to its philosophy, OPM facilitates the integration of the various system perspectives into a single model that is being gradually developed throughout the life-cycle, establishing a solid framework that supports the model continuity principle. However, conceptual gaps that exist between the various meta-models that represent the different stages along the system's full life cycle hinder the idea of smooth transition between life-cycle stages.

This work is concerned with analyzing the various information gaps along the development cycle, and proposing a method in the form of a model connecting framework for bridging these gaps, in order to achieve a true life-cycle continuous model. Implementing this idea in an OPM-based system development lifecycle is expected to significantly reduce the inherent complexity of model-based system development process and close the undesired gap that normally exists between the required and the implemented system. In addition, automation through CASE tool support planned as part of the research, will enable improved handling of system development challenges in the areas of requirements engineering, life-cycle verification and validation, reuse, change control, and process management.

After describing the significant problem in this field of research and formulating the research question in the introduction section, the known system modeling methods are discussed in section 2, emphasizing the problem of life-cycle information gaps. These gaps are analyzed in section 3, while section 4 introduces a general approach to bridging the gaps. This method is further developed in section 5 in the light of OPM-based system development. Finally, some closing remarks are presented in section 6.

2 Related Work: System Modeling Methods

As a result of the study of the development of large, complex systems, the software development process has been steadily enriched with more powerful and more comprehensive modeling approaches. However the problem of maintaining a coherent, clear and correct representation of all various aspects of the system (the various stakeholders' views) is recognized for a long time.

The different development/modeling methods can be classified as follows:

- General methods using generic diagrams, generic tables and generic trees. All available graphics can be used and no syntactic diagram constraints are checked and enforced.
- Unified Modeling Language (UML) models for static structure (i.e. class and object) diagrams, use-case diagrams, activity diagrams, statecharts, collaboration diagrams, component diagrams and deployment diagrams.

- Structured analysis (SA) methods for entity-relationship diagrams, data and event flow diagrams, state transition diagrams, function refinement trees, transaction-use tables and function-entity type tables.
- Miscellaneous editors such as for JSD (process structure and network diagrams), recursive process graphs and transaction decomposition tables. These editors are no longer updated but they will remain available.

UML [6] has established itself as the leading Object-Oriented analysis and design methodology. Many tools are available that can be used to manipulate this family of models. Some extensions include view integration that supports describing interactions among multiple views. Capturing the inter-view relationships is essential to enable identification and correction of consistency and conformance mismatches. However these extensions fall short of fully-integrated representation because they focus on specific transitions and do not cover the entire life cycle.

Further analysis of the modeling approaches [5] leads to another classification:

- Special-purpose development models: tend to focus on an individual software development issue or phase, typically formal and are accompanied by powerful analytical tools.
- General purpose models, which address a wider range of development issues, typically resulting in a family of models that span and relate those issues. [6], [7], [8].

The variety of modeling techniques does not offer a good solution to the known problem of system requirements – implementation mismatch. The problem stems from the inherent lack of continuity in the system model representation throughout the system's life cycle, as will be further developed ahead. The key to the solution is by understanding the nature of the gaps and the reasons for their existence.

3 Analysis of the Lifecycle Information Gaps

3.1 The Problem in Lifecycle Model-Based Development

System life-cycle development model is a framework that describes the sequence of engineering activities that take place in the course of a single development effort, and the roles and responsibilities of the system's stakeholders and professional team that are involved. Numerous life-cycle development models were suggested over the years, and they are used in the industry. Although the models differ in their details, they share a common set of core mental and engineering activities [10]. These activities are classified and organized in order to distinguish between the following stages:

1. System Initiation, which includes identifying the needs, requirements elicitation, and creation of the initial description of the system.
2. System development, including specification analysis, design, and implementation.
3. Operation, which includes usage and maintenance.

This analysis is concerned with the activities along the system development cycle, including the transitions across their boundaries (entry and exit points).

The essence of model-based system development is reflecting the system evolution throughout the different stages of the development cycle (see Fig. 1). This includes:

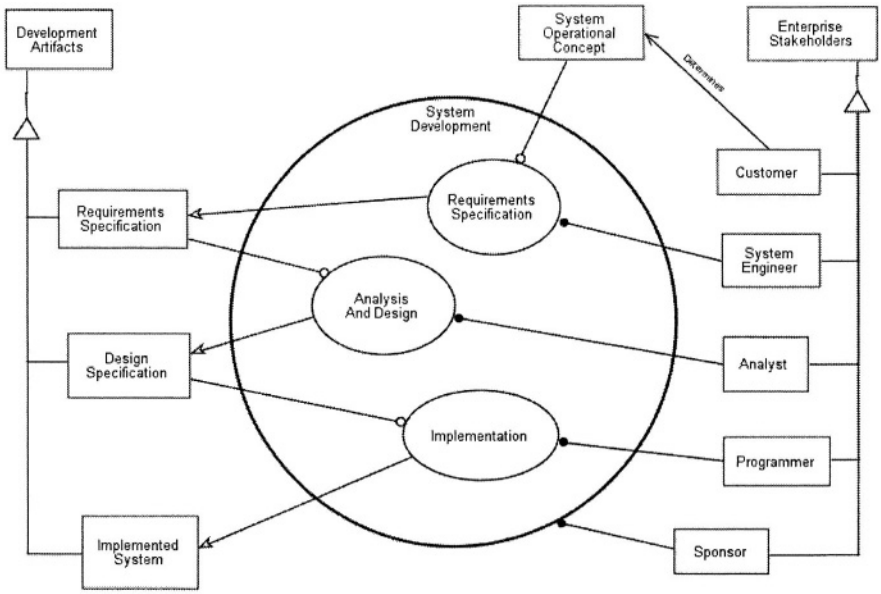


Fig. 1. The system Development Process

- Creation of conceptual models that represent the system in the specific stages
- Production of artifacts that are pertinent to each stage.

Some advanced system development methodologies support modeling of specific stages during the system's lifecycle. The models refer to system function/behavior, Human interfaces, structure/architecture, design, and implementation. Each model highlights a particular view of the system for a certain audience.

Analysis of the nature of the models and their content reveals the roots of the underlying information discontinuity problem: the stage's conceptual models differ in:

- The domain and ontology they are founded upon
- The level of abstraction they express
- Their interface with the relevant stakeholders.

In addition, there is a different group of stakeholders in each part of the development cycle: The work is done by different people and intended for different audience (as depicted in the right-hand side of Fig. 1).

The main factors that contribute to the incomplete/inconsistent representation are:
(1) The specific details of the artifacts of each stage.
(2) The individual professionals and stakeholders who are involved in each stage.
As a consequence, various aspects of the system are reflected and expressed in different ways. These differences create the gaps and discontinuity of information flow along the system's life-cycle evolution process. The information flow disruptions cause the differences to accumulated and the gaps to widen as the development process progresses, resulting in a significant mismatch between the requirements and the implemented system.

3.2 Identifying the Gaps

Analysis of the characteristics of model-based system development as described above highlights the areas of conceptual transition (transition across various representations of the system), as well transition across life-cycle stages. Disruptions of information flow are most likely to occur at these three key points in the development lifecycle:

1. Entry: Moving from freely-expressed initial ideas to a formal specification model. The two initial steps (1) identifying the need and (2) conceiving the system are supported by various process-specific tools that help stakeholders in outlining needs, expectations, goals, and constraints regarding the system. A large amount of information about the system is accumulated at this stage. The essence of the next step – initializing the development process, is generation of the system requirements model that correctly and accurately captures this knowledge. The difficulty: conversion of the semantics of the system specification document into the system requirements model. The nature of the gap is between natural language description and technical/formal specification of a system.

2. Middle: After the initial system model has been constructed, the two following steps are the analyzing and designing processes. The analyzing process is concerned with refinement of the initial (top level) system description, in order to establish a complete and correct specification of the desired appearance and behavior of the system. The design is a detailed plan of how to build a system that follows its requirements. The analysis and design conceptual models (i.e. the system requirements specification and its design/architecture specification) differ as follows:

Analysis/Specification	Design/Implementation
Belongs to the problem domain	Belongs to the solution domain
Describes what needs to be done	Describes how it is done
Handled by business domain experts and system engineers	Handled by implementation experts and software engineers

The transition from analysis to design is not trivial because of the fundamental difference between them: the system's specification is a model of the problem to be solved, where the design models the implemented system, which is the solution of the problem. Hence they use different terminologies and refer to entities that belong to separate domains.

3. Exit: In the implementation stage the system abstraction (the design) is transformed into to a concrete system, and the actual software system is created. This is another situation where a significant transition takes place – from the design model (expressed by the underlying methodology) to the system's actual code.

4 Bridging the Gaps

The proposed method for bridging the information gaps calls for establishing connecting frameworks, based on analysis of the relationships between individual elements of various aspects of the system around the stage transition. The purpose of

these connectors is to enable taking the process of modeling the system through different levels of abstraction along its development life cycle, gradually moving from the highly abstract initial user vision to the concrete results of the implementation stage.

OPM advocates the integration of the various life-cycle system aspects into a single model. Using OPM as the unified methodology for system development makes many of these information gaps inherently disappear, and communication between stakeholders of each phase is greatly improved, since in essence they can all speak the same language.

4.1 The Model Continuity Principle

Regardless of the life-cycle model or development process used for a particular system development effort, there are a number of gaps between the various conceptual models that represent the different phases of the system as it being developed. In order to enable full implementation of the OPM smooth transition philosophy, the system modeling capability needs to be extended in order to capture the relationships between the various models from the creation of the requirements model through the transformation of the design to code. This capability enables bridging the information gaps for complete elimination of information discontinuity along the system's full life cycle,

The ***model continuity principle*** is used as a general guideline for the solution. This principle stipulates that a system modeling and evolution methodology should facilitate seamless transition between the different stages of the lifecycle in a single unified model that supports the entire system lifecycle.

4.2 The Approach

There are two kinds of approaches that may be taken in an attempt to bridge the lifecycle information gaps:

1. Formal transformations - primarily supported by automation.
2. Procedural - a set of tools and methodological guidelines (e.g. RUP)

The proposed solution is a combination of both kinds. According to the analysis shown above, the connecting frameworks will be formulated in the points of rough evolutionary transitions around which the information flow is abrupt. Construction of each connector is based on (a) identification of the individual elements of the representation of both sides of the gap and (b) analysis of the relationships between them. The connector will also be formal in nature and include automated components.

A. Creating the Initial System Model

The nature of the transition is bridging the gap between natural language description and technical/formal specification of a system. This is a human-intensive process.

It should be supported by a comprehensive framework, establishing a structured transition intended to bridge the large semantic gap between high-level, sometimes ambiguous and inconsistent artifacts and the more coherent and clear description of the system. Although it still seems unrealistic to expect complete automation of this

step, partial, semi-automatic solutions that operate under human supervision [4] may already be feasible and may prove to be extremely useful.

B. From Specification to Design
The most significant transition in the system's lifecycle takes place at the heart of the development process. The analysis process conceptually results in a specification of what needs to be done, where the outcome of the design process describes how to do it. The transition in this stage is difficult because it involves the difficult cognitive transition from the problem domain to the solution domain. This includes synthesis of the system architecture, and allocation of functionality to system components. Also, this is difficult transition because it involves creation of the design from the specification details.

C. Implementing the System Design
While the first information transition is at the entry point where the system model is built, another transition occurs at the other end of the model-based system development life-cycle. When the design process is completed, the system is constructed according to the implementation specification - the outcome of the design process. At this point the system design, which is expressed by the relevant system modeling methodology, is converted into an implementation skeleton (i.e. code, database schema, UML model, etc.). This process involves transition between two domains – from the more abstract domain of the design description, presented as the design model, to the implementation domain which results in a concrete system. The details of the transformation are dependant upon the characterization of the source and destination domains.

The properties of OPM allow perfect combination integration of the two approaches, as described in the following section.

4.3 An OPM-Based Solution

OPM is a generic methodology. This is useful for defining goals, entities, actions, constraints, agents and events, and the relationships between them. It also enables modeling of any specific application domain and its environment. One of the most progressive ideas of OPM is the approach that promotes integration of the various system aspects into a single unified model that may reflect the complete system's life cycle [2]. When these capabilities are applied and matched to the model continuity issues presented earlier in the field of model-based software engineering, they enable reducing the difficulty of bridging the gap between (1) the natural language description (initial concept) and technical/formal specification of a system, (2) a formal system requirements model expressed in the problem domain's terminology and a system's architecture/design model expressed in the solution domain's terminology, and (3) a formal design specification and the system's code.

The transitions between the mental model of the system as envisioned or defined by the user into the OPM model should be supported by a procedural and formal framework enhanced by a formal, automated connector, establishing a set of structured transitions intended to bridge the large semantic gap between the different stages of the system development process, implemented as an OPM unifying model.

5 Achieving Model Continuity with OPM

5.1 The Model Continuity Framework

A system modeling and development methodology such as OPM ideally supports the entire system evolution process, from initiating, through development to full implementation. According to the these fundamental goals, rather than applying a separate model for each system aspect, OPM handles the inherent difficulties of large and complex system development by a flexible combination of three scaling mechanisms, which support gradual abstraction/refinement of information and smooth transition across life-cycle stages.

The OPM complexity management mechanism best supports the smooth transitions philosophy idea by allowing for continuous refinement of the system details as represented in the unified model. However there is a significant challenge in bridging the semantic gap that exists between the various conceptual models that represent the different stages along the system's full life cycle, from requirements/analysis (problem domain) through design/implementation (solution domain).

Conclusion: Information gaps still hinder the idea of a unifying life-cycle model-driven development methodology.

Addressing this issue is the key factor in achieving the goal of being able to use a single modeling methodology to support the entire cycle of system evolution with seamless transition between views of the system at different stages along the way.

5.2 The OPM Model Connector

Based on the analysis presented above, the goal of the OPM-based model continuity principle can be defined as follows:

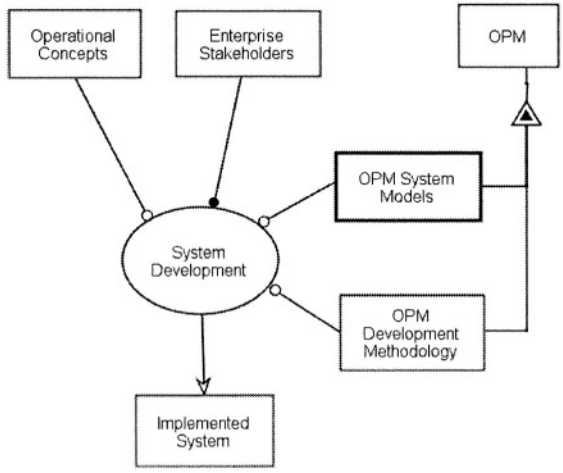


Fig. 2. OPM System Development Environment

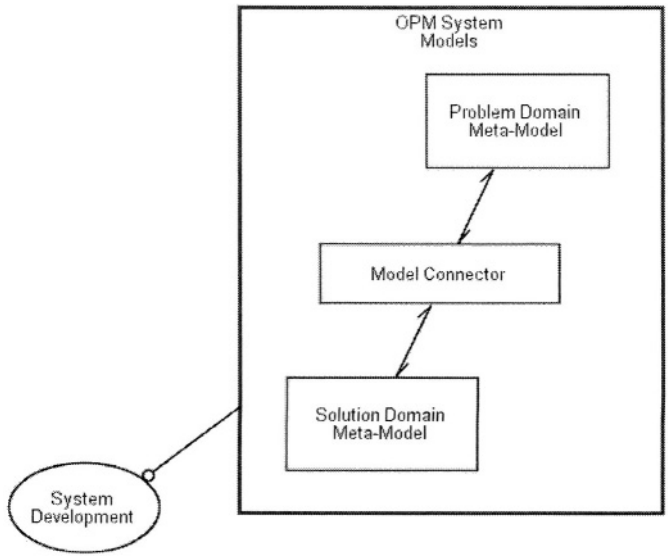


Fig. 3. OPM Model Connector

A. A single unified model that supports the complete system lifecycle evolution
B. Seamless transition between various conceptual models that represent the different stages in the life cycle.

In order to fully implement the model continuity principle in an OPM-based system's life cycle evolution model, there is a need to bridge the semantic gap that separates the problem domain aspects from the solution domain aspects of the system's unified model.

This significant conceptual gap can be bridged by a model transformation framework based on OPM's domain engineering capability, which will facilitate consistent modeling of the system's requirements specification ('what') and its implementation plan ('how').

The construction of the bridging framework includes the following steps:

1. Identification and classification of the components of the two internal models (problem/solution). This will be done using meta-modeling techniques
2. Analysis and refinement of goals, entities, actions, constraints, agents, and the relationships between them.
3. Definition of the transformation between the models based on the details of their content and the relations outlined above.

The following attributes describe the required model connector:

1. Maintaining the relationships between individual elements of the connected models.
2. Support of bidirectional transformation in order to adhere to the OPM complexity management approach. This allows traversal of the different stage models from any point in any direction.
3. The definition and implementation of each transformation is dependent upon the details (ontology/language) of the underlying source and destination models.

6 Closing Remarks

In conclusion, this work is expected to improve OPM's capability to handle the inherent complexity of system development by supporting gradual abstraction/refinement of the information expressed in the system model, and smooth yet complete and correct transition across life-cycle phases. This will minimize the mismatch between the required system and its implementation.

The ideas presented here provide a theoretical foundation for enhancing the current version of OPCAT (the OPM-based CASE tool) to support OPM-based full life-cycle system development and maintenance, as a turn-key Integrated Software Engineering Environment.

The continuous and consistent representation of various types of information that pertains to the different stages in the system's lifecycle and the smooth transitions between them create opportunities for automation and other means of improved handling of system development challenges in the areas of

- requirements engineering
- verification and validation
- change management
- reuse
- round-trip engineering
- process management and control

References

1. Dori, D. Object-Process Methodology, Springer-Verlag, New York, 2002.
2. Dori, D. and Reinhartz-Berger, I. An OPM-Based Metamodel of System Development Process, 22nd International Conference on Conceptual Modeling, October, 2003.
3. Medvidovic, N., Grunbacher, P., Egyed, A. and Boehm, B. Bridging Models across the Software Lifecycle', 13th International Conference on Software Engineering and Knowledge Engineering, June 2001.
4. Dori, D., Korda, N., Soffer, A. and Cohen, S. SMART: System Model Acquistion From Requirements Text, International Conference on Business Process Management, Potsdam, Germany, June 2004.
5. Egyed, A. and Medvidovic, N. A Formal Approach to Heterogeneous Software Modeling, 3rd International Conference on Foundational Aspects of Software Engineering, Berlin, Germany, 2000
6. Booch, G., Rumbaugh, J. and Jacobson, I, The Unified Modeling Language User Guide, Addison-Wesley, 1998
7. OMG, Meta Object Facility (MOF) v1.3.1, http://www.omg.org/technology/documents/formal/mof.htm
8. IBM, Rational Unified Process, http://www-306.ibm.com/software/awdtools/
9. Brooks, F. The Mythical Man-Month, Addison-Wesley, 1975-1995
10. Keller, M. and Shumate, K. Software Specification and Design, John Wiley & Sons, Inc., 1992

Revising and Managing Multiple Ontology Versions in a Possible Worlds Setting

Pieter De Leenheer

Semantics Technology and Applications Research Laboratory
Departement Informatica en Toegepaste Informatica
Vrije Universiteit Brussel
Pleinlaan 2, B-1050 BRUSSEL, Belgium
pdeleenh@vub.ac.be

Abstract. To date, ontology research seems to have come to an equilibrium: there is a wide variety of theories, methods and tools for extracting, representing, storing and browsing ontologies. Furthermore, ontologies have become an integral part of many academic and industrial applications. There are, however, still deep theoretical problems to be resolved. One fundamental problem is ontology evolution. Essentially, changes occur when new knowledge is agreed on by the (not necessarily human) cognitive agents. Each change engages an evolution process from which a new (possibly inconsistent) ontology version may emerge. How will we manage multiple versions of an ontology in a scalable fashion? How will we represent differences and transformations between ontologies, and moreover, can we solve this question independently from any ontology representation model? This paper describes a research project that will investigate all these critical problems in ontology evolution.

1 Introduction

The last ten years ontology research has received a renewed impetus, moreover pushed by the need for taming the wealth of knowledge that is implicitly present on-line on today's web, with the purpose of rendering into a shared resource that allows making more meaningful, and thus more productive use of that knowledge [24]. These knowledge resources are called ontologies. A plethora of research has been spent on how ontologies should be formally defined and represented in a computerisable way [15,16,23], and resulted in a wide variety of theories, methods and tools for extracting, representing, storing, browsing and using ontologies [37, 10].

Meanwhile, ontologies have become an integral part of many academic and industrial applications in the domains such as supporting interoperability, database mediation [6,39], configuration, regulatory compliance [32], semantic-based search [4], Semantic Web [3] applications [11,25,8], etc. It seems we have come to an equilibrium in ontology research, though still several deep theoretical problems need to be resolved. Evidently the methodology for manual construction (or should we say "growing") of standardisable, hence reusable computerized

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 798–809, 2004.
© Springer-Verlag Berlin Heidelberg 2004

ontologies will not be a main feature. At least we can try to enhance this "construction" by searching for algorithms that help us or perhaps automate this process. Alignment and merging of existing ontologies, ontology learning from incompatible information, and internal reorganisation of ontologies in the light of new information by means of transformations and/or versioning are the three main issues for intelligent ontology construction and evolution.

Ontology alignment and merging is not a new problem, and useful results have already been achieved [28,9]. Two agents having different ontologies merge by negotiating in order to come to an agreement on meaning, being a merged ontology. The success of the merging process depends on an (at least non-zero) initial notion of agreement on some part of the Universe of Discourse (UoD). Two agents align if they define an explicit mapping between their meanings.

Ontology learning is extracting semantic knowledge from plain natural text, web documents, dictionaries and so on, which contains (incompatible) redundant mass information. Other important sources are DB schemas, numerous existing thesauri and glossaries, general-purpose lexicons like Wordnet, or possibly even ontologies.

One of the principle characteristics of an ontology is that it should define a *reusable* knowledge component, but in order to create reusable components, evolution is critical, because effective reuse can only be achieved after a component has been evolving over an extended period of time. It is indeed unimaginable to predict all possible uses of an ontology upon its conception. In reality, the UoD will be too complex to enable a complete a priori conceptualisation, that remains valid in the light of new information, or change of pragmatics. Emerging ontologies, domain-specific or general-purpose, will continually re-organise through help of/negotiating with other ontologies (merging and alignment, issue 1), by learning from mining (issue 2) and with or without human support (e.g., change in pragmatics). It is clear that our third issue, re-organisation of ontologies, is an indispensable factor in the issues 1 and 2, and can even be seen as an independent research component. Finally, an intelligent system should be able to accommodate all such issues.

In this doctoral research we concentrate on the problem imposed by (issue 3) re-organisation of ontologies: ontology evolution. Note that in this paper we do not consider the distributed character of ontologies. Until now, relevant results of research on the problem of ontology evolution are sparse [27,18,20,35,36], and there is no consensus about the requirements. However, in currently ongoing EU 6th Framework projects[1] where ontology engineering is being studied, ontology evolution is becoming an ubiquitous part of the work packages. This phenomenon is another indicator for the global interest by the ontology research community and again proves the importance of the subject.

This paper is structured as follows: in Sect. 2 we shortly summarise existing approaches and the current knowledge in the field. In Sect. 3 we state the research question central in this Ph.D., and elicit requirements for an evolution

[1] E.g., Data, Information, and Process Integration with Semantic Web Services (DIP) (EU-FP6 507482) and Knowledge Web (EU-FP6 507482).

framework that is model-independent and manages several ontology versions and their interrelationships concurrently. Finally, we present some possible validation techniques for the proposed hypotheses in Sect. 4.

2 Related Work

Next to related work in ontology evolution [27,18,20,35,36], a considerable amount of research has been done in the data and knowledge engineering community [15,37] that can be fruitful for ontology evolution research. Results of schema integration [2,29] have been proven relevant in the research into ontology alignment and merging [28,9], and a more recent trend *is* that many researchers apply certain principles of data schema evolution [1,22]and versioning [19,31] for ontology evolution and versioning respectively.

The first methodological step in this Ph.D. research project was to make a comprehensive survey [5, manuscript] of relevant approaches and knowledge so far in the field of ontology evolution and evolution, transformations and versioning of data schemas. We also relate principle ideas of belief revision theory and possible worlds semantics.

Ontology Evolution. Not much efforts have been spent so far on ontology evolution as the field is rather new. [35] provides different levels of granularity for change operators, resulting in a taxonomy of compound change operators, inspired by those in [22]. She also suggests that different evolution strategies can be defined according to user's demands. Her semantics of change recognises the potential of cascading changes. [20] focuses on ontology evolution on the Semantic Web. They adopt principles of *pro- and retrospective use* from data schema evolution. Prospective use is the use of a data source conforming to a previous version of the schema, via a newer version of the schema. Other important results are [18], and in medical terminology evolution [27].

Data Schemas versus Ontologies. Although the problem issues in schema evolution are not entirely the same as in ontology evolution, the philosophy and results from schema evolution in general[2] are fruitfully reconsidered for the treatment of the ontology evolution problem. The argumentation behind this comparison is that:

(i) formally, all such kinds of schemas define sets of predicates (data models);
(ii) they describe some UoD by means of a (not necessarily) shared formal language[3].

[2] Evolution of object-oriented (OO), relational, entity-relationship (ER), fact-oriented (NIAM [38], ORM [17]) data schemas in particular.

[3] E.g., [7] presents a language that is able to represent ER, BRM, or relational schemas. She also defines transformations between these different models.

An ontology can be put in this picture by interpreting it roughly as a widely shared conceptual model, although the modeling methodology is different. On the other hand, what really differs is the purpose *(pragmatics)* of a database model. A relational model on the logical level is normalised such in order to get optimal efficiency, based on user-specific demands. On the conceptual level performance is not the issue, there a model should be constructed as detailed as possible in order to give optimal "semantic" performance for a given UoD. An ontology has no performance constraints as in logical models or whatsoever, but it should be more generic than the optimally semantically detailed conceptual schemas that are tailored for one system, in order to act as a common agreement between multiple applications. When evolving ontologies it is thus very important to focus more on the semantics than in data schema evolution.

Ontologies are (in principle and by definition) as generic and task-independent as possible [34, pp. 13]. The resemblance and differences between ontologies and data models are widely discussed in literature such as [24,34,26].

Significant examples of data schema evolution include *transformation rules* to effect change operators on data schemas and change propagation to the data [1], frameworks for managing multiple *versions of data schemas* coherently [19, 31] and models for *compound change operators* [4] [22].

3 Problem Statement and Requirements Elicitation

The central research question of this Ph.D. project is stated as:

> *How can we bootstrap requirements for a model-independent framework that provides methods for changing and managing different ontology versions and their interrelationships?*

In this section, we will propose some ideas that could feature such a framework. As validation we will implement such system and apply it to large-scale problems.

3.1 A Model-Independent Ontology Evolution Framework

We present a step-wise procedure for bootstrapping an ontology evolution framework:

1. select a *representation model* to adopt for evolution;
2. determine the *knowledge elements*[5] that can evolve according to the adopted model;
3. determine the possible *(atomic) change operators* that could engage an *evolution process* or *evolution strategy* on that particular model. In the simplest case, an evolution process would be the application of a finite sequence of atomic change types;

[4] e.g., moving an attribute x from a class A to a class B, means (more than) successively deleting x in A and adding x in B.

[5] a knowledge element could be any (set of) elementary ontology building block(s) such as concept, class, type, relation, slot, attribute, rule, and an instance.

4. optionally compose *compound change operators,* comparable to those discussed in Sect. 2;
5. specify *requirements* for a framework performing and validating the evolution process.

This methodological procedure can be found throughout the literature on schema and ontology evolution: e.g., Heflin [18] takes the definition of Guarino [16] as basis for his model, while Klein et al. [20] refer to the definition of Gruber [15]. Heflin's model is formal, but his definition of an ontology (being a logical theory) is very much alike with the formal definition of a data schema as in [7]. On the other hand, Klein et al. are more pragmatical in a way that they take Gruber's definition quite literally, and infer that there are three parts of the ontology (i.e. the model) to consider: the specification, the shared conceptualisation and the domain. In data schema evolution, e.g., [1] chooses the ORION object-oriented model, defers a taxonomy of possible change operators and finally defines an evolution strategy which they call *transformation rules.*

Considering the procedure above it seems inevitable to tackle the evolution problem without considering a particular knowledge representation model. In order to think representation model independent, we must represent ontologies and evolution processes more abstractly.

Possible Worlds Semantics. For abstraction we choose to adopt *Kripke semantics* (or should we say Leibniz' possible worlds) [21]. Kripke's possible worlds theory consists of a model structure having three components: a set K of *possible worlds* (of which one is privileged to be the real world), an *accessibility relation* $R(u, v)$ defined over K and an *evaluation function* $\Phi(\phi, \Omega)$ of which the details are omitted for now.

A formal ontology can be seen as a mathematical object: a logical theory, a formal language with well-formed constraints – a possible world. So using this analogy, one possible ontology Ω_2 is accessible from another possible ontology Ω_1, if there exists an evolution process (sequence of changes) applied to Ω_1, resulting in Ω_2.

As it is more appropriate in this context, we will label accessibility relations as *transformations. Transformations* are defined as finite sequences of atomic or compound change operators that map one possible ontology onto another possible ontology. Transformations can also be the composite of two other transformations. The result of the application of a transformation to a consistent ontology should again be a consistent ontology. We call such a transformation a *consistency-preserving transformation.*

Transformation Types. We cannot expect that each transformation is consistency-preserving. This statement is too strong: it would imply that we cannot add new knowledge to the ontology that is conflicting with knowledge previously entered in the ontology.

In the field of belief revision a similar problem is tackled. Their quest is: how do you update a knowledge base (belief set) in the light of new information?

What if the new information is in conflict with something that was previously held to be true? A belief set resides potential facts and theories about how the world could be, in logical parlance in fact it models a set of possible worlds [14].

We adopt the three different kinds of belief change presented in the AGM theory of [13, pp. 3] for our ontology transformation types:

1. **Expansion transformation:** a new knowledge element ϕ is added to ontology Ω together with the logical consequences of the addition (i.e. regardless whether the larger ontology Ω^* is consistent).
2. **Revision transformation:** a new knowledge element ϕ that is inconsistent with an ontology Ω is added, but, in order to maintain consistency in the resulting ontology, some of the old sentences in Ω are deleted.
3. **Contraction transformation:** some knowledge element ϕ in Ω is retracted without adding any new knowledge element.

In Fig.1, we visualise transformations in a possible ontology space, by adopting a Lindenbaum lattice [33, pp. 252]. It is an infinite lattice where the nodes represent possible ontologies, and the edges (or paths) transformations between them. A partial order is defined on the paths, that means that the paths going up lead to more general theories, and the paths going down to more specific theories. The more contractions we do, the more general the ontology becomes; ultimately ending up in the empty or universal ontology $\top$. Similarly, multiple expansions would ultimately result into the absurd ontology $\bot$. The lattice represents multiple coherent *ontology versions* and their inter-relationships in terms of differences.

In order to revise the ontology with new knowledge, one must first retract all knowledge that is inconsistent with this new knowledge, and consequently

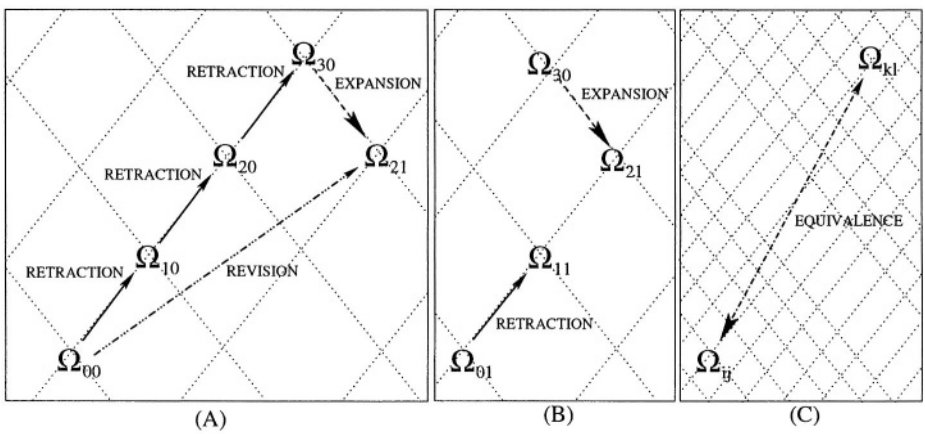


Fig. 1. Different situations might emerge when changing an ontology: (A) illustrates a revision transformation; (B) illustrates consistency-preserving transformations: a clean expansion and retraction; (C) shows equivalence relation between possible ontologies induced by equivalence transformations.

expand the latter with the new knowledge (Fig. 1.A). It is possible that an expansion or a contraction can be performed without a revision. In that case, the transformations are consistency-preserving (Fig. 1.B).

[33, pp. 252] notices that in fact each step through the lattice is simple, but that due to infinity of the number of steps it is almost impossible for humans and even computers to find the best theory for a particular problem. But even if we think we have found the optimal theory, it is possible that there exist other equivalent optimal theories. To represent semantic equivalence we introduce another type of transformation.

Equivalence-preserving Transformations. We introduce a fourth type of transformation: the *equivalence-preserving transformation.* The accessibility relation R defined by this transformation is reflexive, symmetrical and transitive. This makes R an equivalence relation, causing the set of all possible ontologies to be partitioned into disjoint equivalence classes (see Fig. 1.C and 2). Under this interpretation we can argue that an ontology is not one theory or, more appropriate, conceptualisation, but in fact is the set of all equivalent representations for that conceptualisation. In [7,30,17], comparable data schema transformations and semantic equivalence where defined: a transformation is *lossless* if there is no loss of information in the underlying population during the transfomation. In the case of "losslessness" there is a bijective mapping between the two data schemas.

Whether two ontological construct are semantically equivalent has to be decided by agreement between human agents. If the ontology is graphically representable, we could formally define graph morphisms that can be used to compare

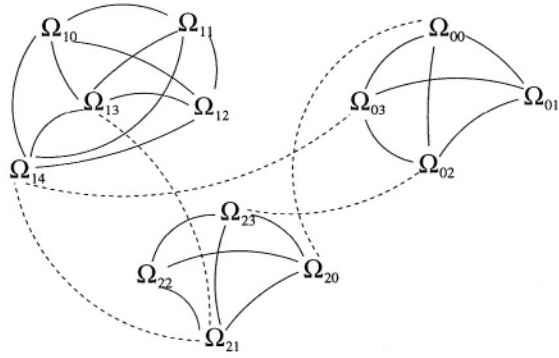


Fig. 2. This figure illustrates an arbitrary set of possible ontologies $\Omega_{i,j}$. The non-directed solid arrows between the ontologies reflect the symmetrical (and transitive) accessibility; each ontology is trivially accessible from itself so reflexive arrows are left implicit. Considering only the regular arrows illustrates three equivalence classes labelled i. The dashed arrows illustrate some possible accessibility between ontologies of different equivalence classes, which happens if we give up the symmetry.

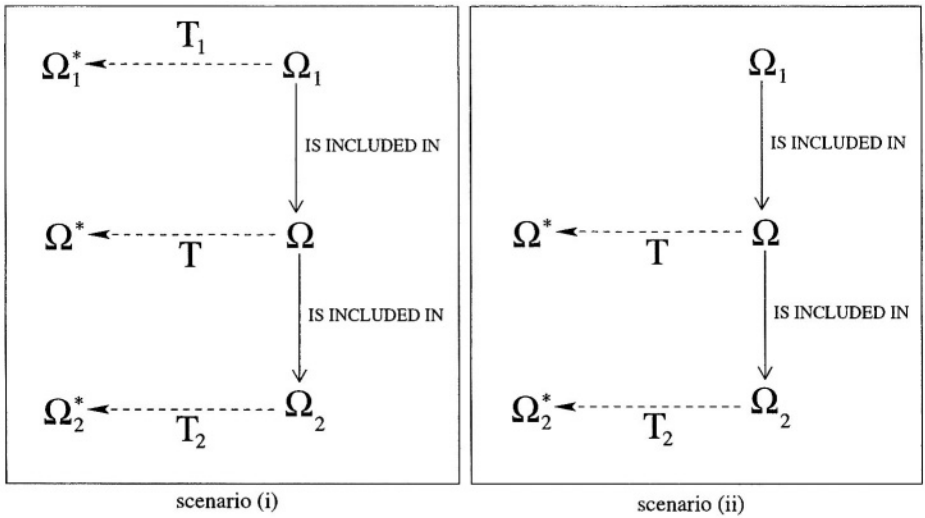


Fig. 3. Cascading changes

ontological constructs on the semantic level[6]. E.g., consider an arbitrary ontology holding the fact "Person has Name". An equivalent could be the union of the two facts: "Person has First-name" and "Person has Last-name".

Cascading Changes. If it is allowed that ontologies include (parts of) other ontologies, a new problem rises. Consider Fig. 3, Ω is defined by including another ontology Ω_1. Scenario (i) illustrates a transformation T_1 from Ω_1 to Ω_1^*, that initiates a cascade on all the ontologies that include Ω_1 such as Ω. T_1 thus implies a transformation T from Ω to Ω^*. Scenario (i) can lead to a chain reaction if Ω is also included in another ontology Ω_2, and Ω_2 is on its turn included in Ω_3, etc.

In Scenario (ii), transformation T from Ω to Ω^* will not cause a cascade on all the ontologies that are included in Ω, such as Ω_1.

Possibility and Necessity. Reconsider the evaluation function $\Phi(\phi, \Omega) \rightarrow \{T, F\}$[7], that was omitted earlier: a knowledge element ϕ is semantically entailed[8] by Ω if $\Phi(\phi, \Omega)$ has value T. We can now determine whether ϕ is neces-

[6] Comparing ontologies on the syntactical level is too naive, because more than one representation can exist for one particular ontology.

[7] Symbols T and F denote the boolean values *true* and *false* respectively.

[8] E.g., suppose the facts "Person is-referred-by Name" and "Employee is-a Person" are elements of Ω, then trivially they are semantically entailed; further due to the specialisation link between Employee and Person: Employee is-referred-by Name is semantically entailed by Ω.

sary or possible by considering the truth of ϕ in the ontologies that are accessible from the current ontology Ω_0 via a consistency-preserving transformation [21]:

- ϕ is *possible* in the current ontology Ω_0 if ϕ is true in *some* ontology Ω accessible from Ω_0 via a consistency-preserving transformation;
- ϕ is *necessary* in the current ontology Ω_0 if ϕ is true in *every* ontology Ω accessible from Ω_0 via a consistency-preserving transformation.

So when a knowledge element is possible in an ontology, then we can simply add it to the ontology without losing consistency.

3.2 Requirements for a Model-Independent Evolution Framework

As part of the project, an *ontology versioning system* will be implemented on top of our existing ontology server. On the client side, it will provide 2 components:

1. a graphical ontology version browser that is inspired on the lattices in Fig. 1–2. This *"lattice of ontologies" browser* provides basically an hyperbolic view with as first-class citizens all the ontologies and their inter-relationships stored in the server. Further, a zoom feature on all first class-citizens enabling:
   - zooming on an ontology label in the lattice returns the representation of the ontology in an appropriate representation;
   - zooming on an inter-relationship provides detailled information on the associated transformation and meta-information, such as the transformation type, etc.
2. an *editor* where the engineer can define transformations. Assuming the ontology is representable by a graph, he defines the mapping by selecting a subgraph as source of the revision, and defines a new subgraph as target of the revision. Further, he determines the type of the transformation. Research in graph morphisms can be useful here to ensure well-defined transformations. If the transformation is a revision he has to define a retraction followed by an expansion in order to express the change.
3. A *notification agent* that notifies the engineer when his change has cascading effects elsewhere in the lattice. The system should accommodate and control the chain reactions mentioned in previous subsection.

4 Validation

Once a prototype of the proposed framework is implemented, we will validate it by applying it to some case studies. We will validate each of its components (as specified in the requirements elicitation) separately in order to be able to finetune locally.

The **first case study** tests the *effectiveness* of the proposed framework to manage and perform sequential transformations on an emerging ontology versions set. The ideal case study would be a distributed ontology modelling environment where a large corpus of knowledge is conceptualised by a distributed team

of engineers. The **second case study** must verify the *preciseness* of the transformation editor. The **third case study** will test whether the lattice browser is *scalable* up to very large versions sets.

5 Discussion and Conclusion

In this paper we presented a model-independent framework for the management and revision of multiple ontology versions. In related work, basically a particular ontology representation model is chosen, and consequently, based on the evolvable knowledge elements in that model, the possible change operators that could engage an evolution process are determined .

In order to remain representation model independent, we have chosen to represent ontologies and evolution processes more abstractly. By adopting possible worlds semantics, a formal ontology can be seen as a logical theory with well-formed constraints or a possible world. A possible ontology Ω_2 is accessible from another possible ontology Ω_1 if there exists a transformation, that is a finite sequence of changes, from Ω_1 to Ω_2.

Further, inspired by belief revision, we have distinguished different types of transformations: revision, expansion, contraction and equivalence-preserving transformations. The possible ontology space and the different types of transformations are visualised by a Lindenbaum lattice.

Finally, we present requirements for our ontology evolution framework, where the ideas above are central: a Lindenbaum lattice inspired ontology versions browser and an editor where the knowledge engineer can define transformations of any type. We also suggest a methodology for validating our ideas in the implementation.

References

1. Banerjee, J. and Kim, W. (1987) Semantics and Implementation of Schema Evolution in Object-oriented Databases. *In ACM SIGMOD Conf., SIGMOD Record* 16(3):311–322.
2. Batini, C., Lenzerini, M. and Navathe, S. (1986) A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys,* 18(2):323–364, ACM Press.
3. Berners-Lee, T., Fischetti, M. and Dertouzos, M. (1999) Weaving the Web – The Original Design and Ultimate Destiny of the World Wide Web, Harper Collins.
4. Chiang, R., Eng Huang Chua, C. and Storey, V. (2001) A Smart Web Query Method for Semantic Retrieval of Web Data. *Data and Knowledge Engineering* 38:63–84.
5. De Leenheer, P. (2004) Reusing Certain Principles from Schema Evolution, Schema Transformation and Belief Revision for Ontology Evolution, manuscript.
6. Deray, T. and Verheyden, P. (2003) Towards a Semantic Integration of Medical Relational Databases by Using Ontologies: a Case Study. *In On the Move to Meaningful Internet Systems 2003 (OTM 2003) Workshops, LNCS 8889 (Catania, Italy),* pp. 137–150, Springer Verlag.

7. De Troyer, O. (1993) On Data Schema Transformation, PhD. thesis, University of Tilburg (KUB), Tilburg, The Netherlands.
8. Dhraief, H., Nejdl, W. and Wolpers, M. (2001) Open Learning Repositories and Metadata Modeling. *In Proc. of the 1st Int'l Semantic Web Working Symposium (SWWS01) (Stanford, CA),* pp.495–514.
9. Doan, A., Madhavan, J., Dhamankar, R., Domingos, P. and Halevy, A. (2003) Learning to Match Ontologies on the Semantic Web. *VLDB Journal* 12(4) :303–319, Springer Verlag.
10. Farquhar, A., Fikes, R. and Rice, J. (1997) The Ontolingua Server: a Tool for Collaborative Ontology Construction. *Int'l Journal of Human-computer Studies* 46(6):707-727.
11. Fensel, D. (2000) The Semantic Web and its Languages. *IEEE Computer Society* 15(6):67–73.
12. Foo, N. (1995) Ontology Revision, *in Proc. of 3rd Int'l Conf. Conceptual Structures (Berlin, Germany),* pp. 16–31, Springer-Verlag.
13. Gärdenfors, P. (ed.) (1992) Belief Revision, Cambridge Tracts in Theoretical Computer Science no. 29, Cambridge University Press.
14. Grove, A. (1988) Two modellings for theory change, *Journal of Philosophical Logic* 17, pp. 157–170.
15. Gruber, T.R. (1993) A translation approach to portable ontologies. *Knowledge Acquisition* 5(2):199-220.
16. Guarino, N. (1998) Formal Ontology and Information Systems. *In Proc. of the 1st Int'l Conf. on Formal Ontologies in Information Systems (FOIS98) (Trento, Italy),* pp. 3–15, IOS Press.
17. Halpin, T. (2001) Information Modeling and Relational Databases (From Conceptual Analysis to Logical Design), Morgan Kauffman.
18. Heflin, J. (2001) Towards the Semantic Web: Knowledge Representation in a Dynamic, Distributed Environment. PhD thesis, University of Maryland, Collega Park, MD, USA.
19. Kim, W. and Chou, H. (1988) Versions of Schema for Object-oriented Databases. *In Proc. of the 14th Int'l Conf. on Very Large Data Bases (VLDB88) (L.A., CA.),* pp. 148–159, Morgan Kaufmann.
20. Klein, M., Fensel, D., Kiryakov, A. and Ognyanov, D. (2002) Ontology Versioning and Change Detection on the Web. *In Proc. of the 13th European Conf. on Knowledge Engineering and Knowledge Management (EKAW02) (Siguenza, Spain),* pp. 197–212, Springer-Verlag.
21. Kripke, S. (1963) Semantic Analysis of Modal Logic I. *in Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* 9, pp. 67–96.
22. Lerner, B. (2000) A model for compound type changes encountered in schema evolution. *ACM Transactions on Database Systems (TODS),* 25(1):83–127, ACM Press, New York, NY, USA.
23. Meersman, R. (1999) The Use of Lexicons and Other Computer-linguistic Tools in Semantics, Design and Cooperation of Database Systems. *In Proc. of the Conf. on Cooperative Database Systems (CODAS99),* pp. 1–14, Springer Verlag.
24. Meersman, R. (2001) Ontologies and Databases: More than a Fleeting Resemblance. *In Proc. of the OES/SEO 2001 Workshop (Rome, Italy),* LUISS Publications.
25. Motik, B., Maedche A. and Volz, R. (2002) A Conceptual Modeling Approach for Semantics-driven Enterprise Applications. *In On the Move to Meaningful Internet Systems 2002 (OTM02) Workshops, LNCS 2519 (Irvine, CA, USA),* pp. 1082–1092, Springer Verlag.

26. Noy, N.F. and Klein, M. (2003) Ontology evolution: Not the same as schema evolution, *in Knowledge and Information Systems, 5. in press.*
27. Oliver, D., Shahar, Y., Musen, M. and Shortliffe, E. (1999) Representation of change in controlled medical terminologies. *In AI in Medicine,* 15(1):53–76.
28. Pinto, H., Gómez-Pérez, A. and Martins, J. (1999) Some Issues on Ontology Integration. *In Proc.of IJCAI99 Workshop on Ontologies and Problem Solving Methods: Lessons Learned and Future Trends,* pp.7.1–7.12.
29. Pottinger, R. and Bernstein, P. (2003) Merging Models Based on Given Correspondences. *In Proc. of 29th Int'l Conf. on Very Large Data Bases (VLDB03) (Berlin, Germany),* pp.826–873, Morgan Kaufmann.
30. Proper, H.A. and Halpin, T.A. (1998) Conceptual Schema Optimisation: Database Optimisation before sliding down the Waterfall. *Technical Report 341, Department of Computer Science, University of Queensland, Australia.*
31. Roddick, J. (1995) A Survey of Schema Versioning Issues for Database Systems, *in Information and Software Technology* 37(7):383–393.
32. Ryan, H., Spyns P., De Leenheer, P. and Leary R. (2003) Ontology-Based Platform for Trusted Regulatory Compliance Services. *In On the Move to Meaningful Internet Systems 2003 (OTM03) Workshops, LNCS 2889 (Catania, Italy),* pp. 675–689, Springer Verlag.
33. Sowa, J.F. (2000) Knowledge Representation - Logical, Philosophical and Computational Foundations, Ph.D. thesis, Vrije Universiteit Brussel (VUB), Brussel, Brooks/Cole Publishing Co., Pacific Grove, CA.
34. Spyns, P., Meersman, R., and Jarrar, M. (2002) Data Modelling versus Ontology Engineering, *in SIGMOD Record Special Issue on Semantic Web, Database Management and Information Systems 2000* 31(4):12–17.
35. Stojanovic, L., Maedche, A., Motik, B. and Stojanovic, N. (2002) User-driven Ontology Evolution Management. *In Proc. of the 13th European Conf. on Knowledge Engineering and Knowledge Management (EKAW02) (Siguenza, Spain),* pp. 285–300, Springer-Verlag.
36. Stojanovic, L., Stojanovic N., Gonzalez, J. and Studer, R. (2003) OntoManager – A Sytem for usage-based Ontology Management. *In On the Move to Meaningful Internet Systems 2003 (OTM03), LNCS 2888 (Catania, Italy),* pp. 858–875, Springer Verlag.
37. Ushold, M. and Gruninger, M. (1996) Ontologies: Principles, Methods and Applications. *The Knowledge Engineering Review* 11(2):93–136.
38. Verheijen, G. and Van Bekkum, J. (1982) NIAM, aN Information Analysis Method, *in Proc. of the IFIP TC-8 Conference on Comparative Review of Information System Methodologies (CRIS-1),* North-Holland.
39. Wiederhold, G. (1994) Interoperation, Mediation, and Ontologies. *In Proc. of the Int'l Symposium on 5th Generation Computer Systems (FGCS94), Workshop on Heterogeneous Cooperative Knowledge-Bases (Tokyo, Japan),* Vol.W3, pp.33–48, ICOT.

On the Verification of Web Services Compatibility: A Petri Net Approach

Manu De Backer

K.U.Leuven, Department of Applied Economic Sciences,
Naamsestraat 69, B-3000 Leuven, Belgium
Manu.DeBacker@econ.kuleuven.ac.be

Abstract. The evolution of the internet from an information provider to a service provider drives the need for more and better verification and validation mechanisms, especially in a Web services context, where services will be integrated and linked at runtime. The use of a formal semantics, like Petri Nets, enables us to create automatic verification procedures. In this paper, we discuss two types of conformance verification. First of all, we look at the conformance of a business process with the domain model of the information systems that are used to support business process activities. Secondly, we look at the similar problem of compatibility between two Web services. For these two problems algorithms will be created based on the Petri Net language theory.

1 Introduction

In recent years Business Modeling has gained momentum. In [16], two drivers for the development of Business Modeling have been identified. First of all, the formalization that needs to be taken when developing information systems for the support of business activities. Secondly, both external forces, in the form of customers and, competitors, as well as the internal requirements to improve efficiency in the organization drive the need for both more and more detailed business information. Any business has to improve its external effectiveness and internal efficiency continuously. The goal of business modeling is to gain substantial information about the business. A model is usually a simplified (graphic) description of some aspect of a business and it is therefore a vehicle for communication and mutual understanding of the business [16].

This research is inspired by the problems that arise when organizations change their business and subsequently need to change their information systems accordingly. We advocate that in some cases, but not always, the alteration of the information systems could be avoided. A survey by Andersson [2] showed results that one third of the companies questioned have changed their information systems. Not all these changes were caused by a change in the business, other causes were discussed too. This survey also showed that formal methods for business process development are used to a rather limited extent. One of the reasons for this limited use could be that working with a modeling method with a rich and expressive model syntax is of course advantageous in many situations - but

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 810–821, 2004.
© Springer-Verlag Berlin Heidelberg 2004

the parties involved need to be very well trained to use this method. In business modeling, the specialists involved are mostly specialists of the business who have no or very little training in modeling. Therefore, the tools and techniques used to model the business should be simple, self-evident and easy to learn. In the past, Entity Relationship modeling and its variants have been widely used for modeling business data models. For the modeling of business process aspects, Data-flow modeling, Petri Nets and Petri Net like techniques have proven to be very efficient. In the remainder of this paper we will concentrate on Petri Nets and Petri Net like techniques for modeling Business Process aspects.

One of the problems with information system requirements engineering is that of validation. Validation of design against requirements is fairly well understood by researchers and practitioners today, but validation of requirements against stakeholder needs is still a comparatively unknown area [18]. Nellborn uses the "Berlin Wall" as a metaphor to refer to the barrier that exists between the business developers and the information systems developers. Both have been working in the same world but with very different frames of reference and knowing very little of the life on the other side. Specific properties of both domains are the main drivers of their independent evolvement. For instance, on the business development side, flexibility is often a high priority whereas on the information system development side stability and persistence are key issues. The relation between the enterprise and its information system is complex and has many aspects. Because of the largely independent development of the two domains, not much research can be found on the application of business process modeling principles in Object-Oriented systems development. In [26], several advantages of the link between business process modeling and information systems design have been identified. This relation can help in designing more flexible and adaptable systems. In addition, when business processes are not modeled separately, they are often (hard-coded) baked in the procedural logic of the class-methods. Due to the flexible nature of business processes this often means that information systems need to be modified when business processes are redesigned. In [26], business events are proposed to serve as the link between the process aspects and the behavioral aspects of the business domain object types. Figure 1 shows the relation between the Business Process Modeling Concepts (BPM) and the Business Domain Modeling Concepts (DM), with the relation between the two represented as business events. A business process is modeled in terms of workflow activities or tasks. Indirectly, these tasks can be mapped on the business events, as the execution of a task triggers a set of business events. The implementation of business events in the information system depends on the paradigm. In the Object-Oriented paradigm for instance, the business events can be defined as one operation or a group of coordinated operations in the classes. The notion of event as a unit of action that can group more than one class operation is defined in several methodologies, e.g. Syntropy [10], MERODE [25], Remora [24] and Catalysis [11](where they are called "Actions").

The research that we propose will concentrate on the problems caused when business process concepts and domain modeling concepts are used in Object-

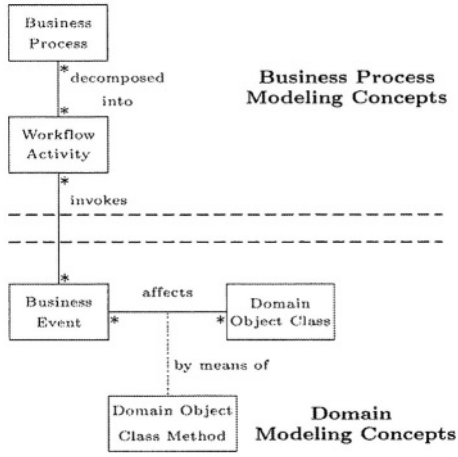


Fig. 1. Meta-model for conceptual modeling [26].

Oriented development. More specifically, we focus on the consistency checking of the constraints leveraged, by the business process model and the domain model, on the business events. These problems will be further refined and clarified in the rest of the paper. More importantly, this verification task proves to be a very challenging quest in the domain of Web services and Enterprise Application Integration (EAI) too. The ability to integrate and assemble individual Web services into standards-based business processes is an important element of the overall Web service technology stack. This can best be pointed out by the abundance of proposed standards in this research area (e.g. WSFL [15], XLANG [27], BPMN [9], BPML [9], BPEL4WS [3]).

The rest of the paper is structured as follows. In the next section, the specific research questions will be explained. The third section elaborates on the technologies used to answer the questions. The fourth section presents a case study about a complaint handling business process. The fifth section enumerates the contributions of this work.

2 Research Questions

This paragraph elaborates and describes the two research questions of this paper:

1. (RQ1) Verify the conformance of the business process model with the domain model.
2. (RQ2) Verify the semantic compatibility of two business processes.

As mentioned in the previous section, this research focuses on the consequences arising when both business developers and information systems developers model their view of the system independently. Independent development can easily lead to inconsistencies and hence a correspondence between the two models need to

be defined in order to keep them consistent. The use of business events can fulfill this purpose [25]: in the business process model activities will be mapped onto business events and in the domain model, business events will map onto class operations. Each model will impose in its own way sequence constraints on the execution order of the business events. It is just this, that is the source of the problems. Each model defines the sequence constraints slightly differently. In the domain layer, sequence constraints for the events are imposed by the definition of sequences of class operations. In UML for instance, besides the class diagrams specifying the static structure of the system, state charts can be used to describe the behavioral aspects of objects. These state charts describe sequence constraints imposed on the methods of the domain object classes. We assume that these operations can be directly mapped on the events, defining in this way sequence constraints on these events. On the other hand, the business process model will also impose sequence constraints on the business events, although in a more indirect way: the business process model imposes sequence constraints not on events but rather on tasks or activities. These tasks can then be further refined as business events. The problem we are investigating here, is the fact that when two models impose sequence constraints independently on the business events, we should be able to verify whether these constraints are consistent with each other. The first research question (RQ1) can be defined as follows: how can we analyze in an automatic way the consistency between the business process model and the domain model. Although the problem has been formulated as a general modeling problem, this research question is also particularly relevant in the context of Web services. Web services are often used as wrappers to make the functionality of existing information systems accessible through the Internet. In this context the first research question can be rephrased as the question whether the behavioral description of a Web service is compatible with the behavior of the underlying information systems.

The same line of thought brings us to our second research question about distributed business processes. A distributed business process models the behavior of several local business processes. In this work we assume that the local business processes are implemented as Web services. When several Web services work together a compatibility question is raised. Are the two (or more) Web services able to work together? This question is not only related to the fact whether two Web services *can* understand each other in terms of message syntax and form, but also to the question whether the sequence constraints of the Web services imposed on the business events can coincide. The first problem is the *syntactic* compatibility of Web services, whereas the second refers to an aspect of *semantic* compatibility. The syntactic compatibility question will not be treated in this research. The semantic compatibility problem that we consider can be reduced to the problem of comparing the sequence constraints of both processes. The second research question (RQ2) can be rephrased as follows: if two business processes want to work together, then they have to impose compatible sequence constraints on the set of business events. How can we verify the compatibility of the sequence constraints?

3 Methodology

Various Workflow Management Systems are being developed. Each of them using different technologies and different languages. Van der Aalst proposes in a series of articles the use of Petri Nets in workflow modeling [28,29,30,31,32,33,34] The decision to use Petri Nets in this research is mainly motivated by the existence of a formal semantics (and graphical counterpart) and the abundance of analysis techniques, which enabled the research community to define mathematical verifiable properties of Petri Nets. For more information we would like to refer to the elaborate Petri Net literature [22,23,17,1]. In Web service research, the upcoming Business Process Modeling Notation (BPMN)[9] is not a pure Petri Net notation, but more similar to the UML Activity Diagrams [14]. But as it can be formalized by means of Petri Nets, we will initially rephrase our research questions in terms of Petri Nets and analyzable properties of Petri Nets. In a later stage, the research results will be adapted to the specific characteristics of the BPMN notation. Comparison with UML Action Semantics [4,5,6,7,8] is also planned.

The quality of the business process is of great impact on the quality of service offered to the customers. The successful use of Petri Nets in the business process modeling domain has been mentioned in the previous section: Petri Net modeling allows for a variety of quality measurements for *individual* business processes, such as turnaround time, throughput, bottlenecks, deadlocks, livelocks, soundness [28]. The goal of both research questions, consistency checking between the BPM and the DM and consistency between two BPMs, offers the possibility to define augmented (verifiable) quality properties *across* business process and domain classes and *mutual consistency* between two or more business processes.

Transforming both research questions in terms of Petri Net theory requires some subtle adaptations of the problem. First of all, it is important to translate the business process model and the business domain model to a common representation, Petri Nets in our case. In the Web services world, business processes are modeled through the use of a Web service choreography language (e.g. BPEL4WS [3], BPML [9]), or a graphical representation of the process (e.g. BPMN [9]). The Business Process Modeling Notation (BPMN) provides businesses with the capability of defining and understanding their business procedures through a Business Process Diagram. The compatibility of BPMN with other BPM languages and the relationship with several standards organisations (e.g. OMG [20], WfMC [36], OASIS [19], W3C [35]) allows us to use BPMN as a platform independent notation for BPM. As a first step to answer our research questions we need to find a way to translate the BPM to a Petri Net representation. After this translation we have a Petri Net which models the equivalent behavior of the BPM. The Petri Net model of the BPM defines the control relationship between activities of the business process. As mentioned before, each task can in its turn lead to a triggering of events, which in its turn triggers class operations. It is therefore necessary, to map the tasks to their relevant set of events. To simplify the model we will, in the first stages of the research, assume that each task can be directly mapped on a business event.

In the Business Domain Model (DM) sequence constraints are modeled using State Charts. We make similar simplifying assumptions in the initial stages of our research. First, we will assume that a business event can be mapped to several class operations in different classes, but to at most one operation for a given class. Secondly, we assume that the modeling technique for class behaviour are pure Finite State Automaton (FSA) rather than the more complex Harel Statecharts that are used in UML. For the transformation of the FSA to a Petri Net the following result, which has been proven in [22] plays a crucial role, for every FSA an equivalent Petri Net can be generated. Therefore, we need to transform the FSA to its Petri Net equivalent, which is explained in [22].

3.1 RQ1 – Consistency Between DM and BPM.

For the first research question, the result of the transformations gives us two Petri Nets, which define the sequence constraint of the activities in the BPM on the one hand and on the operations of a Business Domain class on the other hand. Because of the assumptions we made, an activity in the BPM maps to a single event, which in turn maps to a single operation in a business domain class. We will additionally assume that activity, event and operation carry an identical name. In a later stage, these assumptions will be relaxed and a more complex mapping from activity to event and from event to operation will be introduced.

Checking the compatibility of the BPM with the DM, requires the comparison of the BPM sequence constraints against each DM class's sequence constraints. To compare these sequence constraints we will use the Petri Net language theory. Petri Net language theory has been developed in analogy with other formal language theory [22]. This paragraph gives a short introduction in formal language theory, with its application to the sequence constraints modeling on business events. We define an *alphabet* (Σ) as a finite set of symbols. This alphabet will refer to the set of business events a certain object participates in. A *string* is a sequence of finite length of symbols from the alphabet and defines a sequence of business events. If Σ is an alphabet, then Σ^* is the set of all possible strings, i.e. all possible sequences of events, over the alphabet Σ. A *language* is a set of strings over an alphabet, for instance defined by a FSA or Petri Net.

Every regular expression (RE), FSA or Petri Net defines a set of scenarios (sequence of business events), its language, over the alphabet (set of business events). If two Petri Nets share a number of events, they each define their own scenarios on this set of business events. It is clear that in order to combine both Petri Nets in a single system, this will only be possible if they are consistent with each other. Consequently, we need a way to verify whether the language defined by two Petri Nets is the same with respect to the common events. Actually, this problem can be refined. An important element is that the second Petri Net is equivalent to a FSA. First of all, we could test whether all the scenarios defined in the BPM are supported in the DM. This can be summarized as checking whether the language of the Petri Net is contained in the language of the FSA. In more requirements oriented terms, it means that we check whether the domain objects accepts the business processes. This problem has proven to be decidable. A

second type of check, would be to verify whether all scenarios defined in the DM are accepted by the BPM, or, in other words, whether there are some scenarios in the domain model that will be rejected by the business processes. In its general form, the problem of checking whether the language of a FSA is contained in the language of a Petri Net has been proven to be undecidable for L-type Petri Nets. However, the class of Petri Nets that we consider is much more restricted than the general class of L-type Petri Nets: Business Process Modeling quality usually implies some restrictions on the well-formedness of Petri Nets such as S-coverability. In addition, BPMN also defines a more restricted category of Petri Nets. Notice that of the two tests, it is the first (decidable) one that is the most interesting test: it is important to ensure that each scenario defined in the business process layer will be supported by the business domain layer. The question whether some scenarios of the business domain remain unused because they are refused by the business processes is relevant, but does not yield an immediate practical problem.

The goal of this research is therefore to create an algorithm that takes as inputs a BPM and a FSA, and decides whether the BPM and the DM are compatible. The algorithm will check whether the BPM is included in the FSA and the research will investigate whether it is possible to define an algorithm that checks whether the FSA is also included in the BPM. If possible the algorithm should be able to identify the events on which the problem occurs.

3.2 RQ2 – Compatibility of two Business Processes

The second research question is situated in the field of Web service composition and dynamic linking and invocation of Web services. More specifically, the question is closely related to problems that occur with distributed business processes. A distributed business process models the behavior of several (local) business processes working together. Each of these local business processes is implemented as a Web service. Again, we are aware of the difficulty of creating good (distributed) business processes, but we will not elaborate on that. Comparing business processes becomes a relevant topic, when they are composed in one new (global) business process. The composition of Web services will only be meaningful, if the execution of the distributed business process succeeds without deadlocks. Each local business process can be deadlock-free but with the composition of the processes a deadlock can occur. Therefore, it could be interesting to know beforehand whether two Web services are compatible.

As mentioned before, the compatibility of business processes can be verified from two point of views. First of all, the syntactical compatibility refers to the conformance of signatures between the two Web services. For instance, if one Web service invokes an operation of the second Web service it is necessary that the parameters, the number of parameters and the type of parameters match. In this research, this type of compatibility will not be dealt with. The second compatibility aspect deals with the semantic compatibility of two business processes. How can we verify that the composition of both business processes will be deadlock-free? Or more specifically, how can we verify that the sequence

constraints imposed by both business processes are compatible? There is an extensive research community dealing with the problems of comparing Petri Nets and Petri Net languages. Although, deciding language equivalence of Petri Nets has proven to be undecidable [13,12] for classical Petri Nets, we believe that certain restrictions on the construction of Petri Nets enables us to verify semantic compatibility. For instance, language equivalence for deterministic Petri Nets is decidable [21].

4 Case Study

In this section, an example is given to illustrate the first research question (RQ1). The example used, deals with the handling of complaints in a company. In the business domain model, the concepts of customer, complaint and product are defined as well as their relationships. A *customer* can have several *complaints* but each complaint is about one *product,* see Figure 2(a). Together with the object classes a set of events is defined, Figure 1. These "real-world" events also act as intermediaries between the DM and BPM, as shown later. Every object class participates in certain events. We assume that each event in which an object participates is implemented as an operation in the specific object. More importantly, events are not allowed to occur in random order during the life cycle of an object. In order to specify these constraints an object is equipped with a description of the allowed sequences of events, which can be modeled as a finite state automaton (FSA), see Figure 2(b). For instance, it is necessary that a complaint is first investigated before a refund is done. Every object class in the domain model will have an object life cycle, and therefore reduces the allowed scenarios.

As will be demonstrated later on, the problem is that in the modeling of the behavior of the object class, certain events and restrictions can be added which will contradict with the business process model.

The second phase, consists of creating the business process model. Let us assume that the initial business process was the following: *When a client has a complaint about a certain product, he creates a complaint and sends it to the company. The company accepts the complaint and prepares it for investigation. After this investigation, the complaint can be rejected or there can be a refund or the complaint is accepted but no refund is necessary. Finally the complaint is archived.* Due to the high number of complaints the company decided to transform the business process as follows. The company was able to deduct a number of rules out of the high number of complaints. The rules enabled the company to apply an automatic first filter to the complaints, deciding between acceptance and rejection of the complaint. The new business process can be defined as follows. *Depending on the value of the complaint, two different paths can be taken. First of all, if the value is low then the complaints are accepted or rejected automatically (using the rules). If the complaint is accepted it will be investigated, and if necessary a refund is paid. If the value is high, the complaint is directly investigated and a decision on its outcome is made. Reject, refund*

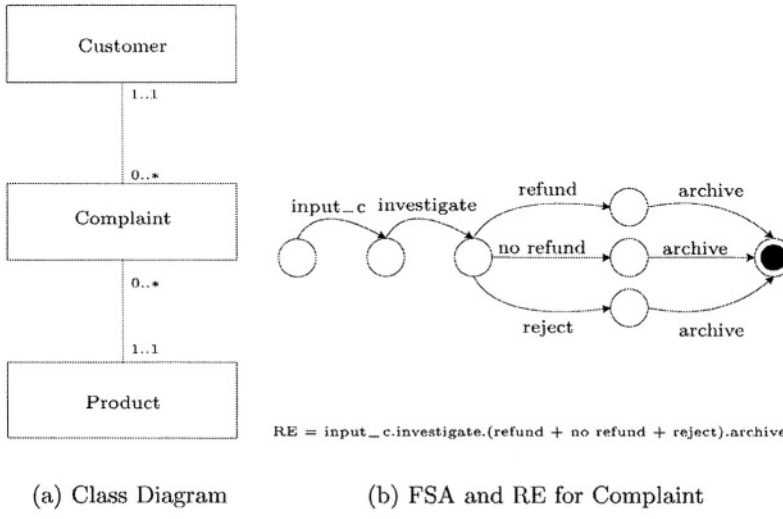


Fig. 2. Partial Business Domain Model.

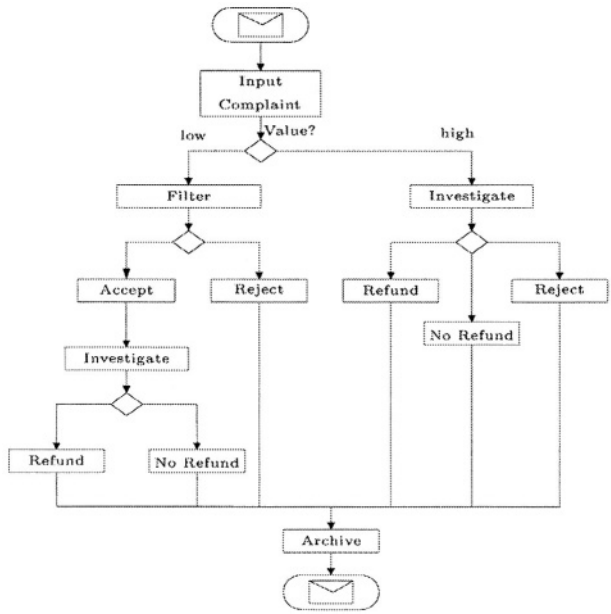


Fig. 3. Business Process Model for complaint handling.

or no-refund are the options. Both business processes are modeled in BPMN in Figure 3.

Both the DM and the BPM put sequence constraints on the business events. In this example it is clear that the DM and BPM are not totally compatible, the sequence constraint on the events *investigate* and *reject* are too strict defined in the DM. In the second business process a complaint can be rejected before any investigation has taken place. Consequently, the business process is not entirely supported in the domain model. At this point we could wonder whether it is a good idea to model a constraint on investigate and reject in the DM at all, but that is another question. The goal of this research is to develop a method to (automatically) verify the consistency of the sequence constraints and if possible direct the system developers to the sequence constraints that are not compatible. Transforming the DM and BPM to their equivalent Petri Nets enables us to verify the sequence constraints. This means that using the Petri Net theory an automatic consistency checking algorithm can be created.

5 Contributions and Conclusions

The contributions of this research are discussed in this paragraph. This research has identified two problems in the business modeling domain, the consistency checking between the DM and BPM and the compatibility issues between two business processes. Therefore, the contributions of this thesis can be classified as to pertaining to one of these two problem areas. The contribution in these domains are threefold. First of all, the problems are translated to a theory with formal semantics, viz. Petri Net theory. This enables us to define verifiable properties on the business models. Secondly, automatic consistency checking algorithms are developed for these problems. The third contribution, which is more indirect, is that, through the use of these results, business process development tools (e.g. Biztalk, Collaxa, Openstorm) can be made more intelligent. The use of these smarter tools will enable the business developer to create his business processes with a higher degree of correctness. The applicability of the results will be proven by a prototype implementation of the algorithms.

Acknowledgment. I would like to thank prof. dr. Monique Snoeck for the comments and discussions on this text. I would also like to thank prof. dr. Guido Dedene and prof. dr. Jacques Vandenbulcke for the overall support. This research is partially sponsored by Microsoft Belgium and Luxemburg.

References

1. The petri nets bibliography. http://www.informatik.uni-hamburg.de/TGI/pnbib/.
2. J. Andersson. Business process development and information technology in small and medium-sized companies. *In Perspectives on Business Modelling,* pages 117–130. Springer-Verlag, 1999.

3. T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic and S. Weerawarana. Business process execution language for web services. http://www-106.ibm.com/developerworks/library/ws-bpel/, 2003.
4. C. Bock. Uml 2 activity and action models. *Journal of Object Technology,* 2(4):43–53, 2003.
5. C. Bock. Uml 2 activity and action models part 2: Actions. *Journal of Object Technology,* 2(5):41–56, 2003.
6. C. Bock. Uml 2 activity and action models part 3: Control nodes. *Journal of Object Technology,* 2(6):7–23, 2003.
7. C. Bock. Uml 2 activity and action models part 4: Object nodes. *Journal of Object Technology,* 3(1):27–41, 2004.
8. C. Bock. Uml 2 activity and action models part 5: Partitions. *Journal of Object Technology,* 3(7):37–56, 2004.
9. BPMI.org. Business process management initiative, http://www.bpmi.org/.
10. S. Cook and J. Daniels. *Designing Object Systems: Object-Oriented Modelling with Syntropy.* Prentice Hall, 1994.
11. D. D'Souza and A.C. Wils. *Components, and Frameworks with UML, the Catalysis approach.* Addison Wesley, Reading MA, 1999.
12. J. Esparza. Decidability and complexity of petri net problems - an introduction. *Lecture Notes in Computer Science: Lectures on Petri Nets I: Basic Models,* 1491:374–428, 1998.
13. J. Esparza and M. Nielsen. Decidability issues for petri nets. *Petri Net Newsletter,* (47):5–23, 1994.
14. Object Management Group. Unified modeling language, http://www.uml.org/.
15. F. Leymann. Web services flow language. http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.pdf, 2001.
16. C. G. Lindström. Lessons learned from applying business modelling: Exploring opportunities and avoiding pitfalls. *In Perspectives on Business Modelling,* pages 151–164. Springer-Verlag, 1999.
17. K. H. Mortensen and H. Rölke. Petri nets world. http://www.daimi.au.dk/PetriNets/.
18. C. Nellborn. Business and systems development: Opportunities for an integrated way-of-working. *In Perspectives on Business Modelling,* pages 195–213. Springer-Verlag, 1999.
19. OASIS. Organization for the advancement of structured information standards. http://www.oasis-open.org.
20. OMG. Object management group. http://www.omg.org/.
21. E. Pelz. Closure properties of deterministic petri nets. *Lecture Notes in Computer Science: 4th Annual Symposion on Theoretical Aspects of Computer Science, Passau,* 247:371–382, 1987. NewsletterInfo: 25,26.
22. J. L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, 1981.
23. W. Reisig. *Petri Nets - An introduction.* Springer-Verlag, 1982.
24. C. Rolland and C. Richard. *The REMORA Methodology for Information System Design and Management.,* pages 369–426. Information Systems Design Methodologies: A Comparative Review, 1982.
25. M. Snoeck, G. Dedene, M. Verhelst and A. M. Depuydt. *Object-oriented Enterprise Modeling with MERODE.* Leuven University Press, 1999.

26. M. Snoeck, S. Poelmans and G. Dedene. A layered software specification architecture. In F. Laendler, S.W. Liddle and V.C. Storey, editors, *Conceptual Modeling,* volume 1920 of *Lecture Notes In Computer Science,* pages 454–469. Springer, Oktober 2000.
27. S. Thatte. Xlang. http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm, 2001.
28. W. M. P. van der Aalst. The application of petri nets to workflow management. *Journal of Circuits, Systems and Computers,* 8(1):21–66, 1998.
29. W. M. P. van der Aalst. Inheritance of interorganizational workflows to enable business-to-business E-commerce. *Electronic Commerce Research,* 2(3): 195–231, 2002.
30. W. M. P. van der Aalst. Making work flow: On the application of petri nets to business process management. *In Lecture Notes in Computer Science: 23rd International Conference on Applications and Theory of Petri Nets, Adelaide, Australia, June 24-30, 2002 / J. Esparza, C. Lakos (Eds.),* volume 2360, pages 1–22. Springer Verlag, June 2002.
31. W. M. P. van der Aalst. Making work flow: On the application of petri nets to business process management. *In Lecture Notes in Computer Science: 23rd International Conference on Applications and Theory of Petri Nets, Adelaide, Australia, June 24-30, 2002 / J. Esparza, C. Lakos (Eds.),* volume 2360, pages 1–22. Springer Verlag, June 2002.
32. W. M. P. van der Aalst and A. H. M. ter Hofstede. Workflow patterns: On the expressive power of (petri-net-based) workflow languages. *In Proc. of the Fourth International Workshop on Practical Use of Coloured Petri Nets and the CPN Tools, Aarhus, Denmark, August 28-30, 2002 / Kurt Jensen (Ed.),* pages 1–20. Technical Report DAIMI PB-560, August 2002.
33. W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski and A. P. Barros. Workflow patterns. *QUT Technical report, FIT-TR-2002-02,* 2002. (Also see http://www.tm.tue.nl/it/research/patterns.).
34. W. M. P. van der Aalst, K. M. van Hee and G. J. Houben. Modelling workflow management systems with high-level petri nets. In De Michelis, G., Ellis, C. and Memmi, G., editors, *Proceedings of the second Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms,* pages 31–50, 1994.
35. W3C. The world wide web consortium. http://www.w3.org/.
36. WfMC. The workflow management coalition. http://www.wfmc.org/.

Usability of Visual Information Retrieval Metaphors for Object-Oriented Databases

Mariusz Trzaska[2] and Kazimierz Subieta[1,2]

[1]Institute of Computer Science, Ordona 21, Warsaw, Poland
[2]Polish-Japanese Institute of IT, Koszykowa 86, Warsaw, Poland

Abstract. We present visual metaphors of Mavigator, a visual information retrieval system for Web-oriented database applications. The metaphors allow the naive users (computer non-professionals) to retrieve information from object-oriented databases in an easy and intuitive form. Novel features of Mavigator include coherent combination of a few paradigms: intentional navigation, extensional navigation, baskets for recording retrieval results and active extensions. The latter are programmed in a programming language thus allow a programmer to extend functionalities available to end users with no limitations. Due to the flexible architecture Mavigator is able to work not only with a proprietary object-oriented database but also with any information source. The objective of the corresponding PhD thesis is investigation into usability of different visual retrieval metaphors in realistic Web applications. In this paper we present some of the metaphors and discuss their potential for users.

1 Introduction

Databases, among other features, should provide an easy access to information. However, the "easiness" is relative to a user kind and his/her experience in computer technologies. Nowadays more and more non-professionals use computers, especially various applications based on Web browser-oriented interfaces. Their requirements with respect to user-friendliness of the entire computer systems are much stronger than requirements of computer professionals. Non-professionals usually do not accept formalized syntax, strict sophisticated rules of user action, long and specialized training or following professional manuals. As a consequence, computer tools, including interfaces for information retrieval, must evolve into non-sophisticated and easy-to-use applications.

On the other hand, the interface should be simple, but not simpler than the inherent complexity of the task that the user has to solve. For some tasks the typical full-text retrieval (as e.g. in Google) is not sufficiently precise because of lack of facilities to specify semantic meaning of data items stored on Web. The XML technologies (including RDF, OWL, Semantic Web and other proposals) aim at putting data into nested labelled structures with well-defined semantic meaning. There are also query languages such as XQuery to express the user needs much more precisely than it is possible in full-text retrieval engines.

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 822–833, 2004.
© Springer-Verlag Berlin Heidelberg 2004

In the database domain there are many query languages proposed, in particular, SQL as a major language for accessing relational databases. SQL, like all textual query languages, is very powerful, but its audience, due to its complexity, is rather limited to computer professionals. The same concerns more recent textual languages for object-oriented databases, such as OQL [1] or SBQL [2].

Attempts to define more user-friendly interfaces have been noticed for many years. One of the most successful was QBE [3] based on a tabular view of relational tables and specific retrieval conditions inserted by the user into the tables. QBE was perhaps the first graphical query interface based on visualization of a database schema and specific visual manipulations of the user on the graphical interface. Due to a better hardware and popularity of easy application programming interfaces for graphical manipulations we observe recently extensive development of visual metaphors for information retrieval. Some of them are counterparts to their textual predecessors. Other, like Value Bars [4] and the interface presented in [5] are based on specific graphical ideas.

As noted in [6], the key issue behind such proposals is *usability* in real applications prepared for users who are not computer professionals. Unfortunately, usability cannot be predicted in advance during design and implementation of an interface, because it depends on many factors such as the readiness of the user to get some training, inherent complexity of the task that he/she has to accomplish and adequacy of the interface to this task, supporting various forms of user awareness during long sessions, and others. Therefore the only way to check the usability is to implement a particular metaphor and then, to measure some user-oriented factors (such as the number of errors, the entire time to reach the goal, etc.) in real applications.

In this paper we present the visual querying interface Mavigator which has been developed to allow non-computer professionals to work with object-oriented database systems. Its key concepts include: intentional navigation, extensional navigation, persistent baskets for recording temporary and final results of querying, and active extensions. The first kind of navigation (intentional) is based on navigation in a database schema graph. The user moves from vertex to vertex via edges, where vertices denote classes (more precisely, collections of objects) and edges denote associations among classes (in UML terms). Additional functionalities allow the user to build quite complex queries. The second kind of navigation (extensional) is similar but the user navigates in a graph of objects rather than in a graph of classes. This mode enables the user to move (in mind) from an object to an object (vertices of the graph) via a specific link (an edge of the graph).

After the user has retrieved some result he/she may require to present it visually in some friendly way, e.g. as a tabular report, as a chart, as a distribution map, etc. This is a critical issue for visual querying interfaces, as the trade-off between simplicity of the end user interface and complexity of a possible visualization form. The number of options and the general complexity of the interface that may be required to visualize a querying result seem to be unacceptable for naive users. Therefore for achieving the required goal we assume some contribution of computer professionals. The last feature, called active extensions, allows a computer professional to add functionalities required by a particular naive user. We assume that the functionalities will be coded

in a programming language (currently C#) thus the range of the functionalities is unlimited.

The remainder of this paper is organized as follows. In Section 2 we discuss related work on visual information retrieval facilities and explain how Mavigator differs from the existing solutions. In Section 3 we give a detailed description of the Mavigator metaphors. Section 4 describes two prototypes: SKGN [7, 8] implemented for the European project ICONS, and Mavigator extending and improving SKGN. Mavigator is currently under construction. Section 5 concerns implementation and architecture of prototypes. Section 6 briefly discusses proposed evaluation procedures and Section 7 concludes.

2 Related Work

Due to the limited space this paper gives only an outlined overview of the related work. The final version of the PhD thesis will contain more detailed description.

Roughly speaking, visual metaphors to information retrieval can be subdivided into two groups: based on graphical query languages and graphical browsing interfaces. This subdivision is not fully precise because many systems have features from both groups. An example is Pesto [9] having possibilities to browse through objects from a database. Otherwise to Mavigator, the browsing is performed from one object to a next one object. For instance, the user can display a Student class object, but to see another student, he/she needs to click next (or previous) button and replace current visualization. Besides browsing, Pesto supports quite powerful query capabilities. It utilizes the query-in-place feature, which enables the user to access nested objects, e.g. courses of particular students, but still in the one-by-one mode. Another advantage concerns complex queries with the use of existential and universal quantification. Such complex features may however compromise usability for some kinds of users and kinds of retrieval tasks.

In our opinion an essential issue behind such interfaces is how the user uses and accumulates information during querying. For instance, the user may see all the attributes even those, which are not required for the current task. Otherwise, the user can hide non-interesting attributes, but this requires from him/her some extra action. Therefore, from the user point of view, there is some trade-off between actions that have to prepare the information necessary for querying and actions of further querying. To accomplish complex queries without putting them explicitly, the system should support any sequences of both types of actions.

Typical visual querying system are Kaleidoscape [10], based on its language Kaleidoquery [11], and VOODOO [12]. Both are declared to be visual counterparts of ODMG OQL [1] thus graphical queries are first translated to their textual counterpart and then processed by an already implemented query engine. The first one uses an interesting approach to deal with AND/OR predicates (another proposal can be found in [13]). It is based on a flow model described previously by Schneiderman [14]. We find it very useful and intuitive thus we have adopted it to our metaphor.

Polaris [15], designed for relational databases, has some querying capabilities, but it seems that the major emphasis has been put on data visualization.

A typical example of a browsing system is GOOVI [16] developed by Cassel and Risch. Unfortunately, selecting of objects is done via a textual query editor. A strong point of the system is the ability to work with heterogeneous data sources.

Another interesting browser is [17], which is dedicated to Criminal Intelligence Analysis. It is based on an object graph and provides facilities to make various analyses. Some of them are: retrieving all objects connected directly/indirectly to specified objects (i. e. all people, who are connected to a suspected man), finding similar objects, etc. Querying capabilities include filtering based on attributes and filter patterns. The latter allow filtering links in a valid path by their name, associated type, direction or a combination of these methods. The manner of work is similar to the metaphor that we have called extensional navigation.

Browsing systems relay on manual navigation from one object to a next one. During browsing the user can read the content of selected objects. Browsing should be an obligatory option in situations when the user cannot define formally and precisely the criteria concerning the search goal.

3 Mavigator Metaphors

Mavigator is made up of four metaphors: intentional navigation, extensional navigation, persistent baskets, and active extensions. The subdivision of graphical querying to "intentional" and "extensional" can be found in [18] and [19]. We have adopted these terms for the paradigm based on navigation in a graph. The user can combine these metaphors in an arbitrary way to accomplish a specific task.

Intentional and extensional navigation are based on navigation in a graph according to semantic associations among objects. Because a schema graph (usually dozens of nodes) is much smaller than a corresponding object graph (possibly millions of nodes), we anticipate that intentional navigation will be used as a basic retrieval method, while extensional navigation will be auxiliary and used primarily to refine the results. Next subsections contain description of the methods.

3.1 Intentional Navigation

Intentional navigation utilizes a database schema graph. Fig. 1 shows a part of such a graph for the Structural Fund Knowledge Portal implemented within the European project ICONS. A graph consists of the following primitives:

- Vertices, which represent classes or collections of objects. With each of them we associate two numbers: the number of objects that are marked by the user (see further) and the number of all objects in the class,
- Edges, which represent semantic associations among objects (in UML terms),
- Labels with names of association roles. They are understood as pointers from objects to objects (like in the ODMG standard, C++ binding).

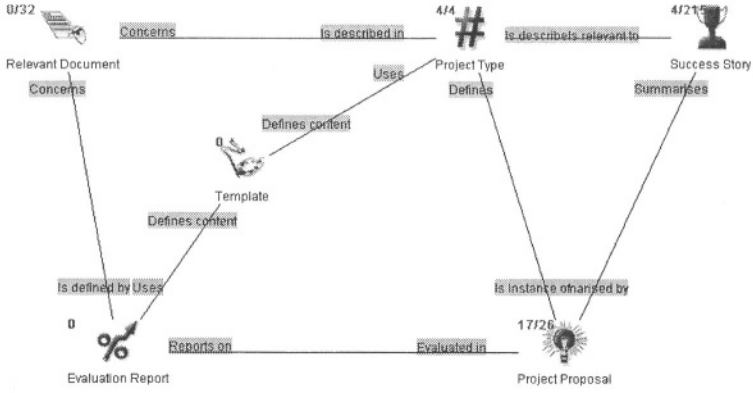


Fig. 1. Intentional navigation graph

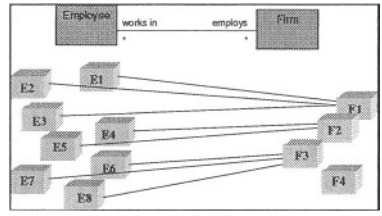


Fig. 2. Intentional navigation

The user can navigate through vertices via edges. Objects which are relevant for the user (candidates to be within the search result) can be marked, i.e. added to the group of marked objects. There are a number of actions, which cause objects to be marked:

- Filtering through a predicate based on objects' attributes. The action cause marking those objects for which the corresponding predicate is true. There are two options: objects to mark are taken from a set of already marked objects or from entire extension of the class. Filtering objects through a predicate is analogous to the SQL *select* clause.
- Manual selection. Using values of special attributes from objects (identifying objects by comprehensive phrases) it is possible to mark particular objects manually. It is especially useful when the number of objects is not too large and there are no common properties among them.
- Navigation (Fig. 2) from marked objects of one class, through a selected association role, to objects of another class. An object from a target class became marked if there is an association link to the object from a marked object in the source class. Fig. 2 explains the idea. Let's assume that the *Firm* set of marked objects has four marked objects: *F1,...,F4.* Than, navigating from *Firm* via *employs* cause marking eight objects: *E1,...,E8.* This activity is similar to using path expressions in query languages. A new set of marked objects (the re-

sult of navigation) replaces existing one. It is also possible to perform a union or intersection of new marked objects with the old ones. Notice that the later options allow one to perform a query like: get all employees working in the "Main" department **and** earning more than 5000. The options are easy to understand (even for naive users) and greatly enhance the retrieval capabilities.

- Basket activities. Dragging and dropping the content of a basket on a class icon causes some operation on the marked objects of the class. A new set of marked objects taken from the basket can replace the existing set of marked objects, can be summed with it, can be intersected with it, or can be subtracted from it (equivalents to OR, AND and NOT).
- Active extensions. In principle, this capability is introduced to process marked objects rather than to mark objects. However, because all the information on marked objects is accessible from a C# program the capability can also be used to mark objects.

Intentional navigation and its features allow the user to receive (in many steps but in a simple way) the same effects as through complex, nested queries. Integrating these methods with an extensional navigation, manual selection and other options supports the user even with the power not available in typical query languages.

An open issue concerns functionalities that are available in typical query languages, such as queries involving joins and aggregate functions. There is no technical problem to introduce them to Mavigator (except some extra implementation effort) but we want to avoid situation when excess options will cause our interface to be too complex for the users. We hope that during evaluation we will find answers on such questions.

3.2 Extensional Navigation

Extensional navigation takes place inside extensions of classes. Graph's vertices represent objects, and graph's edges represent links. When the user double clicks on a vertex, an appropriate neighbourhood (objects and links) is downloaded from the database, which means "growing" of the graph.

Extensional navigation is useful when there are no common rules (or they are hard to define) among required objects. In such a situation the user can start navigation from any related object, and then follow the links. It is possible to use basket for storing temporary objects or to use them as starting points for the navigation.

3.3 Baskets

Baskets are persistent storages of search results. They store two kinds of entities: unique object identifiers (OIDs) and sub-baskets (Fig. 3). The hierarchy of baskets is especially useful for information categorization and keeping order. Each basket has its name that is typed in by the user during its creation. The user is also not aware



Fig. 3. Visualization of the basket containing two sub-baskets and three objects

OIDs, because special objects labels are used. During both kinds of navigation it is possible to drag an object (or a set of marked objects) and to drop them onto a basket. The main basket (holding all the OIDs and sub-baskets) is assigned to a particular user. At the end of a user session all baskets are stored in the database.

Below we list all basket activities:

- Create a new basket.
- Change basket properties (name, description).
- Remove selected items (sub-baskets or objects).
- Perform operations on two baskets: sum of baskets, intersection of baskets, and set-theoretic difference of baskets. The operation result can be stored in one of the participating baskets or in a new one.
- Drag an object and drop it onto an extensional navigation frame. As the result, the neighbourhood (other objects and links) of a dropped object will be downloaded from the repository.
- Drag a basket and drop it onto class's visualization in the intentional navigation window. As the result a new set of marked object can be created (replace, add, intersect, subtract). Only objects of that class are considered.

Baskets allow storing selected objects in a very intuitive and structured way. Navigation could be stopped at any time and temporary results (currently selected/marked objects) can be named, stored and accessed at any time.

3.4 Active Extensions

Active extensions provide a way to add new functionalities operating on marked objects or objects recorded in baskets. In order to achieve the maximal power and flexibility active extensions are based on a fully-fledged programming language. Such a solution requires collaboration of an end user with a programmer who will write the code accomplishing the required functionalities.

The prototype, currently under construction, will be based on Microsoft C# as a language for active extensions. A programmer will be aware of the Mavigator metadata environment, which will allow him/her to write a source code of the required functionality in C#. Then the end user will be supported with one click button causing execution of the written code. The code will process the visually selected set of

marked objects or objects recorded in a user basket. The functionality of such programs is unlimited. Next two sub-sections present its particular applications.

3.4.1 Active Projections

Active projections allow visualizing a set of marked objects where position (in terms of x, y coordinates) of each of them will be based on value of particular objects' attributes. By using two or three axes it is possible to visualize dependencies of two or three attributes. The number of attributes could be increased due to other visual features. In particular [15, 20, 21] propose to use the features of visual icons such as shape, size, orientation, colour, texture and correlate them with values of additional attributes. For instance, the size of an employee icon could be proportional to his/her salary. Such an approach makes it possible to identify some groups of marked objects. For instance it will be easy to see the group of employees who are paid less than $2000 and have some specified experiences.

Besides the visual analysis of objects dependencies it is also possible to utilize projections in more active fashion. Object taken from a basket could be dropped on projection's surface, which cause right (based on attributes values) placement. It is also possible to perform reverse action: drag object from surface onto the basket (which cause recording object in a basket). We also consider some kind of extensional navigation on projection's surface: clicking on object's visualization causes visualizing its neighbourhood. To avoid misunderstanding, objects from the neighbourhood should be visually distinguished (e.g. because they could have types different from the type of visualized objects).

3.4.2 Objects Exporters

Objects exporters allow cooperating with other software systems. By having a selected set of marked objects, it is possible to send it to other programs, such as Excel, Crystal Reports, etc. That approach makes it possible a subsequent processing of Mavigator's results of querying/browsing.

4 User Interface for Mavigator

There are two implemented prototypes, which follow the Mavigator metaphors. First one, called Structural Knowledge Graph Navigator (SKGN), have been developed during the European project ICONS (Fig. 4). It utilizes almost all (except Active projections) concepts described in the previous sections.

SKGN has a lot of features, which make them easy to use by casual users. For instance, during intentional navigation all activities involving marked objects could be undone or redone. This is similar to browser back/forward buttons. Each step is stored and shown in the list (see left side of Fig. 4). There is also the possibility to go back to any previous step from the recorded history of navigation.

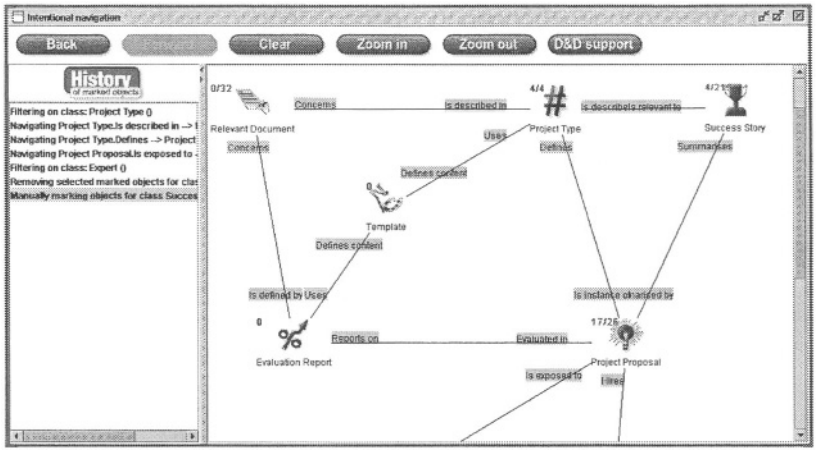


Fig. 4. SKGN window during intentional navigation

An extensional navigation window allows to utilize facilities such as zoom in/out and scroll bars. All of these functionalities improve navigation and legibility of the graph. Because of legibility, permanent showing of link labels is avoided (however it is possible to turn it on in the user's options). Instead, ontology, auto ontology and specialized tool tips features are introduced. The ontology functionality means that when an object is selected, its class is selected too (on a classes graph – the intentional navigation window). Auto ontology is similar to ontology with one difference – selecting an object's class proceeds when object's tool tip is shown (without selecting the object). A similar functionality has been implemented for links.

In Mavigator we plan to implement all functionalities described above and many others. Some of them are listed below:

- Create a new user interface based on filter flow model [14] for visual definition of predicates.
- Additional activities for marked objects, which could be implemented using Active Extensions (see chapter 3.4). New functions, such as average, maximum, minimum, sum, etc., allow one to perform other kinds of queries.
- Stickers. A user may want to annotate particular object(s). To do this we would like to introduce little stickers, which can hold any kind of information, which are important to users. Stickers will be persistent and assigned to a particular user.
- Extending the utilized data model by generalization. We realize, however, that excessive complexity of a data model could have impact on the easy-of-use and in consequence, on usability.
- Visual joins. They should allow to group objects from two or more classes, for instance, an employee object with a company object. A group of objects would be treated as an ordinary object, with possibilities of filtering, navigating, marking, storing in a basket, etc.

Two later ideas will be carefully evaluated whether they could be acceptable and useful for a sufficiently wide range of users.

5 Software Architecture and Implementation

The Structural Knowledge Graph Navigator has been implemented as a Java applet. Due to the flexible architecture, it is possible to connect it to any data source (through a proprietary wrapper).

The Mavigator prototype, currently under construction, is implemented as a Windows Form Application in C# language. Its architecture (Fig. 5) consists of the following elements:

- GUI – contains implementation of the user interface,
- Business logic – includes implementation of the Mavigator metaphors and some additional routines,
- Database wrapper – ensures communication, via defined AbstractDatabase2 interface, with any database. Currently we are working with Matisse [22] post-relational database. In future we plan to develop wrappers to other object-oriented database management systems, including our own prototype ODRA (Object Database for Rapid Application development).

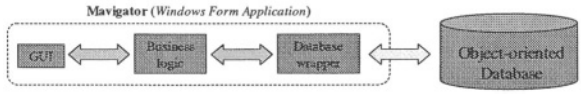


Fig. 5. Architecture of the Mavigator prototype

The Mavigator prototype will utilize active extensions written in Microsoft C#. The functionality requires compiling and running source code (which will start particular extension) during execution (runtime) of the Mavigator. We have already recognized how to implement such a functionality in C#. The solution needs to define interface having methods to be invoked during user's activities and to determine a way to compile the user's code. Both problems are currently solved and we start to implement corresponding functionalities.

6 Evaluation

The SKGN prototype has been used for the Structural Fund Projects Portal thus we have some informal response from the users, generally very positive. SKGN, as a part of European project ICONS, has also been positively assessed by the Project Officers.

Mavigator will be tested according to prepared formal scenarios from the point of view of usability, efficiency, easy-of-use and expressive power. Two (or more) groups of users will get typical queries; sample queries can be found e.g. in [10, 23]. One of the groups will use the Mavigator, the rest other methods (such a textual query languages or other visual tools). The results will be judged from various points of view. More information about evaluating visual tools and graphical user interfaces can be found in [24, 25, 26].

7 Conclusion

This paper describes the Mavigator prototype and its visual metaphors that offer new quality for visual querying of object-oriented databases. Our ideas have been presented from the PhD thesis point of view. We have stressed the items, which should be included in the dissertation. We have also enumerated some open issues, which should be carefully investigated. We hope that the OTM PhD Symposium will be a great opportunity to discuss problems related to visual querying and our particular approach to solve them.

References

1. Object Data Management Group: The Object Database Standard ODMG, Release 3.0. R.G.G.Cattel, D.K.Barry, Ed., Morgan Kaufmann, 2000
2. Subieta K., Beeri C., Matthes F., Schmidt J.W.: A Stack-Based Approach to Query Languages. Proc. 2nd East-West Database Workshop, 1994, Springer Workshops in Computing, 1995, 159-180
3. Zloof M. M.: Query-by-Example: A Database Language. IBM Syst. Journal, 16(4), 1977, 324-343
4. Chimera, R.: Value Bars: An Information Visualization and Navigation Tool for Multi-Attribute Listings. Proc. ACM CHI '92, pp. 293-294, (1992).
5. Kumar, H., Plaisant, C., Shneiderman, B.: Browsing Hierarchical Data with Multi-Level Dynamic Queries and Pruning. IJHCI, vol. 46, pp. 103-124, (1997).
6. Catarci T.: What Happened When Database Researchers Met Usability. Information Systems 25(3), 2000, 177-212.
7. Trzaska M., Subieta K.: Structural Knowledge Graph Navigator For The Icons Prototype. Proc. of the IASTED International Conference on Databases and Applications (DBA 2004)
8. Trzaska, M., K.Subieta, K.: The User as Navigator. Proc. 8th East-European Conference on Advances in Databases and Information Systems (ADBIS), September 2004, Budapest, Hungary, to appear
9. Carey M.J., Haas L.M., Maganty V., Williams J.H.: PESTO: An Integrated Query/Browser for Object Databases. Proc. VLDB (1996) 203-214
10. Murray N., Goble C., Paton N.: Kaleidoscape: A 3D Environment for Querying ODMG Compliant Databases. In Proceedings of Visual Databases 4, L'Aquila, Italy, May 27-29, 1998
11. Murray N., Paton N.W., Goble C.A., Bryce J.: Kaleidoquery - A Flow-based Visual Language and its Evaluation. Journal of Visual Languages and Computing 11(2), 2000, 151-189
12. Fegaras L.: VOODOO: A Visual Object-Oriented Database Language For ODMG OQL. ECOOP Workshop on Object-Oriented Databases 1999, 61-72
13. Jones S., McInnes S.: A graphical user interface for boolean query specification. International Journal on Digital Libraries Special Issue on User Interfaces for Digital Libraries, 2(2/3):207–223, 1999

14. Shneiderman, B.: Visual user interfaces for information exploration. In Proceedings of the 54th Annual Meeting of the American Society for Information Science, pages379–384, Medford.NJ, 1991.Learned Information Inc.
15. Stolte Ch., Tang D., Hanrahan P.: Polaris: A System for Query, Analysis and Visualization of Multidimensional Relational Databases. IEEE Transactions on Visualization and Computer Graphics, Vol 8, No 1, January-March 2002
16. Cassel K., Risch T.: An Object-Oriented Multi-Mediator Browser. 2nd International Workshop on User Interfaces to Data Intensive Systems, Zürich, Switzerland, May 31 - June 1, 2001
17. Smith M., King P.: The Exploratory Construction Of Database Views. Research Report: BBKCS-02-02, School of Computer Science and Information Systems, Birkbeck College, University of London, 2002
18. Batini C., Catarci T., Costabile M.F., Levialdi S.: Visual Strategies for Querying Databases. Proc. of the IEEE Int. Workshop on Visual Languages, Japan, October 1991.
19. Derthick M., Kolojejchick J., Roth S.F.: An Interactive Visual Query Environment for Exploring Data, Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '97), ACM Press, (1997) 189-198
20. Bertin J.: Semiology of Graphics. Madison, Wis.: Univ. of Wisconsin Press, 1983. Translated by W. J. Berg
21. Nowell L.T.: Graphical Encoding for Information Visualization: Using Icon Color, Shape, and Size To Convey Nominal and Quantitative Data. PhD dissertation, Virginia Polytechnic Institute and State University, 1998
22. Matisse – a post-relational database. www.matisse.com
23. Chavda M., Wood P. T.: Towards an ODMG-Compliant Visual Object Query Language. In Proc. 23rd Int. Conf. on Very Large Data Bases, 1997, pp. 456-465
24. Ivory M. Y., Hearst M. A.: The state of the art in automating usability evaluation of user interfaces. ACM Comput. Surv. 33(4): 470-516 (2001)
25. Plaisant C.: The Challenge of Information Visualization Evaluation. In Proc. of Conf. on Advanced Visual Interfaces AVI'04 (2004) (to appear)
26. Rivadeneira W., Bederson B. B.: A Study of Search Result Clustering Interfaces: Comparing Textual and Zoomable User Interfaces. Human-Computer Interaction Lab / Univ. of Maryland, Report no HCIL-2003-36, October 2003

Automated Compatibility Tests for Business Related Aspects of Software Components

Johannes Maria Zaha

Business Informatics and Systems Engineering
Business Faculty, University of Augsburg,
Universitätsstraße 16, 86135 Augsburg, Germany
johannes.maria.zaha@wiwi.uni-augsburg.de

Abstract. The focus of the thesis is to develop algorithms for enabling and executing automated compatibility tests for business related aspects of software components. Compatibility tests are necessary for the identification of required components, which are traded on component markets. Compositional reuse of software components requires standardized specification techniques if applications are created by combining third party components. Adequate techniques need to be used in order to specify not only technical but also business related aspects of software components. The use of formal specification techniques is a prerequisite for compatibility tests on component specifications. Thereby a specification framework with informal, semiformal and formal specification techniques for different levels of abstraction is used. Based on this framework compatibility tests have been developed for some of the technical aspects of software components in preparatory work. Thereby the requirements on specification techniques concerning the necessary degree of formalization in order to execute automated compatibility tests were examined. Since business related aspects of software components are informal or semiformal in order to enable experts of a certain business domain to specify their demands, a well-defined algorithm has to be developed, allowing the specification techniques of the domain experts to be transferred into a formalized specification language in order to execute automated compatibility test.

Keywords: Compatibility test, software components, software reuse.

1 Business Components

The development of compatibility tests for business related aspects of software components is the final goal of this thesis. These compatibility tests are the prerequisite for the long lasting idea of reusing black-box software components. The idea of developing application systems from prefabricated software components has been traced at least since the publication of McIllroy in 1968 [1]. Since then different techniques for reusing software artefacts, like code and design scavenging [2] or generative techniques [3], have been developed. Compositional reuse of software instead is a technique to combine the advantages of both standard software and individually programmed software by a plug-and-play-like reuse of black-box-

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 834–841, 2004.
© Springer-Verlag Berlin Heidelberg 2004

components which are traded on component markets. To achieve this goal, developers and users need to express the characteristics of a software component in a standardized way. Web Service standards like the Web Service Description Language (WSDL) [30] and Universal Discovery and Description (UDDI) [31] are approaches for reaching this goal. As stated in [10], the need for specifying Web Services is comparable to the one of software components, but there is a lack of specification techniques for behaviour related, quality related and partly business related aspects.

An approach considering specification of technical and business related aspects on different levels of abstraction for business components, which are components that offer a certain set of services of a given business domain, has been proposed by [4] (cp. Fig. 1). This specification framework establishes the basis for component compatibility tests in this thesis.

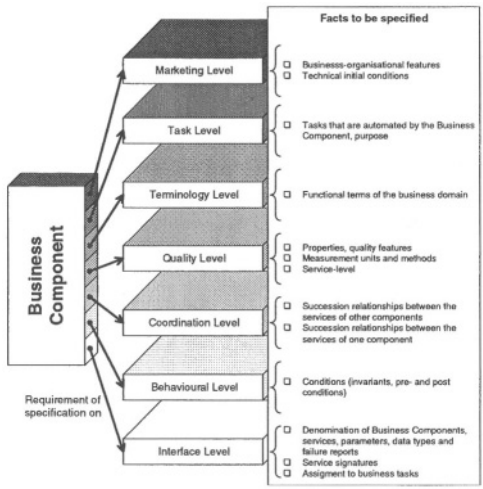


Fig. 1. Software contract levels and facts to be specified (cp. [4])

In this framework standardized techniques for the specification of business components on different levels of abstraction have been chosen, like the Interface Definition Language (IDL) [5] on Interface Level, the Object Constraint Language (OCL) [6] on Behavioural Level or the Reconstructed Business Language [7] on Task and Terminology Level. With this specification framework it is possible to enter into a contract on a piece of software in a standardized way. The concept of software contracts was introduced by Meyer [8] with his programming language Eiffel. This idea of programming by contract was later on extended to the concept design by contract [9], proposing that the design phase should end with a detailed specification of the components using standardized specification techniques. Thereby the efforts during specification are increased, but costs will be charged off in the later phases in the process of software development (cp. [24], p. 207). With this concept it is possible to decrease the effort to build application systems by reusing existing components, which leads not only to lower costs but to other benefits like improved software quality or a shortened time-to-market (cp. [23], p. 102).

To enable the reuse of black-box components, the components traded on component markets have to be evaluated concerning the usability for a certain task. Therefore compatibility tests on all abstraction layers are necessary. However the focus of the thesis is a compatibility test for Task and Terminology Level. On Task Level, the tasks supported by the described business component are specified. To specify these tasks, only predefined sentence construction plans can be used. Therewith schemas are constituted, which can be filled with terms, whereby each used term has to be defined unequivocally on Terminology Level. An example for a sentence construction plan is depicted top left in figure 3. The sentence construction plan *after executing _follows executing _* can be filled with terms like *A, B, C,* or *D.* These terms, in the example functions in a specific business process, have to be defined unequivocally on Terminology Level.

In order to determine the necessary degree of formalization of a specification technique for executing compatibility tests, algorithms for tests on the technical layers of the introduced framework have been generated as preparatory work. Thereby the Interface Definition Language (IDL) has been stated as language with an adequate degree of formalisation by implementing an algorithm for a test on equivalence of two specifications on Interface Level and the subsequent generation of adapters for the two interfaces, if possible [12]. Different approaches exist in the area of compatibility tests, but respecting all work done in this area, none of the approaches provides an algorithm for testing compatibility of specifications or for generating respective adapters. Zaremski and Wing introduced a type system for functions and modules [13] in order to test compatibility of software components by deriving information of their interfaces. Their approach was extended for differing signatures [14] and pre- and post-conditions [15]. Other work extended these approaches for protocols needed for interaction between components [16] and introduces the generation of component adapters [17]. Thereby the possibility of developing compatibility tests was proved in a formal way, but no algorithm was provided.

The prerequisite of a formal language is not given for the test on Task and Terminology Level. E.g., the Web Service standards mentioned above are due to their medium degree of formalization neither suitable for the use as informal or semiformal language for domain experts nor as formal language for the execution of compatibility tests. Since no representation between these extreme characteristics – concerning the degree of formalization – is necessary in order to perform the transformation of the language artefacts, this attempt will not be followed for the specification of business related aspects of software components.

Whereas the language provided by the specification framework [4] used in the thesis is suitable for one of the extremes. The Reconstructed Business Language [7] is not suitable for automated compatibility tests because of its semiformal syntax, but it is suitable as specification technique for domain experts. The central elements of this language are sentence construction plans. According to [7], these sentence construction plans can be defined depending on the application scenario. Instances of an existing sentence construction plan are filled with terms, which have to be defined unequivocally in a dictionary on Terminology Level. Some attempts to define a set of sentence construction plans have been made (cp. [25] and [26]), but an approach defining a generic set of sentence construction plans to describe any kind of domain is lacking.

To achieve the stated goal of automated compatibility tests on Task and Terminology Level, a generic set of sentence construction plans needs to be defined

and transferred into a formal language. The ontology language DAML+OIL [11] has been identified as an adequate language and will be used in the thesis. A suggestion for algorithms necessary for transforming sentence construction plans into DAML+OIL-constructs have been proposed and will be refined in the further progress of the thesis. Parts of the existing algorithm are defined in the following section for a simple example.

2 Compatibility Tests and Multidimensional Specification

An initial set of sentence construction plans has been developed by analyzing the domain of retail information systems [22]. Thereby pre-structured information was formalized. This pre-structured information was gained out of a mainly graphical reference model [21], which represents a domain model. Considering the definition of the initial sentence construction plans only the specification language for describing the domain was changed. This initial collection of sentence construction plans should be satisfactory for describing any domain, but has to be consolidated in future work by experiments in other domains.

Primary aim of domain modelling is to identify structure- and behaviour-oriented features of a domain, which are independent from a concrete application system [27]. According to [28] the knowledge of domain experts is needed to implement the model of the domain. These experts are not the target group of adequate specification techniques, because they merely serve as deliverers of domain knowledge. Process models like the Feature-Oriented Domain Analysis (FODA) [29] are distinguishing between roles providing knowledge and roles using this knowledge for modelling.

In the presented approach of domain modelling and finding software components it is essential that the person describing the domain with the initial set of sentence construction plans is the same as the one searching for software supporting tasks in the described domain. The integrative process of finding reusable software components (cp. Fig. 2) requires the transformation of the specification languages used. Therefore algorithms have to be defined, describing how the specification techniques of the domain experts can be transferred into a formalized specification language (DAML+OIL) in order to execute automated compatibility tests.

The way of transforming the specification techniques for (domain) experts into those for compatibility tests is defined separately for each layer of the specification framework with a grammar-mapping (cp. Fig. 2). This grammar-mapping becomes simpler, when getting closer to technical specification levels. On Interface Level the specification languages used are even identically. Interface Definition Language (IDL) [5] is used as specification technique as well as for compatibility tests. Therefore no grammar-mapping is needed.

The higher the level of abstraction gets by moving up the specification framework introduced above, the stronger is the need for a semi formal or informal specification technique fitting the needs of domain experts. Therefore the mapping of the grammars gets more complicated.

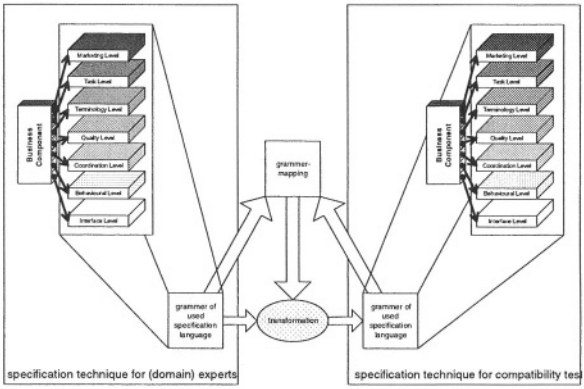


Fig. 2. Transformation process of different specification languages

Besides the existing multidimensionality in [4] by introducing seven specification layers of different abstraction, a second kind of multidimensionality is introduced, namely for specifying one layer with different intentions (domain specification or compatibility tests). As stated above, there is no additional representation needed to transfer the specification of business related aspects of a domain expert into a formal representation to execute automated compatibility tests. Nevertheless not a two-dimensionality but a second multidimensionality is created, because experiments showed, that domain experts prefer additional graphical notations such that there are three or more specification techniques used for one layer. The mapping of these different grammars is not trivial, because these different notations usually have different degrees of formalization. Then it is very difficult to achieve a bijective transformation of one language to another. However, a one-to-one and onto transformation is important, because for the searching and finding of business components on component markets, in addition to the transformation of informal and semiformal into formal descriptions, the inverse function is also required.

In theses experiments the initial set of sentence construction plans was created as described above. Additionally the sentence construction plans were transformed in DAML+OIL-statements. In figure 3 a simple example for the succession of different functions in a business process is depicted.

Starting with the process model in graphical notation on the right side, sentence construction plans were built to which the images could be mapped. In the depicted example, all relationships between the four functions *A, B, C* and *D* of the business process could be represented with only one sentence construction plan: *after executing _follows executing _*. The AND-conjunctions are only implicit existent in the four sentence construction plans, which describe the relationships between two direct successive functions. In DAML+OIL-notation at the bottom of the image, each function is represented by a labelled resource. The relationships between the different functions are presented in DAML+OIL by annotating possibly existent predecessor at each function. The logical principles of the notation are different from those of the sentence construction plans. Due to this fact and the relative low abstraction layer of the specification artefacts, the mapping of one specification language to another, regardless if text-based or graphical noted, is not bijective. This problem occurs especially with complex and complicated domains or rather models.

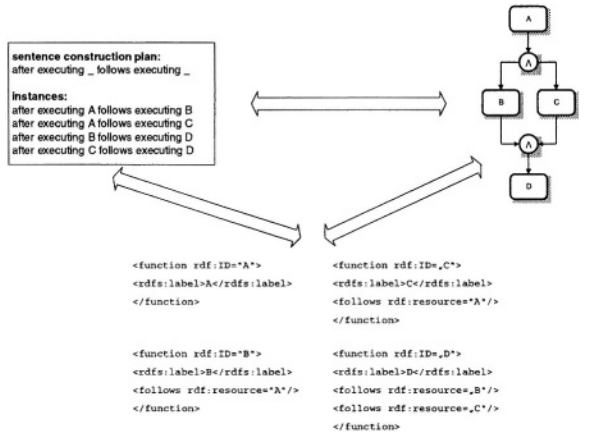


Fig. 3. Example of transforming different specification languages

In this experiment a set of sentence construction plans claimed above was generated, providing the starting point for the consolidation of this collection and the mapping of them to DAML+OIL-constructs, stated in the respective grammar-mappings. However, most work will occur in the field of implementing compatibility tests. Different kinds of tests are necessary, e.g. tests analogue to the existing tests for IDL-signatures on equivalence or tests based on a knowledge pool related to a certain domain. Users searching for reusable software components could then ask questions, which could be answered with knowledge contained in the knowledge pool.

3 Conclusion and Outlook

Recapitulating, the compatibility tests of business related aspects of software components are completely dependant on the existence of an algorithm to transform specification artefacts from an (domain) expert language to a language for compatibility tests. The existing experiences have to be iterated, because the actual set of sentence construction plans has to be consolidated and fixed. This is a prerequisite for a stable transformation process between the varying dimensions of one specification layer. For this reason, tests in other domains are planned, where the information is not pre-structured but has to be gathered by the modellers of the domain. Thereby new sentence construction plans will be identified, which have to be integrated into the existing set.

The sentence construction plans, which have been developed by this means, have to be transferred in DAML+OIL. This specification language has appeared as adequate for this purpose, because it is notwithstanding to the satisfactory degree of formalization relative easy to read and therefore suitable for the creation of transformation rules. In the latest experiments, DAML+OIL was substituted by the Web Ontology Language (OWL), because this language has supplementary features and is currently a recommendation of the World Wide Web Consortium (W3C). Since

DAML+OIL was the starting point for the development of OWL, the transformation of already achieved results in the new language is simple.

For the final implementation of compatibility tests other languages will be used, because existing tools that offer inference machines (cp. [18], p.357) for the generation of new knowledge are working with established ontology languages like Ontolingua [19] or FLogic [20].

The final goal of this research project is to enhance existing tools – which already support the specification of software components according to [4] – towards an integrated development environment, where all users are able to specify their demands and search for existing software components or Web Services that can be reused in order to decrease the effort and increase the quality of future application systems.

References

[1] McIlroy, M.D. *Mass Produced Software Components.* In: Software Engineering: Report on a Conference by the NATO Science Committee. 1968. Brussels: NATO Scientific Affairs Devision.

[2] Sametinger, J., *Software engineering with reusable components.* 1997; Berlin, New York: Springer. xvi, 272 p

[3] Czarnecki, K. and U. Eisenecker, *Generative programming : methods, tools, and applications.* 2000, Boston: Addison Wesley. xxvi, 832 p.

[4] Turowski, K. (Ed.), *Standardized Specification of Business Components.* 2002, Gesellschaft für Informatik, Working Group 5.10.3 - Component Oriented Business Application Systems: Augsburg.

[5] OMG (Ed.), *The Common Object Request Broker: Architecture and Specification: Version* 2.5. 2001, Framingham.

[6] OMG (Ed.), *Unified Modeling Language Specification: Version 1.4.* 2001, Needham.

[7] Ortner, E., *Methodenneutraler Fachentwurf: Zu den Grundlagen einer anwendungs-orientierten Informatik.* 1997, Stuttgart.

[8] Meyer, B., *Object-Oriented Software Construction.* 1988, Englewood Cliffs.

[9] Meyer, B., *Applying "Design by Contract".* IEEE Computer, 1992. 25(10): p. 40-51.

[10] Krammer, A., Turowski, K., *Spezifikationsbedarf von Web-Services.* In: Ortner, E., Overhage, S. (Eds.), 1. Workshop Entwicklung von Anwendungen auf der Basis der XML Web-Service Technologie. 2002, Darmstadt, pp. 1-16.

[11] W3C (Ed.), *DAML+OIL Reference Description.* 2001, http://www.w3.org/ TR/daml+oil-reference (download 2004-07-11).

[12] Zaha, J.M., Geisenberger, M., Groth, M., *Compatibility Test and Adapter Generation for Interfaces of Software Components.* 2004, University of Augsburg, Chair of Business Information Systems and Systems Engineering, Internal Report.

[13] Zaremski, A. M., Wing, J. M., *Signature Matching: A Key to Reuse.* Proceedings of SIGSOFT '93, 1993.

[14] Zaremski, A. M., Wing, J. M., *Signature Matching: a Tool for using Software Libraries.* ACM Transactions on Software Engineering and Methodology (TOSEM), 1995

[15] Zaremski, A. M., Wing, J. M., *Specification Matching of Software Components.* Proceedings of 3rd ACM SIGSOFT Symposium on the Foundations of Software Engineering, 1995.

[16] Nierstrasz, O., *Regular Types for Active Objects.* Proceedings of the eighth annual conference on Object-oriented programming systems, languages, and applications, 1993.

[17] Yellin, D. M., Strom, R. E., *Protocol Specifications and Component Adapters.* ACM Transactions on Programming Languages and Systems (TOPLAS), 1997.
[18] Gómez-Pérez, A., Fernández-López, M., Corcho, O., *Ontological Engineering.* 2004, London: Springer Verlag.
[19] Farquhar, A., Fikes, R., Rice, J., *The Ontolingua Server: A Tool for Collaborative Ontology Construction.* In: International Journal of Human Computer Studies 46 (6), pp. 707–727.1997.
[20] Kifer, M., Lausen, G., Wu, J., *Logical Foundations of Object-Oriented and Frame-BAssed Languages* In: Journal of the ACM 42 (4), pp. 741 – 843. 1995.
[21] Becker, J., Schütte, R., *Handelsinformationssysteme – Domänenorientierte Einführung in die Wirtschaftsinformatik.* 2004, Frankfurt am Main: Redline Wirtschaft.
[22] Brocke, J. v., *Referenzmodellierung: Gestaltung und Verteilung von Konstruktions-prozessen.* In: Advances in Information Systems and Management Science. 2003, Berlin: Logos Verlag.
[23] Lim, W. C., *Managing Software Reuse: A Comprehensive Guide to Strategically Reengineering the Organization for Reusable Components.* 1998, London: Prentice Hall.
[24] Sommerville, I., *Software Engineering.* 2001, München: Pearson Stadium.
[25] Ortner, E., *Ein Multipfad-Vorgehensmodell für die Entwicklung von Informations-systemen – dargestellt am Beispiel von Workflow-Managemnt-Anwendungen.* In: Wirtschaftsinformatik 40 (1998) 4, pp. 329-337.
[26] Schienmann, B., *Objektorientierter Fachentwurf.* 1997, Stuttgart: Teubner Verlag.
[27] Prieto-Díaz, R., *Domain Analysis for Reusability.* In: IEEE (Ed.): Proceedings, COMPSAC 87, The eleventh Annual International Computer Software and Applications Conference. 1987, Tokyo, pp. 23-29.
[28] Prieto-Díaz, R., *Domain Analysis: An Introduction.* Software Engineering Notes 15(2). 1990, pp. 47-54.
[29] Kang, K. C.; Cohen, S. G.; Hess, J. A.; Novak, W. E.; Peterson, A. S., *Feature-Oriented Domain Analysis (FODA) Feasibility Study.* Technical Report Nr. CMU/SEI-90-TR-21. 1990, Software Engineering Institute, Carnegie Mellon University, Pittsburgh.
[30] W3C (Ed.), *Web Service Description Language (WSDL).* 2004, http://www.w3.org/TR/wsdl (download 2004-07-11).
[31] UDDI Organisation (Ed.), *UDDI Technical White Paper.* 2004, http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm (download 2004-07-11).
[32] W3C (Ed.), *Web Ontology Language (OQL).* 2004, http://www.w3.org/TR/owl (download 2004-07-11).

Formal Infrastructure for Modelling Intelligent Agents with Agent UML and Petri Nets

Marina Bagic

Department of Telecommunication, Faculty of Electrical Engineering and Computing,
University of Zagreb
10000 Zagreb, Croatia
marina.bagic@fer.hr

Abstract. The complexity of today's software systems demands specific approach to software development. While *Unified Modelling Language* (UML) has become standard notation for analysis and design of software systems, and its extension *Agent UML* (AUML) is yet to become a standard, current (A)UML specifications have many limitations because they are initially intended to be general-purpose and suitable for many different kinds of applications. On the other hand, Petri nets are a formal concept suitable for simulation, validation and verification of software system execution. This thesis is focused on analysing the current AUML specifications and the concepts of Petri nets in order to find connecting points between them and to propose the means for extending AUML with Petri nets. In order to practically examine the possibilities for extending AUML with the concept of Petri nets this thesis applied the proposed methodology for extending AUML with Petri nets to development of multi-agent system and uses AUML as a main modelling tool for multi-agent system specification and design, and Petri nets as a formal verification and validation tool before actual implementation.

1 Introduction

Intelligent software agents are becoming one of the most important topics in distributed and autonomous decentralized systems. The increasing interest in multi-agent system research is due to the significant advantages inherent in such systems, including their ability to solve problems that may be too large for a centralized single agent, to provide enhanced speed and reliability, and to tolerate uncertain data and knowledge [1].

Formal methods have been frequently adopted in the requirements analysis phase to specify a system and its desired properties, e.g., behavioural properties; however, to create formal models in the design phase and therefore verify their correctness is rare. This is not only because of the infancy of the techniques and the apparent difficulty of the notations used, but also due to a lack of support for modularisation in most of the formal approaches. This observation has motivated research on using

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 842–853, 2004.
© Springer-Verlag Berlin Heidelberg 2004

formal methods in concurrent object-oriented design, and further derived model-based approach for development of agent-oriented software systems [2].

Object-orientation software paradigm offers formalism for highly reusable and modular systems, but partially lacks concurrency features. Effective modelling of complex concurrent systems requires a formalism that can capture essential properties such as non-determinism, synchronization and parallelism. Petri nets concept efficiently supports specification of these features. Due to their well-defined semantics they are also powerful tool to both simulate and analyse system characteristics once the system has been completely specified. Petri nets offer a clear formalism for concurrency, however they partially lack thorough modularisation techniques. However, modularisation is very well introduced as an inherent feature of Agent UML modelling technique that also support extended object-oriented constructs. Therefore, an idea to merge these two concepts into one seems natural, so the developer profits from the benefits from the strengths of both approaches.

Saldhana, J. A. and Shatz, S. M [7] have built an approach to both extend Petri nets onto Object Petri nets and connect them with UML diagrams. They used Statechart and Collaboration UML Diagram to express the idea of translation mechanism. They used Statechart Diagram to represent internal entity's actions and activities and Collaboration diagram to connect different communication actors into one system. Such system was designed to separate communication entities' interfaces from their internal implementations to adopt object-orientation approach. It was a first step toward extension of Petri nets onto modularization support. Such attempts were also works written by Pooley, R. and King, P. [12] and Bergenti and F., Poggi, A.[13]. Nevertheless, these attempts were mainly focused on translation of some specific UML diagrams onto Petri nets.

Hughes, P. and Pahl, C [14] have given a generic model for state-based agent systems as formal method to describe multi-agent infrastructure with both of its static structure and internal behaviour.

Xu, H., Shatz, S. M. and Smith, A. B. [1] have given a framework for model-based design of agent-oriented software which Xu, H. [2] has built a complete model-based approach for development of multi-agent software systems encompassing all software development phases, from requirements through specification and design to final implementation.

Sertic, H. [15] has given an insight into UML diagrams translation onto Petri nets with the application in real-time system modelling.

Also, there have been a number of papers proving significance in simulation, validation and verification of Coloured Petri nets usage (Kristensen, L. M., Christensen and S., Jensen, K. [3], Jensen, K. [4], Jensen, K. [8]). In further text Petri nets would mean Coloured Petri Nets (CPN, or CP-nets) since the basic idea of the thesis considers only Coloured Petri nets.

This thesis should encompass all these enumerated aspects of research into one, offering unique translation mechanism from AUML diagrams to Petri nets at the same time proving importance of such connection and giving an insight of such mechanism generality of application. Main focus should be on exploring both AUML and Petri nets' semantics regarding multi-agent system derivation.

So far, a new approach to AUML diagrams has been achieved and is presented in the next chapter. Then, after shortly explaining the significance of Petri nets, an idea of focusing elements of translation onto Petri nets is given (Chapter 4). And, finally, a path for further work and research considering applications, and further directions of exploring theoretical concepts are given.

2 Formal Definitions of Agent UML Diagrams

An Agent is a computer system that is situated in some environment, and is capable of autonomous action in this environment in order to meet its design objectives [10]. Its characteristics are mostly extracted into several ones: autonomous, situated, reactive, pro-active, social, intelligent, organized and mobile. To support such variety of qualities *Foundation for Intelligent Physical Agents* (FIPA) has developed and adopted several diagrams extending UML diagrams regarding the idea that an agent extends notion of an object. Here is proposed different insight into such diagrams giving their form as n-tuples and focusing mainly on their semantics. However, these definitions should not be considered as final since diagrams are not given their final standard form by FIPA. Based on FIPA Modelling drafts [9], [10] and [11] it is possible to give next definitions;

2.1 Agent Class Diagram

Definition 2.1. Agent class diagram is 4-tuple:

$$ACD := (I, R, O, AD)$$

where I is a set of agent identifiers, R a set of their roles and O, set of organizations. AD is an optional set of additional agents' associative classes, i.e. additional description consisting of a collection of subsets from a set AD:= (CD, SD, PD, ACLD, ACAD, OD, KD, BMD, GD, AD', ED, MD, PD), where CD is a capability description, SD a service description, PD a protocol description, ACLD an agent communication language description, ACAD an accepted communicative act description, OD an ontology description, KD a knowledge description, BMD a *Beliefs–Desires-Intentions* (BDI) modalities description, GD a goals description, AD' an actions description, ED events description, MD mobility description, and PD persistence description.

2.2 Sequence Diagram

Definition 2.2. Sequence diagram is 5-tuple:

$$SD := (I, P, M, O, A)$$

where I is a non-empty set of agents or agent roles. P is a non-empty set of protocols where one is major (sd Protocol), and the others can be either interleaved or nested protocols. M is a set of messages either synchronous or asynchronous. Each message

is specified as 2-tuple m := (a, options), where $a \in \{a_s, a_a\}$, a_s - synchronous message, a_a - asynchronous message, and options specify whether a message is intended to be received by more than a one receiver (n >= 1), or a range of possible receivers ([m, n] equals m <= r <= n, where r is a number of receivers). Also, an option can specify some special conditions on which a message sending is performed or to explicitly identify a particular agent. O is a set of operators O := {alternative, option, break, parallel, weak, strict, negative, critical, ignore, consider, assertion, loop} such that can be easily extended to an appropriate number of newly introduced operators. A is a non-empty set of directed arcs connecting communication actors, therefore it is $A \subseteq I \times I$, and $I \times I$ is a set of Cartesian products $(i1, i2) \in I \times I$, for $i1, i2 \in I$. Shape of arc must correspond to character of message, thus arcs are either synchronous or asynchronous [5].

2.3 Activity Diagram

Definition 2.3. Activity diagram AD is 3-tuple:

$$AD := (S, A, C)$$

where S is a non-empty set of agent states, A is a set of relations between the states where $A \subseteq S \times S$, and $S \times S$ is a set of Cartesian products $(s_1, s_2) \in S \times S$, for $s_1, s_2 \in S$. C is a set of conditions or events that decide whether agent can change its state or not. These expressions are represented as informal set of text strings [5].

2.4 Interaction Overview Diagram

Definition 2.4. Interaction Overview diagram ID is 2-tuple:

$$ID := (SD, AD)$$

where a SD is a set of interactions provided for inter-agent communication as seen in Sequence diagram, i.e. ID is a collection of few Sequence diagrams connected into one diagram via AD, elements of Activity diagram.

Interaction Overview Diagrams focus on the overview of the flow of control where the nodes are Interactions or Interaction Occurrences. The Lifelines and the Messages do not appear at this overview level. Interaction Overview Diagrams are specialization of Activity Graphs that represent Interactions, and differ from Activity Diagrams in some respects [9].

2.5 Communication Diagram

Definition 2.5. Communication diagram CD is 3-tuple:

$$CD := (I, R, M)$$

where I is a non-empty set of agent identifiers (or agent roles), R is a set of relations among actors (agents) where $R \subseteq I \times I$, and $I \times I$ is a set of Cartesian products $(i_1, i_2) \in I \times I$ (i_1, i_2) for $i_1, i_2 \in I$. M is a set of messages where each message is represented as m := (k, s) where s is a string value representing message name or content of mes-

sage in its simplest instantiation, and $k \in N$ is a complex number $k_1.k_2...k_n$ representing the sequential order of the message within the next higher level of procedural calling. Messages that differ in one integer term are sequentially related at that level of nesting.

Communication diagram can also be defined as strongly connected directed graph where each node represents communication actor of interaction and each (virtually, two-headed) arrow represents relation between two actors. Messages supplement arrows with semantics of relation [9].

2.6 Agent World Diagram

Definition 2.5. Agent World diagram AWD is 4-tuple:

$$AWD := (A, R, M, \pi)$$

where a set $A := \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n\}$ consists of ε_i, an extended objects enriched with agent specific properties focused mainly on agent mental state, its autonomy and internal-motivation; R, a set of relation between ε_i such that $R \subseteq AxA$. A function $\pi(\varepsilon_i) : \varepsilon_i \rightarrow (m_1, m_2, \ldots, m_k)$, dedicates each agent a set of independent modules from a set $M := (m_1, m_2, \ldots, m_k)$, where each m_i has its own functionality, a set of internal attributes, methods and interfaces.

This diagram is proposed as one corresponding to UML Use Case diagram which is not so far defined in AUML. Agent World diagram depicts multi-agent system at its higest level of abstraction regarding agents and relations among them.

3 Coloured Petri Nets

Coloured Petri Nets (CPNs) provide a framework for the design, specification, validation and verification of systems. Coloured Petri Nets are based on a well-defined semantics, which unambiguously defines the behaviour of each CP-net. Such a property forms the foundation for the formal analysis methods. CPNs' semantics is built upon true concurrency instead of interleaving [3].

CPNs offer generality that is to be used to describe a large variety of different systems. On the other hand, CPNs have very few, but powerful primitives which attracts their users to wide area of usage [4]. Being graphically appealing, CPNs provide an explicit description of both states and actions. They also provide hierarchical descriptions to enable user to specify quasi-modules. This concept has enhanced ordinary Petri net approach to an outstanding dimension that made a step toward object-orientation. This has also introduced reusability feature into the CP-net as the same "module" can be reused by another entity. This mechanism lead us to class modelling since building a model for module structure and its behaviour we build a pattern which provides a set of different module instances containing the same features. Extending this approach further more, we are able to provide inheritance by adding new features onto these instantiated modules, i.e. objects, therefore introducing augment

inheritance. Restrictive and replacement inheritance could also be introduced, but due to their complexity none of them are of major interest here.

So, CPNs support integration of description of control and synchronization with the description of data manipulation. They also provide stability towards minor changes of the modelled system [8].

3.1 Formal Definition of Coloured Petri Nets

Definition 3.1. A CP-net is a 9-tuple [4]:

$$CPN = (\Sigma, P, T, A, N, C, G, E, I)$$

where Σ is a finite set of non-empty colour sets, P is a finite set of places, T is a finite set of transitions, A is a finite set of arcs such that: $P \cap T = P \cap A = T \cap A = \emptyset$, N is a node function. It is defined from A into $P \times T \cup T \times P$. C is a colour function. It is defined from P into Σ.

G is a guard function. It is defined from T into expressions such that: $\forall t \in T$: $[Type(G(t)) = B \wedge Type(Var(G(t))) \subseteq \Sigma]$. Here, Type(V) stands for type of a variable /expression V; Var(expr) a set of variables in an expression expr; B, a set of $b(v) \in Type(V)$, bindings of variable v (mapping a variable v from expression expr to its value).

E is an arc expression function. It is defined from A into expressions such that: $\forall a \in A$: $[Type(E(a)) = C(p)_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma]$ where p is the place of N(a), and $C(p)_{MS}$, a C function from initial CP-net definition regarding multi-set (MS, defined in [4]) in p place.

I is an initialisation function. It is defined from P into closed expressions such that: $\forall p \in P$: $[Type(I(p)) = C(p)_{MS}]$.

The set of types determines the data values and the operations and functions that can be used in the net expressions (i.e. arc expressions, guards and initialization expressions) [4].

4 Translation Mechanism from Agent UML to Petri Nets

4.1 Agent Class Diagram Translation

Based on definitions as they are declared in Chapter 2, translation mechanism from AUML diagrams' elements onto Petri nets elements is proposed.

An atomic operation (as part of agent's method supporting either agent's behaviour or some additional description) is directly translated onto Petri net place. A translation between atomic operation and Petri net place is bijection because each operational step (domain of translation) defines one and only one Petri net place (codomain of translation). Elements of Agent's Class are not translated onto Petri net transitions. Petri net transition is used for definition of control flow of operation execution by activating itself after finishing atomic operation.

An attribute value (as part of agent's structure or some additional description) is directly translated onto one Petri net place. A translation between attribute value and Petri net place is a bijection because each attribute value defines one and only one Petri net place. Attributes are not translated onto Petri net transitions. Petri net transition is used for definition of control flow of operation execution changing the attribute's value.

Given a set of agents $\alpha := \{\alpha_1, \alpha_2, \ldots, \alpha_q\}$, we define a set of relations between them $\rho := \{\rho_1, \rho_2, \ldots, \rho_q\}$, where two agents α_e and α_f are related with relation ρ_j if and only if there exists a Cartesian product (α_e, α_f): $\rho_j \in \alpha \times \alpha$. Such formal expressions of agents and their relations can be translated onto Petri net places and transitions;

$$\Psi(\alpha_i) : \alpha_i \rightarrow PP_i$$
$$\zeta(\rho_j) : \rho_j \rightarrow PT_j$$

Ψ, a function translates each agent α_i onto its respective Petri place PP_i, whereas ζ a function translates each relation ρ_j onto its respective Petri transition PT_j.

4.1.1 Agent's Hierarchy Decomposition - Inheritance Modelling

There is no logically equivalent Petri net construct to represent parent-child relationship between the two classes. It is nevertheless possible to express certain aspects of such relationship (Figure 4.1).

Generalization relationship among classes can be depicted by dedicating each class in a structure a Petri net place, so the very relationship can be expressed as a transition between those places. A class involved in generalization is directly translated onto Petri net place. A translation between a class and Petri net place is a bijection because each class defines one and only one Petri net place.

Generalization relationship is expressed via two transitions where each exhibits one direction (Figure 4.2). Momentary active class holds tokens therefore representing that this one is invoking an active method.

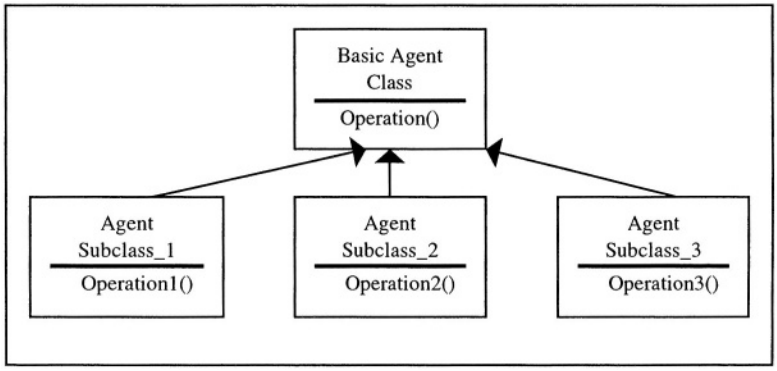


Fig. 4.1. Building Inheritance Modelling by Agent UML

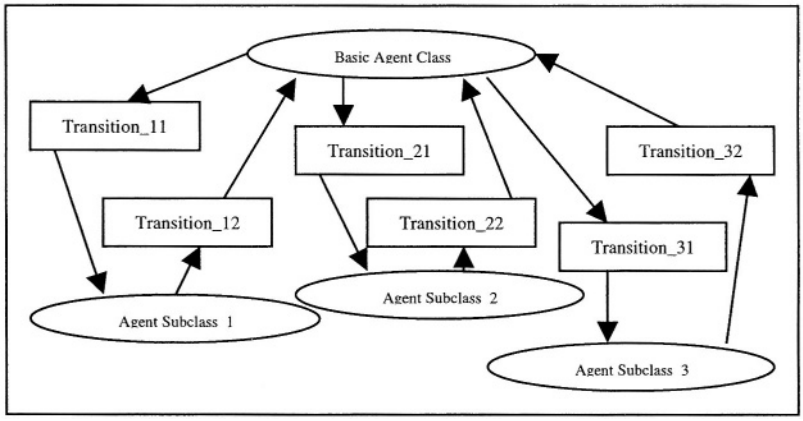


Fig. 4.2. Translation of Agent's Hierarchy onto Petri net Structure

4.2 Sequence Diagram Translation

Sequence diagram does not support semantics to define behaviour execution expressed in diagram. Petri nets are appropriate to extend AUML semantics by formal definition of ordered sequence of messages within interaction and actions of sending and receiving those messages.

Global Petri net dedicated to the whole Sequence diagram represents all agents from sequence diagram and is composed of few sub-Petri nets representing agents involved in communication.

Each message sent by Agent_1 to Agent_2 is represented by following actions (Figure 4.3):

- Communication place represents message sending state of interaction;
- agent *Agent_1* is dedicated to place-transition-place sequence (*State_1i – Message Sending - State_1k*) representing message sending where *State_1i* represents agent's state before message sending, and *State_1k* its state after message sending;
- agent *Agent_2* is dedicated to place-transition-place sequence (*State_2i – Message Receiving - State_1k*) representing message receiving where *State_2i* represents agent's state before message receipt, and *State_1k* its state after message receipt.

Each transition has its entry and exit place corresponding to agents states before and after message sending or receiving. Each Petri net representing agent has its start and finishing place corresponding to starting and finishing agent state of sequence diagram. Each agent is represented with its own thread of control.

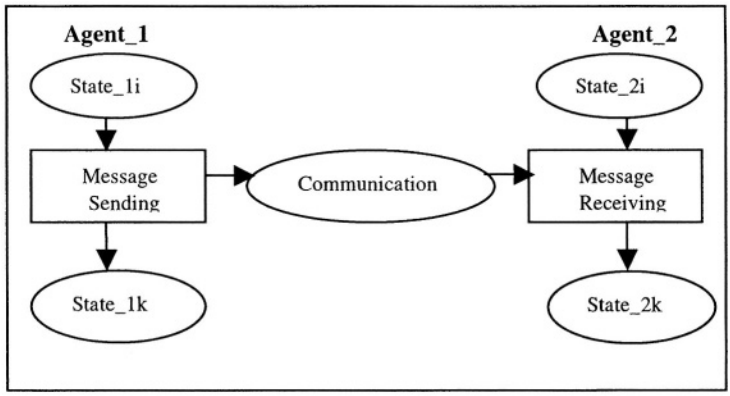


Fig. 4.3. Sequence Diagram as Petri Net

4.3 Activity Diagram Translation

Activity state $\alpha_i \in S$, a set of agent's states, is directly translated onto Petri net place PP_i. Translation Ω between activity state α_i and Petri net place PP_i is a bijection because each activity state α_i defines one and only one Petri net place PP_i:

$$\Omega(\alpha_i) : \alpha_i \rightarrow PP_i.$$

Relation ρ_j between activity states is directly translated onto Petri net transition PT_j. Translation Ξ between relation ρ_j and Petri net transition PT_j is a bijection because each relation ρ_j between the two states defines one and only one Petri net transition PT_j:

$$\Xi(\rho_j) : \rho_j \rightarrow PT_j$$

The synchronization line is directly translated onto Petri net transition with as much entry places as there are exits from activity state leading to synchronization line and as many exit places as there are relations exiting synchronization line.

The token of Petri net place corresponding to activity state expresses execution of activity state.

4.4 Communication Diagram Translation

Entities involved in communication $\varepsilon_i \in (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)$ are directly translated onto Petri net places. Such translation Π is a bijection because each entity defines only one Petri net place:

$$\Pi(\varepsilon_i) : \varepsilon_i \rightarrow PP_i$$

A relation among entities ρ_i is directly translated onto Petri net transition PT_j. Such translation Θ is a bijection because each relation ρ_j among two entities defines only one Petri net transition PT_j:

$$\Theta(\rho_j) : \rho_j \rightarrow PT_j$$

A message sent between two communication entities translates onto transition activation, so token exchange represents message sending.

4.5 Agent World Diagram Translation

Agent World diagram is translated onto CP-net specification using substitution transitions. Substitution transitions are presented as components that hide their internal structure from the outside the system, but provides it with its dedicated functionality. It is possible to reuse each component for different purposes.

A translation of each of agents' modules onto a set of substitution transitions is done by a function Ψ:

$$\Psi(\{m_1, m_2, \ldots, m_n\}): \{m_1, m_2, \ldots, m_n\} \rightarrow \{ST_1, ST_2 \ldots, ST_p\}$$

Agents $\varepsilon_i \in (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)$ are directly translated onto Petri net places. Such translation ϑ is a bijection because each entity defines only one Petri net place:

$$\vartheta(\varepsilon_i) : \varepsilon_i \rightarrow PP_i$$

A relation among agents $\rho_j \in R$ is directly translated onto Petri net transition PT_j. Such translation Λ is a bijection because each relation ρ_j among two entities defines only one Petri net transition PT_j:

$$\Lambda(\rho_j) : \rho_j \rightarrow PT_j$$

5 Case Study: Network Management System

All the theoretical achievement from the above chapters should be applied to a case study multi-agent system and tested to show the final results. Since intelligent telecommunications agents express a certain amount of improvement in network management activities, network management agents are to be explored and tested in simulation environment using current CPN tools.

Also, a software tool regarding the above (partial) translation should be built as practical part of this thesis and tested on such case study.

6 State of the Thesis and Future Plans

This thesis should present formal infrastructure for specification of multi-agent system using both Agent UML and Petri nets. It points out the importance of well-defined formal specification and design to support an early stage of software system development. Such formalism should improve system development before its implementation phase by verifying and validating the system. Petri nets-based approach has been applied to system specification and design because of their benefits in system construction, simulation, functional and performance analysis.

Thesis is focused on multi-agent system analysis due to wide application of agent technologies in distributed artificial intelligence. Agent-oriented models support autonomous, reactive and internally motivated agents.

Due to so far achieved theoretical results in this thesis further research is mainly focused on their practical implementation. Theoretical translation mechanism from Agent UML specification to Petri net formalism should be explored in more detail

level. These insights should analyse semantics of each translated element. Such evaluation could suggest more specific expression of such formal system specification. Since there already exists Petri net .xml file format, similar step could be made onto Agent UML diagrams representation. Such representation could be then easily read and translated partially automatically onto each other.

Another aspect of future research shall be focused on intelligent agent applications in real-time systems. Tasks regarding network management in distributed environment should be dedicated to mobile intelligent agents and tested to prove better performance results as a reason to be accepted.

Results achieved by this thesis should clearly show that there are many aspects of AUML that can be extended with Petri nets to formally define run-time behaviour of modelled system. This thesis should also show that AUML notation extended with Petri nets can efficiently be used for verification and validation of specific real-time software solutions due to Petri net properties.

References

1. Xu, H., Shatz, S. M., Smith, A. B.: A framework for Model-Based Design of Agent-Oriented Software, IEEE Transactions on Software Engineering, Vol. 29, No. 1 (2003)
2. Xu, H.: A model-based approach for development of multi-agent software systems, Ph. D. Thesis submitted in the Graduate College of rhe University of Illinois at Chicago, Chicago, Illinois (2003)
3. Kristensen, L. M., Christensen, S., Jensen, K.: The practitioner's guide to Coloured Petri nets, IEEE Transactions on Software Engineering, Int J STTT 2, Springer-Verlag (1998), 98-132
4. Jensen, K.: An introduction to the Theoretical Aspects of Coloured Petri nets, A Decade of Concurrency, Lecture Motes in Computer Science vol. 803, Springer-Verlag (1994) 230-272
5. Kunstic, M., Bagic, M.: Verification and Validation of Multiagent System Specified by Agent UML, Proceeding of 1st International Conference on Information & Communication Technologies from Theory to Applications – ICTTA '04, Damascus, Syria (2004)
6. Kunstic, M., Bagic, M.: Formal Infrastructure of Multi-agent System Specified by Agent UML and Petri Nets, International Conference on Computer, Communication and Control Technologies CCCT '04, Proceedings of Computer, Communication and Control Technologies, IIIS, Austin, Texas, USA (2004)
7. Saldhana, J. A., Shatz, S. M.: UML diagrams to Object Petri Net Models: An approach for Modeling and Analysis (2000)
8. Jensen, K.: An introduction to the Practical Use of Coloured Petri nets, Advanced Course on Petri Nets, Dagstuhl, Germany (1996)
9. Foundation for Intelligent Physical Agents: FIPA Modelling: Interaction Diagrams, working draft, version 2003-03-03 11:05EST, Geneva, Switzerland (2003)
10. Foundation for Intelligent Physical Agents: FIPA Modelling: Agent Class Diagrams, working draft, version 2003-03-03 11:05EST, Geneva, Switzerland (2003)
11. Foundation for Intelligent Physical Agents: FIPA Modelling: Agent Class Superstructure Metamodel, initial draft for review, London, UK (2003)

12. Pooley, R., King, P.: Derivation of Petri Net Performance Models from UML Specifications of Communications Software, Proceedings of the fifteenth annual UK Performance Engineering Workshop (1999) 45-56
13. Bergenti, F., Poggi, A.: Exploiting UML in the Design of Multi-Agent Systems, Lecture Notes in Computer Science, In A. Omicini, R. Tolksdorf and F. Zambonelli, Engineering Societiea in the Agents World, Springer (2000) 106-113
14. Hughes, P., Pahl, C.: A Generic Model for State-based Agent Systems, Proceedings of the 1st Irish Workshop on Formal Methods, Dublin (1997)
15. Sertic, H.: Uporaba jezika UML i Petrijevih mreza za modeliranje sustava za rad u stvarnom vremenu, Master Thesis, Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb (2004)
16. Pooley, R., King, P.: Derivation of Petri Net Performance Models from UML Specifications of Communications Software, Proceedings of the fifteenth annual UK Performance Engineering Workshop (1999) 45-56

Model-Driven Design of Distributed Applications

João Paulo A. Almeida

Centre for Telematics and Information Technology, University of Twente
PO Box 217, 7500 AE Enschede, The Netherlands
almeida@cs.utwente.nl

Abstract. The objective of the Ph.D. work discussed in this paper is to define a methodology for the design of distributed applications, in line with the Model-Driven Architecture (MDA). An important characteristic of this methodology is that it leads to models of distributed applications that withstand the impact of change in (middleware) platform technologies. These models are organized into different levels of platform-independence that are defined using the notion of abstract platform. An abstract platform is an abstraction of infrastructure characteristics assumed for models of an application at some point of (the platform-independent phase of) the design process. We aim at providing methodological guidelines for the definition of abstract platforms and their representations in modelling languages.

1 Introduction

The timely development of distributed applications is a costly effort. Therefore, an important quality of these applications is their ability to withstand the impact of change, both with respect to changes in application requirements and with respect to changes in the technologies used to build the application.

In the last decades, the development of distributed applications has been facilitated to some extent by the introduction of middleware platforms (e.g., CORBA/CCM [20], .NET [17], JMS [29], and Web Services [30, 31]). These platforms offer generic (distribution) support for distributed applications, masking from applications some details and differences in the support offered by programming languages, operating systems and network protocols. Since a significant amount of development effort is spent on overcoming problems related to distribution (e.g., remoteness, partial failures, heterogeneity) and in exploiting distribution beneficially (e.g., to achieve performance and dependability), the reuse of middleware platforms significantly increases the efficiency of application development.

Different middleware platforms have been developed, to satisfy a variety of needs. The current distributed application scenario is populated by multiple platform standards, implementations from different vendors, proprietary platforms and ad hoc infrastructures, standard and proprietary extensions to platforms, etc. These infrastructures provide different constructs from which applications can be built, and exhibit different quality characteristics. Recently, it has become clear that different parts of a distributed application may be built using various middleware platforms, and that the set of platforms used may change over time. In addition, it has also

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 854–865, 2004.
© Springer-Verlag Berlin Heidelberg 2004

become clear that middleware platforms may evolve during the lifetime of applications. The use of a single immutable distribution infrastructure is therefore not envisioned as a long term solution for the support of distributed applications.

The Object Management Group (OMG) has identified the need to address some of these issues in its Model Driven Architecture (MDA) [19], [21]. This architecture proposes the separation of platform-independent and platform-specific aspects of distributed applications into platform-independent models (PIMs) and platform-specific models (PSMs). A common pattern of MDA development is to define a platform-independent model (PIM) of an application, and to apply (parameterised) transformations to this PIM to obtain one or more platform-specific models (PSMs). The potential benefits of this approach stem from the possibility to derive different PSMs from the same PIM, and to partially automate the model transformation process and the realization of the distributed application on specific target platforms. While this may reduce development costs and improve software quality, it also forms the basis for facilitating the evolution of software solutions, hence contributing to the containment of maintenance costs for distributed applications.

Nevertheless, the appropriateness of MDA as an approach for the development of distributed applications can be criticized on a number of points:

- there is a lack of guidelines to select abstraction criteria and modelling concepts for platform-independent models;
- there is little methodological support to distinguish between platform-independent and platform-specific concerns, which is detrimental to the beneficial exploitation of the PIM-PSM separation of concerns;
- the distinction between platform-independent and platform-specific models is coarse and insufficient to cope with the diversity of application requirements and infrastructure characteristics;
- little attention is given to the role of platform characteristics throughout the development trajectory, possibly leading to models with unacceptable levels of platform-independence and applications with unacceptable quality attributes;
- design operations are not clearly defined, thus inhibiting their effective application along a design trajectory; and
- the focus on a particular design language (UML) constrains the designer in some respects. Currently, it is unclear when and where such constraints apply.

In order to obtain the potential benefits of the model-driven approach to the development of distributed applications, we aim at addressing the issues above in an effective model-driven design methodology. The objective of our work is to propose such a methodology for the design of distributed applications so that:

- available and future distribution infrastructures can be (re-)used, improving the efficiency of the design process;
- the knowledge used to perform various design operations can be captured and re-used to improve the overall efficiency of the design process;
- designs of distributed applications remain stable in face of changes in platform technologies; and,
- designs can be reused to target different middleware platforms.

The methodology is defined so as to be generic with respect to application domains and platform characteristics.

This paper is further structured as follows: section 2 introduces the role of models and discusses the concept of abstract platform; section 3 discusses important activities of the proposed methodology, including abstract platform definition, abstract platform representation and transformation definition; section 4 discusses some work-in-progress; section 5 reviews related work; and, finally, section 6 presents concluding remarks.

2 The Role of Models

2.1 Platform-Independent Models

The development of a distributed application can be regarded as the process of building a realization of the application that satisfies user requirements. In most traditional development cultures, application developers are instructed to produce intermediate models to facilitate bridging the gap between requirements and realization. These intermediate models are mainly regarded as a means to obtain a realization of the system, with different models addressing different design concerns. The ultimate product of the development process is the realization, which can be deployed on available implementation technologies (platforms). Any intermediate models produced during the development processes are considered means and not ends.

In the case of Model-Driven Architecture (MDA) development [21], however, intermediate models that are used to produce the final realization are also considered final products of the development process. These models are carefully defined so as to remain stable in face of changes in platform technologies, and are therefore called platform-independent models (PIMs).

A platform-independent model can be refined or implemented into a number of technology platforms. For the purpose of our work, we assume that a platform corresponds ultimately to some specific middleware technology, such as CORBA/CCM [20], .NET [17], and Web Services [30, 31].

When pursuing platform-independence, one could strive for PIMs that are absolutely neutral with respect to all different classes of middleware platforms. This is possible for models in which the characteristics of supporting infrastructure are irrelevant, such as, e.g., conceptual domain models [7] and RM-ODP Enterprise Viewpoint models [13] (which can be considered Computation Independent Models [21]). However, when the application is described as a decomposition of interacting application parts, different sets of modelling concepts may be used, each of which is better suited for specific classes of target middleware platforms. For example, a designer may describe the interaction between application parts using either request-response invocations or event queues.

The implicit assumption of platform characteristics may result in models that cannot be reused for different platforms. Furthermore, it may lead to models of different applications that cannot be directly compared and integrated. We conclude that platform characteristics assumed in platform-independent models are better understood and controlled by designers if they are explicitly represented. In our design approach, these characteristics are embodied in an abstract platform.

2.2 Abstract Platforms

The notion of abstract platform, as we have proposed initially in [2], supports a designer in defining levels of platform-independence explicitly. An abstract platform is an abstraction of infrastructure characteristics assumed in the construction of PIMs of an application. Alternatively, an abstract platform defines characteristics that must have proper mappings onto the set of concrete target platforms that are considered for a design.

For example, if a platform-independent design contains application parts that interact through operation invocations, then operation invocation is a characteristic of the abstract platform. Capabilities of a (concrete) platform are used during platform-specific realization to support this characteristic of the abstract platform. For example, if CORBA is selected as a target platform, this characteristic can be mapped onto CORBA operation invocations.

The use of the abstract platform concept may be reflected in an abstract platform model, as depicted in the in Figure 1. The PIM of a distributed application depends on an abstract platform model, in the same way as the PSM depends on a (concrete) platform model.

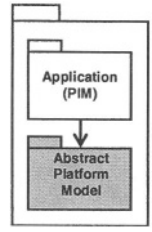


Fig. 1. PIM depends on abstract platform model

3 Methodological Aspects of Model-Driven Design

3.1 Abstract Platform Definition

The number of levels of platform-independence and the characteristics of the models at each level depend on a number of design goals to be balanced, including those of maximizing the efficiency of the design process and maximizing the reusability of models. Different application domain requirements and platform characteristics may also lead to the definition of different levels of platform-independence. We propose the levels of platform-independence should be identified explicitly in an early stage of the design process, which we call *preparation phase* [10]. In the preparation phase, (MDA) experts define the required levels of models as well as define the modelling language(s) to be used in the *execution phase.*

The definition of an abstract platform is supported by two observations [3]:

1. *platform characteristics may play a role in early (platform-independent) designs,* and;
2. *platform-independence must be balanced against platform-specific realization*

The first observation leads us to the conclusion that platform characteristics that play a role in platform-independent designs should be reflected in the abstract platform.

The second observation recognizes that achieving platform-independence is a requirement that must be considered in a larger context, where other relevant design goals play an important role. An MDA design process should lead efficiently to a (platform-specific) application running on a concrete platform.

The next subsections examine these observations and their implications.

Role of Platform Characteristics

Defining an abstract platform requires the ability to identify what abstract platform characteristics are relevant at a platform-independent level. Some platform characteristics become relevant when identifying application parts and their interactions. This is the case for the characteristics of the support for interactions between system parts. Some other platform characteristics play a more subtle, but not necessarily negligible, role. Platform characteristics that may have impact in early stages of the definition of a distributed application's architecture are likely to qualify as abstract platform characteristics.

This is best illustrated by an example, in which the design of a groupware service is considered. This service facilitates the interaction of users residing in different hosts. Initially, the service designer describes the groupware service solely from its external perspective, possibly stating quality-of-service requirements on the service, e.g., that the service should have high availability. At subsequent stages of development, the designer is confronted with design decisions. In this example, we consider the following alternatives: (i) a centralized (server-based) design, and (ii) a distributed (peer-to-peer) design.

Figure 2 depicts these two solutions. In solution (i), a server facilitates the interaction between users. In solution (ii), symmetric components facilitate the interaction without the support of a centralized application-level component.

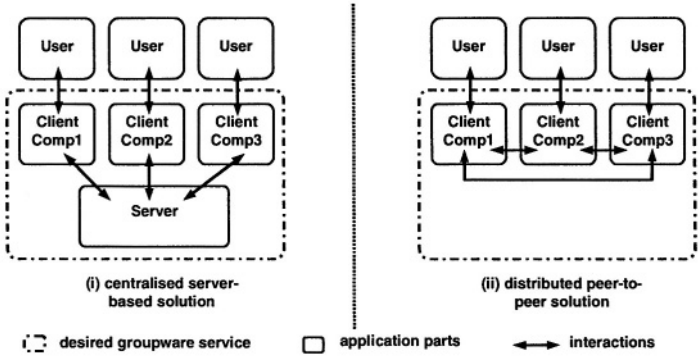


Fig. 2. Alternative designs for the groupware service

In order to improve the reusability of platform-independent models, stable aspects of a system's architecture should be captured in platform-independent models. Therefore, it would be desirable to select between alternative models (i) and (ii) during platform-independent modelling. Nevertheless, some platform-specific aspects play an important role in the selection of an adequate architecture. For example, solution (i) would introduce a single point of failure in the architecture, unless the platform provides support for replication transparency (as defined in the Reference Model for Open Distributed Processing (RM-ODP) [12]).

Apparently, this places the designer in a dilemma, since platform selection would affect platform-independent design. In order to solve this, a designer should be able to express, at a platform-independent level, requirements on platform-specific realizations that would allow all design decisions that are relevant for platform-independent modelling to be captured. In our groupware service example, this would mean that requirements on the reliability of individual components should be stated at the platform-independent level, justifying the selection of a centralized or a distributed design (possibly through application of aspect-oriented modelling [11]).

Requirements expressed at a platform-independent level should justify design decisions for the design at that level and provide input for platform-specific realization. If these requirements invalidate portability requirements for platform-independent designs, then it is impossible to consider the design at the current level of platform-independence. In this case, we envision two different contrasting solutions:

1. to consider the design at a higher level of abstraction, at which the platform characteristics are no longer relevant for design decisions taken at that level; or,
2. to relax portability requirements, lowering the degree of platform-independence for the design. This solution reflects on the characteristics of the abstract platform being defined.

For our groupware service example, possible applications of these solutions would be:

1. to describe the groupware service solely from its external perspective. At this level of abstraction, the reliability characteristics of the supporting infrastructure are irrelevant. Details on the service's internal design are only addressed at platform-specific modelling, and hence cannot be re-used for different target platforms; and,
2. to restrict the set of potential target platforms, e.g., to include only platforms that provide support for highly available components. In this case, it is possible to describe the groupware service's internal design at the newly defined level of platform-independence, while still guaranteeing the satisfaction of the service requirements. The abstract platform considered provides support for highly available components.

In [6], we have presented thoroughly an example of solution (2), where an abstract platform that supports dynamic reconfiguration of components is used at some point of the design process in order to satisfy availability requirements.

Platform-Independence Balanced with Platform-Specific Realization

Defining an abstract platform brings attention to *balancing* between two conflicting goals: (i) platform-independent modelling, and (ii) platform-specific realization. On the one hand, an abstract platform indicates directly the support available for designers during platform-independent modelling, and therefore, reflects the needs of

application designers, including the needs to handle complexity in application design and portability requirements. On the other hand, an abstract platform is established by considering the set of potential target platforms and their (common and diverging) characteristics [2]; this bottom-up knowledge is useful to reduce the design space to be explored for platform-specific realization. Large design spaces are less amenable to automatic exploration, and require more intervention of designer, e.g., through extensive parameterization of transformations. Reducing the design space contributes to increasing the efficiency of the design process.

3.2 Abstract Platform Representation

Designs must be represented using suitable design languages. In a model-driven design process, several design languages may be used, e.g., to produce models at different levels of abstraction and platform-independence. Alternatively, a single "broad spectrum" design language [9] may be used. The design language adopted for a design has an important role in defining characteristics of an abstract platform assumed for the design.

In the *implicit abstract platform definition* approach characteristics of an abstract platform are implied by the set of design concepts used for describing the platform-independent model of a distributed application. These concepts are often inherited from the adopted modelling language. For example, the exchange of "signals" between "agents" in SDL [14] may be considered to define an abstract platform that supports reliable asynchronous message exchange. The restricted use of particular constructs in a design language or the use of certain modelling styles or design patterns can serve as a means to select subsets of a language's design concepts.

Instead of implying an abstract platform definition from the adopted set of design concepts for platform-independent modelling, it may be useful or even necessary to define the characteristics of an abstract platform explicitly, resulting in one or more separate and reusable design artefacts. We call this approach *explicit abstract platform definition.* During platform-independent modelling, parts of a pre-defined abstract platform model may be composed with the model of the distributed application. For example, although group communication is not a primitive design concept of UML 2.0, it is possible to specify the behaviour of a group communication sub-system using UML2.0. This sub-system is then re-used in the design of a distributed application. Other examples of pre-defined artefacts that may be included in abstract platforms are the ODP trader [12] and the OMG pervasive services [21] (yet to be defined). The set of design concepts of a design language is still relevant in this approach, since the distributed application and the abstract platform model are described in the language.

In both the implicit and explicit abstract platform definition approaches, there is some overlap between language characteristics and abstract platform characteristics. This leads to the formulation of an important requirement for a design language to support platform-independent design: *the concepts underlying the design language should be precisely defined, so that the characteristics of the abstract platform can be unambiguously derived from these concepts.* This is important for at least two reasons: (1) designers need to know the characteristics of the abstract platform when

defining platform-independent models of an application; and (2) abstract platforms are a starting point for platform-specific realization.

Furthermore, a comprehensive MDA design approach should allow designers to select or define suitable abstract platforms for their platform-independent designs. This leads to the formulation of a second requirement for design languages suitable for MDA: *a design language should enable the definition of appropriate levels of platform-independence.*

In [5], we have discussed how the two approaches to the definition of abstract platforms can be supported using MDA standards, namely UML 2.0 [26] and MOF 2.0 [22].

3.3 Transformation Definition

When multiple levels of platform-independence are adopted, successive (automated) transformations may be used that lead to models at lower levels of platform-independence and, ultimately to platform-specific models (i.e., models at the lowest level of platform-independence with respect to a particular definition of platform).
A transformation is straightforward when the selected target platform (either a concrete or an abstract platform) corresponds (directly) to the source abstract platform. When this is not the case, more effort has to be invested in the transformation.

In general, we distinguish two contrasting extreme approaches to proceed with the design step:

1. *Adjust the target platform,* so that it corresponds directly to the abstract platform.
2. *Adjust the (scope of the) application model during transformation, such that* the requirements specified at source platform-independent level are satisfied by the composition of the application model and target platform model.

In approach 1, the boundary between abstract platform and platform-independent application model is preserved during the transformation step. This implies the introduction of some platform-specific *abstract platform logic* to be composed with the target platform. The nature of this composition depends on the particular requirements for the abstract platform. For example, it may be possible to implement abstract platform logic on top of a concrete platform. Nevertheless, this composition may also imply the introduction of platform-specific (e.g., quality-of-service) mechanisms, possibly defined in terms of internal components of the concrete platform. Extension in a non-intrusive manner is often the preferred way to adjust a concrete platform. Techniques that can be used for non-intrusive extension include interceptors [20], aspect-oriented programming and composition filters [8]. In [6], we have presented an example this approach, using CORBA portable interceptors to extend the CORBA platform with dynamic reconfiguration functionality.

Approach 2 may imply the introduction of (e.g., quality-of-service) mechanisms in the model of the application. This approach may be suitable in case it is impossible to adjust the target platform, e.g., due to the cost implications of these adjustments or the lack of extension mechanisms in a concrete platform.

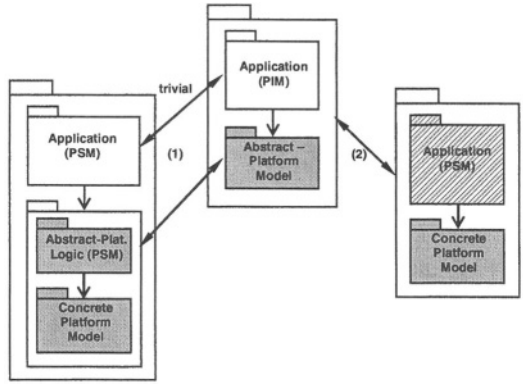


Fig. 3. Alternative approaches to platform-specific realization

Figure 3 illustrates these approaches to transformation. We consider a transformation from platform-independent models to platform-specific models.

Both approaches allow us to target different concrete platforms from the same platform-independent model, with different quality characteristics [2]. Approach 1 can be generalized as a recursive application of service definition (external perspective) and the service's internal design, resulting in a hierarchy of abstract platforms and a concrete target platform. At each step of the recursion, both approaches to transformation can be chosen.

4 Towards a Reference Architecture for Abstract Platform Definition

In MDA development, opportunities for reuse of transformations play an important role in deciding the organization of the execution phase in terms of levels of models and transformations. A single transformation from high-level models to implementations may be costly to develop and is rendered useless in the face of technology platform changes. Given that technology platforms are generally regarded as unstable, it is important to attempt to recognize (intermediate) stable abstract platforms that can be used for a large number of applications. This allows transformations to and from this intermediate abstract platform to be reused.

The proliferation of different abstract platforms reduces the opportunities for large-scale reuse of intermediate models and transformations to and from intermediate models. This calls for the agreement on a small number of abstract platforms that are, to a great extent, application-domain-neutral and platform-independent.

Ideally, a reference architecture with a small set of canonical abstract-platform-elements should be used to compose abstract platforms that suit the needs of particular projects. We intend to define such a reference architecture, based on concepts of the computational viewpoint of the RM-ODP [12]. We believe that using a well-founded reference model (RM-ODP) to refer to abstract platform enables agreement on the concepts for the description of abstract platforms, and may prove to be an initial step towards a comprehensive framework for the definition of abstract

platforms. An initial discussion on the relation between the RM-ODP concepts and the notion of abstract platform can be found in [4].

An example of the composition of abstract platforms can be found in [5], where we have used UML to combine a number of abstract platforms defined both through the implicit and the explicit abstract platform definition approaches.

5 Related Work

The MDA Guide [21] provides some examples of "generic platform types" and mentions briefly the need for a "generic platform model", which "can amount to a specification of a particular architectural style." Nevertheless, the introduction of these concepts is superficial: for example, the term "generic platform" is not even defined explicitly. In our interpretation of that documentation, we position our notion of abstract platform as subsuming that of generic platform. Abstract platforms can have other relevant characteristics in addition to defining a "particular architectural style", as we have shown in section 3.1. Furthermore, we have focussed on providing guidelines for a designer to define and represent these abstract platforms. The MDA Guide also states that a PIM "exhibits a specified degree of platform independence so as to be suitable for use with a number of different platforms of similar type." Our concept of abstract platform defines the degrees of platform independence for a PIM.
Explicit abstract platform definition is comparable to the definition of (the behaviour of) connectors in Architecture Description Languages (ADLs), such as Rapide [15], [16] and Wright [1], when considering exclusively the characteristics of interaction support. While the role of middleware platform characteristics in ADLs have been recognized in [18], approaches to systematically separate and relate platform-independent and platform-specific descriptions have not been proposed in the scope of the work on Software Architecture.

6 Concluding Remarks

While, in the context of MDA, much effort has been invested in meta-modelling [22, 23], language definition [26, 28], model transformation specification [24], and tool support, the methodological implications of platform-independence have been largely overlooked. The objective of the Ph.D. work discussed in this paper is to fill this gap by defining a methodology for model-driven design of distributed applications.

We have argued that the architectural concept of abstract platform should have a prominent role in this design methodology. An abstract platform defines platform characteristics that are considered at the particular level of platform-independence, and may also serve as starting point for platform-specific realization.

Design language concepts and characteristics of abstract platforms are interrelated. Therefore, careful selection of a design language is indispensable for the beneficial exploitation of the PIM/PSM separation and the definition of abstract platforms.

Often, some platform characteristics are assumed implicitly in platform-independent designs. This may lead to PIMs that cannot be reused for different

platforms or it may lead to PIMs that cannot be directly compared and integrated. It may also lead to transformations that cannot be reused. Platform characteristics assumed in platform-independent designs are better understood and controlled by designers if the characteristics of the abstract platform are explicitly represented in abstract platform definitions.

Work-in-progress includes the elaboration of a reference architecture for abstract platform definition and the application of the proposed methodology and reference architecture in a case study. In this case study, we will define a number of related levels of platform-independence and their abstract platforms, as well as the representation of these abstract platforms in UML. In addition, we will define transformations from the abstract platforms to different concrete platforms.

Acknowledgements. Luís Ferreira Pires, Marten van Sinderen and Chris Vissers should be acknowledged for their invaluable contribution to the Ph.D. work described in this paper. This work is part of the Freeband A-MUSE project. Freeband (http://www.freeband.nl) is sponsored by the Dutch government under contract BSIK 03025. This work is also partly supported by the European Commission within the MODA-TEL IST project (http://www.modatel.org).

References

1. Allen, R.J., Garlan, D.: A Formal Basis for Architectural Connection. ACM Transactions on Software Engineering and Methodology, Vol. 6, No. 3 (1997) 213–219
2. Almeida, J. P. A., van Sinderen, M., Ferreira Pires, L., Quartel, D.: A systematic approach to platform-independent design based on the service concept. In: Proceedings 7th IEEE Intl. Enterprise Distributed Object Computing Conference (EDOC 2003). IEEE Computer Society, Los Alamitos, CA (2003) 112–123
3. Almeida, J.P.A., Dijkman, R., van Sinderen, M., Ferreira Pires, L.: On the Notion of Abstract Platform in MDA Development. In: Proc. 8th IEEE Intl. Enterprise Distributed Object Computing Conference (EDOC 2004), IEEE Computer Society, Los Alamitos, CA (2004)
4. Almeida, J.P.A., van Sinderen, M., Ferreira Pires, L.: The role of the RM-ODP Computational Viewpoint Concepts in the MDA approach. In: Proceedings of the 1st European Workshop on Model-Driven Architecture with Emphasis on Industrial Applications (MDA-IA 2004), University of Twente, The Netherlands (2004) 43–51
5. Almeida, J.P.A., Dijkman, R., van Sinderen, M., Ferreira Pires, L.: Platform-independent modelling in MDA: supporting abstract platforms. In: Proceedings Model-Driven Architecture: Foundations and Applications 2004 (MDAFA 2004), Linköping University, Linköping, Sweden (June 2004) 219–233
6. Almeida, J.P.A., van Sinderen, M., Ferreira Pires, L., Wegdam, M.: Platform-independent Dynamic Reconfiguration of Distributed Applications. In: Proceedings IEEE 10th International Workshop on Future Trends in Distributed Computing Systems (FTDCS 2004), Suzhou, China, (May 2004) 286–291
7. Arango, G.: Domain Analysis: from Art Form to Engineering Discipline. ACM SIGSOFT Software Engineering Notes, Vol. 14, No. 3 (1989) 152–159
8. Elrad, T., Filman, R. E., Bader, A. (eds.), Communications of the ACM, Special Section on Aspect-Oriented Programming, Vol. 44, No.10 (2001) 29–97

9. Ferreira Pires, L.: Architectural Notes: a framework for distributed systems development, Ph.D. Thesis. University of Twente, Enschede, the Netherlands (1994)
10. Gavras, A., Belaunde, M., Ferreira Pires, L., Almeida, J.P.A.: Towards an MDA-based development methodology for distributed applications. In: Proceedings of the 1st European Workshop on Model-Driven Architecture with Emphasis on Industrial Applications (MDA-IA 2004), University of Twente, Enschede, The Netherlands (March 2004) 43–51
11. Gray, J., Bapty, T., Neema, S., Schmidt, D.C., Gokhale, A., Natarajan, B.: An Approach for Supporting Aspect-Oriented Domain Modeling. In: Proceedings Generative Programming and Component Engineering (GPCE 2003), Lecture Notes in Computer Science, Vol. 2830, Springer-Verlag (Sept. 2003) 151–168
12. ITU-T / ISO: Open Distributed Processing - Reference Model - All Parts, ITU-T Recommendations X.901, X902, X903, X.904 | ISO/IEC 10746-1, 2, 3,4 (1995)
13. ITU-T / ISO: Open Distributed Processing - Reference Model - Enterprise Language, ITU-TX.911 | ISO/IEC 15414 (2001)
14. ITU-T: Recommendation Z.100 - CCITT Specification and Description Language. International Telecommunications Union (2002)
15. Luckham, D., Kenney, J., Augustin, L., Vera, J., Bryan, D., Mann, W.: Specification and Analysis of System Architecture Using Rapide. IEEE Transactions on Software Engineering, Vol. 21, No. 4 (1995) 336–355
16. Luckham D., Vera, J.: An Event-Based Architecture Definition Language. IEEE Transactions on Software Engineering Vol. 21, No. 9 (1995) 717–734
17. Microsoft Corporation: Microsoft .NET Remoting: A Technical Overview (2001), available at http://msdn.microsoft.com/library/en-us/dndotnet/html/hawkremoting.asp
18. Di Nitto, E., Rosenblum D.: Exploiting ADLs to Specify Architectural Styles Induced by Middleware Infrastructures. In: Proceedings of the 21st International Conference on Software Engineering (ICSE'99). Los Angeles, CA (1999)
19. Object Management Group: Model driven architecture (MDA), ormsc/01-07-01 (2001)
20. Object Management Group: Common Object Request Broker Architecture: Core Specification, Version 3.0, formal/02-12-06 (2002)
21. Object Management Group: MDA-Guide, Version 1.0.1, omg/03-06-01 (2003)
22. Object Management Group: Meta Object Facility (MOF) 2.0 Core Specification, ptc/03-10-04 (2003)
23. Object Management Group: Meta Object Facility (MOF) Specification, Version 1.4, formal/02-04-03 (2002)
24. Object Management Group: MOF 2.0 Query / Views / Transformations RFP, ad/2002-04-10 (2002)
25. Object Management Group: Unified Modelling Language: Object Constraint Language Version 2.0, Draft Adopted Specification, ptc/03-08-08 (2003)
26. Object Management Group: UML 2.0 Superstructure, ptc/03-08-02 (2003)
27. Object Management Group: UML Profile for Enterprise Distributed Object Computing Specification, ptc/02-02-05 (2002)
28. Object Management Group: Unified Modelling Language (UML) Specification: Infrastructure, Version 2.0, ptc/03-09-15 (2003)
29. Sun Microsystems: Java(TM) Message Service Specification Final Release 1.1 (2002)
30. World Wide Web Consortium: SOAP Version 1.2 Part 1: Messaging Framework, W3C Proposed Recommendation (2003), available at http://www.w3.org/TR/soap12-part1
31. World Wide Web Consortium: Web Services Description Language (WSDL) 1.1, W3C Note (2001), available at http://www.w3.org/TR/wsdl

INTEROP NoE: Interoperability Research for Networked Enterprises Applications and Software

Hervé Panetto[1], Monica Scannapieco[2], and Martin Zelm[3]
on behalf of the INTEROP consortium

[1] CRAN UMR 7039, University Henri Poincaré Nancy I, BP 239,
F-54506 Vandoeuvre-les-Nancy, France
Herve.Panetto@cran.uhp-nancy.fr
http://www.interop-noe.org
[2] DIS, Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
monscan@dis.uniroma1.it
[3] Cimosa Association, Gehenbuehlstr.18,
D-70499 Stuttgart, Germany
Martin.Zelm@cimosa.de

Abstract. This paper reports on the goals, rationale and first results of the European INTEROP Network of Excellence concerned with Interoperability Research for Networked Enterprises Applications and Software. The paper presents the interoperability issues that arise between and inside enterprises systems, applications and software, and the different activities of the network to solve problems of fragmentation in this domain.

1 Introduction

World-class competitiveness of European economy strongly depends on enterprises' ability in the near future to concretise networked dynamic organisations massively and rapidly. Today, networked business encounters recurrent difficulties due to the lack of interoperability between enterprise systems. The role of research in the field is to create upstream conditions of technological breakthrough to avoid that enterprise investment be simply pulled by the incremental evolution of IT offer.

Specific research on interoperability of enterprise applications and software does not exist as such at the European level. The purpose of INTEROP Network of Excellence [1] is to set-up a durable European-wide virtual laboratory dedicated to Enterprise Interoperability with both academic and industrial audience. The roadmap for interoperability research emphasises the need for integrating three key research areas: i) Architectures and Platform, ii) Enterprise Modelling and iii) Enterprise Ontologies.

INTEROP aims to extract new knowledge from the integration of these thematic components. INTEROP Work programme deploys collaborative activity with regard to three aims:

R. Meersman et al. (Eds.): OTM Workshops 2004, LNCS 3292, pp. 866–882, 2004.
© Springer-Verlag Berlin Heidelberg 2004

- To integrate the knowledge in Ontology, Enterprise Modelling, and Architectures to give sustainable sense to interoperability,
- To structure the European research community and influence organisations' programmes to achieve critical research mass,
- To animate the community and spread industrially significant research knowledge outside the network.

To ensure efficient industrial impact, in particular through future standardisation, INTEROP-NoE specifically and explicitly interacts with the FP6 IST ATHENA Integrated Project "Advanced Technologies for interoperability of Heterogeneous Enterprise Networks and their Applications". The joint INTEROP-NoE and ATHENA-IP is a common initiative from European key research organisations and industrial actors to push interoperable solutions for networked business to market.

INTEROP-NoE project duration is 3 years from Nov. 2003. The network gathers 50 organizations from 15 countries.

2 Background

In order to produce goods and services more quickly, at lower cost, while maintaining higher levels of quality and customization, enterprises and organizations must be able to adapt to market changes through efficient outsourcing and collaboration strategies.

2.1 Why Interoperability?

Collaborative business requires reliable exchange of commercial, financial and technical data as well. Legacy ERP, SCM, LCM and CRM enterprise applications commonly manage the information required for collaboration, but the software itself was for the most part conceived and programmed to be run within specific organizational boundaries (Fig. 1).

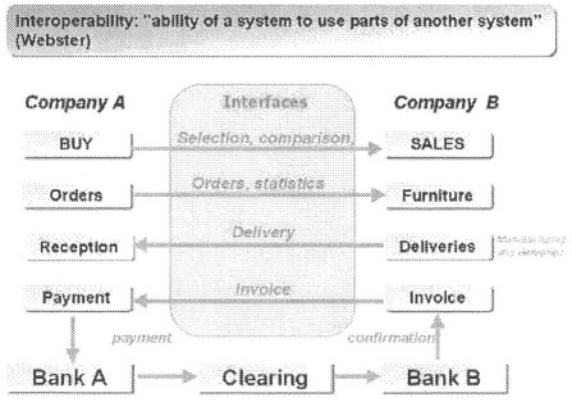


Fig. 1. Interoperability across the enterprise boundaries

Even if many applications use unified technologies (e.g. XML to describe data), business and data models remain heterogeneous. Even as common notions as "order" or "customer" may vary greatly among applications. Despite standardisation efforts, describing and orchestrating business processes across multiple systems is at best a semi-manual process. All this is going on as IT budgets are shrinking and new intrinsically interoperable systems are unfeasible for most IT directors. In Europe, the part of IT budgets due to the lack of application interoperability is up to between 30-40% (Fig. 2).

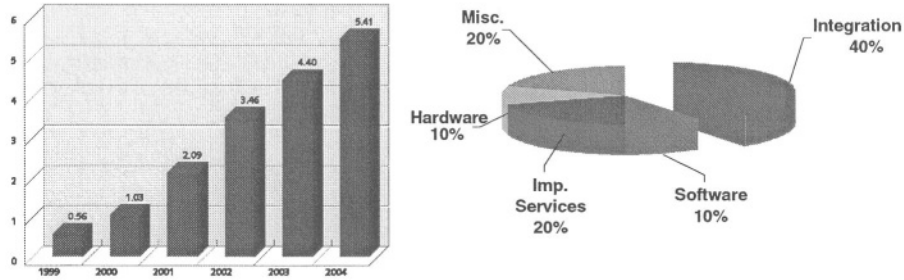


Fig. 2. Application integration licence revenue *(Source: the Yankee Group 2001)* and systems implementation budget

2.2 European Policy for ICT for Business

To intensify business-driven research in Europe, the Sixth Framework Programme supported by the European Commission provides new instruments dedicated to the emergence of critical mass of activities, resources and stakeholders needed to achieve ambitious, clearly defined scientific and technological objectives.
FP6 supports different instruments that concern eBusiness research (Fig. 3):

- *Integrated Projects:*
  - *DBE* (Digital Business Ecosystem) which aims at proving Europe with a recognized advantage in innovative software application development by its SME industry,
  - *ECOLEAD* (European Collaborative Networked Organizations Leaderships Initiative) which aims to create the necessary strong foundations and mechanisms for establishing the most advanced collaborative and network-based industry society in Europe,
  - *TrustCoM* which aims to develop a framework for trust, security and contract management in dynamically-evolving virtual organizations,

- *ATHENA* (Advanced Technologies for interoperability of Heterogeneous Enterprise Networks and their Applications) which aims to contribute to enabling enterprises to seamlessly interoperate with others, by providing reference architectures, methods, and infrastructure components that solve interoperability difficulties.

- *STREPS Projects:*
  - *Spider-win* (Supply Information Dynamic Exchange and Control by Web-based Interaction Network) which aims to achieve context-aware SME co-operation with low-level local software requirements,
  - *Crosswork* (Cross-Organisational Workflow Formation and Enactment) focused on distributed, cross-organisational processes in the automotive industry,
  - No-Rest (Networked Organisations - Research into Standards) which investigates the applicability of standards in the e-business and e-government sectors,
  - *Mytreasury* (The European Treasury Platform) which aims to integrate and demonstrate the technology that will support the delivery of multi-bank, multi-instrument corporate treasury platform,
  - *SATINE* (Semantic-based Interoperability Infrastructure for Integrating Web Service Platforms to Peer-to-Peer Networks) which develops a secure semantic-based interoperability framework for exploiting Web service platforms.

- *Coordination Actions*
  - *V-CES* (Virtual Cost Engineering Studio) which implements, validates and improves a virtual cost engineering studio service,
  - *Co-DesNet* (Supply Networks) which focuses on designing and managing multi-functional multi-agent Collaborative Demand and Supply Networks,
  - *XBRL* (Speeding up the development and adoption of XBRL in Europe) promoting the open standard XBRL for identifying and better communicating the complex financial information in corporate business reports,
  - *DIFFUSE* (Dissemination of InFormal and Formal Useful Specifications and Experiences to research, technology development & demonstration communities) promoting standards and specifications that are suitable for electronic interchange of multimedia data over networks.

One specific action, *VE-FORUM* (The Portal for Projects and Communities In the Virtual Organisation Domain) which provides an integrating portal for all researchers and professionals in the domain of networked businesses and governments.

To this end, *Networks of Excellence* are specifically designed to strengthen scientific and technological excellence by integrating at European level the critical mass of resources and expertise.

INTEROP is the only NoE funded by the European Commission in ICT for Business area. INTEROP-NoE is in charge of providing the set of projects with upstream concepts for Interoperability, with specific synergy with ATHENA Integrated Project.

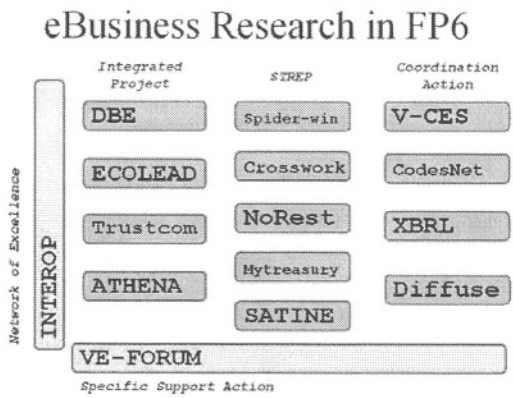


Fig. 3. eBusiness research in FP6-IST

Interoperability is one of FP6 Priority 2 (IST) keywords. The concept originally rose in 2001 as the outcome of a dedicated Working Group supported by DG INFSO e-business unit. FP5 IDEAS Thematic Network (Interoperability Developments for Enterprise Applications and Software Roadmaps, IST-2001-37368) [2] was then in charge of the definition of a roadmap for research in interoperability of enterprise applications and software and FP5 UEML Thematic Network (Unified Enterprise Modelling Language, IST 2001-34229) [3] had the objective to define a common framework and language to enable enterprise models exchange between various enterprise applications. Moreover, the foundations of INTEROP Work programme aims to demonstrate the need for basing Interoperability research on the integration of three key thematic components (Fig. 4):

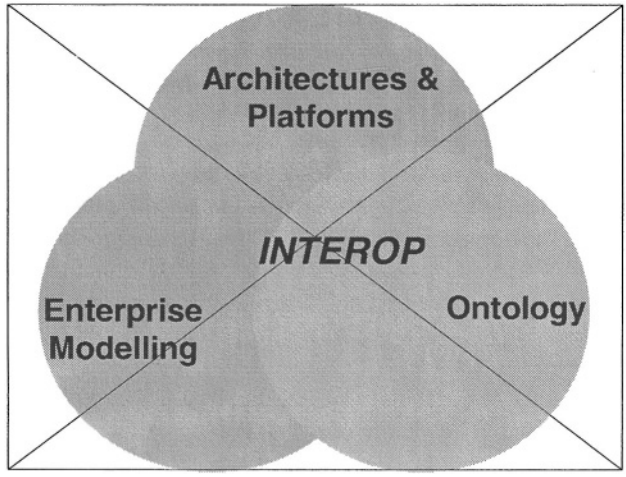


Fig. 4. Knowledge integration for Interoperability research

1. Architectures & Platforms to provide implementation solutions,
2. Enterprise Modelling to define interoperability requirements and to support solution implementation,
3. Enterprise Ontologies to identify interoperability semantics in the enterprise.

Consequently, Interoperability of Enterprise Applications and Software is approached through a knowledge integration process that involves the best European specialists in each of the three components mentioned above. As such, INTEROP project is fully in line with the spirit of FP6 Networks of Excellence.

3 Goals and Objectives

The success of INTEROP's mission will be assessed by measuring the quality rate the three following goals are matched:

1. emergence of a lasting European research community on interoperability of enterprise applications and software;
2. As a research virtual lab on Enterprise application and software interoperability, INTEROP has to create the conditions of an innovative and competitive technology transfer by bringing upstream conceptualisation of business-based interoperability
3. INTEROP has to achieve by the end of the project the integration process which will assemble three knowledge-components (Ontology, Enterprise Modelling, Architectures and Platforms) and prepare a lasting centre of competence on Enterprise Interoperability with maximum research and industrial audience.

These goals are declined in operational objectives, which are provided in the INTEROP Description of Work [1].

4 Joint Program of Activities

In respect with goal #3, there are four kinds of actions concerning: integration, joint research and spreading of excellence, and management of INTEROP (Fig. 5).

4.1 Integrating Activities – IA

INTEROP is not as such a research project but a research integration project. Then, according to goal #3, INTEROP will concentrate partner's efforts on the creation of added-value through the integration of existing knowledge-components. It is the purpose of integrating activities.



Fig. 5. The four kinds of INTEROP activities

4.1.1 IA1. INTEROP Knowledge Map

One of the major objectives of the INTEROP project is to tackle the problem of research fragmentation. Indeed, INTEROP aims at having a long-term integration effect on the research in the field of interoperability and at building long lasting collaborations and mechanisms for coordination of research. One of the tools that INTEROP will put in place to concur to this objective is the INTEROP Knowledge Map (KMap) [4], which is under the responsibility of a specific integrating activity (IA1). The KMap aims at drawing a picture of the status of research in interoperability and to keep this picture up-to-date in the future. Within the first 18 months of the project, a work package (WP1) will create a first version of the KMap with the more specific objective to provide a clear picture of research performed inside the INTEROP project.

The first main objective of the KMap is to support a periodic diagnostic of the extent of research collaboration and coordination among INTEROP partners. This diagnostic will allow formulating recommendations to strengthen this collaboration and better orient the future research efforts of the various NoE partners in order to avoid gaps in research topics, to join efforts of partners working on the same topics and to avoid redundancies of efforts. This will allow improving synergies and collaboration with INTEROP research activities. The KMap will take the form of a database containing data about research activities, results and collaborations within INTEROP (Fig. 6).

The KMap will contain knowledge about existing research and industry institutions, groups and individuals with skills and competencies in interoperability. In order to meaningfully compare elements of this knowledge, they have to be classified according to the different interoperability research domains which they address. For this reason, a classification framework for interoperability research is needed. This common framework will allow the identification of similar research elements coming from the different research communities involved in INTEROP (namely ontology, enterprise modelling and software architectures and platforms).



Fig. 6. The INTEROP Knowledge map

The data requirements define the expected KMap contents. It described the knowledge to be stored about interoperability research, including data about the INTEROP classification framework and its use to classify interoperability research. It was defined in the form of a UML class diagram. This diagram is summarised Fig. 7.

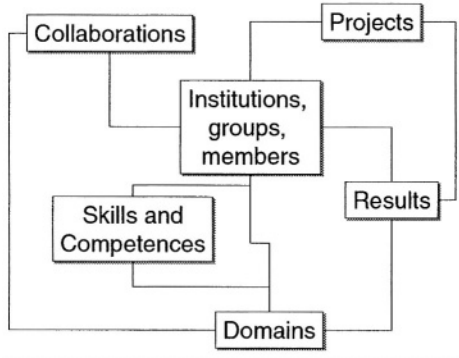


Fig. 7. Summary of KMap data requirements

The KMap will contain knowledge about existing research and industry institutions, groups and individuals with skills and competencies in interoperability. Their collaborations, projects and results (publications, products, software...) will also be documented. In order to meaningfully compare these elements, they have to be classified according to a specific INTEROP classification framework defined as a set of research domains and a number of relationships among them.

4.1.2 IA2. INTEROP Method of Work and Platform Enabling Collaboration, Knowledge Sharing, and Dissemination

The purpose of IA2 is to provide INTEROP with appropriate methods for collaborative work, to capitalise the knowledge added by integration and make it open to all, and to develop common platforms and tools to support the INTEROP Knowledge.

Actions to be processed by IA2 are as follow [5]:

- To make the common body of knowledge produced in INTEROP easily accessible and structured according to domain ontology;
- To disseminate the knowledge by providing training sessions and by sharing network results;
- To accelerate and facilitate the diffusion of the knowledge produced, to support ontology-based semantic searching on the knowledge base and accurate retrieval of the most relevant information, to host and support communities of practice established in INTEROP;
- To support real-time sessions among partners to discuss and enhance knowledge issues, to automatically send notifications on relevant topics to subscribers, to provide training sessions to partners and outside participants;
- To introduce a sustainable platform to gather and increase the scientific and technological excellence produced by INTEROP, to provide the means to disseminate the knowledge beyond the INTEROP partnership by providing training sessions and by sharing network results;
- To establish, customise (if needed), roll-out and maintain a platform based on a common repository which integrates a set of tools to enable real-time collaborative sessions, to share knowledge and to provide training (e-learning) for the entire length of INTEROP.

An analysis of real-time collaboration methods to enable synchronous sessions among geographically distributed work group member have been carried out in order to allow the creation and discussion of topics to improve and refine the knowledge produced in INTEROP. Knowledge management approach analysis has been carried out in order to allow accurate semantic indexing and navigation of the body of knowledge produced in INTEROP in order to guarantee effective sharing and usage/application of information to partners and external participants.

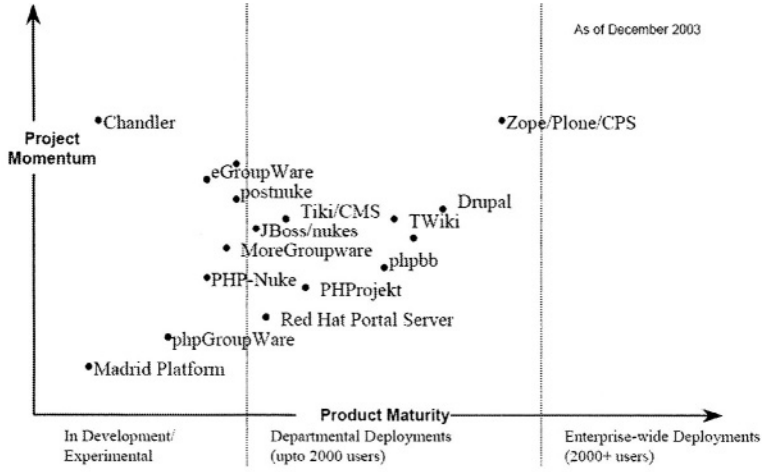


Fig. 8. Open source software map

This activity has identified a set of user requirements in order to select the most suitable tools among those available from previous projects or open source (Fig. 8), or among those available on the market (Fig. 9), described in [5]. These requirements have been classified following these categories: Portal Services, Document Management, Collaboration, Content Management, Project Management, E-learning.

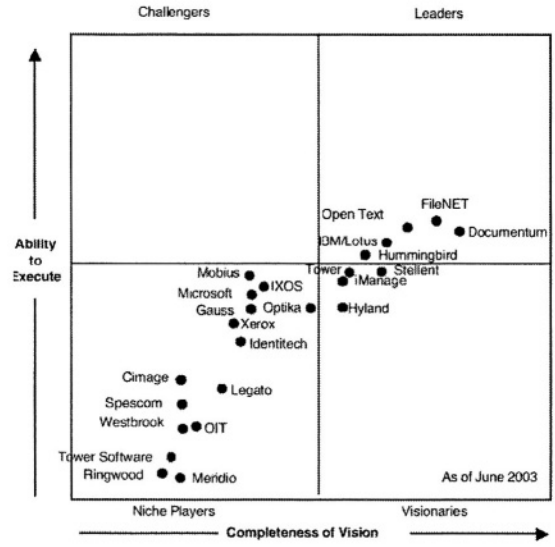


Fig. 9. Commercial Software Map *(Source: Gartner Group)*

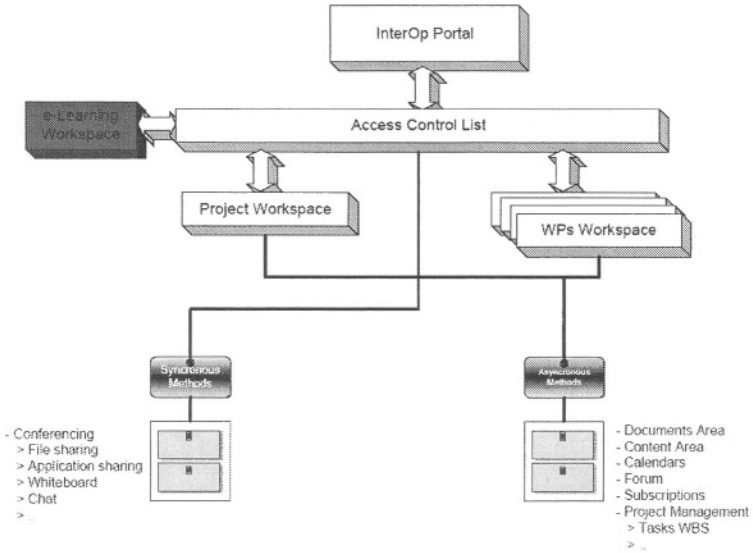


Fig. 10. The INTEROP collaborative platform structure

Taking into account the INTEROP requirements together with financial considerations in terms of licensing costs, the resulting choice has been the open source ZOPE/PLONE which appears more reassuring with a great community of users developing modules and add-on and a lot of companies working on it. The collaborative platform (http://www.interop-noe.org) is structured as shown in Fig. 10.

It is actually used by all INTEROP partners to share documents, knowledge and work. Next stage will be to complete it with some missing requirements and to feed the public part with valuable contents for the world wide scientific community.

4.1.3 IA3. Mobility of Researchers

The mobility of researchers within the virtual European laboratory provided by INTEROP is clearly one of the keys to improve the de-fragmentation of the research resources in the field.

This activity is to be strongly deployed to make the European research area on interoperability a sustainable reality by creating suitable scientific, administrative, financial and practical conditions for scientific stays of both researchers and doctoral students. The mobility of researchers within INTEROP is seen as a major action to achieve sustainable knowledge integration. The scientific stays will be of different nature and duration depending on researcher's needs:

- Researchers are encouraged to spend a significant time in one hosting institution (partner in INTEROP or not, company) for knowledge interchange and common research. Co-written publications should concretise the value added by the integration of expertise.
- Doctoral students are able to spend a period of time in a host laboratory to complete his knowledge on defined topics. Knowledge integration will also be performed through co-directed PhDs once suitable conditions are provided.

4.1.4 IA4. Method for Scientific Integration and Assessment

The objective of IA4 is to manage the performance of the project with implementing performance indicators and make the performance indicators system alive by disseminating the performance among the partners.

The overall objective is to elaborate and demonstrate a domain paradigm based "methodology backbone" for INTEROP, and to apply this methodology for the planning and assessment of INTEROP work.

Results expected from IA4 are as follow:

- To give a conceptual definition of what "scientific integration" is, covering such topics as circulation of ideas, the diffusion of technical and scientific documents and knowledge, researchers mobility, knowledge deployment, and means through which researchers meet and exchange ideas (workshops, tutorials, schools, seminars, conferences, and scientific exchanges and visits);

- To validate the definition in reference to:
  1. the situation pre-existing to INTEROP in term of scientific integration (current operations regarding scientific integration),
  2. the INTEROP work programme and expected results in term of scientific integration,
  3. the gap in-between.
- To specify and formalize scientific integration gaps and assist INTEROP in positioning its various performance criteria in relation to those gaps; in evaluating, fine-tuning or revising the performance criteria to better target gaps; in identifying integration gaps and in prioritizing INTEROP actions regarding these gaps;
- To assist INTEROP in anchoring its future workplans, methodologically and with respect to future measures of scientific integration;
- To assist INTEROP in specifying the architecture of a scientific integration infrastructure, and in migrating INTEROP operations into "INTEROP scientific integration infrastructure operations".

4.2 Joint Research Activities – JRA

According to Goal #3, INTEROP focuses key excellent actors' research efforts on the emergence of an interoperability research corpus through the fusion of three knowledge components which are ontology, enterprise modelling and architectures & platforms.

This is the purpose of joint research activities to study these three knowledge components and to realize such a fusion.

4.2.1 JR1. Common Enterprise Modelling Framework in Distributed Environments

The target of JR1 is to integrate the different aspects available in the state-of-the-art (model data, responsibilities, motivation, knowledge, social aspects, etc.) usually fragmented in different domains of sciences and applications to provide a complete reference model of the distributed enterprise. More specifically, the meta-model resulting from UEML thematic network will be maintained and adjusted with respect to activity IA1.

The mission is to provide, within distributed organisations, a common framework in which modelling, simulation, analysis, management for designing interoperability solutions are explored and integrated.

Distributed organisations (inter and intra) are characterised by distributed modelling and distributed heterogeneous models. Therefore, these distinct models and distributed modelling need clear procedures, standards and easy ways to adapt, build, connect, managing and operate models in heterogeneous, distributed simulation and execution environments, and compositional approaches to analysis of models.

Distributed modelling and distributed heterogeneous models need to be arranged into interoperable components that can be defined by a set of principles bringing

together good design practices and design patterns which allow, according to some modelling objectives, to derive focused solutions.

The framework will be based on a common language for enterprise modelling. More specifically, the meta-model resulting from UEML thematic network will be maintained and adjusted with respect to other INTEROP activities.

The final result will be a framework including:

- a widely accepted Unified Enterprise Modelling Language (UEML) as the base for modelling with its reusable and traceable "Strategy for UEML Extension";
- procedures for model management, model interconnection and model synchronisation;
- design patterns and best practices for interoperability in distributed organisations; showcases illustrating the use and benefits of the framework as well as gabs for further development, also in relation to selected Integrated Projects;
- easy and goal-oriented access to the information within the framework by a interactive role based representation (e.g. Topic Map).

4.2.2 JR2. Generation of Customised Enterprise Software from Enterprise Modelling

JR2 is justified by the need for federating the disparate research approaches to develop enterprise modelling tools and software in relation with international standardization. The purpose of JR2 is to improve modelling techniques to allow interoperability between enterprise models and to facilitate the customisation of suitable software.

Enterprise software tools as Enterprise Resources Planning (ERP), Supply Chain Management (SCM) or Customer Relationship Management (CRM) are today strategic investment for all types of companies. But it must be recognized that the implementation of such Enterprise Software Applications (ESA) is very difficult, takes a long time, cost a lot of money and introduces a lot of disturbances inside the company.

In addition, the customization of such ESA is not easy and sometimes leads the enterprise to modify strongly its organization and decrease its performances.

The last but very important point is the issue of the interoperability of such ESA in various situations such as:

- introduction of a new type of ESA in the enterprise and connection on the existing ones,
- connection of ESA between several enterprises,
- merge of enterprises and preparation of the merging by using enterprise modelling techniques,...

To answer this problematic, the long term goal of JR2 is to generate customized Enterprise Software Applications (ESA) starting from the model of the enterprise at a conceptual level and using what a so-call "ESA generator " (Fig. 11).

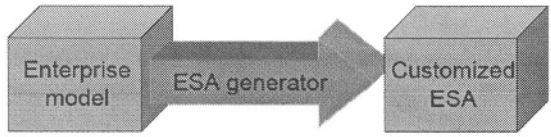


Fig. 11. The ESA generation principle

4.2.3 JR3. Ontology-Based Integration of Enterprise Modelling and Architecture and Platforms

As a support for activity IA1, the purpose is to define the paradigm of "interoperability for enterprise applications and software" from the three research perspectives already mentioned: enterprise modelling, ontology and architectures & Platforms. The
output of JR3 will be a complete approach to make interoperability operational, including research, methodological and in-practice points of view.

Architectures for achieving interoperability are built up on the base of interoperability definition given by goal #1 and include automatic support, cooperation models, non ambiguous common understanding of what is exchanged and task goals.

The nowadays widely distributed and obviously heterogeneous enterprise living and operating environment (variety of "enterprise cultures", business rules, supporting technology, dynamics of enterprise structures through new alliances or enterprise merging and acquisition, etc.) increases the needs for making explicit (i.e. modelling) enterprise structure and organization, enterprise processes, intra and inter-enterprise communication and so on, while ensuring fast enterprise adaptation to the changing environment together with preserving enterprise privacy, autonomy and communication security.

Founding enterprise modelling and support on commonly agreed enterprise ontology will obviously impact modelling techniques and will improve intra and inter-enterprise entities connectivity. Moreover, it will very probably impact the underlying supporting architecture and technology by making them more knowledge-based.

JR3 will obviously consider concepts and tools for enterprise ontology management (ontology modelling, elaboration, exploitation and maintenance).

4.2.4 JR4. New Architectures and Platforms for Interoperability

This research area is focusing on harmonizing and synthesizing existing research around new flexible and adaptive architectures for interoperability, - such as the model driven approach, service-oriented architecture approaches, peer-2-peer architectures, agent architectures and federated architectures.

The focus will be on both functional and non-functional aspects of interoperability. A common framework in terms of an architectural reference model and common description techniques will be a vehicle for unification and common understanding of the advantages and disadvantages of the various architectural approaches.

In particular a focus will be put on the possibilities for mappings and transformations from and to the enterprise model level and the usage of ontologies to support

semantic interoperability and for a shared usage of tools and methodologies for this purpose.

4.3 Spreading of Excellence Activities – SEA

With respect to goals #2 and #3, INTEROP should ensure the fertilisation of the largest research community. This is the purpose of Spreading of Excellence activities.

4.3.1 SE1. Training by E-learning

The aim of SE1 is to provide training facilities to spread the knowledge added by INTEROP inside (between members) and outside the network, during and beyond the EU-funded period. Training activities have been differentiated in internal and external, addressed to researchers, PhD students or enterprises and use suitable e-techniques.

A big challenge inside SE1 is the creation and co-ordination of teaching programmes. A two-level curriculum is proposed:

- a specialized master on enterprise modelling, architecture & platform and ontologies supported by e-learning methodologies,
- a European PhD programme (graduate) on interoperability based on exchange of teachers between European universities and e-learning methodologies.

Another task is to create a structure in order to gather materials concerning interoperability and put them at the disposal of researchers and teachers:

- virtual Library of papers about interoperability (Interoperability Explicit Knowledge Repository-IEKR),
- tutorials (interactive courses on CDROM) and courses on the web about interoperability,
- best practices applying interoperability,
- glossaries on interoperability matters in order to unify semantics.

4.3.2 SE2. Dissemination and Communication

A special effort will be devoted to the external communication and knowledge dissemination to increase awareness on interoperability. The activity both addresses the policy and means required [6].

This activity aims to structure the networked laboratory in terms of communication and dissemination of the results within INTEROP itself and outside INTEROP in the European research community.

To gain the full benefit, the work done by INTEROP must be accepted by a wide community of researchers, tool developers and users of interoperability solutions concerning both process centred organisational and technical enterprise integration.

Acceptance and awareness will be significantly leveraged if it is largely supported by the academic world and standardization bodies.

Therefore, results expected from SE2 are as follow [6]:

- structure of the networked laboratory in term of communication and dissemination of results within INTEROP itself and outside INTEROP within the European research community,
- increase of awareness on technology for organisational interoperability with emphasis on enterprise modelling,
- definition of a common consortium strategy for the dissemination and evaluation of the network results,
- provision of means for a wide dissemination of and opportunities in interoperation of markets, industries and research,
- identification and evaluation of interoperability requirements of software applications with users and standardisation bodies, linked with the JRA activities.

4.3.3 SE3. Transfer of Research to Industry

An Interoperability Laboratory Network (ILN) will be created to promote Interoperability services towards SMEs. ILN staff will be composed by experts with backgrounds in research and in industry business. Studies on economic potential impact and business opportunities will be led. Take-up actions will also result from this generic activity [7].

In order to spread the adoption of interoperability technology in SME's, INTEROP will provide an instrument to stimulate the transfer of research results to the industrial world.

In this way, INTEROP will found an Interoperability Laboratory Network (ILN) whose goal is to provide to SME's support their technological needs related to interoperability technologies and know-how in their own countries. The efficiency of ILN will be ensured thanks to the wide geographical area covered with the different partners in Europe, and the close links with enterprises and universities.

ILN will act service providers for SME wishing to implement the interoperability technology innovation results of INTEROP research activities. ILN will be operated by INTEROP partners by means of national organisations (universities and research centres) with established reputations in national research and industrial communities. ILN staff will be composed by experienced specialists with backgrounds in research as well as in industry business.

5 Conclusions

Competition is changing the way organisations are managed and interact in dynamic global markets. An organisation's ability to enter into commercial and functional alliances more quickly under better conditions with partners to execute contracts presents a commercial advantage. However from a technological as well as business process point of view, there are numerous gaps between the existing paradigms and the comprehensive interoperable systems required to enable true networked enterprises. The shift from the total integrated approach to interoperability development is

not only a technical change, but reflects organisational, economical and social trends and requirements. To successfully elaborate on this very complex endeavour, it is necessary to develop research involving knowledge and competencies of all domains concerned. The originality of the European approach is to integrate the three technical domains Enterprise Modelling, Architectures and Platforms as well as Ontology which contribute strongly to developing interoperability in networked enterprises.

The INTEROP approach of bringing together leading academics, research centres and industrial stakeholders is considered as a first step towards a multidisciplinary research (not only technical, but also social and economical) leading to a sustainable re-organisation of the research activities in Europe and to fruitful international co-operations.

Acknowledgements. This work is supported by the Commission of the European Communities under the sixth framework programme (INTEROP Network of Excellence, Contract N° 508011, <http://www.interop-noe.org>). The authors would like to thank all members of the INTEROP NoE project for their scientific contributions to this work.

References

1. INTEROP NoE: IST-508 011 Network of Excellence, Interoperability Research for Networked Enterprises Applications and Software, http://www.interop-noe.org, (2003)
2. Chen, D., Doumeingts, D.: European Initiatives to develop interoperability of enterprise applications - basic concepts, framework and roadmap. Journal of Annual reviews in Control, 27 (3), 151-160, Elsevier, December (2003)
3. Panetto, H., Berio, G., Benali, K., Boudjlida, N., Petit, M.: A Unified Enterprise Modelling Language for enhanced interoperability of Enterprise Models. Proceedings of the 11th IFAC INCOM2004 Symposium, April 5th-7th, (2004), Bahia, Brazil
4. Petit, M., et al.: Knowledge map of research in interoperability in the INTEROP NoE, Deliverable D1.1, INTEROP IST-508 011 Network of Excellence, August 2004, http://www.interop-noe.org
5. Ippolito, M., et al.: Collaboration methods of work and Dissemination methods, Deliverable D2.1, INTEROP IST-508 011 Network of Excellence, March 2004, http://www.interop-noe.org
6. Panetto, H., Zelm M., et al.: Plan for using and disseminating knowledge, Deliverable D11.1, INTEROP IST-508 011 Network of Excellence, August 2004, http://www.interop-noe.org
7. Doumeingts, G., et al.: The Methodology to implement services and develop take up actions towards SMEs, Deliverable D12.1, INTEROP IST-508 011 Network of Excellence, March 2004, http://www.interop-noe.org

Author Index

Adam, Otmar 483
Afsarmanesh, Hamideh 14
Albani, Antonia 418, 556, 778
Almeida, João Paulo A. 854
Aloisio, Giovanni 211
Alonso, Diego 246
Álvarez, Bárbara 246
Alvaro, Alexandre 38
Angelova, Galia 700
Arana, Jesus 741
Ardaiz, Oscar 187
Aufaure, Marie-Aude 30
Ayed, Dhouha 36

Bacon, Jean 1, 198
Bagic, Marina 842
Baiardi, Fabrizio 123
Balette, Gerd 758
Bartolini, Roberto 593
Basanta-Val, Pablo 359
Batista, Thais 34
Bauer, Bernhard 544
Bazijanec, Bettina 556
Becker, Leandro B. 292
Beltrán, Marta 135
Benmerzoug, Djamel 495
Bennacer, Nacera 30
Bernard, Guy 36
Bignon, J. Claude 3
Böhme, Helge 303
Borg, Andrew 315
Bosque, Jose L. 135
Bouattour, Mohamed 3
Boufaida, Mahmoud 495
Boufaida, Zizette 495
Brasher, Andrew 723
Butner, Scott 26

Caballé, Santi 175
Cafaro, Massimo 211
Cai, Hao 278
Calsavara, Alcides 46
Cannataro, Mario 232
Cánovas Reverte, Óscar 165
Chao, Isaac 187
Choi, Daesik 7
Choi, Jaeyoung 222
Choi, Yongsun 17
Comito, Carmela 88
Coppens, Frederick 605
Corsaro, Angelo 244, 394
Cotroneo, Domenico 44
Cowell, Andrew 26
Cullot, Nadine 30
Cuzzocrea, Alfredo 144

da Costa, Juarez 46
Dang Tran, Frédéric 256
Daradoumis, Thanasis 175
de Almeida, Eduardo Santana 38
De Backer, Manu 810
Decouchant, Dominique 12
De Leenheer, Pieter 798
Delgado, Jaime 581
De Nicola, Antonio 773
Dietz, Jan L.G. 533
di Flora, Cristiano 383
Di Stefano, Antonella 406
do Prado, Antonio Francisco 38
Duval, Erik 689,713

Eder, Johann 662
Estévez-Ayres, Iria 359

Fargetta, Marco 406
Fassino, Jean-Philippe 256
Favela, Jesus 12
Ferrarini, Luca 346
Fessl, Kurt 23
Ficco, Massimo 383
Fiore, Sandro 211
Fomkin, Ruslan 99
Freisleben, Bernd 40
Furfaro, Filippo 144

Gangemi, Aldo 568, 674
Gans, Günter 520
García Clemente, Félix J. 165
Garcia-Valls, Marisol 359
Gensel, Jérôme 9
Golze, Ulrich 303

Gómez-Pérez, Jose Manuél 5
Gómez Skarmeta, Antonio F. 165
Guelfi, Nicolas 468
Guevara-Masis, Victor 14
Guzmán, Antonio 135
Guzzi, Pietro Hiram 232

Habing, Nathalie 533
Halin, Gilles 3
Halstenberg, Silke 59
Hassas, Salima 741
Heo, Sung-Phil 32
Herrero, Pilar 48, 111
Hertzberger, Louis O. 14
Higuera-Toledano, M. Teresa 268
Hofer, Anja 483
Hofreiter, Birgit 507
Holze, Marc 19
Huedo, Eduardo 50
Huemer, Christian 507
Hwang, Seogchan 222

Jablonski, Stefan 19
Jarke, Matthias 520
Jarrar, Mustafa 568

Kalaydjiev, Ognian 700
Kammüller, Reiner 40
Kang, Dazhou 731, 763
Kerremans, Koen 605
Kim, Jangha 7
Kim, Kwanghoon 419
Kim, Sangwook 7
Kingston, John 605
Kirsch-Pinheiro, Manuele 9
Klerkx, Joris 713
Koehler, Jana 444
Kolb, Carlos 46
Koncilia, Christian 662
Krieg-Brückner, Bernd 753
Kwon, Jagun 333

Lakemeyer, Gerhard 520
Le Cousin, Guillaume 468
Lee, Jee Yeon 621
Lee, Kanghee 7
Lee, Keon Myung 21
Lee, Sang Ho 21
Lenci, Alessandro 593
Lepinette, Alain 50
Li, Yanhui 731, 763
Liebig, Thorsten 28
Llorente, Ignacio M. 50
Llorente, Silvia 581
Lobry, Olivier 256
Lu, Jianjiang 731, 763
Lucrédio, Daniel 38
Lüder, Arndt 346

Mahnke, Achim 753
Mami, Najla 79
Martin, Hervé 9
Martinelli, Fabio 123
Martínez Enríquez, Ana María 12
Martínez Pérez, Gregorio 165
Mazza, Tommaso 232
Mazzeo, Giuseppe M. 144
McAndrew, Patrick 723
Medeiros, Adelardo 34
Meersman, Robert 605
Mehandjiev, Nikolay 23
Meire, Michael 713
Melz, Eric 652
Mendoza, Sonia 12
Migliaccio, Armando 44
Mirto, Maria 211
Missikoff, Michele 773
Montemagni, Simonetta 593
Montero, Rubén S. 50
Morán, Alberto L. 12
Moreau, Rachel 758
Mori, Paolo 123
Müller, Jörg P. 544
Muñoz-Hernández, Susana 5

Najjar, Faiza 79
Najjar, Jehad 713
Noda, Agnaldo K. 46
Novatnack, John 444
Nunes, Leonardo 46

Paal, Stefan 40
Paik, Woojin 621
Palinginis, Alexios 778
Panetto, Hervé 866
Pappalardo, Giuseppe 406
Pastor, Juan A. 246
Paulson, Patrick 26
Peña, José M. 111
Pereira, Carlos Eduardo 292

Pérez, María S. 48, 111
Petrov, Ilia 19
Pirrelli, Vito 593
Prié, Yannick 741
Pulou, Jacques 256

Quemada, Juan 690

Radenkovic, Milena 48
Ries, Benoît 468
Risch, Tore 99
Rivera, Germán M. 690
Rivierre, Nicolas 256
Robles, Víctor 48, 111
Rodríguez, Eva 581
Roser, Stephan 544
Roth, Bram 634
Roumani, Ali M. 154
Russo, Stefano 44, 383
Ryba, Michael 456
Ryoo, Hyukjae 419

Sabri, Nawel 36
Saccà, Domenico 144
Sagri, Maria-Teresa 674
Salim, Diogo 34
Salvachúa, Joaquín 690
Sánchez, Alberto 111
Sangüesa, Ramon 187
Santoro, Corrado 244, 394
Scannapieco, Monica 866
Schiappelli, Frederica 773
Schmidt, Andreas 779
Schmitz, Dominik 520
Schoeberl, Martin 371
Schreurs, Jeanne 758
Schwab, Christian 346
Schwiderski-Grosche, Scarlet 68
Seo, Jungtaek 7
Simon, Bernd 690
Skillicorn, David B. 154
Slimani, Yahya 79
Smrž, Pavel 768
Soffer, Avi 787
Song, Byungwook 7
Soria, Claudia 593
Sotnykova, Anastasiya 30
Specht, Günther 28
Spyns, Peter 689, 778
Stalker, Iain 23
Strintzis. Michael G. 569
Strupchanska, Albena 700
Stucky, Karl-Uwe 59
Subieta, Kazimierz 822
Süß, Wolfgang 59

Taconet, Chantal 36
Talia, Domenico 88
Tangermann, Marcus 346
Temmerman, Rita 605
Thomson, Judi 26
Tiscornia, Daniela 674
Tomlinson, Allan 68
Tramontana, Emiliano 406
Trzaska, Mariusz 822
Tsampoulatidis, loannis 569
Turowski, Klaus 418, 556
Tzovaras, Dimitrios 569

Vaccarelli, Anna 123
Valente, Andre 652
Vangenot, Christelle 30
Vázquez, Luis 50
Veber, Carlo 346
Veltri, Pierangelo 232
Verbert, Katrien 713
Verheij, Bart 634
Verlinden, Ruben 605
Villanova-Oliver, Marlène 9

Wang, Peng 731, 763
Wehrmeister, Marco A. 292
Weichhart, Georg 23
Weithöner, Timo 28
Wellings, Andy 278, 315, 333
Whiting, Mark 26
Wickler, Gerhard 456
Winnewisser, Christian 556
Won, Jaekang 419

Xhafa, Fatos 175
Xu, Baowen 731, 763

Yoneki, Eiko 1, 198

Zaha, Johannes Maria 834
Zang, Sven 483
Zdravkovic, Jelena 431
Zelm, Martin 866
Zhao, Gang 605
Zhao, J. Leon 17
Zhou, Nan 175

This page intentionally left blank

Lecture Notes in Computer Science

For information about Vols. 1–3192

please contact your bookseller or Springer

Vol. 3305: P.M.A. Sloot, B. Chopard, A.G. Hoekstra (Eds.), Cellular Automata. XV, 883 pages. 2004.

Vol. 3293: C.-H. Chi, M. van Steen, C. Wills (Eds.), Web Content Caching and Distribution. IX, 283 pages. 2004.

Vol. 3292: R. Meersman, Z. Tari, A. Corsaro et al. (Eds.), On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops. XXIII, 885 pages. 2004.

Vol. 3291: R. Meersman, Z. Tari et al. (Eds.), On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE. Part II. XXV, 824 pages. 2004.

Vol. 3290: R. Meersman, Z. Tari et al. (Eds.), On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE. Part I. XXV, 823 pages. 2004.

Vol. 3287: A. Sanfeliu, J.F.M. Trinidad, J.A. Carrasco Ochoa (Eds.), Progress in Pattern Recognition, Image Analysis and Applications. XVII, 703 pages. 2004.

Vol. 3286: G. Karsai, E. Visser (Eds.), Generative Programming and Component Engineering. XIII, 491 pages. 2004.

Vol. 3284: A. Karmouch, L. Korba, E.R.M. Madeira (Eds.), Mobility Aware Technologies and Applications. XII, 382 pages. 2004.

Vol. 3280: C. Aykanat, T. Dayar, İ. Körpeoğlu (Eds.), Computer and Information Sciences - ISCIS 2004. XVIII, 1009 pages. 2004.

Vol. 3274: R. Guerraoui (Ed.), Distributed Computing. XIII, 465 pages. 2004.

Vol. 3273: T. Baar, A. Strohmeier, A. Moreira, S.J. Mellor (Eds.), <<UML>> 2004 - The Unified Modelling Language. XIII, 454 pages. 2004.

Vol. 3271: J. Vicente, D. Hutchison (Eds.), Management of Multimedia Networks and Services. XIII, 335 pages. 2004.

Vol. 3270: M. Jeckle, R. Kowalczyk, P. Braun (Eds.), Grid Services Engineering and Management. X, 165 pages. 2004.

Vol. 3269: J. López, S. Qing, E. Okamoto (Eds.), Information and Communications Security. XI, 564 pages. 2004.

Vol. 3266: J. Solé-Pareta, M. Smirnov, P.V. Mieghem, J. Domingo-Pascual, E. Monteiro, P. Reichl, B. Stiller, R.J. Gibbens (Eds.), Quality of Service in the Emerging Networking Panorama. XVI, 390 pages. 2004.

Vol. 3265: R.E. Frederking, K.B. Taylor (Eds.), Machine Translation: From Real Users to Research. XI, 392 pages. 2004. (Subseries LNAI).

Vol. 3264: G. Paliouras, Y. Sakakibara (Eds.), Grammatical Inference: Algorithms and Applications. XI, 291 pages. 2004. (Subseries LNAI).

Vol. 3263: M. Weske, P. Liggesmeyer (Eds.), Object-Oriented and Internet-Based Technologies. XII, 239 pages. 2004.

Vol. 3262: M.M. Freire, P. Chemouil, P. Lorenz, A. Gravey (Eds.), Universal Multiservice Networks. XIII, 556 pages. 2004.

Vol. 3261: T. Yakhno (Ed.), Advances in Information Systems. XIV, 617 pages. 2004.

Vol. 3260: I.G.M.M. Niemegeers, S.H. de Groot (Eds.), Personal Wireless Communications. XIV, 478 pages. 2004.

Vol. 3258: M. Wallace (Ed.), Principles and Practice of Constraint Programming – CP 2004. XVII, 822 pages. 2004.

Vol. 3257: E. Motta, N.R. Shadbolt, A. Stutt, N. Gibbins (Eds.), Engineering Knowledge in the Age of the Semantic Web. XVII, 517 pages. 2004. (Subseries LNAI).

Vol. 3256: H. Ehrig, G. Engels, F. Parisi-Presicce, G. Rozenberg (Eds.), Graph Transformations. XII, 451 pages. 2004.

Vol. 3255: A. Benczúr, J. Demetrovics, G. Gottlob (Eds.), Advances in Databases and Information Systems. XI, 423 pages. 2004.

Vol. 3254: E. Macii, V. Paliouras, O. Koufopavlou (Eds.), Integrated Circuit and System Design. XVI, 910 pages. 2004.

Vol. 3253: Y. Lakhnech, S. Yovine (Eds.), Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems. X, 397 pages. 2004.

Vol. 3252: H. Jin, Y. Pan, N. Xiao, J. Sun (Eds.), Grid and Cooperative Computing - GCC 2004 Workshops. XVIII, 785 pages. 2004.

Vol. 3251: H. Jin, Y. Pan, N. Xiao, J. Sun (Eds.), Grid and Cooperative Computing - GCC 2004. XXII, 1025 pages. 2004.

Vol. 3250: L.-J. (LJ) Zhang, M. Jeckle (Eds.), Web Services. X, 301 pages. 2004.

Vol. 3249: B. Buchberger, J.A. Campbell (Eds.), Artificial Intelligence and Symbolic Computation. X, 285 pages. 2004. (Subseries LNAI).

Vol. 3246: A. Apostolico, M. Melucci (Eds.), String Processing and Information Retrieval. XIV, 332 pages. 2004.

Vol. 3245: E. Suzuki, S. Arikawa (Eds.), Discovery Science. XIV, 430 pages. 2004. (Subseries LNAI).

Vol. 3244: S. Ben-David, J. Case, A. Maruoka (Eds.), Algorithmic Learning Theory. XIV, 505 pages. 2004. (Subseries LNAI).

Vol. 3243: S. Leonardi (Ed.), Algorithms and Models for the Web-Graph. VIII, 189 pages. 2004.

Vol. 3242: X. Yao, E. Burke, J.A. Lozano, J. Smith, J.J. Merelo-Guervós, J.A. Bullinaria, J. Rowe, P. Tiňo, A. Kabán, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature - PPSN VIII. XX, 1185 pages. 2004.

Vol. 3241: D. Kranzlmüller, P. Kacsuk, J.J. Dongarra (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface. XIII, 452 pages. 2004.

Vol. 3240:1. Jonassen, J. Kim (Eds.), Algorithms in Bioinformatics. IX, 476 pages. 2004. (Subseries LNBI).

Vol. 3239: G. Nicosia, V. Cutello, P.J. Bentley, J. Timmis (Eds.), Artificial Immune Systems. XII, 444 pages. 2004.

Vol. 3238: S. Biundo, T. Frühwirth, G. Palm (Eds.), KI 2004: Advances in Artificial Intelligence. XI, 467 pages. 2004. (Subseries LNAI).

Vol. 3236: M. Núñez, Z. Maamar, F.L. Pelayo, K. Pousttchi, F. Rubio (Eds.), Applying Formal Methods: Testing, Performance, and M/E-Commerce. XI, 381 pages. 2004.

Vol. 3235: D. de Frutos-Escrig, M. Nunez (Eds.), Formal Techniques for Networked and Distributed Systems – FORTE 2004. X, 377 pages. 2004.

Vol. 3232: R. Heery, L. Lyon (Eds.), Research and Advanced Technology for Digital Libraries. XV, 528 pages. 2004.

Vol. 3231: H.-A. Jacobsen (Ed.), Middleware 2004. XV, 514 pages. 2004.

Vol. 3230: J.L. Vicedo, P. Martínez-Barco, R. Muñoz, M. Saiz Noeda (Eds.), Advances in Natural Language Processing. XII, 488 pages. 2004. (Subseries LNAI).

Vol. 3229: J.J. Alferes, J. Leite (Eds.), Logics in Artificial Intelligence. XIV, 744 pages. 2004. (Subseries LNAI).

Vol. 3226: M. Bouzeghoub, C. Goble, V. Kashyap, S. Spaccapietra (Eds.), Semantics for Grid Databases. XIII, 326 pages. 2004.

Vol. 3225: K. Zhang, Y. Zheng (Eds.), Information Security. XII, 442 pages. 2004.

Vol. 3224: E. Jonsson, A. Valdes, M. Almgren (Eds.), Recent Advances in Intrusion Detection. XII, 315 pages. 2004.

Vol. 3223: K. Slind, A. Bunker, G. Gopalakrishnan (Eds.), Theorem Proving in Higher Order Logics. VIII, 337 pages. 2004.

Vol. 3222: H. Jin, G.R. Gao, Z. Xu, H. Chen (Eds.), Network and Parallel Computing. XX, 694 pages. 2004.

Vol. 3221: S. Albers, T. Radzik (Eds.), Algorithms – ESA 2004. XVIII, 836 pages. 2004.

Vol. 3220: J.C. Lester, R.M. Vicari, F. Paraguaçu (Eds.), Intelligent Tutoring Systems. XXI, 920 pages. 2004.

Vol. 3219: M. Heisel, P. Liggesmeyer, S. Wittmann (Eds.), Computer Safety, Reliability, and Security. XI, 339 pages. 2004.

Vol. 3217: C. Barillot, D.R. Haynor, P. Hellier (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2004. XXXVIII, 1114 pages. 2004.

Vol. 3216: C. Barillot, D.R. Haynor, P. Hellier (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2004. XXXVIII, 930 pages. 2004.

Vol. 3215: M.G.. Negoita, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems. LVII, 906 pages. 2004. (Subseries LNAI).

Vol. 3214: M.G.. Negoita, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems. LVIII, 1302 pages. 2004. (Subseries LNAI).

Vol. 3213: M.G.. Negoita, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems. LVIII, 1280 pages. 2004. (Subseries LNAI).

Vol. 3212: A. Campilho, M. Kamel (Eds.), Image Analysis and Recognition. XXIX, 862 pages. 2004.

Vol. 3211: A. Campilho, M. Kamel (Eds.), Image Analysis and Recognition. XXIX, 880 pages. 2004.

Vol. 3210: J. Marcinkowski, A. Tarlecki (Eds.), Computer Science Logic. XI, 520 pages. 2004.

Vol. 3209: B. Berendt, A. Hotho, D. Mladenic, M. van Someren, M. Spiliopoulou, G. Stumme (Eds.), Web Mining: From Web to Semantic Web. IX, 201 pages. 2004. (Subseries LNAI).

Vol. 3208: H.J. Ohlbach, S. Schaffert (Eds.), Principles and Practice of Semantic Web Reasoning. VII, 165 pages. 2004.

Vol. 3207: L.T. Yang, M. Guo, G.R. Gao, N.K. Jha (Eds.), Embedded and Ubiquitous Computing. XX, 1116 pages. 2004.

Vol. 3206: P. Sojka, I. Kopecek, K. Pala (Eds.), Text, Speech and Dialogue. XIII, 667 pages. 2004. (Subseries LNAI).

Vol. 3205: N. Davies, E. Mynatt, I. Siio (Eds.), UbiComp 2004: Ubiquitous Computing. XVI, 452 pages. 2004.

Vol. 3204: C.A. Peña Reyes, Coevolutionary Fuzzy Modeling. XIII, 129 pages. 2004.

Vol. 3203: J. Becker, M. Platzner, S. Vernalde (Eds.), Field Programmable Logic and Application. XXX, 1198 pages. 2004.

Vol. 3202: J.-F. Boulicaut, F. Esposito, F. Giannotti, D. Pedreschi (Eds.), Knowledge Discovery in Databases: PKDD 2004. XIX, 560 pages. 2004. (Subseries LNAI).

Vol. 3201: J.-F. Boulicaut, F. Esposito, F. Giannotti, D. Pedreschi (Eds.), Machine Learning: ECML 2004. XVIII, 580 pages. 2004. (Subseries LNAI).

Vol. 3199: H. Schepers (Ed.), Software and Compilers for Embedded Systems. X, 259 pages. 2004.

Vol. 3198: G.-J. de Vreede, L.A. Guerrero, G. Marín Raventós (Eds.), Groupware: Design, Implementation and Use. XI, 378 pages. 2004.

Vol. 3196: C. Stary, C. Stephanidis (Eds.), User-Centered Interaction Paradigms for Universal Access in the Information Society. XII, 488 pages. 2004.

Vol. 3195: C.G. Puntonet, A. Prieto (Eds.), Independent Component Analysis and Blind Signal Separation. XXIII, 1266 pages. 2004.

Vol. 3194: R. Camacho, R. King, A. Srinivasan (Eds.), Inductive Logic Programming. XI, 361 pages. 2004. (Subseries LNAI).

Vol. 3193: P. Samarati, P. Ryan, D. Gollmann, R. Molva (Eds.), Computer Security – ESORICS 2004. X, 457 pages. 2004.