

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

University of California, Los Angeles, CA, 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

John F. Roddick V. Richard Benjamins
Samira Si-Saïd Cherfi Roger Chiang
Christophe Claramunt Ramez Elmasri
Fabio Grandi Hyoil Han Martin Hepp
Miltiadis Lytras Vojislav B. Mišić
Geert Poels Il-Yeol Song Juan Trujillo
Christelle Vangenot (Eds.)

Advances in Conceptual Modeling – Theory and Practice

ER 2006 Workshops BP-UML, CoMoGIS
COSS, ECDM, OIS, QoIS, SemWAT
Tucson, AZ, USA, November 6-9, 2006
Proceedings



Springer

Volume Editors

John F. Roddick E-mail: roddick@infoeng.flinders.edu.au
V. Richard Benjamins E-mail: rbenjamins@isoco.com
Samira Si-Saïd Cherfi E-mail: sisaid@cnam.fr
Roger Chiang E-mail: roger.chiang@uc.edu
Christophe Claramunt E-mail: claramunt@ecole-navale.fr
Ramez Elmasri E-mail: elmasri@cse.uta.es
Fabio Grandi E-mail: fgrandi@deis.unibo.it
Hyoil Han E-mail: hyoil.han@ischool.drexel.edu
Martin Hepp E-mail: martin.hepp@deri.org
Miltiadis Lytras E-mail: mdl@aueb.gr
Vojislav B. Mišić E-mail: vmisic@cs.umanitoba.ca
Geert Poels E-mail: Geert.Poels@UGent.be
Il-Yeol Song E-mail: songiy@drexel.edu
Juan Trujillo E-mail: jtrujillo@dlsi.ua.es
Christelle Vangenot E-mail: christelle.vangenot@epfl.ch

Library of Congress Control Number: 2006934617

CR Subject Classification (1998): H.2, H.4, H.3, F.4.1, D.2, C.2.4, I.2, J.1

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

ISSN	0302-9743
ISBN-10	3-540-47703-9 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-47703-7 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media
springer.com

© Springer-Verlag Berlin Heidelberg 2006
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11908883 06/3142 5 4 3 2 1 0

Foreword to ER 2006 Workshops and Tutorials

Welcome to the workshops and tutorials associated with the 25th International Conference on Conceptual Modeling (ER 2006). As always, the aim of the workshops is to give researchers and participants a forum to discuss cutting-edge research in conceptual modeling, both in theory and, particularly this year, in practice.

The change in the nature of the ER workshops to be a balance between research theory and practice has been apparent for a number of years and shows the continual maturing of conceptual modeling over the past 25 years. Now in its silver anniversary year, the ER series continues to be the premier conference in conceptual modeling and the interest shown in the workshops is testament to this. In all, 39 papers were accepted from a total of 95 submitted, an overall acceptance rate of 41%.

The focus for this year's seven workshops, which were selected competitively following a call for workshop proposals, ranges from practical issues such as industrial standards, UML and the quality of information systems, through to workshops focused on managing change in information systems, geographic systems, service-oriented software systems and the Semantic Web. Four have been run previously at an ER conference, three were new this year.

- Best Practices of UML (BP-UML 2006)
- Conceptual Modeling for Geographic Information Systems (CoMoGIS 2006)
- Conceptual Modeling of Service-Oriented Software Systems (COSS 2006)
- Evolution and Change in Data Management (ECDM 2006)
- Ontologizing Industrial Standards (OIS 2006)
- Quality of Information Systems (QoIS 2006)
- Semantic Web Applications: Theory and Practice (SemWAT 2006)

This volume contains the proceedings from the seven workshops. Also included are the outlines for the three tutorials:

- Conceptual Modeling for Emerging Web Application Technologies - Dirk Draheim and Gerald Weber.
- State of the Art in Modeling and Deployment of Electronic Contracts - Kamalakara Karlapalem and P. Radha Krishna.
- Web Change Management and Delta Mining: Opportunities and Solutions - Sanjay Madria.

Although there was a lot to see, the scheduling of the workshops and the main conference were organized so as to maximize the opportunity for delegates to attend sessions of interest.

Setting up workshops such as these takes a lot of effort. I would like to thank the PC chairs and their Program Committees for their diligence in selecting the

papers in this volume. I would also like to thank the main ER 2006 conference committees, particularly the conference Co-chairs Sudha Ram and Mohan Tan-
niru and the conference Publicity Chair and Webmaster, Huimin Zhao, for their
support in putting this programme together.

John F. Roddick
Flinders University
PO Box 2100, Adelaide
South Australia
`roddick@infoeng.flinders.edu.au`

ER 2006 Conference Organization

Honorary Conference Chair

Peter P. Chen (Louisiana University, USA)

Conference Co-chairs

Sudha Ram (University of Arizona, USA, ram@eller.arizona.edu)

Mohan R. Tanniru (University of Arizona, USA, mtanniru@eller.arizona.edu)

Program Committee Co-chairs

Dave Embley (Brigham Young University, USA, embley@cs.byu.edu)

Antoni Olive (Universitat Politècnica de Catalunya, Spain, olive@lsi.upc.es)

Workshop and Tutorial Chair

John F. Roddick (Flinders University, Australia, roddick@infoeng.flinders.edu.au)

Panel Co-chairs

Uday Kulkarni (Arizona State University, USA, uday.kulkarni@asu.edu)

Keng Siau (University of Nebraska, Lincoln, USA, ksiau@unlnotes.unl.edu)

Industrial Program Co-chairs

Arnie Rosenthal (Mitre Corporation, arnie@mitre.org)

Len Seligman (Mitre Corporation, seligman@mitre.org)

Demos and Posters Co-chairs

Akhilesh Bajaj (University of Tulsa, USA, akhilesh-bajaj@utulsa.edu)

Ramesh Venkataraman (Indiana University, USA, venkat@indiana.edu)

Publicity Chair and Webmaster

Huimin Zhao (University of Wisconsin, Milwaukee, USA, hzhao@uwm.edu)

Local Arrangements and Registration

Anji Siegel (University of Arizona, USA, asiegel@arizona.edu)

Steering Committee Liaison

Bernhard Thalheim (Christian-Albrechts-Universität zu Kiel, Germany,
thalheim@is.informatik.uni-kiel.de)

ER 2006 Workshop Organization

ER 2006 Workshop and Tutorial Chair

John F. Roddick (Flinders University, Australia, roddick@infoeng.flinders.edu.au)

BP-UML 2006 - Second International Workshop on Best Practices of UML

BP-UML 2006 was organized within the framework of the following projects: METASIGN (TIN2004-00779) from the Spanish Ministry of Education and Science, DADASMECA (GV05/220) from the Valencia Ministry of Enterprise, University and Science (Spain), and DADS (PBC-05-012-2) from the Castilla-La Mancha Ministry of Science and Technology (Spain).

Program Chairs

Juan Trujillo (University of Alicante, Spain)

Il-Yeol Song (Drexel University, USA)

Program Committee

Doo-Hwan Bae (KAIST, South Korea)

Michael Blaha (OMT Associates Inc., USA)

Cristina Cachero (Universidad de Alicante, Spain)

Tharam Dillon (University of Technology Sydney, Australia)

Dirk Draheim (Freie Universität Berlin, Germany)

Gillian Dobbie (University of Auckland, New Zealand)

Jean-Marie Favre (Université Grenoble, France)

Eduardo Fernández (Universidad de Castilla-La Mancha, Spain)

Jaime Gómez (Universidad de Alicante, Spain)

Anneke Kleppe (Universiteit Twente, The Netherlands)

Ludwik Kuzniarz (Blekinge Tekniska Högskola, Sweden)

Jens Lechtenböörger (Universität Münster, Germany)

Tok Wang Ling (National University of Singapore, Singapore)

Pericles Loucopoulos (University of Manchester, UK)

Hui Ma (Massey University, New Zealand)

Andreas L. Opdahl (Universitetet i Bergen, Norway)

Jeffrey Parsons (Memorial University of Newfoundland, Canada)

Óscar Pastor (Universitat Politècnica de València, Spain)

Witold Pedrycz (University of Alberta, Canada)

Mario Piattini (Universidad de Castilla-La Mancha, Spain)

Ivan Porres (Åbo Akademi University, Finland)
Colette Rolland (Université Paris 1-Panthéon Sorbonne, France)
Matti Rossi (Helsingin kauppakorkeakoulu, Finland)
Manuel Serrano (Universidad de Castilla-La Mancha, Spain)
Bernhard Thalheim (Universität zu Kiel, Germany)
A Min Tjoa (Technische Universität Wien, Austria)
Ambrosio Toval (Universidad de Murcia, Spain)
Antonio Vallecillo (Universidad de Málaga, Spain)
Panos Vassiliadis (University of Ioannina, Greece)

Referees

F. Molina	Ki Jung Lee	A. Sidhu
J. Lasheras	S. Schmidt	

CoMoGIS 2006 - Third International Workshop on Conceptual Modeling for Geographic Information Systems

Workshop Chairs

Christelle Vangenot (EPFL, Switzerland)
Christophe Claramunt (Naval Academy Research Institute, France)

Program Committee

Masatoshi Arikawa (University of Tokyo, Japan)
Natalia Andrienko (Fraunhofer Institute AIS, Germany)
Michela Bertolotto (University College, Dublin, Ireland)
Patrice Boursier (University of La Rochelle, France and Open University,
Malaysia)
Elena Camossi (IMATI-CNR Genova, Italy)
James Carswell (Dublin Institute of Technology, Ireland)
Maria Luisa Damiani (University of Milan, Italy)
Thomas Devogele (Naval Academy Research Institute, France)
Max Egenhofer (University of Maine, USA)
Andrew Frank (Technical University of Vienna, Austria)
Bo Huang (University of Calgary, Canada)
Zhiyong Huang (National University of Singapore, Singapore)
Christian S. Jensen (Aalborg University, Denmark)
Ki-Joune Li (Pusan National University, South Korea)
Dieter Pfoser (CTI, Greece)
Martin Raubal (University of Münster, Germany)
Andrea Rodriguez (University of Concepcion, Chile)

Sylvie Servigne (INSA, France)
Kathleen Stewart Hornsby (University of Maine, USA)
George Taylor (University of Glamorgan, UK)
Nectaria Tryfona (Talent Information Systems, Greece)
Agnes Voisard, (Fraunhofer ISST and FU Berlin, Germany)
Nico van de Weghe (University of Gent, Belgium)
Nancy Wiegand (University of Wisconsin-Madison, USA)
Stephan Winter (University of Melbourne, Australia)
Ilya Zaslavsky (San Diego Supercomputer Center, USA)
Esteban Zimanyi (Free University of Brussels, Belgium)

CoSS 2006 - International Workshop on Conceptual Modeling of Service-Oriented Software Systems

Workshop Chairs

Roger Chiang, (University of Cincinnati, USA)
Vojislav B. Mišić, (University of Manitoba, Canada)

Advisory Committee

Wil van der Aalst, (Technische Universiteit Eindhoven, The Netherlands)
Akhil Kumar, (Penn State University, USA)
Michael Shaw, (University of Illinois at Urbana-Champaign, USA)
Keng Siau, (University of Nebraska at Lincoln, USA)
Carson Woo, (University of British Columbia, Canada)
Liang-Jie Zhang, (IBM, USA)
J. Leon Zhao, (University of Arizona, USA)

Program Committee

Fabio Casati, (HP Labs, USA)
Dickson K. W. Chiu, (Dickson Computer Systems, Hong Kong)
Cecil Eng Huang Chua, (Nanyang Technological University, Singapore)
Haluk Demirkan, (Arizona State University, USA)
Stephane Gagnon, (New Jersey Institute of Technology, USA)
Patrick Hung, (University of Ontario Institute of Technology, Canada)
Qusay H. Mahmoud, (University of Guelph, Canada)
Hye-young Helen Paik, (University of New South Wales, Australia)
Venkataramanan Shankararaman, (Singapore Management University, Singapore)
Benjamin B. M. Shao, (Arizona State University, USA)
Vladimir Tomic, (Lakehead University, Canada)
Harry J. Wang, (University of Delaware, USA)
Lina Zhou, (University of Maryland at Baltimore County, USA)

ECDM 2006 - Fourth International Workshop on Evolution and Change in Data Management

Workshop Chair

Fabio Grandi (University of Bologna, Italy)

Program Committee

Alessandro Artale (Free University of Bolzano-Bozen, Italy)
Sourav Bhowmick (Nanyang Technical University, Singapore)
Michael Böhlen (Free University of Bolzano-Bozen, Italy)
Carlo Combi (University of Verona, Italy)
Curtis Dyreson (Washington State University, USA)
Shashi Gadia (Iowa State University, USA)
Kathleen Hornsby (University of Maine, USA)
Michel Klein (Vrije Universiteit Amsterdam, The Netherlands)
Richard McClatchey (University of the West of England, UK)
Federica Mandreoli (University of Modena and Reggio Emilia, Italy)
Torben Bach Pedersen (Aalborg University, Denmark)
Erik Proper (University of Nijmegen, The Netherlands)
John Roddick (Flinders University, South Australia)
Nandlal Sarda (IIT Bombay, India)
Myra Spiliopoulou (Otto-von-Guericke-Universität Magdeburg, Germany)
Carlo Zaniolo (UCLA, USA)

Additional Referees

M. Golfarelli
P.S. Jørgensen

OIS 2006 - First International Workshop on Ontologizing Industrial Standards

Workshop Chairs

Martin Hepp (University of Innsbruck, Austria)
Miltiadis Lytras (Athens University of Economics and Business, Greece)
V. Richard Benjamins (iSOCO, Spain)

Program Committee

Chris Bizer (Free University of Berlin, Germany)
Chris Bussler (Cisco Systems, Inc., San Francisco, USA)
Jorge Cardoso (University of Madeira)
Oscar Corcho (University of Manchester, UK)
Jos de Bruijn (DERI Innsbruck, Austria)

Doug Foxvog (DERI Galway, Ireland)
 Fausto Giunchiglia (University of Trento, Italy)
 Karthik Gomadam (LSDIS Lab, University of Georgia, USA)
 Michel Klein (Free University of Amsterdam, The Netherlands)
 Paavo Kotinurmi (Helsinki University of Technology, Finland)
 York Sure (University of Karlsruhe, Germany)

QoIS 2006 - Second International Workshop on Quality of Information Systems

Workshop Chairs

Samira Si-Saïd Cherfi (CEDRIC-CNAM, France)
 Geert Poels (University of Ghent, Belgium)

Steering Committee

Jacky Akoka (CEDRIC - CNAM and INT, France)
 Mokrane Bouzeghoub (PRISM, Univ. of Versailles, France)
 Isabelle Comyn-Wattiau (CNAM and ESSEC, France)
 Marcela Genero (Universidad de Castilla-La Mancha, Spain)
 Jeffrey Parsons (Memorial University of Newfoundland, Canada)
 Geert Poels (University of Ghent, Belgium)
 Keng Siau (University of Nebraska, USA)
 Bernhard Thalheim (University of Kiel, Germany)

Program Committee

Jacky Akoka (CEDRIC - CNAM and INT, France)
 Laure Berti (IRISA, France)
 Mokrane Bouzeghoub (PRISM, University of Versailles, France)
 Andrew Burton-Jones (University of British Columbia, Canada)
 Tiziana Catarci (Università di Roma "La Sapienza," Italy)
 Isabelle Comyn-Wattiau (CNAM and ESSEC, France)
 Corinne Cauvet (University of Aix-Marseille 3, France)
 Marcela Genero (Universidad de Castilla-La Mancha, Spain)
 Paul Johannesson (Stockholm University, Sweden)
 Jacques Le Maitre (University of Sud Toulon-Var, France)
 Jim Nelson (Southern Illinois University, USA)
 Jeffrey Parsons (Memorial University of Newfoundland, Canada)
 Óscar Pastor (Valencia University of Technology, Spain)
 Houari Sahraoui (Université de Montréal, Canada)
 Farida Semmak (Université Paris XII , IUT Sénart Fontainebleau)
 Keng Siau (University of Nebraska, USA)
 Guttorm Sindre (Norwegian University of Science and Technology, Norway)
 Monique Snoeck (Katholieke Universiteit Leuven, Belgium)

Il-Yeol Song (Drexel University, USA)
David Tegarden (Virginia Polytechnic Institute, USA)
Bernhard Thalheim (University of Kiel, Germany)
Dimitri Theodoratos (NJ Institute of Technology, USA)
Juan Trujillo (University of Alicante, Spain)

SemWAT 2006 - First International Workshop on Semantic Web Applications: Theory and Practice

Workshop Chairs

Hyoil Han (Drexel University, USA)
Ramez Elmasri (University of Texas at Arlington, USA)

Program Committee

Palakorn Achananuparp (Drexel University, USA)
Yuan An (University of Toronto, Canada)
Paul Buitelaar (DFKI GmbH, Germany)
Stefania Costache (University of Hannover, Germany)
Vadim Ermolayev (Zaporozhye National University, Ukraine)
Fabien Gandon (INRIA, France)
Raul Garcia-Castro (Universidad Politecnica de Madrid, Spain)
Peter Haase (University of Karlsruhe, Germany)
Kenji Hatano (Doshisha University, Japan)
Stijn Heymans (Vrije Universiteit Brussel, Belgium)
Istvan Jonyer (Oklahoma State University, USA)
Esther Kaufmann (University of Zurich, Switzerland)
Christoph Kiefer (University of Zurich, Switzerland)
Beomjin Kim (Indiana University - Purdue University, USA)
SeungJin Lim (Utah State University, USA)
Jun Miyazaki (Nara Advanced Institute of Science and Technology, Japan)
JungHwan Oh (University of Texas at Arlington, USA)
Byung-Kwon Park (Doga University, Korea)
Xiaojun Qi (Utah State University, USA)
Lawrence Reeve (Drexel University, USA)
York Sure (University of Karlsruhe, Germany)
Davy Van Nieuwenborgh (Vrije Universiteit Brussel, Belgium)
Mikalai Yatskevich (University of Trento, Italy)

Table of Contents

ER 2006 Tutorials

Conceptual Modeling for Emerging Web Application Technologies	1
<i>Dirk Draheim, Gerald Weber</i>	
State-of-the-Art in Modeling and Deployment of Electronic Contracts . . .	3
<i>Kamalakar Karlapalem, P. Radha Krishna</i>	

ER 2006 Workshops

BP-UML 2006 - 2nd International Workshop on Best Practices of UML

Preface to BP-UML 2006	5
<i>Juan Trujillo, Il-Yeol Song</i>	

Adopting UML 2.0

Extending the UML 2 Activity Diagram with Business Process Goals and Performance Measures and the Mapping to BPEL	7
<i>Birgit Korherr, Beate List</i>	
UN/CEFACT'S Modeling Methodology (UMM): A UML Profile for B2B e-Commerce	19
<i>Birgit Hofreiter, Christian Huemer, Philipp Liegl, Rainer Schuster, Marco Zapletal</i>	
Capturing Security Requirements in Business Processes Through a UML 2.0 Activity Diagrams Profile	32
<i>Alfonso Rodríguez, Eduardo Fernández-Medina, Mario Piattini</i>	

Modeling and Transformations

Finite State History Modeling and Its Precise UML-Based Semantics	43
<i>Dirk Draheim, Gerald Weber, Christof Lutteroth</i>	
A UML Profile for Modeling Schema Mappings	53
<i>Stefan Kurz, Michael Guppenberger, Burkhard Freitag</i>	
Model to Text Transformation in Practice: Generating Code from Rich Associations Specifications	63
<i>Manoli Albert, Javier Muñoz, Vicente Pelechano, Óscar Pastor</i>	

CoMoGIS 2006 - 3rd International Workshop on Conceptual Modeling for Geographic Information Systems

Preface for CoMoGIS 2006	73
<i>Christelle Vangenot, Christophe Claramunt</i>	

Keynote

Large-Scale Earth Science Services: A Case for Databases	75
<i>Peter Baumann</i>	

Spatial and Spatio-temporal Data Representation

Time-Aggregated Graphs for Modeling Spatio-temporal Networks	85
<i>Betsy George, Shashi Shekhar</i>	
An ISO TC 211 Conformant Approach to Model Spatial Integrity Constraints in the Conceptual Design of Geographical Databases.....	100
<i>Alberto Belussi, Mauro Negri, Giuseppe Pelagatti</i>	
Access Control in Geographic Databases	110
<i>Liliana Kasumi Sasaoka, Claudia Bauzer Medeiros</i>	

Optimizing Representation and Access to Spatial Data

VTPR-Tree: An Efficient Indexing Method for Moving Objects with Frequent Updates	120
<i>Wei Liao, Guifen Tang, Ning Jing, Zhinong Zhong</i>	
New Query Processing Algorithms for Range and k-NN Search in Spatial Network Databases	130
<i>Jae-Woo Chang, Yong-Ki Kim, Sang-Mi Kim, Young-Chang Kim</i>	
A ONCE-Updating Approach on Moving Objects	140
<i>Hoang Do Thanh Tung, Keun Ho Ryu</i>	

Spatio-temporal Data on the Web

A Progressive Transmission Scheme for Vector Maps in Low-Bandwidth Environments Based on Device Rendering	150
<i>David Cavassana Costa, Anselmo Cardoso de Paiva, Mario Meireles Teixeira, Cláudio de Souza Baptista, Elvis Rodrigues da Silva</i>	

Map2Share – A System Exploiting Metadata to Share Geographical Information	160
<i>Luca Paolino, Monica Sebillio, Genny Tortora, Giuliana Vitiello</i>	

CoSS 2006 - International Workshop on Conceptual Modeling of Service-Oriented Software Systems

Preface for CoSS 2006	171
<i>Roger Chiang, Vojislav B. Mišić</i>	
Building Semantic Web Services Based on a Model Driven Web Engineering Method	173
<i>Victoria Torres, Vicente Pelechano, Óscar Pastor</i>	
Choreographies as Federations of Choreographies and Orchestrations	183
<i>Johann Eder, Marek Lehmann, Amirreza Tahamtan</i>	
Designing Web Services for Supporting User Tasks: A Model Driven Approach	193
<i>Marta Ruiz, Vicente Pelechano, Óscar Pastor</i>	
Designing Service-Based Applications: Teaching the Old Dogs New Tricks ... or Is It the Other Way Around?	203
<i>Vojislav B. Mišić</i>	

ECDM 2006 - 4th International Workshop on Evolution and Change in Data Management

Preface for ECDM 2006	207
<i>Fabio Grandi</i>	

Keynote

Reduce, Reuse, Recycle : Practical Approaches to Schema Integration, Evolution and Versioning	209
<i>John F. Roddick, Denise de Vries</i>	

Accepted Papers

A DAG Comparison Algorithm and Its Application to Temporal Data Warehousing	217
<i>Johann Eder, Karl Wiggisser</i>	
Handling Changes of Database Schemas and Corresponding Ontologies	227
<i>Andreas Kupfer, Silke Eckstein, Karl Neumann, Brigitte Mathiak</i>	

Evolving the Implementation of ISA Relationships in EER Schemas	237
<i>Eladio Domínguez, Jorge Lloret, Ángel L. Rubio, María A. Zapata</i>	
Schema Change Operations for Versioning Complex Objects Hierarchy in OODBs	247
<i>Sang-Won Lee</i>	
Representing Versions in XML Documents Using Versionstamp	257
<i>Luis Jesús Arévalo Rosado, Antonio Polo Márquez, Juan María Fernández González</i>	

OIS 2006 - 1st International Workshop on Ontologizing Industrial Standards

Preface for OIS 2006	269
<i>Martin Hepp, Miltiadis Lytras, V. Richard Benjamins</i>	

Accepted Papers

XBRL Taxonomies and OWL Ontologies for Investment Funds	271
<i>Rubén Lara, Iván Cantador, Pablo Castells</i>	
A Semantic Transformation Approach for ISO 15926	281
<i>Sari Hakkarainen, Lillian Hella, Darijus Strasunskas, Stine Tuxen</i>	
Modeling Considerations for Product Ontology	291
<i>Hyunja Lee, Junho Shim, Suekyung Lee, Sang-goo Lee</i>	
Ontologizing EDI Semantics	301
<i>Doug Foxvog, Christoph Bussler</i>	
WSDL RDF Mapping:Developing Ontologies from Standardized XML Languages	312
<i>Jacek Kopecký</i>	

QoIS 2006 - 2nd International Workshop on Quality of Information Systems

Preface for QoIS 2006	323
<i>Samira Si-Saïd Cherfi, Geert Poels</i>	

Introduction to QoIS06

Information Quality, System Quality and Information System Effectiveness: Introduction to QoIS'06	325
<i>Geert Poels, Samira Si-Saïd Cherfi</i>	

Information System Quality

Quality-Driven Automatic Transformation of Object-Oriented Navigational Models.....	329
<i>Cristina Cachero, Marcela Genero, Coral Calero, Santiago Meliá</i>	
HIQM: A Methodology for Information Quality Monitoring, Measurement, and Improvement	339
<i>Cinzia Cappiello, Paolo Ficiaro, Barbara Pernici</i>	
Evaluating the Productivity and Reproducibility of a Measurement Procedure.....	352
<i>Nelly Condori-Fernández, Óscar Pastor</i>	

Data Quality

Quality of Material Master Data and Its Effect on the Usefulness of Distributed ERP Systems.....	362
<i>Gerhard F. Knolmayer, Michael Röthlin</i>	
Towards Automatic Evaluation of Learning Object Metadata Quality	372
<i>Xavier Ochoa, Erik Duval</i>	
Expressing and Processing Timeliness Quality Aware Queries: The DQ2L Approach	382
<i>Chao Dong, Sandra de F. Mendes Sampaio, Pedro R. Falcone Sampaio</i>	

SemWAT 2006 - 1st International Workshop on Semantic Web Applications: Theory and Practice

Preface for SemWAT 2006	393
<i>Hyoil Han, Ramez Elmasri</i>	

Semantic Web Applications (I)

Combining Declarative and Procedural Knowledge to Automate and Represent Ontology Mapping	395
<i>Li Xu, David W. Embley, Yihong Ding</i>	
Visual Ontology Alignment for Semantic Web Applications.....	405
<i>Jennifer Sampson, Monika Lanzemberger</i>	

Automatic Creation of Web Services from Extraction Ontologies 415
Cui Tao, Yihong Ding, Deryle Lonsdale

Semantic Web Applications (II)

An Architecture for Emergent Semantics 425
Sven Herschel, Ralf Heese, Jens Bleiholder

Semantic Web Techniques for Personalization of eGovernment Services . . . 435
*Fabio Grandi, Federica Mandreoli, Riccardo Martoglia,
Enrico Ronchetti, Maria Rita Scalas, Paolo Tiberio*

Query Graph Model for SPARQL 445
Ralf Heese

Author Index 455

Conceptual Modeling for Emerging Web Application Technologies

Dirk Draheim¹ and Gerald Weber²

¹ Institute of Computer Science, Universität Mannheim
`draheim@acm.org`

² Dept. of Computer Science, University of Auckland
`g.weber@cs.auckland.ac.nz`

Abstract. What are the concepts behind state-of-the-art web application frameworks like Websphere on the commercial side or Struts on the open source side? What are the concepts behind emerging formats and technologies like XFORMS, XUL, XAML, Server Faces, Spring? The tutorial is open to working software engineers and decision makers that are involved in web application projects. It also targets researchers in the field of Software Engineering that are interested in a high-level overview on new web technologies.

In a web application project, a large number of advanced technologies must be exploited to get the work done in time. At the same time, modeling is an accepted best practice. Therefore, a high-level understanding of the advanced web application technologies is desired by system analysts and developers.

In this tutorial we analyze web application technologies in a novel modeling technique, form-oriented analysis. We identify the underpinnings of new web application technologies in terms of proven concepts of the modeling community. For example, a web form turns out to be an editable method call, the ubiquitous Model-2-Architecture for web applications turns out to be functional decomposition.

On the basis of the viewpoint of a strict submit/response style system, the participant of this tutorial should be empowered to decide

- which work products are appropriate in his or her concrete project and
- which details of advanced technologies should actually have a footprint in the system documentation.

No single solution fits all sizes – web application projects range from little web shops for a dozen of products to huge business-to-customer portals involving many CRM-related aspects. To trigger the discussion, we propose a well-defined set of documents and work-products for a standard, medium-size web application project. We give the model of an example web shop as a comprehensive example. As a real-world case study we discuss the documentation style guide of an ERP project that introduced a new central university administration system, see Fig. 1.

The tutorial "Modeling Enterprise Applications" held at ER 2005 [5] covered the foundations of modeling of all tiers of enterprise applications. This tutorial is specialized to web applications and has a focus on a conceptual understanding of new technologies in the field of web applications.

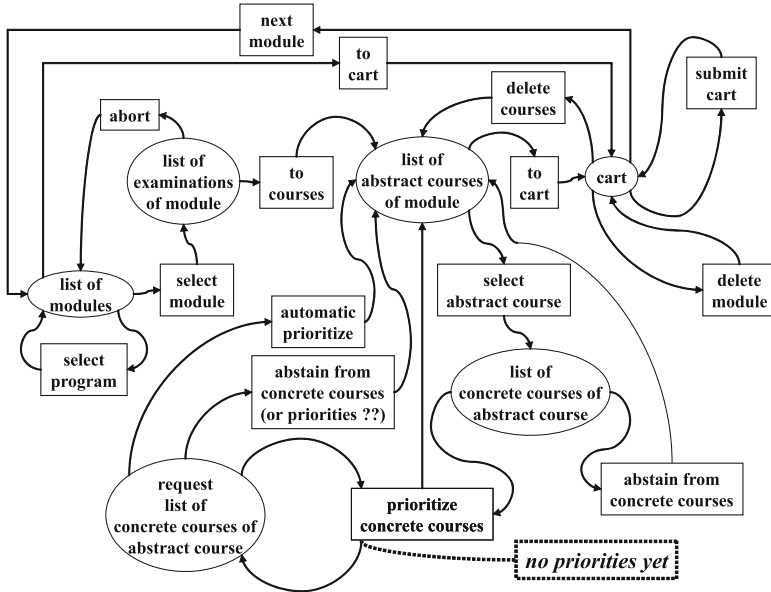


Fig. 1. Real world case study of the tutorial: conceptual modeling of a university administration system in action

References

1. Dirk Draheim, John Grundy, John Hosking, Christof Lutteroth, and Gerald Weber. Realistic Load Testing of Web Applications. In *Proceedings of CSMR 2006 - 10th European Conference on Software Maintenance and Reengineering*. IEEE Press, March 2006.
2. Dirk Draheim, Christof Lutteroth, and Gerald Weber. Revangie - A Source Code Independent Reverse Engineering Tool for Dynamic Web Sites. In *Proceedings of the 9th European Conference on Software Maintenance and Reengineering*. IEEE Press, 2005. to appear.
3. Dirk Draheim and Gerald Weber. Storyboarding Form-Based Interfaces. In *Proceedings of INTERACT 2003 - Ninth IFIP TC13 International Conference on Human-Computer Interaction*. IOS Press, 2003.
4. Dirk Draheim and Gerald Weber. *Form-Oriented Analysis - A New Methodology to Model Form-Based Applications*. Springer, October 2004.
5. Dirk Draheim and Gerald Weber. Modeling Enterprise Applications. In *Perspectives in Conceptual Modeling*, LNCS 3770. Springer, September 2005.
6. Dirk Draheim and Gerald Weber. Modelling Form-Based Interfaces with Bipartite State Machines. *Interacting with Computers*, 17(2):207–228, 2005.

State-of-the-Art in Modeling and Deployment of Electronic Contracts

Kamalakar Karlapalem¹ and P. Radha Krishna²

¹ International Institute of Information Technology, Hyderabad, India
kamal@iiit.ac.in

² Institute for Development and Research in Banking Technology, Hyderabad, India
prkrishna@idrbit.ac.in

Abstract. Modeling and deployment of e-contracts is a challenging task because of the involvement of both technological and business aspects. There are several frameworks and systems available in the literature. Some works mainly deal with the automatic handling of paper contracts and others provide monitoring and enactment of contracts. Because contracts evolve, it is useful to have a system that models and enacts the evolution of e-contracts. This tutorial mainly covers basic concepts of e-contracts, modeling frameworks and deployment scenarios of e-contracts. Specific case studies from current literature and business practices will illustrate the current state of the art, and help enumerate the open research problems that need to be addressed.

Background

Electronic contracts start from being legal documents to processes that help organizations to abide by the legal rules while fulfilling the contracts. Deployment of electronic contracts poses a lot of challenges at three levels, namely conceptual, logical and implementation. Changes in the factors influencing the contract execution require changes at one or more levels.

An e-contract is a contract modeled, specified, executed and deployed (controlled and monitored) by a software system (such as a workflow system). As contracts are complex, their deployment is predominantly established and fulfilled with significant human involvement. This necessitates a comprehensive framework for generic fulfillment of e-contracts. The literature of e-contracts has been spread across various stages, namely contract preparation, negotiation and contract fulfillment. The languages to represent electronic contracts include XML, ebXML, ECML, tpaML, RuleML etc. Further, several projects (COSMOS, CrossFlow, etc) have been undertaken by researches for Business-to-Business e-contracts.

The major problem in modeling e-contracts is the dynamic behavior of e-contracts during their evolution and deployment. Moreover, social and economic factors as well as run-time changes influence the e-contract deployment. This may require changes at conceptual level, at logical model and at the database level. This further requires translation of e-contract instances to deployable workflows. Hence, e-contract deployment necessitates an appropriate model management at multiple levels.

In our earlier work [4,5], we developed an ER^{EC} framework to support model management and deployment of e-contracts. The ER^{EC} framework facilitates designing e-contract processes, a mechanism that allows modeling, management, deployment and monitoring of e-contracts. This framework centers on the ER^{EC} model that bridges between the XML contract document and web services based implementation model of an e-contract.

Angelov and Grefen [1] presented a survey on various projects and systems related to e-contracts. SweetDeal system, developed by Benjamin and Poon [2], allows software agents to create, evaluate, negotiate and execute e-contracts with substantial automation and modularity. This approach represents contracts in RuleML and incorporates process knowledge descriptions based on the ontologies. Chiu, Cheung and Till [3] developed an e-contract deployment system based on multiple layer framework. In this framework, the e-contracts are modeled in UML and the implementation architecture is based on cross-organizational workflows using Enterprise Java Bean and Web services. Xu and Jeusfeld [6] proposed a framework for monitoring e-contracts during the contract execution. Temporal logic has been used to represent the e-contract, which enables the pro-active monitoring of e-contracts. Currently, most of these models are human and system driven prototypes (some of them in the process of developing tool-kits) to popularize e-contracts. These systems reduce the time to learn and deploy new e-contracts and manage workflows for e-contract deployment.

References

1. Angelov, S. and Grefen, P., B2B eContract Handling – A Survey of Projects, Papers and Standards, CTIT Technical Report 01-21; University of Twente, 2001.
2. Benjamin N. Grosz and Terrence C. Poon, SweetDeal: Representing Agent Contracts with Exceptions using XML Rules, Ontologies, and Process Descriptions. In Proceedings of the 12th International Conference on the World Wide Web, 2003.
3. Chiu, D. K. W., Cheung, S. C., Till, S.: A Three-layer Architecture for E-Contract Enforcement in an E-service Environment, Proc. of 36th HICSS36, (2003).
4. Radha Krishna, P., Karlapalem, K., Chiu, D. K. W.: An ER^{EC} Framework for E-Contract Modeling, Enactment and Monitoring, *Data and Knowledge Engineering*, 51 -1, 2004, 31-58.
5. Radha Krishna, P., Karlapalem, K., Dani, A. R.: From Contracts to E-Contracts: Modeling and Enactment, *Information Technology and Management Journal*, 4 –1, 2005.
6. Xu L. and Jeusfeld M.A. : Pro-active Monitoring of Electronic Contracts. Proceedings of 15th Conference On Advanced Information Systems Engineering (CAiSE'03), 2003.

Preface to BP-UML 2006

Juan Trujillo¹ and Il-Yeol Song²

¹ University of Alicante, Spain

² Drexel University, USA

The Unified Modeling Language (UML) has been widely accepted as the standard object-oriented (OO) modeling language for modeling various aspects of software and information systems. UML is an extensible language, in the sense that it provides mechanisms to introduce new elements for specific domains if necessary, such as Web applications, database applications, business modeling, software development processes, data warehouses and so on. Furthermore, the latest version of UML 2.0 got even bigger and became more complicated with a higher number of diagrams. Although UML provides different diagrams for modeling different aspects of a software system, not all of them need to be applied in most cases. Therefore, heuristics, design guidelines, and lessons learned from experiences are extremely important for the effective use of UML 2.0 and for avoiding unnecessary complications.

The Second International Workshop on Best Practices of UML (BP-UML 2006) is a sequel to the successful BP-UML 2005 workshop. BP-UML 2006 was held with the 25th International Conference on Conceptual Modeling (ER 2006), and it intends to be an international forum for exchanging ideas on the best and new practices of UML in modeling and system developments. To keep the high quality of the workshops held in conjunction with ER, a strong International Program Committee was organized with extensive experience in UML as well as relevant scientific production in the area.

The workshop attracted papers from 12 different countries from all continents: Australia, Austria, Chile, France, Germany, India, New Zealand, Spain, Tunisia, Turkey, UK, and USA. We received 17 submissions and only 6 papers were selected by the Program Committee, making an acceptance rate of 35%.

The accepted papers were organized in two sessions. In the first one, three papers explained how to apply UML 2.0 for business process modeling. In the second session, one paper focused on modeling interactive systems, and the other two papers discussed how to use transformations within a model-driven development.

We would like to express our gratitude to the Program Committee members and the external referees for their hard work in reviewing papers, the authors for submitting their papers, and the ER 2006 Organizing Committee for all their support. We also would like to thank Miguel Ángel Varó and Jose-Norberto Mazón for their support in the organization of this workshop.

Extending the UML 2 Activity Diagram with Business Process Goals and Performance Measures and the Mapping to BPEL^{*}

Birgit Korherr and Beate List

Women's Postgraduate College for Internet Technologies
Institute of Software Technology and Interactive Systems
Vienna University of Technology
{korherr, list}@wit.tuwien.ac.at
<http://wit.tuwien.ac.at>

Abstract. The UML 2 Activity Diagram is designed for modelling business processes, but does not yet include any concepts for modelling process goals and their measures. We extend the UML 2 Activity Diagram with process goals and performance measures to make them conceptually visible. Additionally, we provide a mapping to BPEL to make the measures available for execution and monitoring. This profile and its mapping are tested with an example business process.

1 Introduction

Although business process performance measurement is an important topic in research and industry [5], current conceptual Business Process Modelling Languages (BPMLs) do not mirror these requirements by providing explicit modelling means for process goals and their measures [14]. Furthermore, the measures need to be integrated into the process execution and require continuous monitoring. The goal of this paper is to address these limitations by

- extending UML 2 Activity Diagrams with business process goals and performance measures to make them conceptually visible, and by
- mapping the performance measures onto the Business Process Execution Language (BPEL) to make them available for execution and monitoring.

Activity Diagrams are a part of the behavioural models of UML 2 [20] and are used for modelling business processes as well as for describing control flows in software. Activity Diagrams neither have quality nor quantity based elements to measure the performance of a business process. For instance, the modeller of a process has no possibility to express the maximum time limit for processing of a specific action - the basic element of Activity Diagrams - or a group of actions.

UML profiles are an extension mechanism for building UML models for particular domains or purposes [20]. We utilise this well-defined way to extend the UML 2

^{*} This research has been funded by the Austrian Federal Ministry for Education, Science, and Culture, and the European Social Fund (ESF) under grant 31.963/46-VII/9/2002.

Activity Diagram with business process goals and performance measures. In a further step we define its mapping onto BPEL, and thus, provide the following contributions:

- The modelling of goals is a critical step in the creation of useful process models, as they allow a) to structure the process design, b) to evaluate the process design, c) to better understand the broader implication of the process design and, d) to evaluate the operating process [13]. This is made explicitly visible by the UML 2 profile (cf. Section 3).
- The UML 2 profile and its mapping onto BPEL enable the transformation of the business processes models developed in a UML modelling tool into BPEL. Thus, the conceptually described performance measures can be directly transformed into the execution language and can be used to monitor the process instances continuously.
- The business process models as well as the extensions based on the UML profile can be easily created, presented and edited with existing UML modelling tools, as almost all newer UML tools support UML profiles.

In the remainder of the paper, the role of business process goals and performance measures is briefly discussed (Section 2). As a foundation for the UML 2 profile, we have extended the UML 2 metamodel for Activity Diagrams in Section 3. This light-weight extension mechanism provides the concepts to present the business process goals and performance measures. In Section 4, we describe a set of OCL constraints of the UML 2 profile to indicate restrictions that belong to the metamodel. The UML 2 profile as well as the mapping to BPEL is tested with an example business process in Section 5 and Section 6, respectively. We close with related work (Section 7), future work (Section 8), and the conclusion (Section 9).

2 The Role of Goals and Measures in the Business Process

With business process reengineering Davenport, Hammer, and Champy created a new discipline at the beginning of the 1990ies and provided the theoretical background for business process modelling. So far, in the business process modelling community attention has only been given to the modelling of certain aspects of processes (e.g. roles, activities, interactions). These theoretical aspects are mirrored in several BPMLs, for example, in the Business Process Modelling Notation [4], the Event-driven Process Chain [21], the UML 2 Activity Diagram [20], etc. Kueng and Kawalek argued already in 1997 that little attention is paid to the value of making goals explicit [13]. Today, there are quite a lot of conceptual BPMLs available, but they still do not provide modelling means for business process goals and performance measures [14].

A business process is defined as a “group of tasks that together create a result of value to a customer” [7]. Its purpose is to offer each customer the right product or service, i.e., the right deliverable, with a high degree of performance measured against cost, longevity, service and quality [10]. Although process goals and performance measures are available in process theory, they lack the visibility in conceptual BPMLs.

According to [13], the modelling of goals is a critical step in the creation of useful process models for the following reasons:

- We need to be able to state what we want to achieve so that we are then able to define the necessary activities which a business process should encompass (i.e., goals are used to *structure the design*).
- A clear understanding of goals is essential in the management of selecting the best design alternative (i.e., goals are used to *evaluate the design*).
- A clear understanding of goals is essential to evaluate the operating quality of a business process (i.e., goals are used to *evaluate the operating process*).
- A clear expression of goals makes it easier to comprehend the organisational changes that must accompany a business process redesign (i.e., goals help the modeller to better understand the broader implication of design, beyond those of the business process itself).

For all the reasons described above, we capture the business process goals and represent them graphically in a conceptual BPML, namely the UML 2 Activity Diagram. Furthermore, Kueng and Kawalek recommend in [13] to define to which extent the process goals are fulfilled, to measure the achievement of goals either by qualitative or quantitative measures, and to define a target value for each measure. Target values are also very important for Service Level Agreements (SLAs) as well as for business process improvement. Harrington stated “Measurements are the key. If you cannot measure it, you cannot control it. If you cannot control it, you cannot manage it. If you cannot manage it, you cannot improve it.” [8]. In order to support Kueng’s and Kawalek’s statement, and all stages of Harrington’s statement, we need to integrate performance measures into conceptual BPMLs.

3 The UML 2 Profile

In this section, we describe the extended metamodel for Activity Diagrams for the UML 2 profile with business process goals and performance measures. Activity Diagrams are a part of the behavioural set of UML 2 diagrams, and are used for modelling business processes as well as for describing control flows in software. A UML 2 Activity Diagram specifies the control and data flow between different tasks, called actions, which are essential for the realisation of an activity. The UML 2 Activity Diagram currently does not support the graphical representation of business process goals and performance measures. Thus, it is not possible to show, e.g., time restrictions of the business process, its cost or quality requirements. UML offers a possibility to extend and adapt its metamodel to a specific area of application through the creation of profiles. This mechanism is called a light-weight extension. UML profiles are UML packages of the stereotype «profile». A profile can extend a metamodel or another profile [20] while preserving the syntax and semantic of existing UML elements. It adds elements which extend existing metaclasses. UML profiles consist of *stereotypes*, *constraints* and *tagged values*.

A stereotype is a model element defined by its name and by the base class(es) to which it is assigned. Base classes are usually metaclasses from the UML metamodel, for instance the metaclass «Class», but can also be stereotypes from another profile. A

stereotype can have its own notation, e.g. a special icon. Constraints are applied to stereotypes in order to indicate restrictions. They specify pre- or post conditions, invariants, etc., and must comply with the restrictions of the base class [20]. Constraints can be expressed in any language, such as programming languages or natural language. We use the Object Constraint Language (OCL) [19] in our profile, as it is more precise than natural language or pseudocode, and widely used in UML profiles. Tagged values are additional metaattributes assigned to a stereotype, specified as name-value pairs. They have a name and a type and can be used to attach arbitrary information to model elements.

Figure 1 illustrates a section of the UML metamodel for Activity Diagrams and its extension with stereotypes for representing business process goals and their performance measures. The triangle at associations marks the direction of reading of a relationship between the metaclasses to support the clarity of the metamodel.

The UML profile consists of four different stereotypes, namely «*Process Goal*», «*Measure*», «*Alert*» and «*Organisational Structure*». The stereotype «*Process Goal*» describes the specific intension of a business process and is quantified by at least one «*Measure*». The «*Process Goal*» extends the metaclass *Activity*, meaning that a «*Process Goal*» is described at activity level.

The stereotype «*Measure*» can be classified and implemented as «*Quality*», «*Cost*» and «*Cycle Time*» and extends the metaclasses *Activity Partition*, *Structured Activity Node*, and *Control Flow*. This means that the stereotype «*Measure*» can be described in three different ways. It is the modeller's role to choose the most suitable way to best describe a measure for a certain purpose, a user or user group. Moreover, the stereotype «*Measure*» is responsible for the concrete quantification of different goals as well as for measuring the performance of a business process. If the process is not performed according to the «*Measure*», an «*Alert*» is triggered.

A structured activity node has the function to group elements of an activity, in order to structure the activity [20]. A measure located in a structured activity node quantifies the section of the process that is covered. For instance, a structured activity node that is extended with the stereotype «*Cycle Time*» has to finish the processing of its actions within a certain period of time.

A measure positioned in an activity partition quantifies the section of the process that is covered by the role. According to the OMG [20], an activity partition identifies actions that have some characteristics in common. For example, if activity nodes have to be performed within a specific period of time, then they are grouped within an activity partition labelled with the stereotype «*Cycle Time*». It is also possible to nest the stereotypes. A stereotyped structured activity node labelled with «*Working Time*» can be nested in an activity partition, e.g., extending «*Cycle Time*».

A measure based on the control flow quantifies the cycle time, cost or quality between two actions. The OMG [20] defines a control flow as an edge that starts an activity node after the previous one is finished. For example, a control flow that is extended with the stereotype «*Cycle Time*» and connects two activity nodes, means that the stereotype measures a period of time the token requires from the activity node at the beginning of the edge to the activity node at the end of the edge.

The stereotypes «*Quality*», «*Cost*» and «*Cycle Time*» add more detail to the stereotype «*Measure*» and classify it. The stereotype «*Quality*» has the aim to measure the quality of a business process, which can be expressed e.g., by a low number of complaints or a high customer satisfaction.

The stereotype «*Cost*» represents the financial expenses a business process requires e.g., for its execution. Its tagged values and operations are necessary to compute e.g. average values like the total and monthly average cost of a certain process. The performance measures of «*Quality*» and «*Cost*» are in contrast to the measures of the «*Cycle Time*» often more focused on the type level of a process, as the required data is often not available on instance level.

The stereotype «*Cycle Time*» presents a time based measure and defines the duration a business process instance, or part of it requires from the beginning until the end. The stereotype «*Cycle Time*» can be specialised as «*Working Time*» or «*Waiting Time*». «*Working Time*» presents the actual time a business process instance is being executed by a role. «*Waiting Time*» shows the time limit the process instance is allowed to delay further processing. Moreover, «*Cycle Time*» has two tagged values, for representing the target value and the actual value of the process duration or a part of it which is computed by an operation of the stereotype.

The stereotype «*Organisational Structure*» describes the different roles within an Activity Diagram, namely the «*Organisational Unit*» and the «*Organisational Role*». Furthermore, an «*Organisational Unit*» has at least one «*Organisational Role*». The purpose of these stereotypes is besides showing the role that performs certain actions, to make the «*Organisational Structure*» visible that is triggered by the stereotype «*Alert*», if an action or a group of actions is not executed within its performance measures. The stereotype «*Alert*» has two metaclasses, from which it is derived, one for time based measures, namely *AcceptTimeEventAction*, and one for non-time based measures, namely *AcceptEventAction*. An «*Alert*» belongs to exactly one «*Measure*» as well as to one element of the «*Organisational Structure*», and has one tagged value to show on instance level if an alert is caused or not.

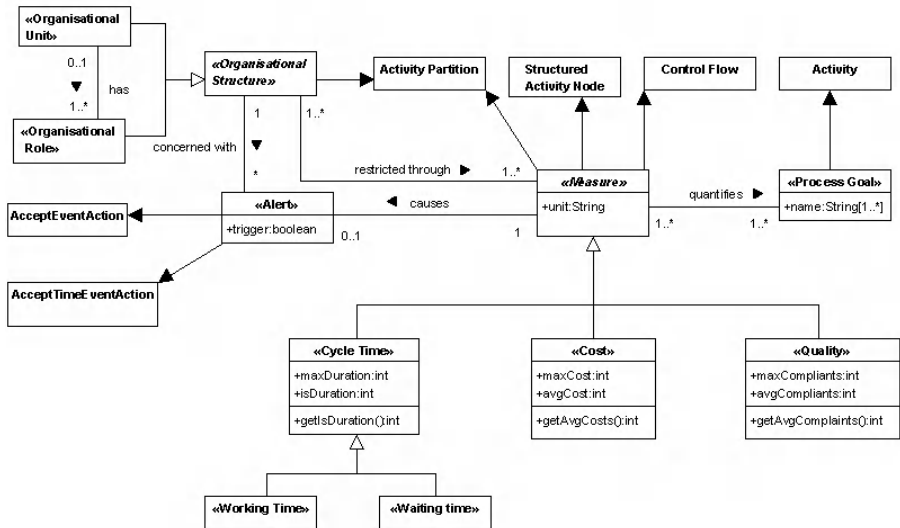


Fig. 1. Extended metamodel of the Activity Diagram for the UML 2 profile with business process goals and performance measures

4 Constraints

Constraints are applied to stereotypes in order to indicate restrictions. They specify pre- or post conditions, invariants, etc., and must comply with the restrictions of the base class [20]. Constraints can be expressed in any language, such as programming languages or natural language. We use the Object Constraint Language (OCL) [19] in our profile, as it is more precise than natural language or pseudocode, and widely used in UML profiles. Table 1 shows the OCL constraints with explanations in natural language for the stereotypes «Measure» and «Alert».

Table 1. OCL Constraints for the stereotypes «Measure» and «Alert»

Stereotype	Constraints
Measure	If a measure is a cycle time based measure, then the occurring alert has the type of an AcceptTimeEventAction, otherwise the alert has the type of an AcceptEventAction. <pre> context Measure inv: if CycleTime.ocIsKindOf(Measure) then Alert.ocIsKindOf(AcceptTimeEventAction) else Alert.ocIsKindOf(AcceptEventAction) </pre>
Alert	If the actual value of the duration is higher then the maximum value of the duration, an alert will be generated. <pre> context Alert inv: if cycleTime.isDuration > cycleTime.maxDuration then trigger = true else trigger = false endif </pre> If the average cost is higher then the maximum cost, then an alert will be generated. <pre> context Alert inv: if Cost.allInstances()->forAll(avgCost > maxCost) then Alert.trigger = true else Alert.trigger = false endif </pre> If the average number of complaints is higher then the maximum number of compliants, an alert will be generated. <pre> context Alert inv: if Quality.allInstances()->forAll(avgCompliants > maxCompliants) then Alert.trigger = true else Alert.trigger = false endif </pre>

5 Applying the UML 2 Profile to an Example Business Process

We demonstrate the practical applicability of the extension of the UML 2 Activity Diagram with business process goals and performance measures in Figure 1 with the example business process of an insurance company: the *Processing of Automobile Insurance Claims* business process (Fig. 2). We have refined the Activity Diagram by including a set of stereotypes, based on the various types of actions specified in the metamodels of actions in the UML superstructure in chapter 11 [20] and inspired by Bordbar et al. in [3].

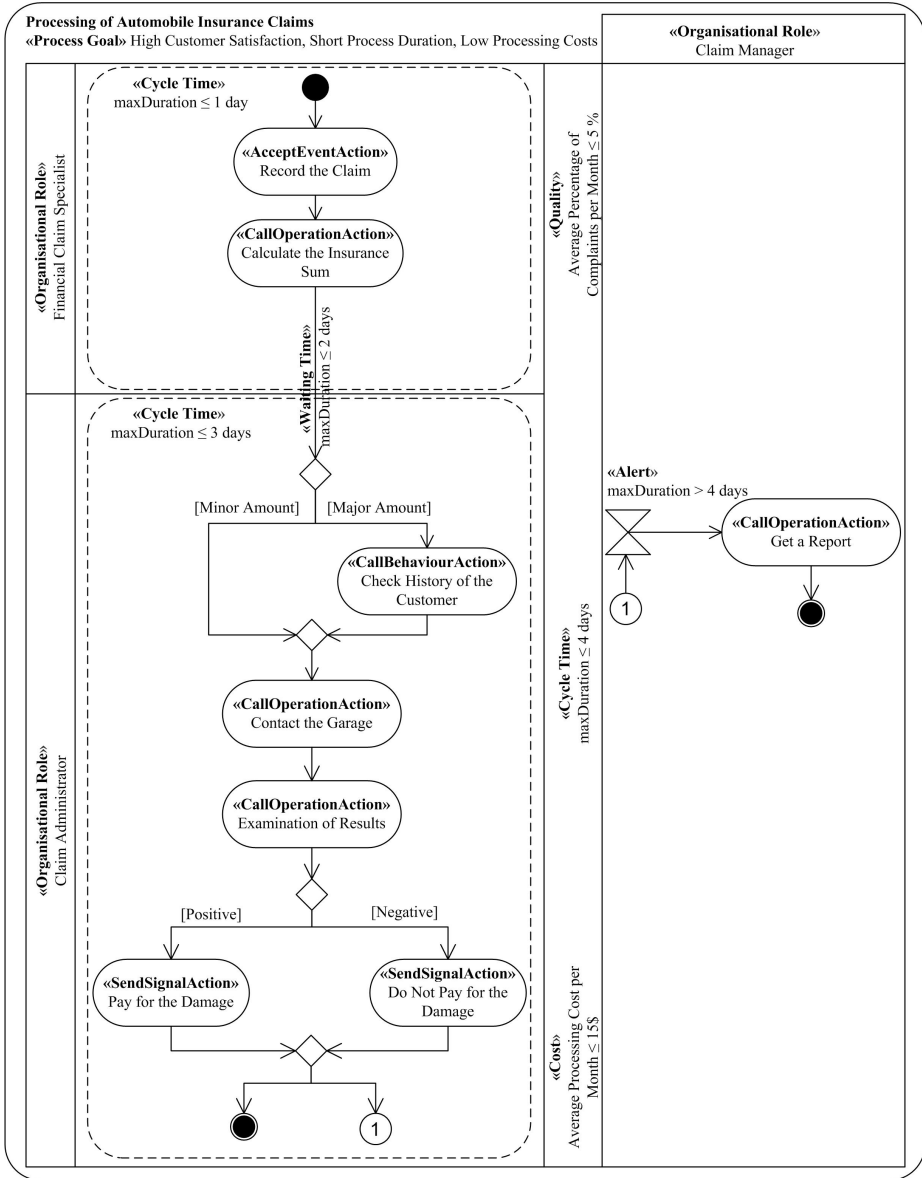


Fig. 2. Example business process based on the UML 2 profile for business process goals and performance measures

The overall goal of the processing of automobile insurance claims business process is to fulfil the «Process Goal» *High Customer Satisfaction, Short Process Duration* and *Low Processing Costs*. At the beginning of the process the «Organisation Role» *Financial Claim Specialist* is responsible for the actions *Record the Claim* and *Calculate the Insurance Sum*. The «Cycle Time» for these actions must not be more

than one day. This is shown in the structured activity node. After a «Waiting Time» of two days maximum that is illustrated on the control flow, the «Organisational Role» of the *Claim Administrator* has to follow up with the process. The claim administrator has a maximum «Cycle Time» of three days for processing its task. If the insurance sum is a major amount, then the claim administrator has to *Check History of the Customer* in the other case no action is required. After starting to *Contact the Garage* for the repairation, the *Examination of Results* has to begin. If the examination is positive, then the insurance has to *Pay for the Damage*, and the case is closed if the «Cycle Time» is not over for days.

The process has to meet the performance measures «Cost», «Cycle Time», and «Quality». The average processing cost per month has to be 15\$ maximum and the complaints should not exceed five percent. Only the «Cycle Time» is measured on process instance level. In the example, if the «Cycle Time» of the process is over four days, then the *Claim Manager* receives an alert and *Gets a Report* about that specific case and the business process terminates.

Figure 2 shows that a business process based on the UML 2 profile can be grasped at a glance. The extensions of the UML 2 Activity Diagram better illustrate the requirements of a certain business process and enhance the expressiveness of a model.

6 Mapping the UML Profile onto BPEL

The Business Process Execution Language (BPEL) is a language for specifying business process behaviour based on Web Services [9]. The UML profile will be mapped onto BPEL, in order to transform a specific business process modelling language and its conceptually described performance measures to an execution language as well as to make it possible to monitor the process instances continuously. Figure 3 shows the extended UML 2 Activity Diagram for BPEL based on [3]. Bordbar et al. present in [3] a transformation of the UML 2 Activity Diagram to BPEL to show the behavioural aspects of web services. We use this approach for mapping the different actions in Figure 2 to the BPEL tags in Figure 3. Furthermore, we are mapping «Cycle Time», «Waiting Time», «Working Time» by using the BPEL tag `<onAlarm>` as well as «Organisational Unit» and «Organisational Role» by using the `<partner link>` tag. We do not map the performance measures cost and quality, because we focus on the instance level of a business process and not on the type level. The web services a business process interacts with are modelled as `<partner links>` in BPEL, in the example business process this is the claim manager, the claim administrator as well as the financial claim specialist. Each partner link is characterised by a `partnerLinkType`, which we do not graphically show in the UML 2 Activity Diagram for BPEL. The tag `<onAlarm>` marks a timeout event, which is a part of the *Event Handler*. Both, the entire process and each scope can be linked with a set of event handlers. An alarm event goes off when the specified time or duration has been reached. The *for* attribute specifies the duration after which the event will be triggered. The alternative attribute *until* describes a specific point in time when the alarm will be fired. The clock for the duration starts at the point in time when the associated scope starts. In the example business process in Figure 3, we illustrate the `<onAlarm>` tags. For the sake of simplicity we do not integrate the whole event

handler into the diagram. Thus, we focus on one alarm event, executed by the role of the claim manager. If the overall process execution exceeds four days, then an alarm event will be created to inform the claim manager, and a report will be generated. Table 2 shows the mapping relations between the stereotypes of the UML 2 Profile and the BPEL tags, which are used in Figure 3.

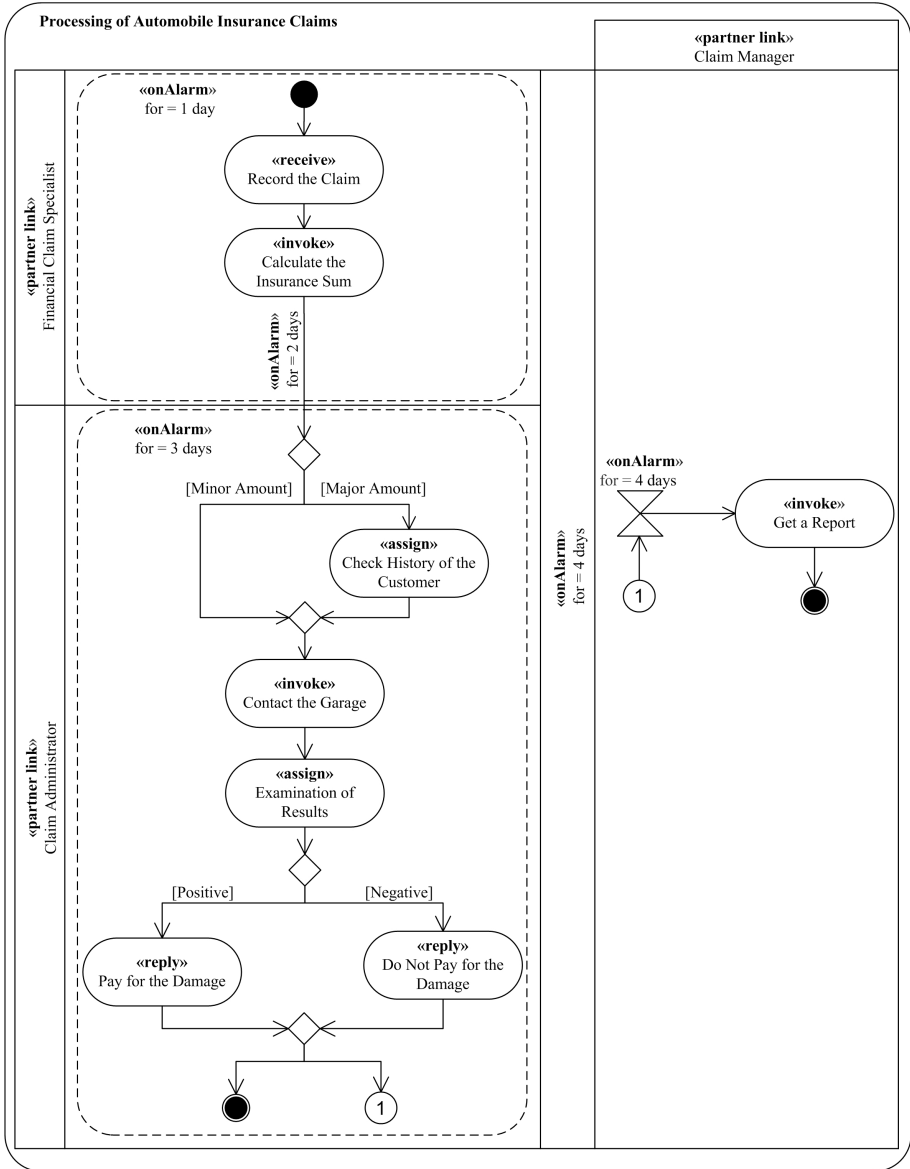


Fig. 3. Example business process based on the UML 2 profile for business process goals and performance measures and the mapping to BPEL

Table 2. Mapping relations between the UML 2 profile and BPEL

UML Base Class	UML Stereotype	BPEL Tag
Action	«AcceptEventAction»	«receive»
Action	«CallOperationAction»	«invoke»
Action	«CallBehaviourAction»	«assign»
Action	«SendSignalAction»	«reply»
Activity Partition / Structured Activity Node / Control Flow	«Cycle Time»	«onAlarm»
	«Waiting Time»	«onAlarm»
	«Working Time»	«onAlarm»
Activity Partition	«Organisational Unit»	«partner link»
Activity Partition	«Organisational Role»	«partner link»

7 Related Work

The related work consists of two parts. The first part is focused on different aspects of the quantification of performance measures, while the second part address to previous proposals for mapping a BPML to BPEL.

Aguilar et al. [1] developed a set of measures to evaluate the structural complexity of business process models on the conceptual level. The authors use the Business Process Modeling Notation (BPMN) [4] for their evaluation. The evaluation of performance measure like time or cost is not important for their work, the focus lies on measuring the core elements of BPMN.

The approach of Vitolins [22] is based on metamodeling according to the Meta Object Facility (MOF) [17] and aims to provide precise definitions of typical process measures for a UML 2 Activity Diagram-like notation. As a contrast to our work, the author annotates cost and time to each action separately as a note. There are no considerations to integrate the performance measures as graphical notation elements, although the approach lacks clarity and explicitness.

Nurcan et al. [15] adopted a goal-perspective, namely the map-driven process modelling approach to master the complexity of process modelling. The authors capture the strategic goals of the organisation as well as the tasks carried out by actors, to establish the importance of goals in process modelling.

There exist quite a lot of proposals for transforming / mapping UML 2 Activity Diagrams to BPEL, rather than for UML profiles to BPEL.

Bordbar et al. [3] present a transformation of the UML 2 Activity Diagram to BPEL to show the behavioural aspects of web services by using a metamodel, based on MOF [17] for BPEL. This work used OCL as a transformation language adapted from [12] at a time where no standard language for transformation definitions existed.

Gardner et al. [4] show a UML Profile for Automated Business Processes which enables BPEL processes to be modelled using an existing UML tool as well as a mapping to BPEL to automatically generate web service artefacts (BPEL, WSDL, XSD) from the UML profile. This work is rather out of date, because of the old UML version 1.4 and BPEL 1.0.

8 Future Work

In the sense of Model Driven Engineering (MDE) [16], the transformation of the Platform Independent Model (PIM), e.g. a UML profile to the Platform Specific Model (PSM), e.g. BPEL has to be the next step. Now the proper model transformation language has to be taken to find the way from the conceptual level to the implementation level. The most well-known transformation approaches [11] are the Query/Views/Transformation (QVT) approach [18], and the ATLAS Transformation Language [2]. The upcoming challenge is to create a MOF-compliant UML profile on metamodel level (M2), because both approaches, ATLAS and QVT operate in the M3 layered MOF-based metamodeling architecture [11], but the extension mechanism for creating profiles in UML is not a part of MOF.

9 Conclusion

In this paper, we have presented a UML 2 profile for integrating business process goals and performance measures into UML 2 Activity Diagrams. The profile provides an explicit illustration of the performance measures time, cost, and quality. Furthermore, it is possible to show the goals a business process must achieve, as well as the organisational structure that is concerned with alerts that belong to a measure. In order to capture these characteristics, we have extended the UML 2 metamodel for Activity Diagrams, and described them with stereotypes. Moreover, we have mapped the UML profile to BPEL, to transform a specific business process modelling language and its conceptually described performance measures into an execution language as well as to make it possible to monitor the process instances continuously. The UML profile and its mapping were tested with an example business process.

References

- [1] Aguilar, E. R., Ruiz, F., Garcia, F., Piattini M.: Evaluation Measures for Business Process Models, Proceedings of the 21st ACM Symposium on Applied Computing (SAC'06), April, Dijon, France, ACM Press, 2006.
- [2] Bezivin, J.: On the Unification Power of Models, Software and System Modeling Journal, Vol. 4 No. 2, pp. 171-188, 2005.
- [3] Bordbar B., Staikopoulos A.: On Behavioural Model Transformation in Web Services, Proceedings of the ER 2004 Workshops CoMoGIS, COMWIM, ECDM, CoMoA, DGOV, and ECOMO, Shanghai, China 2004, Springer Press, 2004.
- [4] BPMI: Business Process Modelling Notation, Business Process Management Initiative, Specification v.1.0, 2004. (06/05/31)
- [5] Casati F.: Industry Trends in Business Process Management – Getting Ready for Prime Time, Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA 2005), First International Workshop on Business Process Monitoring & Performance Management (BPMPM 2005), August 2005, Copenhagen, Denmark, IEEE Press.

- [6] Gardner, T., Amsden J., Griffin C., Iyengar S.: Draft UML 1.4 Profile for Automated Business Processes with a mapping to the BPEL 1.0. IBM alphaWorks (2003), http://www-128.ibm.com/developerworks/rational/library/content/04April/3103/3103_UMLProfileForBusinessProcesses1.1.pdf, (06/05/31)
- [7] Hammer, M.: *Beyond Reengineering – How the process-centered organization is changing our work and our lives*. Harper Collins Publishers, 1996.
- [8] Harrington, J.H.: *Business Process Improvement – The breakthrough strategy for total quality, productivity, and competitiveness*. McGraw-Hill, 1991.
- [9] IBM: *Business Process Execution Language for Web Services version 1.1*, <http://www-128.ibm.com/developerworks/library/specification/ws-bpel/>, (06/05/31)
- [10] Jacobson, I., Ericson, M., Jacobson, A.: *The Object Advantage – Business Process Reengineering with Object Technology*. ACM Press, Addison-Wesley Publishing, 1995.
- [11] Jouault F., Kurtev, I.: *On the Architectural Alignment of ATL and QVT*, Proceedings of the 21st ACM Symposium on Applied Computing (SAC'06), April, Dijon, France, ACM Press, 2006.
- [12] Kleppe, A., Warmer, J., Bast, W.: *MDA Explained. The Model Driven Architecture: Practice and Promise*, Addison-Wesley, April 2003.
- [13] Kueng P., Kawalek P.: *Goal-based business process models: creation and evaluation*, Business Process Management Journal, Vol. 3 No.1., pp. 17-38, MCB Press, 1997.
- [14] List B., Korherr B.: *An Evaluation of Conceptual Business Process Modelling Languages*, Proceedings of the 21st ACM Symposium on Applied Computing (SAC'06), April, Dijon, France, ACM Press, 2006.
- [15] Nurcan, S., Etien, A., Kaabi, R., Zoukar, I., Rolland, C.: *A Strategy Driven Business Process Modelling Approach*, Special issue of the Business Process Management Journal on "Goal-oriented business process modeling", Vol. 11 No. 6, pp. 628-649, Emerald, 2005.
- [16] Object Management Group, Inc.: *MDA Guide V.1.0.1*, <http://www.omg.org/cgi-bin/apps/doc?omg/03-06-01.pdf> (06/05/31)
- [17] Object Management Group, Inc.: *MOF 2.0 Specification*, <http://www.omg.org/cgi-bin/apps/doc?formal/06-01-01.pdf> (06/05/31)
- [18] Object Management Group, Inc.: *MOF QVT Final Adopted Version*, <http://www.omg.org/cgi-bin/apps/doc?ptc/05-11-01.pdf> (06/05/31)
- [19] Object Management Group, Inc.: *OCL 2.0 Specification*, <http://www.omg.org/cgi-bin/apps/doc?ptc/05-06-06.pdf> (06/05/31)
- [20] Object Management Group, Inc.: *UML 2.0 Superstructure*, <http://www.omg.org/cgi-bin/apps/doc?formal/05-07-04.pdf> (06/05/31)
- [21] Scheer, A.-W.: *ARIS – Business Process Modeling*. Springer Verlag, 1999.
- [22] Vitolins, V.: *Business Process Measures*, Proceedings of International Conference Baltic DB&IS 2004, June 2004, Riga, Latvia, Scientific Papers University of Latvia Vol. 673, University of Latvia, 2004.

UN/CEFACT'S Modeling Methodology (UMM): A UML Profile for B2B e-Commerce

B. Hofreiter¹, C. Huemer^{1,3}, P. Liegl², R. Schuster², and M. Zapletal³

¹ University of Vienna

{birgit.hofreiter, christian.huemer}@univie.ac.at

² Research Studios Austria DME

{pliegl, rschuster}@researchstudio.at

³ Vienna University of Technology

marco@ec.tuwien.ac.at

Abstract. The United Nation's Centre for Trade Facilitation and Electronic Business (UN/CEFACT) is an e-business standardization body. It is known from its work on UN/EDIFACT and ebXML. One of its on-going work items is the UN/CEFACT modeling methodology (UMM) for modeling global choreographies of B2B scenarios. The goal of UMM is defining a shared business logic between business partners and fostering reuse of standardized process building blocks. The latest UMM version is defined as a UML 1.4 profile. In this paper we introduce the main concepts of UMM to realize its vision. Furthermore, the paper elaborates on the necessary UML meta model work-arounds we - as part of the specification's editing team - took in order to accomplish the B2B requirements. Then we propose a move towards UML 2 that eliminates some of those workarounds.

1 Introduction

Automating the exchange of business information between business partners exists for a while. In the early days of electronic data interchange (EDI) the focus was limited to standardizing the business document types. However, the business documents must also be exchanged in an agreed order. The business processes between two different organizations participating in a collaborative business process must be defined. For this purpose a commonly accepted methodology is needed.

The United Nation's Centre for Trade Facilitation and Electronic Business (UN/CEFACT), known for its standardization work in the field of UN/EDIFACT and ebXML, took up the endeavor and started research for such a methodology. This on-going work resulted in UN/CEFACT's Modeling Methodology (UMM). UMM enables to capture business knowledge independent of the underlying implementation technology, like Web Services or ebXML. The goal is to specify a global choreography of a business collaboration serving as an "agreement" between the participating business partners in the respective collaboration. Each business partner derives in turn its local choreography, enabling the configuration of the business partner's system.

In order to guarantee user acceptance of the UMM, it must be both effective and easy to understand for the business process modelers and software architects. Due to the growing tool support of the Unified Modeling Language (UML), the decision in favor of UML as notation of UMM was already made in 1998. In the first years, UMM specified its own conceptual meta model and provided guidelines on creating compliant artifacts using the UML. In late 2004 it was decided to define the most recent UMM version as a UML profile [1], i.e. a set of stereotypes, tagged values and constraints - in order to customize the UML meta model for the special purpose of modeling global B2B choreographies. At this time the UML version of choice by UN/CEFACT was UML 1.4 [2]. This paper introduces the most important concepts of the UML 1.4 profile of UMM. Most attention is spent on necessary work-arounds in order to adjust the UML meta model to the special needs of UMM. A future transition of UML 1.4 to UML 2 as the basis of UMM will affect its UML profile. Thus, we will highlight the potential of such a move forward.

2 Related Work

In the world of Web Services a lot of different languages describing business processes exist, e.g. the Business Process Modeling Language (BPML) [3] and the Business Process Execution Language (BPEL) [4]. These languages are limited to orchestrations and local choreographies. The release of the Web Services Choreography Description Language (WS-CDL) draft [5] adds a specification for global choreographies to the family of Web Services which did not exist before. Within the ebXML framework, the Business Process Specification Schema (BPSS) [6] always describes the choreography of a business collaboration from a global perspective.

Since all the above mentioned languages are XML-based, there have been attempts to model them in a graphical syntax and/or to apply a model driven approach leading to them. The model driven approach is also in-line with the Open-edi reference model that became an ISO standard in 1997 [7]. Thereby Open-edi separates the *what* in the Business Operational View (BOV) from the *how* in the Functional Service View (FSV). The BOV covers the business aspects such as business information, business conventions, agreements and rules among organizations. The FSV deals with information technology aspects supporting the execution of business transactions. Accordingly, special UML profiles may be used on the BOV level, whereas the Web Services and ebXML languages are on the FSV level.

Several approaches using UML for business process modeling have been proposed [8] [9] [10]. However, these approaches focus on the modeling of business processes internal to an organization. Other approaches use UML to visualize Web Services and their choreography [11] [12]. More advanced approaches provide a development process for inter-organizational business processes. These are either driven by existing private workflows [13] or they are driven by the inter-organizational requirements instead of the private ones [14].

Also the UMM, which presents the core of this paper, is considered as a BOV-centric methodology. When UN/CEFACT and OASIS started the ebXML initiative, it was UN/CEFACT's vision that UMM is used to create BOV standards and that XML is used as key concept on the FSV layer. Accordingly, UMM is ebXML's modeling methodology, but it is not a mandatory part of ebXML (c.f. [6]). Since UMM stops at the BOV layer, a transformation to an IT solution on the FSV layer is required. In [15] we describe such a mapping from UMM to BPEL. Furthermore, we define a mapping from UMM models to BPSS in [16].

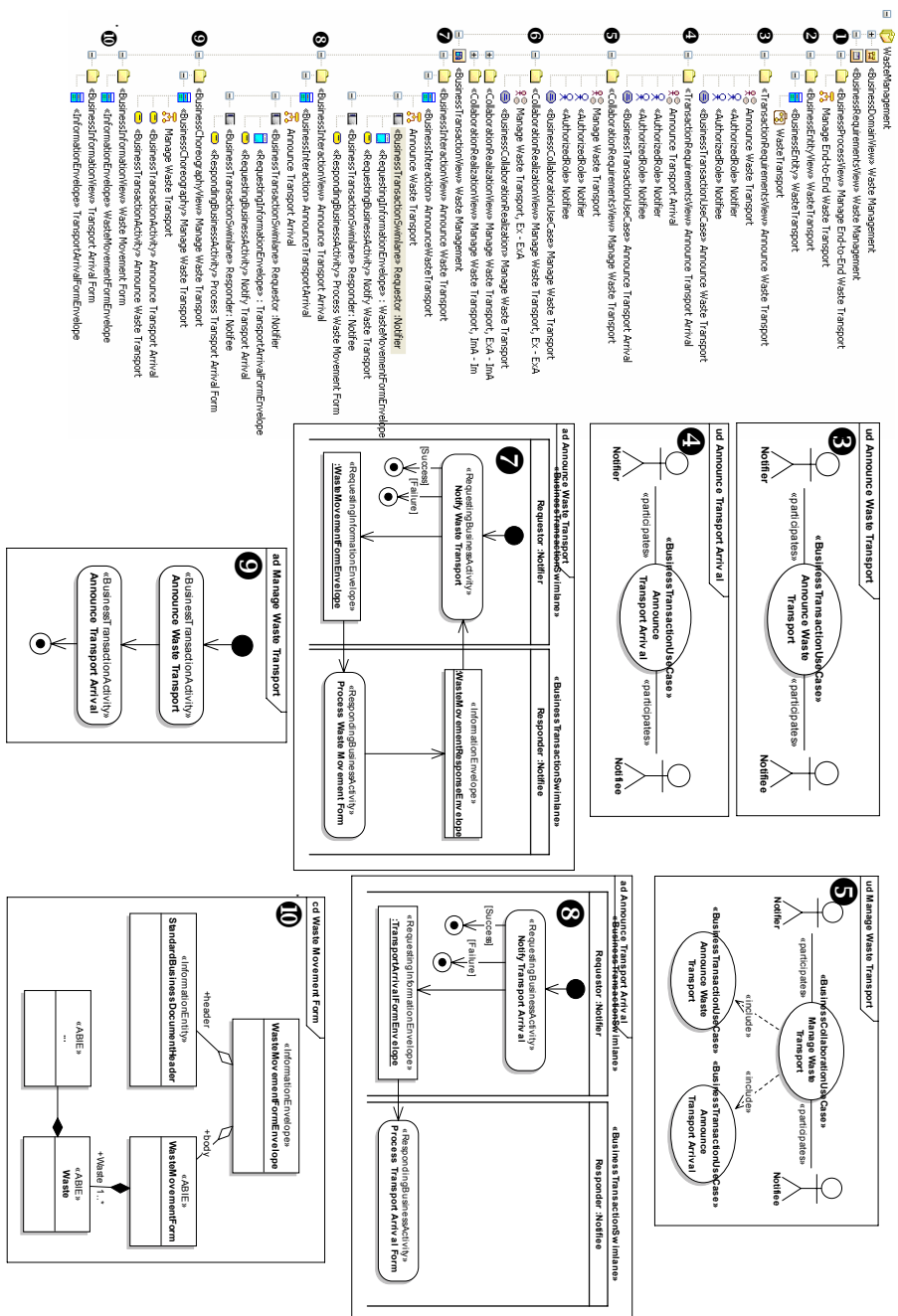
3 UMM by Example

In this section we briefly describe the steps of UMM and the resulting artifacts. For a better understanding we walk through the UMM by the means of a rather simple, but still realistic example. This example is akin to a project in the European **waste management** domain. Crossborder transports of waste - even within the EU - are subject to regulations. A transport must be announced, the receipt of the waste as well as the disposal of the waste must be signaled. Exporter, importer, the competent authorities in their countries and in transit countries interchange this information. In order to keep the example simple we do not consider the competent authorities of transit and we do not include the information about the waste disposal. However, in order to explain all concepts we assume that each individual transport must be approved which is not required in reality.

The UMM comprises three main views: *business domain view* (BDV), *business requirements view* (BRV), and *business transaction view* (BTV). The latter two are split into subviews. A UMM *business collaboration model* reflects this structure by creating packages for all these views and subviews (See left hand side of figure 1).

The BDV is used to gather existing knowledge from stakeholders and business domain experts. In interviews the business process analyst tries to get a basic understanding of the business processes in the domain. The use case descriptions of a *business process* are on a rather high level. One or more *business partners* participate in a *business process* and zero or more *stakeholders* have an *interest in* dependency with the process. The BDV results in a map of *business processes*, i.e. the *business processes* are classified. Thus the BDV package includes *business area* subpackages. UN/CEFACT suggests to use *business areas* according to the classification of Porters value chain (PVC) plus some administrative areas. Each *business area* consists of *process area* packages that correspond to the Open-edi phases (planning, identification, negotiation, actualization, and post-actualization) [7]. In our **waste management** example relevant *business areas* are logistics and regulation, each covering at least the *process areas* of actualization and post-actualization. We do not want to detail here all the processes that may be important to the domain experts and stakeholders in these areas.

Those *business processes* from the BDV that provide a chance for collaboration will be further detailed by the business process analyst in the BRV. The BRV consists of a number of different subviews. The *business process view*



(1 in figure 1) gives an overview about the *business processes*, their activities and resulting effects, and the *business partners* executing them. The activity graph of a *business process* may describe a single partner's process, but may also detail a multi-party choreography. The business process analyst tries to discover interface tasks creating/changing *business entities* that are shared between *business partners* and, thus, require communication with a *business partner*. In our example we detail a multi-party *business process* for a **waste transport**. The **exporter** pre-informs the **export authority** about a waste transport and expects an approval of the waste transport in return. In turn, the **export authority** announces the waste transport to the **import authority** to get the approval, and the **import authority** does the same with the **importer**. Later on when the waste is received by the **importer**, this information goes uni-directional back the chain from the **importer** to the **import authority** to the **export authority**, and finally to the **exporter**.

The information exchanged between *business partners* is about the *business entity waste transport*. Firstly, a **waste transport** entity is created with state **announced**. **Announced** is a kind of pending state because it requires a decision by the other *business partner* to set it either to **approved** or to **rejected**. Once an approved transport has happened it is set to **arrived**. These so-called *shared business entity states* must be in accordance with the *business entity lifecycle* of **waste transport**. This lifecycle is defined in the state chart of the *business entity view* (2).

It is obvious from the requirements described so far that the announcement together with the information of approval/rejection as well as the information of the waste receipt always occur between a different pair of *business partners*. It is not efficient to describe these tasks for each pair again and again. Instead, these tasks are defined between *authorized roles*. A *transaction requirements view* defines the *business transaction use case* for a certain task and binds the two *authorized roles* involved. In our example we have two *transaction requirement views*: **announce waste transport** (3) - which also includes the decision - and **announce transport arrival** (4). The *authorized roles* are in both cases a **notifier** who makes the corresponding announcement and a **notiffee**.

The *collaboration requirements view* includes a *business collaboration use case*. The *business collaboration use case* aggregates *business transaction use cases* and/or nested *business collaboration use cases*. This is manifested by *include* associations. In our example the *business collaboration use case manage waste transport* (5) includes the *business transaction use cases* **announce waste transport** (3) and **announce transport arrival** (4). Furthermore, the *authorized roles* participating in the *business collaboration use case* must be defined. Sometimes it is hard to find a good name for an *authorized role*, like in our example. We call the roles again **notifier** and **notiffee**. The **notifier** is the one who initiates the management of a waste transport and the **notiffee** is the one who reacts on it. A *business collaboration use case* may have many *business collaboration realizations* that define which *business partners* play which *authorized roles*. A detailed discussion of *business collaboration realizations* is provided in section 4.

The BTV builds upon the BRV and defines a global choreography of information exchanges and the document structure of these exchanges. The choreography described in the requirements of a *business transaction use case* is represented in exactly one activity graph of a *business transaction*. A *business transaction* is used to align the states of *business entities* in the information systems of the *authorized roles*. We distinguish one-way and two-way *business transactions*: In the former case, the initiating *authorized role* reports an already effective and irreversible state change that the reacting *authorized role* has to accept. This is the case in the *business transaction announce transport arrival* (8). In the other case, the initiating partner sets the *business entity/ies* into an interim state and the final state is decided by the reacting *authorized role*. It is a two-way transaction, because an *information envelope* flows from the initiator to the responder to set the interim state and backwards to set the final and irreversible state change. In the *business transaction announce waste transport* (7) the *business entity announce waste transport* is set into the interim state *announce* by the *notifier*, whereas the *notifiee* sets the final state of *approved* or *rejected*. Irreversible means that returning to an original state requires compensation by another *business transaction*.

A UMM *business transaction* follows always the same pattern: A *business transaction* is performed between two *authorized roles* that are already known from the *business transaction use case* and that are assigned to exactly one swimlane each. Each *authorized role* performs exactly one activity. An object flow between the *requesting* and the *responding business activity* is mandatory. An object flow in the reverse direction is optional. Both the two-way transaction *announce waste transport* (7) and the one-way transaction *announce transport arrival* (8) follow this pattern.

The activity graph of a *business transaction* shows only the exchange of business information in the corresponding envelopes. It does not show any business signals for acknowledgements. The *acknowledgment of receipt* - sent for a valid document that also passed sequence validation - and the *acknowledgment of processing* - sent for documents that have been checked against additional business rules before importing them into the business application - are specified by maximum time values in the tagged values of the *requesting* and *responding business activity*. Further tagged values are the maximum *time to respond*, and flags for *authorization*, *non-repudiation* of original and of receipt, and a *retry counter* for reinitiating the *business transactions* in case of control failures. The *information envelopes* are characterized by tagged values to signal *confidential*, *tamper proof* and *authenticated* exchanges.

According to the UMM *business transaction* semantics, the *requesting business activity* does not end after sending the envelope - it is still alive. The *responding business activity* may output the response which is returned to the still living *requesting business activity*. This interpretation may be curious for UML purists - however, it was already introduced by the RosettaNet [17] modeling approach and is well accepted by the e-business community.

The requirements described in a *business collaboration use case* are choreographed in the activity graph of a *business collaboration protocol* which is defined in a *business choreography view*. In our example, the **manage waste transport** requirements (5) are mapped to the homonymous *business collaboration protocol* (9). A *business collaboration protocol* choreographs a set of *business transaction activities* and/or *business collaboration activities*. A *business transaction activity* is refined by the activity graph of a *business transaction*. In our example, the *business collaboration protocol* of **manage waste transport** (9) is a simple sequence of two *business transaction activities*: **announce waste transport** and **announce transport arrival**. Each of them is refined by its own *business transaction* (7,8). *Business transaction activities* have tagged values for a maximum *time to perform* and an indicator whether *concurrent* execution is allowed or not. *Business collaboration activities* - which are not used in our example - are refined by a nested *business collaboration protocol*.

Finally, the information exchanged in transactions must be unambiguously defined. Each object in an object flow state is an instance of a class representing an envelope. The aggregates within this envelope are defined in a class diagram. Figure 1 includes a - due to space limitations - very limited extract of the class diagram for the **waste movement form** envelope (10), which is exchanged in the *business transaction announce waste transport* (7). The business document is assembled from reusable building blocks called *core components* [18] [19]. By using a core component in a business document it is adjusted to the document's business context, e.g. by eliminating attributes not needed. Once the core component is adjusted it becomes a so-called *business information entity*. In (10) we just highlight one *business information entity waste* being part of a **waste movement form**. We do not list any attributes, neither we show any other *business information entities* and relationships among them.

4 UMM Meta Model Workarounds

As mentioned before, UMM is based on the UML 1.4 meta model [2]. Accordingly, a UMM *business collaboration model* is a UMM 1.4 compliant model. However, some concepts may appear unfamiliar to a UML modeler who has not used UMM before. These concepts are a result of the specific B2B requirements of UMM.

4.1 Mapping of Authorized Roles

One of the key goals of UMM is to foster reuse. This implies that a *business transaction use case* may be included in many *business collaboration use cases*. Consider for example, the *business transaction use case announce transport arrival* is part of another *business collaboration use case* in the logistics domain.

For the purpose of reuse, the *authorized roles* are defined in the very specific context of a *business transaction*. A *business collaboration use case* that includes the *business transaction* also defines participating *authorized roles* in its specific context. It is the peculiarity of UMM that a certain *authorized role* of the *business*

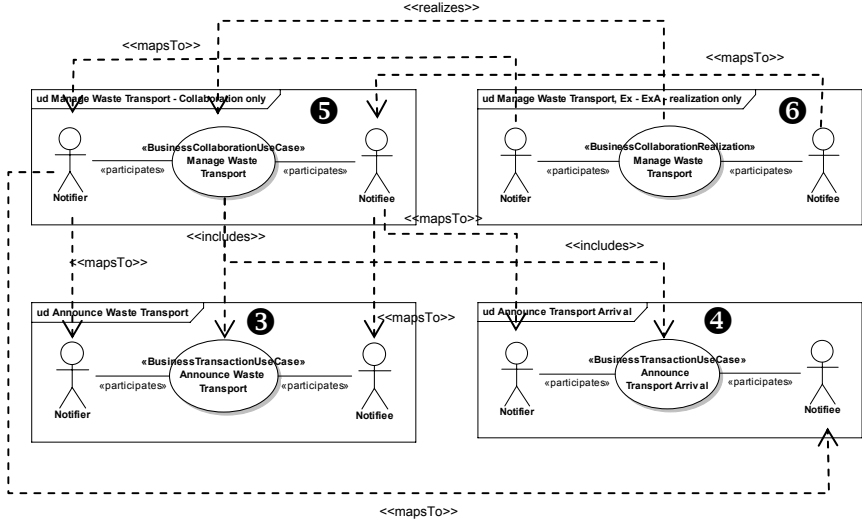


Fig. 2. Mapping of Authorized Roles

collaboration use case must take on an *authorized role* of an included *business transaction use case*. For this purpose UMM uses *maps to* dependencies defining which *authorized role* of a *business collaboration use case* plays which role in an included *business transaction use case* (or nested *business collaboration use case*).

This concept is easily demonstrated by our **waste management** example (see figure 1). The *business collaboration use case* **manage waste transport** (5) includes two *business transaction use cases*: **announce waste transport** (3) and **announce transport arrival** (4). By coincidence, the roles of all three use cases are **notifier** and **notifiee**. However, this does not mean that a **notifier** always maps to a **notifier**. In our example the **notifier** of **manage waste transport** (5) plays also the **notifier** of **announce waste transport** (3), but plays the **notifiee** in **announce transport arrival** (4) since the information flows the other way round. For the **notifiee** of **manage waste transport** it is just the opposite. It is obvious, that **notifier** must be a different *authorized role* in each of the three use cases, however with a homonymous name. Accordingly, *authorized roles* are always defined in the namespace of its *transaction requirements view*. This is easy to recognize in the tree view of our example on the left side of figure 1.

In section 3 we already learned that the same **manage waste transport** *business collaboration* must be realized between different pairs of *business partners*. Thus, we need different *business collaboration realizations* - each defining which *business partner* plays which role in it. Accordingly, our waste example results in three *business collaboration realizations* of **manage waste transport** - one between **exporter** and **export authority**, one between **export authority** and **import authority**, and one between **import authority** and **importer**. Figure 2 depicts the first one as a representative (6). The *business partners* participating in the

business collaboration realization are the ones already defined in the BDV and, thus, are not re-defined in the namespace of the *collaboration realization view*. However, each *business collaboration realization* defines *authorized roles* which are usually - but not necessarily - homonymously named as the ones of the corresponding *business collaboration use case*. The previously introduced concept of *maps to* dependencies is used to map both the *authorized roles* from a *business collaboration realization* to a *business collaboration use case* as well as *business partners* to *authorized roles* of the *business collaboration realization*. In the `manage waste transport` realization (6) of figure 2 the `exporter` plays the `notifier` and the `export authority` acts as `notified`.

4.2 Reusing a Business Transaction in Many Business Collaboration Protocols

The fact that a *business transaction use case* (and a nested *business collaboration use case*) may be included in many *business collaboration use cases* has another implication which is not perfectly met by the UML 1.4 meta model. Each *business collaboration use case* leads to exactly one *business collaboration protocol*. Each included *business transaction use case* will result in a *business transaction* that is part of the corresponding *business collaboration protocol*. In the activity graph of a *business collaboration protocol* the activity graph of a *business transaction* is represented by a *business transaction activity*. Since a *business transaction use case* may be the target of many *include* associations, it follows that the same *business transaction* may be part of different *business collaboration protocols*.

One might think that this concept is reflected in the UML 1.4 meta model. A *business transaction activity* is a UML subactivity state which is refined by a UML activity graph. The UML 1.4 meta model defines a 1:1-relationship between subactivity states and activity graphs. However the relationship between *business transaction activity* and *business transaction* must be n:1. Consequently, UMM uses again a *maps to* dependency to realize the relationship between *business transaction activities* and *business transactions*. A *business transaction activity* is the source for only one *maps to* dependency and a *business transaction* may be the target of many *maps to* dependencies.

4.3 Mapping of Use Cases and Their Activity Graphs in Different Packages

Back in the early days of UMM development, in the late 1990s, the project team decided that the structure of a UMM model follows the UMM views (cf. left hand side of figure 1). In the meantime the user community got used to this structure and changing it would create too much confusion. In UML, an activity graph may go beneath a use case describing its requirements. However, in the UMM structure the *business transaction use cases* and the *business collaboration use cases* are located in other packages than the corresponding *business transaction* or *business collaboration protocol*, respectively. Accordingly, the relationship between a *business transaction use case* and a *business transaction* as well as

the one between a *business collaboration use case* and a *business collaboration protocol* are realized by another *maps to dependency*.

5 Moving Towards UML 2

With the growing acceptance of UML 2, there is a desire to move UMM towards UML 2. As we learned, UMM is based on use cases to model requirements, on classes to model business documents, and on activity diagrams to model choreographies. Since UML 2 made major changes to modeling activity diagrams, major changes must be made to UMM. In UML 1.4 *activity graphs* were specialized *state machines*. In UML 2 they have been replaced by the concept of an *activity*. An *activity* captures user-defined behavior by describing a flow of *actions* and their interaction with *objects* representing data. An *action* is a fundamental unit to describe a step of work within the execution of an *activity*. In this section we propose a transition of UMM *business collaboration protocols* and *business transactions* to UML 2 using the **waste management** example. Furthermore, we will show how to eliminate the workarounds introduced in section 4 using UML 2 concepts.

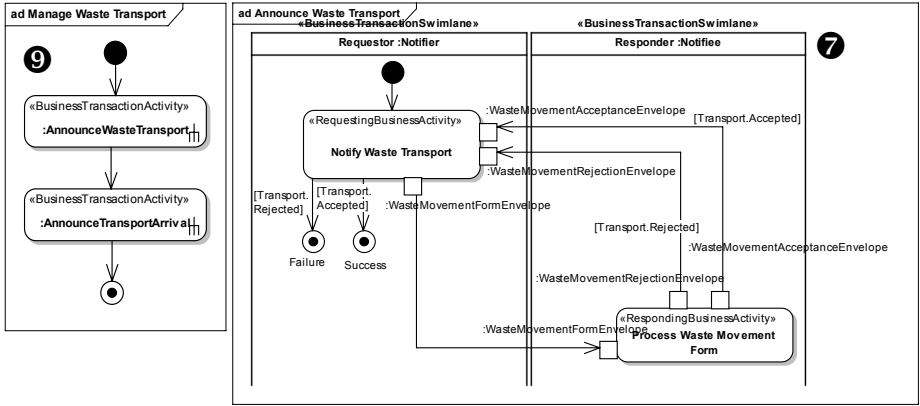


Fig. 3. Waste management example using UML 2

The *business collaboration protocol* **manage waste transport** on the left hand side of figure 3 is composed of two *business transaction activities* - **announce waste transport** and **announce transport arrival**. In UML 2 the *business collaboration protocol* becomes an *activity*. Each of the two *business transaction activities* become an *action*. We already know that *business transaction activities* are refined by a *business transaction*. A *business transaction* also becomes an *activity* in UML 2. In order to refer from a *business transaction activity* to its refining *business transaction* we utilize the predefined action type *call behavior action*. A *call behavior action* - indicated by the rake-symbol in the

lower right corner of a *business transaction activity* in figure 3 - allows the call of another *activity*. This eliminates the corresponding *maps to* dependency in the current UMM. In our example the first *business transaction activity* calls the **announce waste transport business transaction** and the second one calls the **announce transport arrival business transaction**.

A *business transaction* is always composed of a *requesting* and a *responding business activity* - each of them become *actions*. Since the implementation of these activities together with their interfaces within an application are partner specific we use the subtype *opaque action*. This indicates that the "semantics of the action are determined by the implementation" [20]. In UML 2 we prefer to notate the information flows between these two actions by the new pin notation. The right hand side of Figure 3 shows this new notation for the *business transaction announce waste transport*. An output pin of a *requesting business activity* and an input pin of a *responding business activity* are assigned with a *requesting information envelope* object. An output pin of a *responding business activity* and an input pin of a *requesting business activity* are assigned with a *responding information envelope* object.

Considering the response in case of two-way transactions, we suggest an extension to the UMM transaction concept. The current UMM transaction concept allows only one type of *responding information envelope*. Usually, the type of response differs significantly in case of a positive and a negative response. In the current UMM we must use an abstract super type for the positive and negative response. We propose multiple output pins to *responding business activities* and multiple input pins to *requesting business activities* in order to show different types of object flows. These object flows are guarded by mutually exclusive constraints. The **announce waste transport business transaction** on the right hand side of figure 3 defines the exchange of a **waste movement form envelope** between the *requesting business activity notify waste transport* and the *responding business activity process waste movement form*. A **waste movement acceptance envelope** is returned in case of an accepted transport. A **waste movement rejection envelope** is sent back if the transport is rejected. This approach narrows the gap between process- and data modeling in UMM.

Last, but not least UML 2 enables eliminating the *maps to* dependency between a *business transaction* and a *business transaction use case* and also between a *business collaboration protocol* and a *business collaboration use case*. In UML 2, a use case might be associated to an arbitrary *classifier* indicating that the *classifier* realizes the use case. Since *activity* inherits from *classifier*, we are able to connect the activities to the corresponding use cases without *maps to* dependencies.

6 Summary

In this paper we have introduced UN/CEFACT's Modeling Methodology (UMM) which we have co-edited. UMM defines a UML 1.4 profile - i.e. a set of stereotypes, tagged values and constraints - in order to customize the UML meta

model for the special purpose of modeling collaborative business processes from a global view. We demonstrated the steps of the UMM by a simple example of the **waste management** domain. This example reveals most of the stereotypes defined in UMM. Due to space limitation we were not able to go into all the details of the tagged values of each stereotype. Furthermore, we preferred to show the relationships between the stereotypes by means of the example, rather than introducing the equivalent set of OCL constraints as defined in the UMM specification.

Furthermore, we elaborated those concepts that are very specific to UMM's UML profile. These specifics include the mapping of authorized roles participating in a parent use case to the authorized roles participating in an included use case, as well as on the mapping of business partners to authorized roles in a use case realization. Another UMM specialty is that an activity graph may be included in multiple parent activity graphs. Since the UML 1.4 meta model defines a 1:1 relationship between a subactivity state and the refining activity graph, we had to implement a workaround with dependencies between the subactivity state and the refining activity graph to realize an n:1-relationship. Finally, we also used dependencies to trace between a use case and its realizing activity graph located in different packages.

Due to the growing acceptance of UML 2, it is predictable that the UML profile will move towards UML 2 in the near future. Since the concept of modeling activity diagrams changed dramatically in UML 2, we outlined the consequences of such a movement. Furthermore, we have shown the significance of a UMM tool, supporting the modeler in creating a UMM compliant model. The University of Vienna and the Research Studios Austria are committed to the development of the UMM Add-In and will adapt it to future versions of the UMM standard.

References

1. UN/CEFACT Techniques and Methodologies Group: UN/CEFACT's Modeling Methodology (UMM), UMM Meta Model - Foundation Module. (2006) Candidate for 1.0, Final Working Draft, http://www.unece.org/cefact/umm/UMM_Foundation_Module.pdf.
2. Object Management Group (OMG): Unified Modeling Language Specification. (2005) Version 1.4.2, <http://www.omg.org/docs/formal/05-04-01.pdf>.
3. Arkin, A.: Business Process Modeling Language (BPML). Technical report (2002)
4. BEA, IBM, Microsoft, SAP AG and Siebel Systems: Business Process Execution Language for Web Services. (2003) Version 1.1, <ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf>.
5. World Wide Web Consortium (W3C): Web Services Choreography Description Language. (2005) Version 1.0, <http://www.w3.org/TR/ws-cdl-10/>.
6. UN/CEFACT Techniques and Methodologies Group: UN/CEFACT - ebXML Business Process Specification Schema. (2003) Version 1.10, http://www.untmg.org/dmdocuments/BPSS_v110_2003_10_18.pdf.
7. ISO: Open-edi Reference Model. (1995) ISO/IEC JTC 1/SC30 ISO Standard 14662.

8. Penker, M., Penker, M., Eriksson, H.E.: Business Modeling With UML: Business Patterns at Work. Wiley (2000)
9. Vasconcelos, A., Caetano, A., Neves, J., Sinogas, P., Mendes, R., Tribolet, J.: A framework for modeling strategy, business processes and information systems. In: EDOC '01: Proceedings of the 5th IEEE International Conference on Enterprise Distributed Object Computing, IEEE Computer Society (2001)
10. List, B., Korherr, B.: A uml 2 profile for business process modelling. In: ER 2005 Workshops Proceedings. (2005)
11. Gardner, T.: UML Modelling of Automated Business Processes with a Mapping to BPEL4WS. In: 1st European Workshop on Object Orientation and Web Services (EOOWS'03), Springer (2003)
12. Thöne, S., Depke, R., Engels, G.: Process-oriented, flexible composition of web services with uml. In: Conceptual Modeling - ER 2002, 21st International Conference on Conceptual Modeling, Proceedings. LNCS, Springer (2002)
13. Jung, J.Y., Hur, W., Kang, S.H., Kim, H.: Business process choreography for b2b collaboration. IEEE Internet Computing **8**(1) (2004) 37–45
14. Kramler, G., Kapsammer, E., Kappel, G., Retschitzegger, W.: Towards Using UML 2 for Modelling Web Service Collaboration Protocols. In: Proceedings of the First International Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA'05). (2005)
15. Hofreiter, B., Huemer, C.: Transforming UMM Business Collaboration Models to BPEL. In: Proceedings of OTM Workshops 2004. Volume 3292., Springer LNCS (2004) 507–519
16. Hofreiter, B., Huemer, C., Kim, J.H.: Choreography of ebXML business collaborations. Information Systems and e-Business Management (ISeB) (2006) Springer.
17. RosettaNet: RosettaNet Implementation Framework: Core Specification. (2002) V02.00.01, <http://www.rosettanet.org/rnif>.
18. UN/CEFACT Techniques and Methodologies Group: Core Components Technical Specification - Part 8 of the ebXML Framework. (2003) Version 2.01, http://www.unece.org/cefact/ebxml/CCTS_V2-01_Final.pdf.
19. UN/CEFACT Techniques and Methodologies Group: UML Profile for Core Components based on CCTS 2.01. (2006) Candidate for Version 1.0.
20. Object Management Group (OMG): Unified Modeling Language Specification. (2007) Version 2.0, <http://www.omg.org/docs/formal/05-07-04.pdf>.

Capturing Security Requirements in Business Processes Through a UML 2.0 Activity Diagrams Profile

Alfonso Rodríguez¹, Eduardo Fernández-Medina², and Mario Piattini²

¹ Departamento de Auditoría e Informática, Universidad del Bío Bío,
Chillán, Chile
alfonso@ubiobio.cl

² ALARCOS Research Group, Information Systems and Technologies Department,
UCLM-Soluziona Research and Development Institute,
University of Castilla-La Mancha, Ciudad Real, Spain
{Eduardo.FdezMedina, Mario.Piattini}@uclm.es

Abstract. Security has become a crucial aspect for the performance of present organizations since the protected object is the mission of them. Therefore, the management approach oriented to business processes has been a good answer for the current scenarios, changing and complex, where organizations develop their task. Both subjects form a basic requirement to reach not only the mission but also the organizational objectives in a strongly connected global economy. In this work, we will show a microprocess through which it is possible to specify and refine security requirements at a high level of abstraction, in a way that they can be incorporated into the development of a software system. In addition, an extension of UML 2.0 activity diagrams will be presented through which it is possible to identify such requirements.

1 Introduction

The new business scene, where there are many participants and an intensive use of communications and information technologies, implies that enterprises not only expand their businesses but also increase their vulnerability. As a consequence, with the increase of the number of attacks on systems, it is highly probable that sooner or later an intrusion can be successful [22].

Regardless of the importance of the security notion for companies, this is often neglected in business process models, which usually concentrate on modeling the process in a way that functional correctness can be shown [3] mainly due to the fact that the expert in the business process domain is not an expert in security [10]. Typically, security is considered after the definition of the system. This approach often leads to problems, which most of the times are translated into security vulnerabilities [19], which clearly justify the need of increasing the effort in the pre-development phases, where fixing the bugs is cheaper [16].

If we consider that empirical studies show that it is common at the business process level that customers and end users are able to express their security needs [16], then it is possible to capture at a high level, security requirements easily identifiable by those who models business processes. Besides, requirements specification usually results in

a specification of the software system which should be as exact as possible [2], since, effective business process models facilitate discussions among different stakeholders in the business, allowing them to agree on the key fundamentals and to work towards common goals [6].

In our proposal, we consider the definition of a microprocess that complements the requirements capture defined in the Unified Software Development Process [11] and we have defined a UML 2.0 activity diagrams profile to capture security requirements.

The structure of the rest of the paper is the following: in Section 2, we will summarize the main issues about security in business processes. In Section 3, we will present a brief overview of UML 2.0 activity diagrams and profiles. In Section 4, we will propose a microprocess for the security requirements specification and a UML 2.0 profile that allows the business analyst to carry out this task. Finally, in Section 5, we will present an example and in Section 6 our conclusion will be drawn.

2 Security in Business Process

In spite of the importance of security for business processes, we have found out two problems. The first one is that modeling has not been adequate since, generally, those who specify security requirements are requirements engineers that have accidentally tended to use architecture specific restrictions instead of security requirements [7]. And in the second place, security has been integrated into an application in an ad-hoc manner, often during the actual implementation process [3], during the system administration phase [15] or it has been considered like outsourcing [18].

Moreover, capturing the security requirements of a system is a hard task that must be established at the initial stages of system development, and business spruces offer a view of business structure that is very suitable as a basis for the elicitation and specification of security requirements. Business process representations may in this way present in all stages of system development different levels of abstraction appropriate for each stage [16]. Consequently, we believe that business analysts can integrate their view on business security into the business process perspective. In addition, security requirements since any application at the highest level of abstraction will tend to have the same basic kinds of valuable and potentially vulnerable assets [8].

In the review of related works, we have had the possibility to check that not only in those works directly referring to security regarding business processes [3, 10, 17, 23, 24, 27] but also in those that have to do with security and information systems [1, 2, 4, 12, 15, 19, 25, 28], security specifications made by the business analyst are absent. Moreover and in spite of the fact that in some of these works, UML is used for security specifications, none of them use the activity diagrams available in UML 2.0.

3 UML 2.0 Activity Diagrams and UML 2.0 Profiles

Activity diagrams are the UML 2.0 elements used to represent business processes and workflows [13]. In UML previous versions, expressivity was limited and this fact confused users that did not use the orientation to objects as an approach for modeling. Now, it is possible to support flow modeling across a wide variety of domains [5]. An

activity specifies the coordination of executions of subordinate behaviors, using a control and data flow model. Activities may form invocation hierarchies invoking other activities, ultimately resolving to individual actions [20]. The graphical notation of an activity is a combination of nodes and connectors that allow us to form a complete flow.

On the other hand, the Profiles package contains mechanisms that allow meta-classes from existing metamodels to be extended to adapt them for different purposes. The profiles mechanism is consistent with the Meta Object Facility (MOF) [20]. UML profiles consist of Stereotypes, Constraints and Tagged Values. A stereotype is a model element defined by its name and by the base class to which it is assigned. Constraints are applied to the stereotype with the purpose of indicating limitations (e.g. invariants). They can be expressed in natural language, programming language or through Object Constraint Language (OCL). Tagged values are additional meta-attributes assigned to a stereotype, specified as name-value pairs.

Research works related to UML 2.0 profiles and business processes refer to aspects of the business such as Customer, kind of Business Process, Goal, Deliverable and Measure [14], Data Warehouse and its relation to business process dynamic structures [26] or they add semantics to the activities considering organizational aspects that allow us to express resource restrictions during the execution of an activity [13].

4 Microprocess and UML 2.0 Profile for Security Requirements

Requirements specification is a stage that has been taken into account in the most important software construction models such as the traditional waterfall model, the prototype construction, the incremental model, and the spiral model, among others. [21]. In these models, it is considered a stage in which we should obtain the system requirements either from the client or from the interested people in order to start the software construction from that point.

Our proposal studies a microprocess that complements the specification of the system context defined in the Unified Process [11] paying special attention to security requirements capture. To do so, a UML 2.0 activity diagrams profile is proposed.

4.1 SeReS4BP Microprocess

We have considered the use of the Unified Software Development Process stated by Jacobson, Booch y Rumbaugh (2000) since it is a quite consolidated and successful software construction method [9]. This process is composed by a set of activities that allow us to transform a user's requirements into a software system.

In the Unified Process, requirements capture is mainly done during the inception and elaboration stages. The objective of this task is to make a good enough description of the system's requirements (conditions and capabilities that must be fulfilled by the system) to determine what the system must or must not do. To do so, it is considered the performance of an enumeration of the requirements of the candidates, the understanding of the system context, and the capture of both functional and non functional requirements.

Stage	Worker	Tool	Artifact
Construction	 Business Analyst	 UML 2.0 Activity Diagrams	 Business Process Model
Security requirements incorporation	 Business Analyst	 BPsec 1.0	 Business Process Model with Security Specifications (preliminary specifications)
Refining	 Security Expert  Business Analyst	 UML 2.0 Activity Diagrams and BPsec 1.0	 Business Process Model with Security Specifications (final specifications)  Business Process Repository with Security Specifications (update)

Fig. 1. Complete view of the SeReS4BP microprocess

The security requirements specified in the business process can be perfectly linked to the Unified Process. To do so, we propose to complement the task “to understand the system context” with specifications of the domain built by the business analyst. Our proposal is a microprocess that considers the necessary activities that allow us to specify requirements (particularly, security requirements) taking into account the business analyst’s perspective. This microprocess is called SeReS4BP (Security Requirement Specification for Business Process). Figure 1 shows us a view of the main activities performed in this microprocess and Table 1 shows us a details description.

Table 1. SeReS4BP activities

Stages: <i>Construction:</i> whose objective is the business process model construction. To reach this objective, the UML 2.0 activity diagram must be used. <i>Security requirements incorporation:</i> this stage consists of incorporating security requirements, from the business analyst viewpoint, into the business process model that was specified in the previous stage. <i>Refining:</i> This stage corresponds to the review and complementing of the security specifications that have been incorporated into the business process. At this stage, the business analyst and the security expert work together. The specifications that will be finally incorporated into the business process will be agreed at this stage. Workers: <i>Business Analyst:</i> he/she will be responsible for the specifications related to the business itself as well as for incorporating, from his/her point of view, security requirements into the specifications considering a high level of abstraction. <i>Security Expert:</i> he/she will be the responsible for refining the security specifications indicated by the business analyst. Such refining considers the verification of the specifications validity and complementation. Tools: <i>UML 2.0 Activity Diagrams</i> for the business process specification. <i>BPsec 1.0</i> for security requirements specifications Artifacts: <i>Business process model:</i> This artifact is the result of the construction stage. It contains the business process specifications and it can be built using UML. It does not contain security specifications. <i>Business Process Model with Security Specifications.</i> This artifact is the result of the stages of incorporation of security requirements and refining. The first stage contains security preliminary specifications that, after refining, will be converted into definitive security specifications. <i>Business Process Repository</i> that contains security specifications. This repository is composed of a set of business processes that have security requirements already incorporated. This repository must be updated with the business process resulting from the refining stage.

4.2 BPsec Version 1.0 for Modeling Security Requirements in Business Processes

In this section, we will present the main aspects of our profile for representing security requirement in business process. Our proposal allows business analysts to specify security requirements in the business process by using activity diagrams. We have considered the security requirements identified in the taxonomy proposed in [8]. Later on, these requirements will be transformed, by the security experts, into technical specifications including all necessary details for their implementation.

Our Profile will be called BPsec (Secure Business Process) and will be represent as a UML Package. This profile will incorporate new data types, stereotypes, tagged value and constrains. In Figure 2, a high level view is provided.

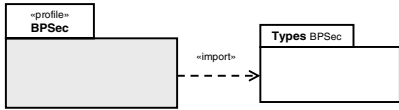


Fig. 2. High level view of BPSec Profile

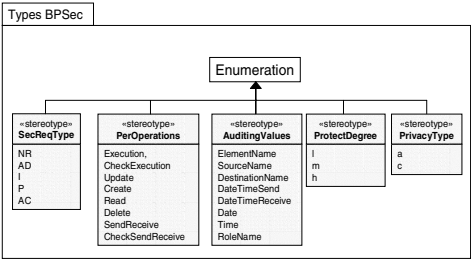


Fig. 3. Value associated to new data type

In addition we need the definitions of some new data types to be used in tagged value definitions. In Table 2, we will show the new data type stereotypes definitions.

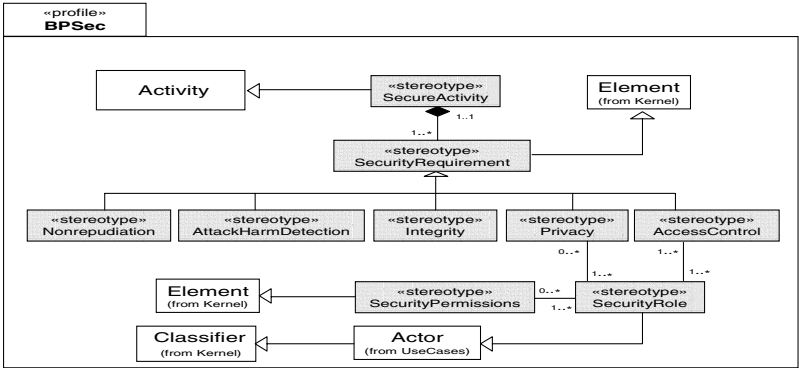


Fig. 4. New Stereotypes

In Figure 3, we can observe the values associated to each one of the necessary type. All the new type must be considered when the business analysts to specify security requirements in business process. We have defined a package that includes all stereotypes that will be necessary in our profile. In Figure 4 we show the stereotypes (in dark) for Secure Activity specifications.

Table 2. New data types

Name	Description	Values associated
SecReqType	It represents a type of security requirement. It must be specified for Non Repudiation, Attack/Harm Detection, Integrity, Privacy or Access Control.	NR, AD, I, P, AC
PerOperations	It is an enumeration for possible operations over objects in activity diagrams. These operations are related to permissions granted over the object	Execution, CheckExecution, Update, Create, Read, Delete, SendReceive, CheckSendReceive
ProtectDegree	It is an abstract level that represents criticality. This degree can be low (l), medium (m) or high (h).	l, m, h
PrivacyType	It consists of anonymity (a) or confidentiality (c).	a, c
AuditingValues	It represents different security events related to the security requirement specification in business processes. They will be used in later auditing	ElementName, SourceName, DestinationName, DateTimeSend, DateTimeReceive, Date, Time, RoleName

A Secure Activity is a stereotype derived from Activity. «SecureActivity» is strongly associated with security requirements stereotypes. «SecurityRequirement» has a composition relationship with «SecureActivity». The proposed notation for

«SecurityRequirement» must be complemented by adding it letters that will allow us to identify the type of requirement that is specified.

The stereotypes derived from «SecurityRequirement» can be *added* to activity diagrams elements. Any security requirement can be added to activity diagram elements (see Table 3). For example, an «Integrity» requirement can be specified over data store, control flow or object flow.

«SecurityRole» and «SecurityPermissions» are related in different ways; because both can be *obtained* from the UML 2.0 element of activity diagrams (see Table 3). For example, «SecurityRole» can be obtained from activities, partitions or regions specifications, but it is not specified in an explicit way over these activity diagrams elements. «SecurityPermission» is a special case, because, permissions depending on each activity diagram element which they are related to. For example, for Actions object, Execution or CheckExecution operations must be specified (see Table 5).

Table 3. Security Requirements and Activity Diagram Elements

Stereotypes for secure activity specification	UML 2.0 element for containment in activity diagrams					
	Activity	Activity Partition	Interruptible Activity Region	Action	Data Store Node	Object Flow
Nonrepudiation						✓
AttackHarmDetection	✓	✓	✓	✓	✓	✓
Integrity					✓	✓
Privacy		✓				
AccessControl	✓	✓	✓			
Security Role	✓	✓	✓			
SecurityPermissions				✓	✓	✓

In Table 4 we show the stereotypes for secure activity specifications extensively. Each stereotype specification contains: name, base class, description, notation (optional), constrains and tagged values (optional).

Table 4. Stereotypes specifications for security requirement

Name	SecureActivity
Base Class	Activity
Description	A secure activity contains security specification related to requirements, role identifications and permissions
Constrains	It must be associated at least with one SecurityRequirement context SecureActivity inv: self.SecurityRequirement->size()>=1
Name	SecurityPermission
Base Class	Element (from Kernel)
Description	It contains permission specifications. A permissions specification must contain details about the objects and operations involved
Constrains	It must be associated with security role specification context SecurityPermission inv: self.SecurityRole->size()>= 1 It must be associated with Actions, DataStoreNode or ObjectFlow context SecurityPermissions inv: self.Actions.size+self.DataStoreNode.size+self.ObjectFlow.size=1 It must be specified such as Objects and Operations pairs. context SecurityPermissions inv: if self.Actions->size()=1 then self.SecPerOperations="Execution" or self.SecPerOperations="Checkexecution" endif if self.Datastorenode->size()=1 then self.SecPerOperations="Update" or self.SecPerOperations="Create" or self.SecPerOperations="Read" or self.SecPerOperations="Delete" endif if self.Objectflow->size()=1 then self.SecPerOperations="Sendreceive" or self.SecPerOperations="Checksendreceive" endif
Tagged Values	SecurityPermissionOperation: SecPerOperations
Name	SecurityRole
Base Class	Actor (from UseCases)
Description	It contains a role specifications. This roles must be obtained from access control and/or privacy specifications

Table 4. (continued)

Constrains	The role in the security role stereotype can be derived from: Activity, ActivityPartition and/or InterruptibleActivityRegion It must be associated with an access control specification and can be associated with privacy and security permissions context SecurityRole inv: self.AccessControl -> size() >= 1 context SecurityRole inv: self.Privacy -> size()>= 0 context SecurityRole inv: self.SecurityPermission -> size()>= 0	
Name	SecurityRequirement	Notation 
Base Class	Element	
Description	Abstract class containing security requirements specifications. Each security requirement type must be indicated in some of its subclasses	
Constrains	A security requirement must be associated with a secure activity context SecurityRequirement inv: self.SecureActivity ->size()=1 The notation must be completed in the subclass specification for each security requirement. It must be used one security requirement type.	
Tagged Values	SecurityRequirementType: SecReqType	
Name	Nonrepudiation	Notation 
Base Class	SecurityRequirement	
Description	It establishes the need to avoid the denial of any aspect of the interaction. An auditing requirement can be indicated in Comment	
Constrains	It can be only specified in the diagram elements indicated in Table 3.	
Tagged Values	AvNr: AuditingValues context Nonrepudiation inv: self.AvNr="ElementName" or self.AvNr="SourceName" or self.AvNr="DestinationName" or self.AvNr="DateTimeSend" or self.AvNr="DateTimeReceive"	
Name	AttackHarmDetection	Notation 
Base Class	SecurityRequirement	
Description	It indicates the degree to which the attempt or success of attacks or damages is detected, registered and notified. An auditing requirement can be indicated in Comment	
Constrains	It can be only specified in the diagram elements indicated in Table 3.	
Tagged Values	AvAD: AuditingValues context AttackHaarmDetection inv: self.AvAD="ElementName" or self.AvAD="Date" or self.AvAD="Time"	
Name	Integrity	Notation 
Base Class	SecurityRequirement	
Description	It establishes the degree of protection of intentional and non authorized corruption. The elements are protected from intentional corruption. An auditing requirement can be indicated in Comment.	
Constrains	It can be only specified in the diagram elements indicated in Table 3. The Protection Degree must be specified by adding a lower case letter according to PDI tagged value.	
Tagged Values	PDI : ProtectDegree AvI: AuditingValues context Integrity inv: self.AvI="ElementName" or self.AvI="Date" or self.AvI="Time"	
Name	Privacy	Notation 
Base Class	SecurityRequirement	
Description	It indicates the degree to which non authorized parts are avoided to obtain sensitive information. An auditing requirement can be indicated in Comment.	
Constrains	It can be only specified in the diagram elements indicated in Table 3. A privacy requirement has one security role specification context Privacy inv: self.SecurityRole -> size() = 1 The Privacy Type must be specified adding a lower case letter according to Pv tagged value. If privacy type is not specified then anonymity and confidentiality are considered.	
Tagged Values	Pv: PrivacyType AvPv: AuditingValues context Privacy inv: self.AvPv="RoleName" or self.AvPv="Date" or self.AvPv="Time"	
Name	AccessControl	Notation 
Base Class	SecurityRequirement	
Description	It establishes the need to define and/or intensify the access control mechanisms (identification, authentication and authorization) to restrict access to certain components in an activity diagram. An auditing requirement can be indicated in Comment.	
Constrains	It can be only specified in the diagram elements indicated in Table 3. It is valid only if it is specified at least one security role. context AccessControl inv: self.SecurityRole -> size() >= 1	
Tagged Values	AvAC: AuditingValues context AccessControl inv: self.AvAC="RoleName" or self.AvAC="Date" or self.AvAC="Time"	

5 Example

Our illustrative example (see Figure 5) describes a typical business process for the admission of patients in a health-care institution. In this case, the business analyst identified the following Activity Partitions: Patient, Administration Area (which is a

top partition that is divided into Admission and Accounting middle partitions), and the Medical Area (divided into Medical Evaluation and Exams).

The business analyst has considered several aspects of security. He/she has specified «Privacy» (confidentiality) for Activity Partition «Patient», with the aim of preventing the disclosure of sensitive information about Patients. «Nonrepudiation» has been defined over the control flow that goes from the action «Fill Admission Request» to the actions «Capture Insurance Information» and «Check Clinical Data» with the aim of avoiding the denial of the «Admission Request» reception. «AccessControl» has been defined over the Interruptible Activity Region.

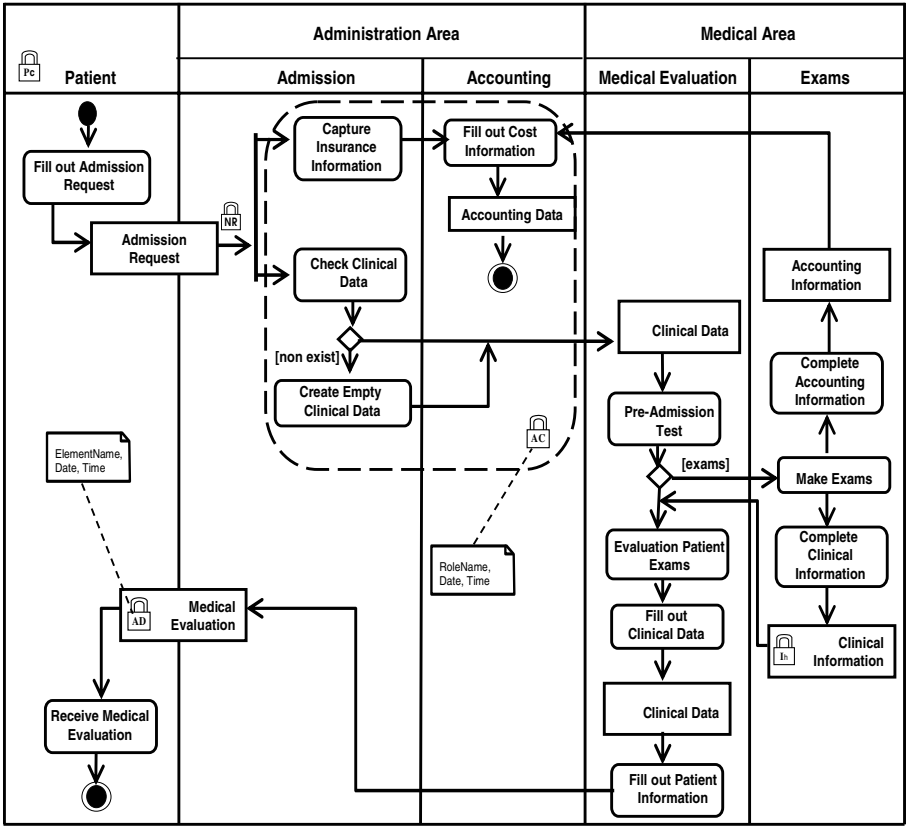


Fig. 2. Admission of Patients in a Medical Institution

Table 5. «SecurityRole» and «SecurityPermission» specifications

Role	Permissions → Objects		Permissions → Operations
Admission/Accounting	Action	Capture Insurance Information Fill out Cost Information Check Clinical Data Create Empty Clinical Data	Execution CheckExecution Execution Execution
	DataStoreNode	Accounting Data	Update

A «SecurityRole» can be derived from this specification. Admission/Accounting will be a role. All objects in an interruptible region must be considered for permissions specification (see Table 5). Access control specification has been complemented with audit requirement. This implies that it must register role name, date and time of all events related to the region interruptible. Integrity (high) requirement has specified for Data Store “Clinical Information”. Finally, the business analyst has specified Attack Harm Detection with auditing requirement. All events related to attempt or success of attacks or damages are registered (names in this case are clinical information, date and time).

6 Conclusions and Ongoing Work

The advantage of early representing requirements, in this case, security requirements, favours the quality of the business process since it provides it with more expressivity and improves the software quality since it considers characteristics that, in other way, would have to be incorporated late. So, we can save on maintenance costs as well as on the total cost of the project. We have defined a microprocess that complements the requirements stage defined in the Unified Process and we have used UML 2.0 to represent security requirements.

The next step should be that of applying an MDA approach to transform the model (including the security requirements) into most concrete models (i.e. execution models). Therefore, future work must be oriented to enrich the security requirements specifications, improving the UML extension specification to complement it with Well-Formedness Rules and OCL.

Acknowledgements

This research is part of the following projects: DIMENSIONS (PBC-05-012-1) and MISTICO, both supported by FEDER and the “Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha”, and COMPETISOFT granted by CYTED.

References

1. Abie, H., Aredo, D. B., Kristoffersen, T., Mazaher, S. and Raguin, T.; *Integrating a Security Requirement Language with UML*, 7th International Conference, The UML: Modelling Languages and Applications. Vol. 3273. Lisbon, Portugal. (2004). pp.350-364.
2. Artelsmair, C. and Wagner, R.; *Towards a Security Engineering Process*, The 7th World Multiconference on Systemics, Cybernetics and Informatics. Vol. VI. Orlando, Florida, USA. (2003). pp.22-27.
3. Backes, M., Pfitzmann, B. and Waider, M.; *Security in Business Process Engineering*, International Conference on Business Process Management (BPM). Vol. 2678, LNCS. Eindhoven, The Netherlands. (2003). pp.168-183.

4. Basin, D., Doser, J. and Lodderstedt, T.; *Model driven security for process-oriented systems*, SACMAT 2003, 8th ACM Symposium on Access Control Models and Technologies. Villa Gallia, Como, Italy. (2003).
5. Bock, C.; *UML 2 Activity and Action Models*, Journal of Object Technology. Vol. 2 (4), July-August. (2003). pp.43-53.
6. Eriksson, H.-E. and Penker, M., *Business Modeling with UML*, OMG Press. (2001).
7. Firesmith, D.; *Engineering Security Requirements*, Journal of Object Technology. Vol. 2 (1), January-February. (2003). pp.53-68.
8. Firesmith, D.; *Specifying Reusable Security Requirements*, Journal of Object Technology. Vol. 3 (1), January-February. (2004). pp.61-75.
9. Fuggetta, A.; *Software process: a roadmap*, ICSE 2000, 22nd International Conference on Software Engineering, Future of Software Engineering. Limerick Ireland. (2000). pp.25-34.
10. Herrmann, G. and Pernul, G.; *Viewing Business Process Security from Different Perspectives*, 11th International Bled Electronic Commerce Conference. Slovenia. (1998). pp.89-103.
11. Jacobson, I., Booch, G. and Rumbaugh, J., *El proceso unificado de desarrollo de software*, . (2000). 464 p.
12. Jürjens, J., *Secure Systems Development with UML*, Springer Verlag, (2004). 309 p.
13. Kalnins, A., Barzdins, J. and Celms, E.; *UML Business Modeling Profile*, Thirteenth International Conference on Information Systems Development, Advances in Theory, Practice and Education. Vilnius, Lithuania. (2004). pp.182-194.
14. List, B. and Korherr, B.; *A UML 2 Profile for Business Process Modelling*, 1st International Workshop on Best Practices of UML (BP-UML 2005) at ER-2005. Klagenfurt, Austria. (2005).
15. Lodderstedt, T., Basin, D. and Doser, J.; *SecureUML: A UML-Based Modeling Language for Model-Driven Security*, The Unified Modeling Language, 5th International Conference. Vol. 2460. Dresden, Germany. (2002). pp.426-441.
16. Lopez, J., Montenegro, J. A., Vivas, J. L., Okamoto, E. and Dawson, E.; *Specification and design of advanced authentication and authorization services*, Computer Standards & Interfaces. Vol. 27 (5). (2005). pp.467-478.
17. Maña, A., Montenegro, J. A., Rudolph, C. and Vivas, J. L.; *A business process-driven approach to security engineering*, 14th. International Workshop on Database and Expert Systems Applications (DEXA). Prague, Czech Republic. (2003). pp.477-481.
18. Maña, A., Ray, D., Sánchez, F. and Yagüe, M. I.; *Integrando la Ingeniería de Seguridad en un Proceso de Ingeniería Software*, VIII Reunión Española de Criptología y Seguridad de la Información, RECSI. Leganés, Madrid. España. (2004). pp.383-392.
19. Mouratidis, H., Giorgini, P. and Manson, G. A.; *When security meets software engineering: a case of modelling secure information systems*, Information Systems. Vol. 30 (8). (2005). pp.609-629.
20. Object Management Group; *Unified Modeling Language: Superstructure*, version 2.0, formal/05-07-04. In <http://www.omg.org/docs/formal/05-07-04.pdf>. (2005).
21. Pressman, R. S., *Software Engineering: A Practitioner's Approach*, 6th Edition, (2006). 880 p.
22. Quirchmayr, G.; *Survivability and Business Continuity Management*, ACSW Frontiers 2004 Workshops. Dunedin, New Zealand. (2004). pp.3-6.
23. Röhm, A. W., Herrmann, G. and Pernul, G.; *A Language for Modelling Secure Business Transactions*, 15th. Annual Computer Security Applications Conference. Phoenix, Arizona. (1999). pp.22-31.

24. Röhm, A. W., Pernul, G. and Herrmann, G.; *Modelling Secure and Fair Electronic Commerce*, 14th. Annual Computer Security Applications Conference. Scottsdale, Arizona. (1998). pp.155-164.
25. Siponen, M. T.; *Analysis of modern IS security development approaches: towards the next generation of social and adaptable ISS methods*, Information and Organization. Vol. 15. (2005). pp.339-375.
26. Stefanov, V., List, B. and Korherr, B.; *Extending UML 2 Activity Diagrams with Business Intelligence Objects*, 7th International Conference on Data Warehousing and Knowledge Discovery (DaWaK2005). Copenhagen, Denmark. (2005).
27. Vivas, J. L., Montenegro, J. A. and Lopez, J.; *Towards a Business Process-Driven Framework for security Engineering with the UML*, Information Security: 6th International Conference, ISC. Bristol, U.K. (2003). pp.381-395.
28. Zulkernine, M. and Ahamed, S. I., *Software Security Engineering: Toward Unifying Software Engineering and Security Engineering*, in: Idea Group (Ed.), Enterprise Information Systems Assurance and Systems Security: Managerial and Technical Issues, M. Warkentin & R. Vaughn, 2006, p.215-232.

Finite State History Modeling and Its Precise UML-Based Semantics

Dirk Draheim¹, Gerald Weber², and Christof Lutteroth²

¹ University of Mannheim, Institute of Computer Science, A5-6
68131 Mannheim, Germany
`draheim@acm.org`

² The University of Auckland, Department of Computer Science, 38 Princes Street
Auckland 1020, New Zealand
`{lutteroth, g.weber}@cs.auckland.ac.nz`

Abstract. This paper discusses the notion of a state history diagram. The concept is directly motivated by a new analysis technique, form-oriented analysis, which is tailored to an important class of interactive systems including web applications. A combined UML metamodeling and framework approach is used to give precise semantics to state history diagrams and the artifacts of form-oriented analysis.

1 Introduction

In this paper we introduce a general notion of finite state modeling that has a tight integration with other modeling views, especially with class diagrams. A state transition diagram in this new notion is called a state history diagram, SHD for short. SHDs can be used in many circumstances in analysis as well as design. They are especially favorable in cases where we model a system by a finite state machine in order to capture a specific aspect, while the system as a whole is modeled by a class diagram as well. Such models are very widespread. Submit/response style interaction is a very important instance. Other examples include the state of processes in operating systems or the life cycle of components in application servers.

We give the operational semantics for general SHDs in order to clarify the general character of the introduced concept. The approach chosen here achieves a sound basis for all the special constructs introduced in form-oriented analysis in a rather short and lightweight way. This is achieved through maximal reuse, mainly because we were able to fully reuse the semantics of class diagrams for our new artifacts. Submit/response style interaction yields an appropriate abstraction from an important class of interactive systems ranging from mainframe/terminal systems to web applications. Form-oriented analysis [3,4] addresses the analysis phase of submit/response style applications. Form-oriented analysis has a special type of state history diagram called formchart. Formcharts are bipartite state transition diagrams with an OCL [9] extension DCL (dialogue constraint language).

We introduce state history diagrams in Sect. 2. We give precise semantics to state history diagrams in Sect. 3 by combining metamodeling techniques [2]

with a notion of semantic framework. In Sect. 4 we model formcharts as bipartite SHDs and give precise semantics to dialogue constraints. Section 5 on related work serves as a general discussion of possible alternative approaches to give semantics to SHDs. The paper finishes with a conclusion in Sect. 6.

2 State History Diagrams and Class Diagrams

Our operational semantics for SHDs are based solely on the semantics of class diagrams. The main idea behind SHDs is the following consideration: Class diagrams define the set of possible object nets, i.e. states over the class diagram. Finite state machines define the set of possible traces for the state machine. Hence, finite state machines can be defined as class diagrams which allow only directed paths as object nets. SHDs are a semantic unification of class diagrams and state transition diagrams. In fact, they are a restriction of class diagrams. We show how SHDs can be defined in the context of UML. Our semantics of SHDs is based only on the fundamental concept of core class diagrams, which serve as the semantic foundation in the modeling universe in which UML is located.

When modeling complex systems, a finite state machine is typically only a part of the model. The behavior may depend further on a classical data model. In many cases the system behavior depends on the history of state transitions, e.g. in systems where certain functions only become available after a user has logged in or completed other required tasks. SHDs give a convenient general modeling tool for such systems by allowing the specification of temporal constraints on finite state automata without any need for further temporal formalisms, solely through the combination of the already defined concepts. One common application are systems with submit/response style interfaces. Such systems can be described by formcharts, the key diagrams of form-oriented analysis. Formcharts are a simple application of SHDs, although we will support it with our own semantic framework. We will also give precise semantics for dialogue constraints introduced in DCL.

If a system is modeled with an SHD, the history or trace of the finite state machine is a part of the actual system state. For each state visit an instance of some class is created, and these instances are arranged in a linked list, forming a log of the state transition process so far. The links represent the transitions between state visits. On this object net we can define constraints concerning the history of state transitions. Hence, we can define certain temporal constraints without having to introduce a temporal extension into the constraint language.

SHDs are based on the idea of choosing the class diagram for this log in such a way that it is isomorphic to the state transition diagram (STD). This is possible since STDs and core class diagrams can be modeled by a similar metamodel. The basic metaclasses are nodes and connectors. In STDs the nodes are states and the connectors are directed transitions. In core class diagrams the nodes are classes and the connectors are binary associations. An SHD is a special class diagram which can be read as an STD at the same time. Consequently, a single diagram describes the state machine on the one hand and serves as a class diagram for the aforementioned

history of state transitions on the other hand. Such a class diagram must, however, adhere to rigorous restrictions that will be given in due course.

We adopt the following rules for speaking about SHDs: the diagram can be addressed as an SHD or as an STD or as a class diagram, emphasizing the respective aspect. The nodes are called *state classes*, the connectors are called *transitions* and they are associations; their instances are called *state changes*. A run of the state machine represented by the STD is called a *process*. The visit of a state during a process over the STD is identified with an instance of the state class and is called a *visit*. Hence a process is the object net over the SHD. This object net describes a directed path from the start visit of the process to the current visit. Each prefix of the path is a part of the whole current path, which matches the semantics of the aggregation. Hence all transitions are aggregations, with the aggregation diamond pointing to the later state. In the SHD, however, the transitions are not drawn with diamonds but with single arrows, and the associated state classes are called source and target.

We will use the SHD for the modeling of formcharts. If a system is used by several clients, each client lives in its own object space. Therefore, the singleton property is local to the client's object space. If one were to model all clients accessing a system in a single object space, this could be done by using several SHDs in a single object space. In that case one has to use a slightly modified framework in which the `StartState` is not a singleton, but there is one `StartState` instance for each run of the finite automaton.

3 Modeling SHDs as Class Diagrams in UML

In the previous section we defined SHDs as a restriction of a class diagram, not as a new diagram type. In this section we give a semantic treatment of this approach in the context of UML. We obtained the definition of SHD's in the following way: SHDs are class diagrams in which all elements are derived from a special semantic modeling framework, the `shdframework`. Note that we use model-level inheritance here. In the UML context a specification alternative would be a package with stereotypes in which the SHD is defined by metamodel instantiation instead of model inheritance. But first we adhere strictly to the economy principle, and argue that modeling is more lightweight than metamodeling, therefore we use modeling wherever possible. Secondly our approach allows for a quite elegant formulation of the central SHD semantics, namely that the object nets are paths. The `shdframework` is depicted in Fig. 1 together with the formchart framework, which extends the `shdframework`. When using SHDs, the stereotype notation can be used for pure notational convenience: for each public element of our framework, a stereotype of identical name is introduced in an auxiliary stereotype package. The stereotype has the metalevel constraint that its instances must inherit the framework class of the same name.

In the `shdframework` we define a hierarchy for classes as well as for associations as shown in Fig. 1. We make intensive use of the concept of association inheritance, in other words the generalization of associations. There have been

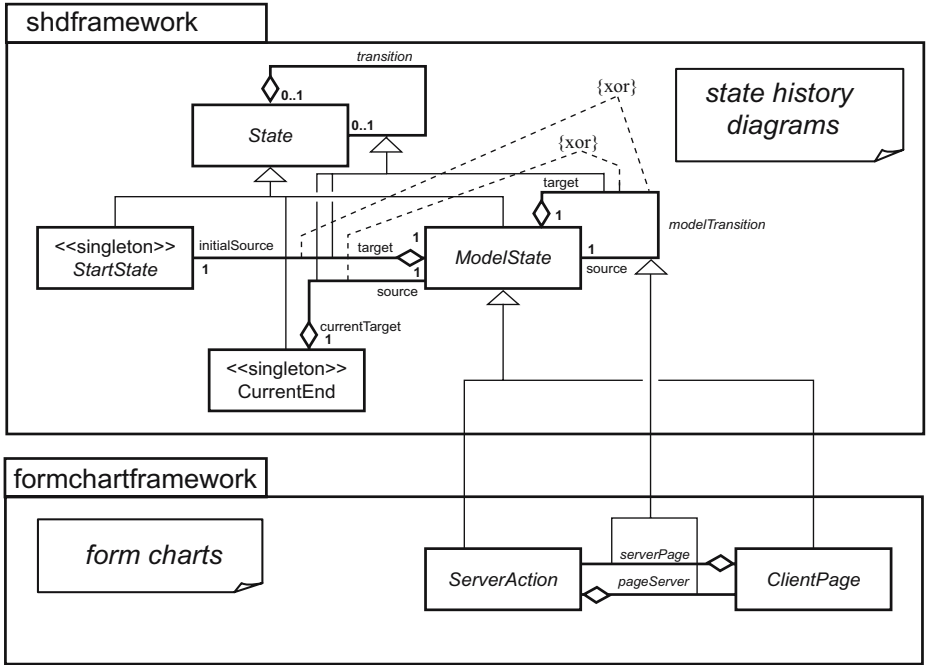


Fig. 1. Frameworks for state history diagrams and formcharts

long debates about the semantics of generalization of associations, therefore we prefer to define the semantics we use here: if an association has n associations, which inherit it directly, then two objects can be connected over only one of the inherited associations. The basic class is **State** and it has an aggregation to itself called **transition**. The ends of **transition** have roles **source** and **target**.

Before we explain how the framework introduces the desired semantics to SHDs, we explain how it should be used in creating SHDs. All state classes in the SHD must be derived from **ModelState** and all transitions must be derived from **modelTransition**. The marker for the start of the SHD is derived from **StartState**, and its transition to the first state is derived from the unnamed transition to **ModelState**. Only these four elements of the shdframework are public. All elements except the class **CurrentEnd** are abstract, so only the classes derived by the modeler can be instantiated. The singleton stereotype of **StartState** requires that the whole class hierarchy derived from **StartState** has only one instance. In a concrete SHD of course the generalization dependencies of SHD elements to the framework are not depicted. Instead we make use of the auxiliary stereotypes mentioned earlier. Therefore we are entitled to change the graphical appearance of stereotyped classes in the SHD, as we will do for formcharts.

Except for the start visit, every visit must have exactly one predecessor. This constraints on the process as the object net over a SHD can be formalized in two different ways. First, one could exempt the start visit from the general rule. The

second way is to use a technique similar to the sentinel technique in algorithms: an artificial predecessor to the start node is introduced. This artificial visit is of a **StartState** class which cannot be revisited. We choose this second method. Both **StartState** and **ModelState** are derived from **State**. In the same way, the current visit has always an artificial successor from the class **CurrentEnd**. All states created by the modeler in the SHD shall be indirectly derived from **ModelState**. The cardinalities are expressed in the class diagram in Fig. 1. Each time a new state A is visited, a new instance of A must be created. This new visit gets the old current state as a predecessor and the current end as a successor.

The model states in SHDs can have attributes and also parts defined by aggregations. Visits of model states are assumed to be deep immutable. Each state has an **enterState()** method, which has to be called on each newly inserted visit. The new visit has to be seen as being the conceptual parameter of its own **enterState()** method. The attribute list of the **ModelState** replaces the parameter list, therefore we have assigned the name *superparameter* to this concept of a single parameter. Each state has a **makeASuperParam()** method, which must be called when the state is left and which constructs the superparameter. The superparameter is passed to the **enterState()** method in sigma calculus style, which means that the **enterState()** method is called on the superparameter without method parameters. State changes are performed by a single method **changeState()** in the old state. The **changeState()** method of one state calls its own **makeASuperParam()** and the **enterState()** of the next state. **makeASuperParam()** and **enterState()** must not be called from any other method. **changeState()** is defined final in **ModelState**. The control logic which invokes **changeState()** is not prescribed; however, a state change can only be caused by calling this method on the current visit. In Java-like pseudocode it would look like this:

```
abstract class ModelState extends State {
    // ....
    abstract ModelState makeASuperParam();
    abstract void enterState();
    final void changeState(){
        ModelState aSuperParam = makeASuperParam();
        superParam.enterState();
    } }
```

SHDs have a new constraint context, which is conceptually placed on the edge between two states. The constraint in this context is called *state output constraint*. From the implementation of **changeState()** it is known that the precondition of **enterState()** is immediately executed after the postcondition of **makeASuperParam()**, so the transition constraint could be placed as post condition of **makeASuperParam()** or as precondition of **enterState()**. However, in the context of **makeASuperParam()** we do, in general, not know the actual type of the object on which **enterState()** is called, and vice versa. This is overcome in the newly introduced transition context: there is no **self** keyword, but the role names of the transition ends can be used, especially the role names **source** and **target** from the general transition.

4 Semantics of Formcharts and Dialogue Constraints

Form-oriented analysis is tailored to systems with a specific interaction style, named form-based, submit/response. For example, web interfaces are form-based. This style allows for abstraction from fine grained user interaction on one page, specifying user interaction as high level requests. From an analysis point of view, the user virtually submits fully typed requests in a strong type system. Using these insights, the interface is modeled by a bipartite state machine, called *form-chart*, which is integrated with a data model and dialogue constraints. A form-chart has two sets of states, which occur alternatingly on a path: *client pages*, in which information is presented to the user and the fine-grained interaction within the page is performed, and *server actions*, which represent the processing of a request on the server side. Client pages are represented as bubbles and server actions as rectangles. A form-oriented specification also comprises a data dictionary, which contains types for the messages sent to and from the server actions during the transitions. The data dictionary is therefore a class diagram in the terms of modern modeling languages like the UML. Each formchart state has a corresponding message type of the same name. Formcharts can be enriched by dialogue constraints, which further specify the transitions between states.

4.1 Formcharts as State History Diagrams

Formcharts are typed, bipartite STDs which can be modeled as SHDs. For this purpose we introduce the `formchartframework`, which specializes the `ModelState` and `ModelTransition` elements of the `shdframework`, as shown in Fig. 1. Form-chart model elements must be derived from the `formchartframework` elements, except the rather technical start elements, which are still derived directly from the `shdframework`. Elements of formcharts are therefore also derived from elements of the `shdframework`, though indirectly, so that formcharts are SHDs.

The formchart framework enforces by itself that formcharts are bipartite and introduces the known names for formchart elements. We introduce two subclasses to `State`, `ServerAction` and `ClientPage`, from which all states in the formchart have to be derived. We derive the aggregations `pageServer` and `serverPage` between them from the transition aggregation in order to enforce that formcharts are bipartite: all transitions must be derived from either `pageServer` or `serverPage`. The usage of the framework for form-oriented modeling is shown in Fig. 2. Only the derivations of the states are shown; the derivations of the transitions is omitted.

We assume again that each formchart framework element is accompanied by a stereotype of the same name. Each stereotype introduces its own graphical representation. The `<<ClientPage>>` stereotype is depicted by a bubble, the `<<ServerAction>>` is depicted by a rectangle. In the formchart, bubbles and rectangles contain only their names. The `<<serverPage>>` and `<<pageServer>>` associations are depicted as arrows, even though they are aggregations.

Figure 3 shows a formchart and, below it, an example object net over that formchart. The start state is omitted. The object net is a path alternating

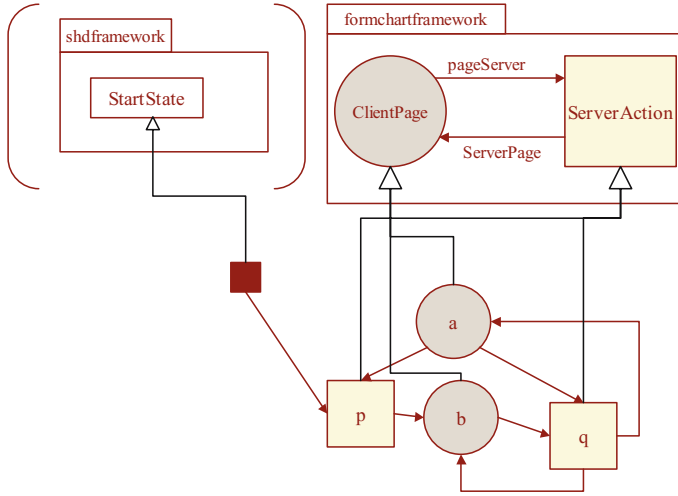


Fig. 2. A formchart is derived from the semantic framework

between client states and server actions. If a **ModelState** in the formchart has no outgoing transition, this state is a terminal state for the dialogue; the dialogue is completed once such a state is entered.

4.2 Semantics of Dialogue Constraints

Now we define the semantics of the different constraint stereotypes introduced in DCL. Only one constraint of the same stereotype is allowed for the same context.

Enabling Conditions can be used to restrict the outgoing transitions of a **ClientPage**. For each outgoing transition, the page offers a form where users may be able to enter data. Often a certain form shall be offered only if certain conditions hold, e.g. a bid in an auction is possible only if the auction is still running. Since the page shown to the user is not updated unless the user triggers a page change, the decision whether to show a form or not has to be taken in the **changeState()** method leading to the current **ClientPage** visit. The enabling condition is mapped to a part of a precondition of **enterState()**. Alternatively each enabling condition can be seen as a query that produces the boolean value which is assigned to **formXenabled**. Typically, the same constraint has to be reevaluated after the user interaction. In the example above, the auction may end while the user has the form on the page. Then the same OCL expression is also part of another constraint stereotype, especially **<<serverinputconstraint>>** or **<<flowcondition>>**.

Server Input Constraint appear only in incomplete models, or models labeled as TBD, to be defined [8]. A server input constraint expresses that the **ServerAction** is assumed to work correctly only if the server input constraint

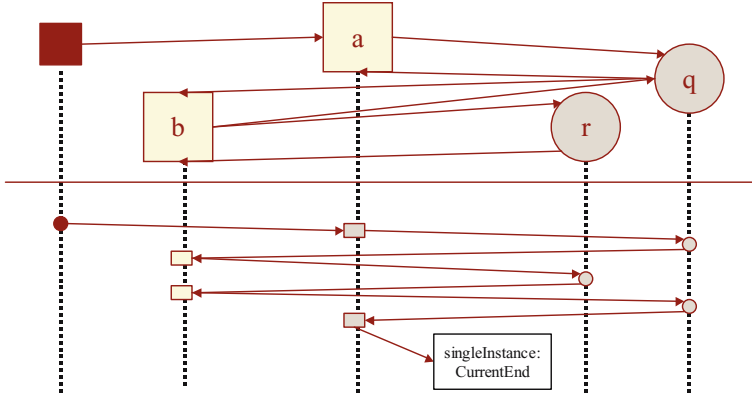


Fig. 3. The object net over a formchart is a path

holds. In a late refinement step the server input constraint has to be replaced by transitions from the **ServerAction** to error handlers. Context of the server input constraint is the **ServerAction** visit. Server input constraints are not preconditions in a design by contract view, since server input constraint violations are not exceptions, but known special cases.

Flow Conditions are constraints on the outgoing transitions of a server action. All but one flow condition must be numbered, and the numbers on flow conditions must be unique, although they need not be strictly ascending in order. Context of flow conditions is the **ServerAction** visit. The semantics of flow conditions can be given by mapping all flow conditions of a state onto parts of a complex postcondition on `makeASuperParam()`. This postcondition has an **elsif** structure. In the **if** or **elsif** conditions the flow conditions appear in the sequence of their numbering. In the **then** block after a flow condition, it is assured that a visit of the targeted **ClientPage** is the new current state. In the final **then** block the same check is performed for the target of the **serverPage** transition without a flow condition.

Client Output Constraints and Server Output Constraints are specializations of state output constraints and live in the new transition context.

5 Related Work

As we mentioned earlier, a modeling alternative would be a pure metamodel for formcharts, i.e. purely as a framework of stereotypes for metamodel elements of class diagrams [1]. The intention of this approach would be still to define formcharts (or more generally SHDs) as a variant of class diagrams in such a way that it is guaranteed that the object net is a path, but the constraints would have to be expressed within the stereotype description. A related topic is the use of

the composition notation. One could think of expressing certain multiplicities in the framework implicitly by using composition instead of aggregation. However, composition would be able to express only one direction of the multiplicity at best; furthermore there are different opinions about the exact meaning of composition. Therefore it seems more convenient to stick to aggregation, and to use explicit multiplicities, as was done in the framework. The diagram in Fig. 3 resembles a message sequence chart. However, the diagram is completely different. Above the horizontal line are classes, not instances. Below the line are instances, not method invocations.

UML has its own diagram type for state automata called state machines, which are based on David Harel's statecharts [6]. However, state machines have no operational semantics defined within UML; indeed no formal operational semantics are part of the specification. Instead, operational semantics are given by reference to a state machine that is described verbally. The semantics of statecharts based on pseudo-code are given in [7]. Our definition of SHDs has operational semantics based solely on core class diagrams, the semantic core of UML. One could model formcharts with UML state machines, but one would have to drop the SHD semantics and hence loses support for temporal path expressions. The fact that formcharts use only flat STDs in contrast to hierarchical statecharts does not restrict the expressibility of formcharts due to the fact that formcharts are coupled with an information model. Formcharts can be easily combined with statechart notation if the statechart notation is interpreted as the visualization of parts of the session objects of the information model.

A proposal for representing states by classes can also be found in the state design pattern [5], which describes how an object can delegate its state change to a state object. The object appears to change its state within a finite set of states. The set of states is finite, since each state is represented by a class. The pattern does not prescribe how to model a finite automaton over the state instance set, but discusses procedural implementations. First it proposes implementing the transitions in the so-called Context class using the pattern. An alternative proposal is a table lookup. Both are non-equivalent to the SHD model using associations to model state transitions.

Petri nets are a state transition formalism based on bipartite graphs. Formcharts resemble Petri nets due to the fact that server actions resemble Petri net transitions. Petri nets as a finite state machine model are classically conceived as being never in a state corresponding to a transition. The main difference between Petri nets and bipartite state diagrams is therefore that the state of a Petri net is not necessarily a single bubble, but possibly a compound state, depending on the type of Petri net, e.g. a function from bubbles to numbers for place/transition nets. It is possible to give formcharts Petri net semantics by defining only client pages as places and introducing a Petri net transition for every path of length two to another client page. Such Petri net semantics, however, involves only trivial transitions, i.e. transitions with one ingoing and one outgoing edge.

6 Conclusion

The notion of state history diagram has been introduced. A state history diagram is a state transition diagram and a class diagram at the same time. The object net over the class diagram view is the history of the visits to the states of the state transition diagram view. The motivation is the fact that in important applications of state transition diagrams a structurally identical class diagram has to be considered. Formcharts have been characterized as state history diagrams. Precise semantics have been given to state history diagrams, formcharts and the dialogue constraint language by means of combining a new semantics framework approach with convenient metamodeling techniques.

References

1. Colin Atkinson and Thomas Kühne, "Strict Profiles: Why and How", In: Proceedings of UML 2000: 3rd International Conference, Lecture Notes in Computer Science 1939, Springer, 2000.
2. Colin Atkinson and Thomas Kühne, "The Essence of Multilevel Metamodelling". In: Proceedings of UML 2001: 4th International Conference, Lecture Notes in Computer Science 2185, Springer, 2001, pp. 19–33.
3. Dirk Draheim and Gerald Weber, "Form-Oriented Analysis - A New Methodology to Model Form-Based Applications", Springer, 2005.
4. Dirk Draheim and Gerald Weber, "Modelling Form-Based Interfaces with Bipartite State Machines", Journal Interacting with Computers, vol. 17, no. 2. Elsevier, 2005, pp. 207-228.
5. Erich Gamma et al., "Design Patterns", Addison-Wesley, 1995.
6. David Harel, "Statecharts: a Visual Formalism for Complex Systems", Science of Computer Programming, Elsevier Science Publishers B.V., 1987, pp. 231-274.
7. David Harel and Amnon Naamad, "The Statemate Semantics of Statecharts", In: ACM Transactions on Software Engineering and Methodology, vol. 5, no. 4, 1996, pp. 293-333.
8. IEEE Std 830-1993, "Recommended Practice for Software Requirements Specifications", Software Engineering Standards Committee of the IEEE Computer Society, New York, 1993.
9. Jos Warmer, Anneke Kleppe, "The Object Constraint Language", Addison Wesley, 1999

A UML Profile for Modeling Schema Mappings

Stefan Kurz, Michael Guppenberger, and Burkhard Freitag

Institute for Information Systems and Software Technology (IFIS)
University of Passau, Germany

Stefan.Kurz@uni-passau.de, Michael.Guppenberger@uni-passau.de,
Burkhard.Freitag@uni-passau.de

Abstract. When trying to obtain semantical interoperability between different information systems, the integration of heterogeneous information sources is a fundamental task. An important step within this process is the formulation of an integration mapping which specifies how to select, integrate and transform the data stored in the heterogeneous local information sources into a global data store. This integration mapping can then be used to perform the data integration itself.

In this paper, we present a UML-based approach to define integration mappings. To this end, we introduce a UML profile which can be used to map local information schemata onto one global schema thus eliminating schema conflicts. We claim that this is the first time that the integration mapping can be specified within the UML model of the application and that this model can be used to generate a working implementation of the schema mappings using MDA-transformations.

Keywords: Data Integration, Schema Mapping, Model Driven Architecture (MDA), UML Profiles.

1 Introduction

The integration of heterogeneous information sources is an important task towards the achievement of semantical interoperability. To perform data integration, it has to be determined how to select, integrate and transform the information stored in local data sources. During the formulation of the integration mapping, possible integration conflicts have to be recognized and eliminated, like data-level conflicts, e.g. inconsistent attribute ranges, or schema-level conflicts, e.g. different data models. Our approach addresses schema-level conflicts concerning semantical and structural heterogeneity.

The integration of legacy information systems is usually done in four phases:

1. The local data sources to be integrated and especially their data schemata are analysed in detail. The goal of this first step is to determine the semantics of the data to be integrated as completely as possible.
2. The heterogeneous representations of local data (e.g. Entity-Relationship-Models or XSchemata) are transformed into a common global data model to overcome conflicts resulting from varying modeling concepts.

3. Further structural and semantical schema-level conflicts have to be uncovered. Whilst structural conflicts can be detected directly by analyzing the schemata, the detection of semantical conflicts is more complicated since the discovery of the model's semantics on the basis of a schema is only possible to a limited extent. Basically, the various causes for schema diversity arise from different perspectives, equivalences among constructs, and incompatible design specifications. To solve schema-level conflicts, a *schema integration* has to be performed. In brief, schema integration is the activity of first finding correspondences between the elements of the local schemata and next integrating these source schemata into a global, unified schema [1].
4. Finally, the results of the third phase are used to consolidate the data stored in the local sources in a way that the integrated data is accessible based on the global data schema.

In this paper, we focus on the third phase. As we assume that the global schema and the local schemata are given, we have to specify a *schema mapping*. To avoid the problem of handling different modeling formalisms, we also assume that both the global and the local schemata are specified as UML models [2].¹

We present a newly developed UML profile providing a set of different constructs (as explained in section 4), which can be used to specify the integration mapping between source and target schema. Our approach helps keeping the model consistent and readable. Even more important, it also allows us to use new MDA techniques to automatically generate a fully functional implementation of the mapping, using only the UML model(s) and a set of generic transformations.

The remainder of the paper is organized as follows: section 2 gives an overview of existing approaches to schema-level integration. Section 3 gives an outline to the fundamental structure of a schema mapping. In section 4 we show how these ideas have been transferred into the UML profile. To indicate the practical applicability of the proposed profile, section 5 very briefly describes an example we used to evaluate the profile. The paper ends with a conclusion in section 6.

2 Related Work

There exist various approaches to schema-level data integration. Most of them use a mediator-based architecture to access and integrate data stored in heterogeneous information sources [4]. From the perspective of the applications, a mediator can be seen as an overall database system which accepts queries, selects necessary data from the different sources and, to process the queries, combines and integrates the selected data. A mediator does not access the data sources directly, but via a so-called wrapper [5].

An early representative of this architectural concept is the TSIMMIS project [6]. Heterogeneous data from several sources is translated into a common object model (Object Exchange Model, OEM) and combined to allow browsing the

¹ This, in fact, is no real restriction, since tools like AndroMDA's Schema2XMI [3] are able to generate UML representations from e.g. relational data sources.

information stored in the different sources. Unlike our objective, TSIMMIS only allows the execution of a predefined set of queries (so-called query-templates).

To overcome this obvious handicap, the Garlic project [7] introduced a common global schema and allows to process general queries according to this unified schema. Both the local schemata as well as the global schema are represented in a data definition language similar to ODL [8].

The MOMIS project [9] uses a schema matching approach to build a global schema in a semi-automatic way. Again an object-oriented language derived from ODL is used to describe heterogeneous data sources.

The Clio project [10] of IBM Research also tries to support data integration by facilitating a semi-automatic schema mapping. However, Clio only allows the integration of relational data sources and XML documents and therefore generates data transformation statements in SQL and XSLT/XQuery, respectively.

Apart from research projects, there are also many commercial software systems and tools available that support data integration. Most of them allow a graphical, but proprietary definition of a mapping between data schemata. We name Oracle's Warehouse Builder [11] and Altova's MapForce [12] as examples.

It is obvious that it would be a good idea to combine the commercial software solutions mainly addressing the users' needs and the research projects considering advanced technical developments. Our approach allows user-friendly graphical modeling of the schema mapping using the de-facto standard UML (in contrast to Garlic and MOMIS which use other object-oriented description languages). Furthermore, our method can be integrated into a mediator-based architecture serving as a platform for the model-driven implementation of a schema mapping modeled according to our proposal. With our approach, various target platforms can be supported. In contrast to approaches tied to a specific data model, any format of data transformation statements like SQL or Java can be generated by our method. Also, various kinds of data sources (e.g. relational databases, semi-structured information sources or flat files) can be integrated. Finally, our architecture offers interfaces to external schema matching tools thus supporting a semi-automatic schema mapping.

3 Modeling Schema Mappings

An overview of our approach is shown in fig. 1. Assume that some local, possibly heterogeneously represented data schemata (bottom of fig. 1) are to be integrated into a common global schema (top of fig. 1).

Assume further that the global schema already exists and is represented in UML. First, the local schemata are re-modeled using UML (middle of fig. 1). Afterwards, for each UML representation of a local schema a mapping onto the global schema is defined. Based on these mappings, the necessary data access and data integration procedures can be generated using transformation techniques from model-driven software technology.

In the following, the local schemata are denoted as *source schemata* and the global schema as *target schema*. In general, the objective of schema mapping is

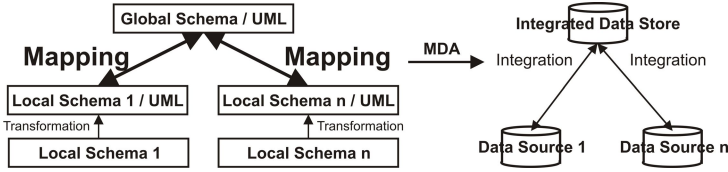


Fig. 1. Overview of our approach

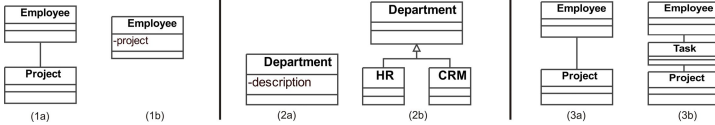


Fig. 2. Sample structural schema-level conflicts

to find the correspondences between the target schema and the source schemata. During this phase, conflicts similar to those shown in fig. 2 have to be resolved [1]. The left part of fig. 2 illustrates a frequent structural conflict: a project associated with an employee is modeled as a designated class in one schema (1a) and by an attribute in the other schema (1b). As another example, a structural conflict arises because in one schema (2b) a generalization hierarchy is introduced whereas the other schema (2a) simply uses different attribute values to distinguish between different departments. The right part of fig. 2 finally shows a structural conflict that is caused by different representations of an association: in one schema two classes are associated directly (3a) whereas in the other they are associated indirectly via another class (3b). Of course, also semantical schema-level conflicts have to be followed and solved.

According to [13], we consider a mapping between two schemata as a set of *mapping elements*. Each mapping element correlates specific artefacts of the source schema to the corresponding artefacts of the target schema.

In general, a mapping element may refer to element-level schema components like attributes or to structure-level artefacts like classes and associations. At structure-level, we define which classes of the source schema and the target schema correspond to each other. At element-level, we define how the structure-level specifications work in detail, i.e., how target elements are computed from their corresponding source elements.

Consider fig. 3 for an example: there are two schemata each basically modeling an employee associated with a project. We are interested in merging the two source classes (left hand side) into the one target class (right hand side) to solve the structural conflict as shown in fig. 2 (parts 1a and 1b). To achieve this we will define a $n : 1$ structure-level mapping. At element-level `source.Employee.name` maps onto `target.Employee.lastName` and `source.Project.name` onto `target.Employee.project` thus defining a $n:m$ element-level mapping.

Of course, the semantics of a mapping must be defined in more detail. To this end, a mapping element can be associated with a so-called *mapping expression*.

In our example above, the mapping expression could be defined by a SQL-query like `target.Employee.lastName, target.Employee.project = SELECT e.name, p.name FROM source.Employee e, source.Project p WHERE e.project = p.id`.

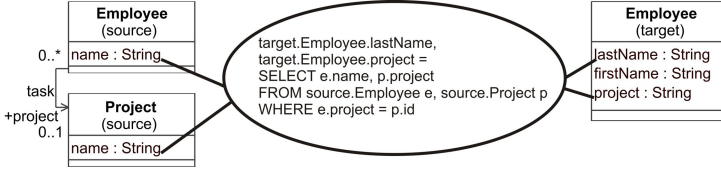


Fig. 3. Sample source and target schema

At a first glance it seems to be obvious that a mapping can have any cardinality. However, for simplification we allow only $1 : 1$ and $n : 1$ mapping cardinalities at structure-level. This restriction guarantees that each mapping element can be related to exactly one target class which is important for the implementation of the mapping.² Furthermore, $1 : n$ and $n : m$ structure-level mappings can be replaced by appropriate $1 : 1$ and $n : 1$ mappings if needed.

In the following, we assume that a given target class originates from one or more source classes. Consequently, we call the target class a mapping element refers to the "originated" target class. We further assume that from the set of source classes associated with a target class one is selected as the main "originating" source class; the other source classes are seen as "dependent" source classes. In fig. 3, for example, the class `target.Employee` originates from the classes `source.Employee` and `source.Project` whereas the latter can be seen as dependent.

In the following section we introduce a profile which extends the UML meta-model and allows for the representation of schema mappings using UML modeling primitives. The core constructs of our extension are the *MappingElement* and *MappingOperator* stereotypes used to define mapping elements and associated mapping expressions. We will use these concepts in conjunction with UML dependencies to graphically specify which source schema artefacts are mapped onto which target schema artefacts and how this schema mapping is to be performed.

4 The Profile

An overview of the stereotypes introduced by the profile can be found in fig. 4. To clarify the practical aspects of the proposed profile, we also give some simple examples.³

² The functional correlation of mapping elements with target classes, i.e., $1 : 1$ or $n : 1$, allows the implementation of each mapping element to be coded as an implementation of its associated target class.

³ These examples are only meant to illustrate the descriptions of the profile, not to provide a detailed survey of how to model the elimination of arbitrary schema-level conflicts. All of them are based on the two simple schemata introduced in fig. 3.

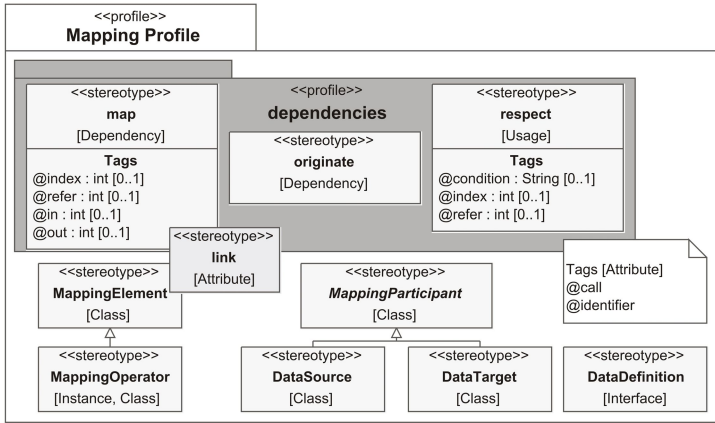


Fig. 4. Overview of stereotypes introduced by the profile

Fig. 5 shows the elimination of the semantical conflict between two elements having different names but modeling the same concept (problem of synonyms), here the attributes `name` and `lastName`. We will give a step-by-step illustration of how this conflict can be solved using our UML profile.

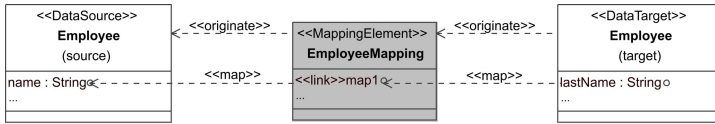


Fig. 5. Sample 1:1 element-level mapping

First, we describe the stereotypes which have been introduced to tag the source schema and the target schema.

MappingParticipant. The (abstract) stereotype *MappingParticipant* (see fig. 4) is used to tag classes which participate in the mapping. As a mapping is always defined between classes which are tagged with the stereotypes *DataSource* or *DataTarget*, *MappingParticipant* is only used implicitly as a generalization of the stereotypes *DataSource* and *DataTarget*.

DataSource and DataTarget. The stereotypes *DataSource* and *DataTarget* are used to tag the source and target classes participating in the mapping. In our running example, we tag the source class `source.Employee` as *DataSource* and the target class `target.Employee` as *DataTarget* (cf. fig. 5).

DataDefinition. According to the principles of object-oriented software design, a class is commonly implemented against interfaces. Especially in case of a

DataTarget, we assume that among these interfaces one is available which specifies the methods needed to access the *DataTarget*. The *DataDefinition* stereotype is used to tag this particular interface.⁴

We will now explain how to specify a mapping between these two schemata.

MappingElement. This stereotype is used to tag a class defining the association of originating *DataSources* with an originated *DataTarget*. In our running example, we introduce the *MappingElement* *EmployeeMapping* to relate the *DataSource* *source.Employee* to the *DataTarget* *target.Employee* (cf. fig. 5).

Originate. To specify the structure-level associations of a *MappingElement*, we use dependencies tagged with the stereotype *originate*. The already mentioned restriction, i.e., that at structure-level we allow only 1 : 1 and $n : 1$ mapping cardinalities, is checked by appropriate OCL constraints.

Map. The stereotype *map* allows to tag dependencies which specify the element-level relationships of a *MappingElement*. In our example (cf. fig. 5), two linked⁵ *map*-dependencies define the attribute *name* of the *DataSource* *source.Employee* to be mapped onto the attribute *lastName* of the *DataTarget* *target.Employee*.

MappingOperator. The stereotype *MappingOperator* tags classes defining functions that can be used to specify the mapping expression of a *MappingElement*. This way it is possible to define more complex relationships between a *DataTarget* and its corresponding *DataSources*. Note that a class tagged *MappingOperator* merely *defines* a function. The mapping itself must be modeled using instances of a *MappingOperator* class.

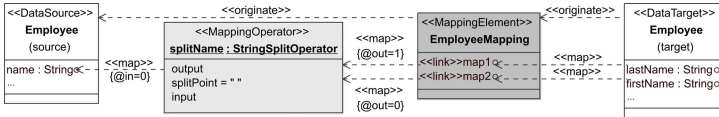


Fig. 6. Sample 1:n element-level mapping

Fig. 6 illustrates how even more complicated semantical conflicts can be resolved. As an example, consider the conflict of relating the attribute *name* to the attributes *lastName* and *firstName*. We use the instance *splitName* of the *MappingOperator* *StringSplitOperator* that associates the attribute *name* of the *DataSource* with the attributes *lastName* and *firstName* of the *DataTarget* using

⁴ When implementing the mapping, this means that the tagged interface of a target class remains unchanged, whereas the implementation of the target class can be replaced according to the specified mapping definitions.

⁵ The stereotype *link* is used to tag the attributes of a *MappingElement* which act as "connectors" between *map*-dependencies and additionally relate *map*-dependencies to a *MappingElement*.

a blank as separator. These input/output parameters of the *MappingOperator* are defined by *map*-dependencies in conjunction with appropriate tagged values. For example, the source data "John Smith" could be transformed into the target data "John" and "Smith".

The implementation of the *MappingOperator*, here the class *StringSplitOperator*, can be provided by the user. This offers a very flexible and simple method to define complex mappings by introducing new mapping operators.

Respect. The stereotype *respect* is mainly used to tag dependencies relating dependent *DataSources* to their originating *DataSource*. Fig. 7 shows a *respect*-dependency indicating how to navigate from the originating *DataSource* source.Employee to the dependent *DataSource* source.Project (see also fig. 3).

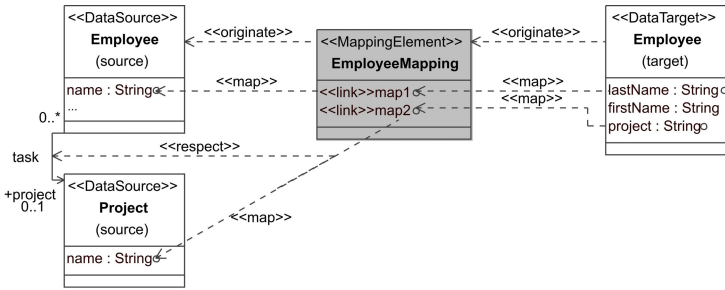


Fig. 7. Sample n:m element-level mapping (n:1 structure-level mapping)

Fig. 8 illustrates how to resolve the structural conflict of fig. 2 (parts 2a,b). A *respect*-dependency with an appropriate tagged value specifies that each time the value of the source attribute `description` is "development", the *DataTarget* `DevelopmentDepartment` is instantiated.

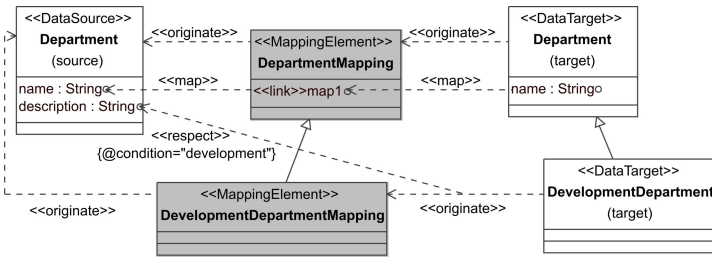


Fig. 8. Mapping concerning generalization hierarchy

The structural conflict shown in fig. 2 (parts 3a,b) can be resolved similarly.

5 The Profile in Practice

To evaluate the practical applicability of the proposed profile, we defined a mapping between two realistic heterogeneous schemata. The sample schemata cover

almost all of the structural and semantical schema-level conflicts listed in [1] and [13], in particular the structural conflicts of fig. 2.

To define the mapping between the source schema (consisting of four classes) and the target schema (containing four classes with a generalization hierarchy), four *MappingElements* and one *MappingOperator* had to be used. Although the profile proved to be suitable for real-life integration scenarios, it became obvious that a complete integration mapping easily becomes complex (which is also the reason why we do not show the complete mapping here). However, such an integration mapping can be decomposed into several smaller parts, which increases readability and understandability a lot.

6 Conclusion and Summary

The results of our work make it possible to specify mappings for the integration of heterogeneous data sources directly within the UML model(s) of the application in a user-friendly graphical and standardized way.

By using UML, we are able to apply the MDA approach [14] to generate code from our models implementing the modeled schema mappings. So it is possible to generate code which defines data transformation statements that allow us to access integrated local data according to a global schema. Furthermore, as our models are independent from any implementation details (which is one of the core concepts of MDA), we are also able to generate code that satisfies the needs of any target platform. To prove that claim, we also developed a mediator-based architecture which can be seen as a framework to execute generated code from UML models built according to our profile, thus allowing homogeneous access to the integrated data sources [15]. The code generation itself is done by an AndroMDA cartridge [16]. A lot of problems (e.g. with handling associations and generalization hierarchies) - whose explanation is out of the focus of this paper - are also solved by our framework, proving that the UML profile we proposed in this paper is applicable.

However, there are still several open issues: Currently, we are working on extending our framework by integrating a schema matching tool which proposes initial mapping elements. This would help the user to understand the schemata which have to be mapped and would support modeling (larger) mappings. Furthermore, we intend to transform the mapping specification into the native query languages of the integrated data sources (SQL, XQuery, ...) to gain efficiency.

Finally, one of the most important advantages of our approach is that it is not limited to the integration of data sources, but can also be used to specify operations on the data in a unified way. This can, for instance, be used to specify semantical notification information as proposed in [17] already on the high level of the integrated schema instead of having to use different rules for every integrated source.

Altogether, the bottom line is that our approach provides a standardized and adequate means to not only integrate data sources, but to specify integration mappings that can be used for a variety of requirements whenever information systems have to deal with several legacy data sources.

References

1. Batini, C., Lenzerini, M., Navathe, S.B.: A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys* **18**(4) (1986) 323–364
2. The Object Management Group: UML 1.4.2 Specification. <http://www.omg.org/cgi-bin/doc?formal/04-07-02> (last access: 05/2006)
3. AndroMDA: Schema2XMI Generator. <http://team.andromda.org/docs/andromda-schema2xmi/> (last access: 05/2006)
4. Wiederhold, G.: Mediators in the Architecture of Future Information Systems. Computer, IEEE Computer Society Press **25**(3) (1992) 38–49
5. Roth, M.T., Schwarz, P.M.: Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources. *Proceedings of the 23rd International Conference on Very Large Data Bases* (1997) 266–275
6. Chawathe, S., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J.D., Widom, J., Garca-Molina, H.: The TSIMMIS Project: Integration of Heterogeneous Information Sources. 16th Meeting of the Information Processing Society of Japan (1994) 7–18
7. Haas, L.M., Miller, R.J., Niswonger, B., Roth, M.T., Schwarz, P.M., Wimmers, E.L.: Transforming Heterogeneous Data with Database Middleware: Beyond Integration. *IEEE Data Engineering Bulletin* **22**(1) (1999) 31–36
8. Berler, M., Eastman, J., Jordan, D., Russell, C., Shadow, O., Stanienda, T., Velez, F.: The Object Data Standard: ODMG 3.0. Morgan Kaufmann (2000)
9. Bergamaschi, S., Castano, S., Vincini, M., Beneventano, D.: Semantic Integration of Heterogeneous Information Sources. *Data & Knowl. Eng.* **36**(3) (2001) 215–249
10. Miller, R.J., Hern?andez, M.A., Haas, L.M., Yan, L., Ho, C.T.H., Fagin, R., Popa, L.: The Clio Project: Managing Heterogeneity. *SIGMOD Record (ACM Special Interest Group on Management of Data)* **30**(1) (2001) 78–83
11. Oracle: Integrated ETL and Modeling. White Paper, <http://www.oracle.com/technology/products/warehouse/pdf/OWB.WhitePaper.pdf> (2003)
12. Altova: Data Integration: Opportunities, challenges, and MapForce. White Paper, <http://www.altova.com/whitepapers/mapforce.pdf> (last access: 05/2006)
13. Rahm, E., Bernstein, P.A.: A Survey of Approaches to Automatic Schema Matching. *VLDB Journal: Very Large Data Bases* **10**(4) (2001) 334–350
14. Kleppe, A., Warmer, J., Bast, W.: MDA Explained. The Model Driven Architecture: Practice and Promise. Addison-Wesley Longman (2003)
15. Kurz, S.: Entwicklung einer Architektur zur Integration heterogener Datenbest?ande. Diploma thesis, University of Passau; in German (2006)
16. AndroMDA: Model Driven Architecture Framework. <http://www.andromda.org/> (last access: 05/2006)
17. Guppenberger, M., Freitag, B.: Intelligent Creation of Notification Events in Information Systems - Concept, Implementation and Evaluation. In A. Chowdhury et al., ed.: *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM)*, ACM, ACM Press (2005) 52–59

Model to Text Transformation in Practice: Generating Code from Rich Associations Specifications

Manoli Albert, Javier Muñoz, Vicente Pelechano, and Óscar Pastor

Department of Information Systems and Computation

Technical University of Valencia

Camino de Vera s/n

46022 Valencia (Spain)

{malbert, jmunoz, pele, opastor}@dsic.upv.es

Abstract. This work presents a model to code transformation where extended UML association specifications are transformed into C# code. In order to define this transformation the work uses a conceptual framework for specifying association relationships that extends the UML proposal. We define a set of transformation rules for generating the C# code. The generated code extends an implementation framework that defines a design structure to implement the association abstraction. The transformation takes as input models those which are specified using the conceptual framework. Transformations have been implemented in the Eclipse environment using the EMF and MOFScript tools.

1 Introduction

Model to text transformations play a key role in MDA based methods for the development of software systems. The assets that are produced by this kind of methods usually are source code files in some programming language. Therefore, model to text transformation are used in most of the projects that apply the MDA. Currently, there is a lack of specific and widely used techniques for specifying and applying this kind of transformations. The OMG “*MOF Model to Text Transformation Language RFP*” aims to achieve a standard technique for this task. Anyway, guidelines and examples of model to text transformations are needed in order to improve the way this step is performed in MDA based methods. In this work, we introduce a model to text transformation for a specific case: the generation of code to implement in OO programming languages (C# in our example) extended UML association relationships specified at the PIM level.

In order to do this, we use a conceptual framework that was introduced in [8] for precisely specifying association relationships in Platform Independent Models. This conceptual framework defines a set of properties that provide to the analyst mechanisms for characterising the association relationships. Then, we propose a software framework for implementing them. The framework, which has been implemented using the C# programming language, applies several design patterns in order to improve the quality of the final application. Using these two items, we define transformations for automatically converting the PIMs that are defined using the

conceptual framework, into code that extends the implementation framework. We implement this model to text transformation using Eclipse plug-ins for model management. Concretely, we use EMF to persist and edit the models and MOFScript to specify and apply the model to text transformations.

In short, the main contribution of this paper is a practical application of model to text transformations for automatically generating code from PIMs. In addition, we provide knowledge (a conceptual framework, an implementation framework and a transformation mapping) for specifying and implementing association relationships. This proposal has been developed in the context of a commercial CASE tool (ONME¹), but the knowledge can be integrated in other MDA based methods, since association relationships are widely used in OO approaches.

The paper is structured as follows: Section 2 briefly presents the conceptual framework that is used in the paper for specifying association relationships. In Section 3 we show our proposal to implement association relationships in OO languages. Section 4 describes the model to text mapping and Section 5 introduces the implementation using Eclipse and MOFScript. Finally, Section 6 contains the conclusions and our future works.

2 A Conceptual Framework for Association Relationships

The meaning of the association construct, central to and widely used in the OO paradigm, is problematic. The definitions provided in the literature for this construct are often imprecise and incomplete. Conceptual modelling languages and methods, such as Syntropy[1], UML[2], OML[3] or Catalysis[4], include partial association definitions that do not solve some relevant questions. Several works have appeared highlighting these drawbacks and answering many important questions regarding associations [5, 6, 7].

To define a precise semantics for the association abstraction, we present a Conceptual Framework [8] that identifies a set of properties that have been extracted and adapted from different OO modelling methods. These properties allow us to characterize association relationships in a conceptual model.

Fig. 1 shows the metamodel for specifying associations using our approach. The three basic elements that constitute an association (the *participating classes*, the *association ends* and the *association*) are represented by metaclasses. The attributes of the metaclasses represent the properties of the conceptual framework that is introduced in the next section.

2.1 Properties of the Conceptual Framework

In this section we briefly present the properties. We introduce the intended semantics of each property in a descriptive way, and its possible values.

Dynamicity: Specifies whether an instance of a class can be dynamically connected or disconnected (creating or destroying a link) with one or more instances of a related class (through an association relationship) throughout its life-time. The property is applied to the associated ends. The values are: *Dynamic* (the connection and disconnection is

¹ <http://www.care-t.com>

possible), *Static* (the connection and disconnection are no possible), *AddOnly* (only the connection is possible) and *RemoveOnly* (only the disconnection is possible).

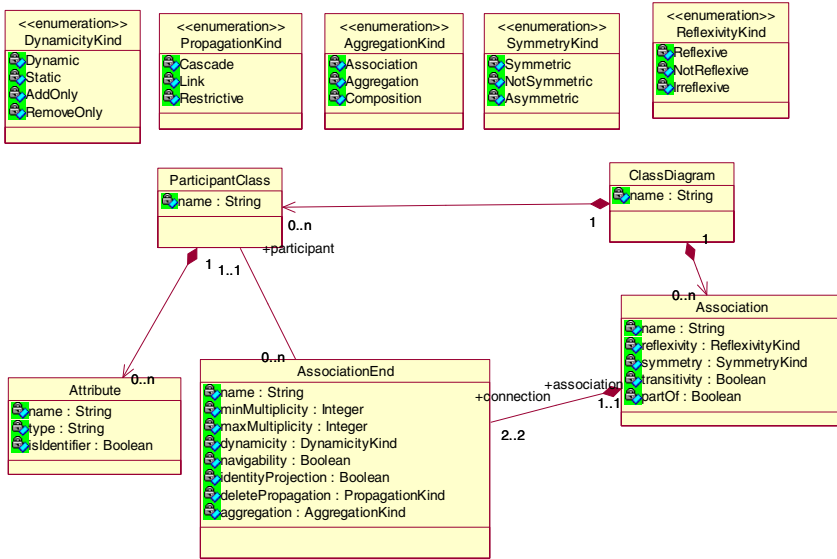


Fig. 1. Metamodel for specifying association following our conceptual framework

Multiplicity (maximum and minimum): Specifies the maximum/minimum number of objects of a class that must/can be connected to one object of its associated class. The property is applied to the associated ends.

Delete Propagation: Indicates which actions must be performed when an object is destroyed. The property is applied to the associated ends. The possible values are: *Restrictive* (the object cannot be destroyed if it has links), *Cascade* (the links and the associated objects must also be deleted) and *Link* (the links must be deleted).

Navigability: Specifies whether an object can be accessed by its associated object/s. The property is applied to the associated ends. The property value is *true* if the objects of the opposite end can access the objects of the class; otherwise the value is *false*.

Identity Projection: Specifies whether the objects of a participating class project their identity onto their associated objects. These objects are identified by their attributes and by the attributes of their associated objects. The property is applied to the associated ends. The property value is *true* if the class of the opposite end projects its identity; otherwise the value is *false*.

Reflexivity: Specifies whether an object can be connected to itself. The property is applied to the association. The possible values are: *Reflexive* (the connection is mandatory), *Irreflexive* (the connection is not possible) and *Not Reflexive* (the connection is possible but not mandatory).

Symmetry: Specifies whether a **b** object can be connected to an **a** object, when the **a** object is already connected to the **b** object. The property is applied to the association.

The possible values are: *Symmetric* (the connection is mandatory), *Antisymmetric* (the connection is not possible) and *Not Symmetric* (the connection is possible but not mandatory).

Transitivity: Specifies whether when an **a** object is connected to a **b** object, and the **b** object is connected to a **c** object, it implies that the **a** object is connected to the **c** object. The property is applied to the association. The property value is *true* if the implicit transition exists; otherwise the value is *false*.

Using this conceptual framework we can specify associations in a very expressive way. Furthermore, these properties have been used in [8] for characterizing the association, aggregation and composition concepts in the context of a commercial tool that follows the MDA proposal (the ONME Tool).

In the next section, we present the software representation of an association relationship that is characterized by the framework properties.

3 Implementing Association Relationships

Most object oriented programming languages do not provide a specific construct to deal with associations as first level citizens. Users of these languages (like C# and Java) should use reference attributes to implement associations between objects. Following this approach, an association is relegated to a second-class status. In order to solve this situation, several approaches have been proposed to implement association relationships (as it has been presented in [9]). Nevertheless, in these approaches some expressivity is missed in order to support those properties which are widely used for specifying association relationships.

3.1 Design Patterns

Our proposal for implementing associations provides a software framework that combines a set of design patterns [10]. The goal of our framework is to provide quality factors like **loose coupling** (since most of the implementation proposals introduce explicit dependencies which make difficult the maintainability of the application), **separation of concerns** (since the objects of the participating classes could have additional behaviour and structure to those specified in the domain class) and **reusability and genericity** (since most of the association behaviour and structure can be generalized for all associations.).

In order to achieve these goals, we present a solution that combines three design patterns: the **Mediator**, the **Decorator** and the **Template Method**. The next section shows how these patterns are applied to implement associations.

3.2 Framework Structure

We combine the design patterns selected to obtain a composite structure of design classes that implements an association relationship. Taking into account the association relationship and the participating classes, in this section, we present the design classes that constitute the framework.

Fig. 2 shows the generic structure of design classes that represents an archetype association (two participant classes connected through an association). Next, we describe the elements of the figure.

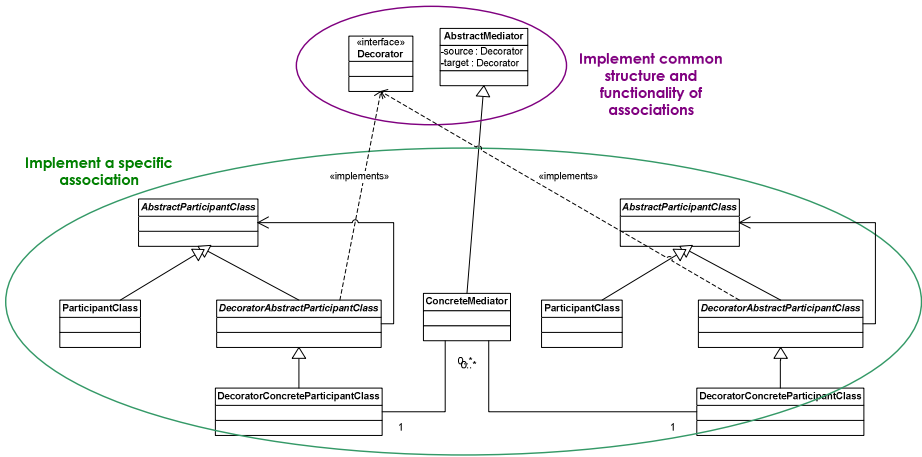


Fig. 2. Design Classes of the Implementation Framework

Participant classes (ParticipantClass in the figure), implement the participant classes of the conceptual model according to their specifications. The application of the **Decorator Pattern** implies the definition of decorated classes (DecoratorConcreteParticipantClass) that wrap the participant classes and represent the association end in which they participate. These classes implement the structure and behaviour that is added to the participant classes as a consequence of their participation in associations. Moreover, the application of this pattern results in the definition of an abstract decorator class (DecoratorAbstractParticipantClass) that generalizes every decorators (association ends) of a participant class. Finally, the pattern entails the definition of an abstract participant class (AbstractParticipantClass), from which the participant and the decorator abstract classes inherit. The decorator abstract classes keep a reference to an object of the abstract participant class, representing the decorated object. This class structure allows to use in the same way a decorated object (an object of a participant class) and a decorator object (an object of a decorator class).

The **Mediator Pattern** is applied in order to encapsulate the interaction of the participant objects. The application of this pattern results in the definition of a mediator class (ConcreteMediator) that implements the structure and behaviour of an association (independently of the participant classes). This class connects the concrete decorator classes that represent the association ends (since the participant classes are implemented in an isolated way from the association in which they participate).

The **Template Pattern** is applied in the context of the mediator class. We define an abstract class (`AbstractMediator`) for the concrete mediator class to implement the common structure and behaviour of the associations, describing common execution strategies. This class defines the template methods for the link creation and destruction and includes a reference to each participant object.

Finally, we define an interface for the decorator classes in order to specify those methods that must be implemented in the decorator classes. The abstract decorator classes implement this interface.

3.3 Functionality Implementation

In this section we present the part of the framework that regards to the functionality. The definition of an association between two classes implies the implementation of new functionality. This functionality is the following:

- *Link Creation*: allows the creation of links between objects of the participating classes. The implementation of this functionality requires checking the **reflexivity**, **symmetry**, **transitivity** and **maximum multiplicity** properties.
- *Link Destruction*: allows the destruction of links between objects of the participating classes. The implementation of this functionality requires checking the **reflexivity**, **symmetry**, **transitivity**, **minimum multiplicity** and **delete propagation** properties.
- *Participant Object Creation*: allows the creation of decorator objects independently of the creation of their decorated objects. The implementation of this functionality requires checking the **minimum multiplicity** and **reflexivity** properties.
- *Participant Object Destruction*: allows the destruction of decorator objects independently of the creation of their decorated objects. The implementation of this functionality requires the checking the **delete propagation** property.

The next section introduces the mappings between the association specification and its implementation. We also present how the properties affect the implementation of the methods that have been introduced in this section.

4 Mapping Association Specifications into Code

This section describes the transformation from models that are specified using our conceptual framework into C# source code files. First of all, we are going to intuitively describe the mapping. Then, we show the implementation of this model to text transformation.

4.1 Metaclasses Mapping

In order to describe the mapping, we introduce the implementation classes that are generated from the metaclasses of the PIM metamodel.

- **ParticipantClass:** Every *ParticipantClass* element in the model generates three classes:
 1. *AbstractDomainClass*: this class defines the attributes and operations specified in the *ParticipantClass*. Note that all the information in this class is independent of the class associations.
 2. *DomainClass*: this class implements the operations specified in the *AbstractDomainClass*.
 3. *AbstractDecorator*: this class implements the methods which are used for the management of the links (create and delete a link).
- **AssociationEnd:** Every *AssociationEnd* element in the model generates one class:
 1. *ConcreteDecorator*: this class extends the *AbstractDecorator* class, which has been generated from the *ParticipantClass* element.
- **Association:** Every *Association* element in the model generates one class:
 1. *Mediator*: this class extends the *AbstractMediator* class from the implementation framework.

The contents of these implementation classes and their methods depend on the values of properties specified in the PIM level. Next, we present briefly (due to space limitations) the representation of the properties in the framework.

4.2 Properties Mapping

Identity Projection. The value of this property determines how are implemented the identifier attributes in the *AbstractDomain C#* classes.

Dynamicity. Depending on the value of this property, methods for adding and deleting links are included in the opposite concrete decorator *C#* class.

Navigability. The value of this property determines if it is necessary to limit the access to the objects of the end by their associated objects.

Reflexivity, Symmetry, Transitivity, Multiplicity and Delete Propagation. The constraints that are imposed by these properties are checked by specific methods. The implementation of these methods depends on the values assigned to the properties. In section 3.3 we have described when these properties must be checked.

5 Transforming Models to Code. The Tools

We have implemented the transformation that has been introduced in this paper using the Eclipse environment. Eclipse is a flexible and extensible platform with many plug-ins which add functionality for specific purposes. In this work we have used the Eclipse Modelling Framework (EMF)² for the automatic implementation of the metamodel shown in Fig. 1. This metamodel provides the primitives for specifying association relationships using the properties that are defined in our conceptual framework. The EMF plug-in automatically generates the Java classes which

² <http://www.eclipse.org/emf>

implement functionality for creating, deleting and modifying the metamodel elements, and for the models serialization.

In order to implement the model to text transformation, we have used the MOFScript tool that is included in the Generative Model Transformer (GMT)³ Eclipse project. The MOFScript tool is an implementation of the MOFScript model to text transformation language. This language was submitted to the OMG as response to the “*MOF Model to Text Transformation Language RFP*”.

In this work, **we have selected the MOFScript language/tool for several reasons**: (1) MOFScript is a *language specifically designed for the transformation of models into text files*, (2) MOFScript *deals directly with metamodel descriptions* (Ecore files) as input, (3) MOFScript transformations *can be directly executed from the Eclipse environment* and (4) MOFScript *provides a “file” constructor* for the creation of the target text files.

MOFScript provides the “*texttransformation*” constructor as the main language primitive for organizing the transformation process. A transformation takes as input a metamodel, and it is composed of one or several rules. Every rule is defined over a context type (a metamodel element). Rules can have arguments and/or return a value. The special rule called “main” is the entry point to the transformation.

5.1 Implementing the Transformation Using the MOFScript Tool

We have structured our transformation in several modules:

- We define a specific transformation for each kind of class (file) (ConcreteDecorator, Mediator, etc.).
- The root transformation is in charge of navigating the model and invoking the specific transformations.

Next we show the root transformation, which takes as input a model that is specified using our metamodel. The main rule iterates (using the *forEach* MOFScript constructor) over *ParticipantClass* and *Association* elements. Moreover, the rule iterates over the *AssociationEnd* elements of every *ParticipantClass*. The files are generated following the mapping described in Section 4.

```
import "ParticipantAbstract.m2t"
import "Participant.m2t"
import "DecoratorAbstract.m2t"
import "DecoratorConcrete.m2t"
import "MediatorConcrete.m2t"
import "Mediator.m2t"

texttransformation Association2CSharp (in asso:"http://associationmodel.ecore"
) {

  asso.ClassDiagram::main(){

    self.ParticipantClass->forEach(c:asso.ParticipantClass) {
      file (c.name+"Abstract.cs")
      c.generateParticipantAbstractClass()
      file (c.name+".cs")
      c.generateParticipantClass()
      file ("Decorator"+c.name+"Abstract.cs")
      c.generateDecoratorAbstractClass()
      c.AssociationEnd->forEach(end:asso.AssociationEnd){
        file ("Decorator" + c.name + end.association.name + ".cs")
        end.generateDecoratorConcreteClass()
      }
    }
  }
}
```

3 <http://www.eclipse.org/gmt/>

```

    }
}
self.Association->forEach(a:asso.Association){
    file("Mediator"+ a.name +".cs" )
    a.generateMediatorConcreteClass()
}
file("Mediator.cs")
self.generateMediatorClass()
}
}

```

The description of the transformations that generate every file can not be included in this paper due to space constraints. An Eclipse project with the transformation can be downloaded from <http://www.dsic.upv.es/~jmunoz/software/>. Next, we (partially) show the transformation that is in charge of generating the decorator concrete classes.

```

01 texttransformation DecoratorConcrete (in asso:"http://associationmodel.ecore")
02 {
03   asso.AssociationEnd::generateDecoratorConcreteClass(){
04     <%using System;
05     using System.Collections;
06     using System.Text;
07     namespace org.oomethod.publications {
08       public class Decorator%> self.participant.name + self.association.name
09     <%:Decorator%>self.participant.name<%Abstract {
10
11     %>
12     //Definition of the collection reference
13     self.association.conexion->forEach(con:asso.AssociationEnd | con <> self ) {
14     <%
15       ArrayList %> con.participant.name <%sCollection;
16     %>
17     }
18     //Constructor
19     //...
20     //Insert Link Method
21     self.association.conexion->forEach(con:asso.AssociationEnd | con<>self ){
22       if ( con.dynamcity=="Dynamic" or con.dynamcity=="AddOnly" ){
23         <%
24           public Mediator insert%>
25           print(con.participant.name)<% (Decorator%> con.participant.name +
26           self.association.name
27           <% object) {
28             return (new Mediadora%> self.association.name <%(object, this));
29           }
30     %>
31       }}
32     //...
33   }

```

This transformation creates a decorator concrete class for each association end associated to a class. Next, we describe the most relevant issues of the transformation:

- Lines 08-09: The class inherits from its corresponding abstract decorator class.
- Lines 12-17: A collection is defined to maintain a reference to the links. The name of this collection is based on the name of the class at the opposite end.
- Lines 20-31: An insert link method is created depending on the value of the dynamcity property of the opposite end. If the value is *Dynamic* or *AddOnly* the method is defined. Otherwise the method is not defined. The name of the method is based on the name of the class at the opposite end.

6 Conclusions

In this work we have introduced a practical case study of model to text transformation. This transformation takes as input models that are specified using the primitives of a conceptual framework for precisely specifying association relationships. The results of the transformation are C# classes which implement the association relationships using design patterns.

In the context of MDA, the transformation introduced in this paper is a PIM to Code transformation. We do not explicitly use intermediate PSMs for representing the C# classes. Currently, we are working on the development of the PIM-PSM-Code implementation of the transformation that has been introduced in this paper. This work will provide a precise scenario for the comparison of both approaches.

Another line of research includes the implementation of the association-to-C# transformation for the persistence and presentation layers. These layers are currently implemented by hand, but we have defined the correspondence mappings. Our goal is to automate these mappings using a similar approach to the one that has been introduced in this paper.

References

1. S. Cook and J. Daniels. *Designing Objects Systems*. Object-Oriented Modelling with Syntropy. Prentice Hall, 1994.
2. Object Management Group. *Unified Modeling Language Superstructure*, Version 2.0. 2005
3. Firesmith, D.G., Henderson-Sellers, B. and Graham, I. *OPEN Modeling Language (OML) Reference Manual*, SIGS Books, 1997, New York, USA.
4. D.F. D'Souza and A.C. Wills. *Objects, Components and Frameworks with UML*. Addison-Wesley, 1998.
5. Gonzalo Genova. "Entrelazamiento de los aspectos estático y dinámico en las asociaciones UML" PhD thesis, Dept. Informática. Universidad Carlos III de Madrid. 2003.
6. Monika Saksena, Robert B. France, María M. Larrondo-Petrie. "A characterization of aggregation.", In Proceedings of OOIS'98, Springer editor, pp 11-19. C. Rolland, G. Grosz, 1998
7. Brian Henderson-Sellers and Frank Barbier. "Black and White Diamonds". In Proceedings of UML'99. The Unified Modeling Language Beyond the Standard, 1999, Springer-Verlag, R.France and B.Rumpe editors, pp 550-565.
8. Manoli Albert, Vicente Pelechano, Joan Fons, Marta Ruiz, Oscar Pastor. "Implementing UML Association, Aggregation and Composition. A Particular Interpretation Based on a Multidimensional Framework". In Proceedings of CAISE 2003, LNCS 2681 pp 143-158.
9. M. Dahchour. "Integrating Generic Relationships into Object Models Using Metaclasses", PhD thesis, Dept. Computing Science and Eng., Université Catholique de Louvain, Belgium, Mar. 2001.
10. E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading, MA, 1994.

Preface for CoMoGIS 2006

Christelle Vangenot¹ and Christophe Claramunt²

¹ EPFL, Switzerland

² Naval Academy Research Institute, France

These proceedings contain the papers selected for presentation at the third edition of the International Workshop on Conceptual Modeling in GIS, CoMoGIS 2006, held in November 2006, in Tucson, Arizona, USA, in conjunction with the annual International Conference on Conceptual Modeling (ER 2006). Following the success of CoMoGIS 2005 (held in Klagenfurt, Austria) and CoMoGIS 2004 (held in Shanghai, China), its aim was to bring together researchers investigating issues related to conceptual modeling for geographic and spatial information handling systems, and to encourage interdisciplinary discussions including the identification of emerging and future issues. The call for papers attracted 19 papers, many of them of excellent quality and most of them very related to the topics of the workshop. Each paper received three reviews. Based on these reviews, eight papers were selected for presentation and inclusion in the proceedings. The accepted papers cover a wide range of topics from spatial and spatio-temporal data representation, to indexing methods for moving objects, to optimization of spatial queries and to spatio-temporal data on the Web. Our keynote presentation, given by Peter Baumann, discussed large-scale raster services.

The workshop would not have been a success without the efforts of many people. We wish to thank the authors who contributed to this workshop for the high quality of their paper and presentation. We would also like to thank all the Program Committee members for the quality of their evaluations and the ER06 workshop and local organizers for their help. Furthermore we would like to thank Peter Baumann for accepting to be our keynote speaker.

Large-Scale Earth Science Services: A Case for Databases

Peter Baumann

International University Bremen
Campus Ring 12, D-28759 Bremen
p.baumann@iu-bremen.de
<http://www.faculty.iu-bremen.de/pbaumann>

Abstract. Earth sciences are about to follow the mapping domain where raster data increasingly get integrated in online services, contribute by far the largest volume. Interestingly, although more and more raster services are getting online, there is few work on a comprehensive model of raster services, rather architectures are of ad-hoc style and optimized towards very narrow applications, such as fast zoom and pan on seamless maps.

We claim that databases introduce a new quality of service on high-volume multi-dimensional earth science raster data, characterized by clear and understandable concepts, extensibility, and scalability. To support this, we present a comprehensive conceptual model for raster data in earth science and discuss how an efficient architecture can be derived from it, which is implemented in the rasdaman system. Further, we show how such concepts play a role in the development of OGC's geo raster services, WCS and WCPS. Finally, we discuss some research challenges.

Keywords: Raster service, coverage service, rasdaman, OGC.

1 Motivation

Raster data recently increasingly receive attention not only for scientific applications, but even for everyday's convenience such as Internet map services. Advances in storage technology, processing power, and data availability make online navigation on large data volumes feasible. Hence, more and more remote sensing image services are getting online, however usually based on ad hoc implementations. Interestingly there is few work on a comprehensive theory of raster services, rather architectures are optimized towards very narrow applications, such as fast zoom and pan on seamless maps.

In this contribution we claim that database concepts and methods can contribute to an increased quality of service which is characterized by clear (hence, easy to handle) concepts, extensibility, and a potential for high-performance implementations. This opens up new avenues into online data analysis and, generally, advanced services.

In Section 2 we describe the state of the art. Our viewpoints are illustrated through the rasdaman raster middleware for retrieval of n-D raster data stored in relational

databases [2, 17, 22] in Section 3. In Section 4 we report the state of standardization in the field of geo raster services looking at OGC's Web Coverage Service (WCS) and Web Coverage Processing Service (WCPS). Section 5 concludes the plot.

2 State of the Art

Traditionally, raster data have been stored in sets of files. As the file system has no idea about the semantics (pixel type, number of dimensions, etc.), all selection and processing is burdened to the application developer, leading to tedious, repetitive work and an ill-defined consistency state of the data. Moreover, file-based storage tends to favour particular access patterns (e.g., x/y selection on time series) while conveying disastrous performance on all others (e.g., z/t selection).

For fast zoom and pan on mosaicked image file sets, many products are available. Access is done through low-level API libraries instead of high-level, model-based query support with internal optimisation and without flexible image extraction functionality, such as hyperspectral channel extraction, overlaying, and *ad hoc* thematic colouring. Most important, file-based solutions *per se* do not scale very well and are inflexible wrt. new requirements. Optimisations done to speed up performance consist in adopting, not to say: re-inventing, one or the other of the set of techniques known in the database community since long – for example, spatial indexing, load balancing, preaggregation, and materialized views.

Relational DBMSs, designed to scale well indeed, traditionally store multi-dimensional arrays as unstructured BLOBs (“binary large objects”) introduced by Lorie as “long fields” [12]. This technique cannot give any support for operations beyond line-by-line access, something clearly not feasible for large archives. Tiling, a technique stemming from imaging, has therefore been introduced to databases by *rasdaman* [1] and recently has been adopted by ESRI's ArcSDE [7].

Object-relational database systems (ORDBMSs) allow to add new data types, including access operations, to the server [20]; examples for such a data type is Oracle GeoRaster [15]. Arrays, however, are not a data *type*, but a data type *constructor* (“template”), parametrized with cell type and dimension – see Section 3.1. Such templates are not supported by ORDBMSs, hence a separate data type has to be defined for 2-D grayscale ortho images, 2-D hyperspectral MODIS images, 4-D climate models, etc. Furthermore, server internal components are not prepared for the kind of operations occurring in MDD applications, therefore important optimization techniques like tile-based pipelining and parallelization of array query trees are difficult to implement.

A literature review specifically on raster databases has been conducted in [21]. Interesting research focuses on specific raster database aspects, such as mass storage support for extreme object sizes [19, Reiner-02] and data models [11, 13]. To the best of our knowledge, *rasdaman* currently is the only system which combines a formal framework, a declarative, optimizing query language, a system architecture streamlined to large n-D raster objects, and an implementation that is in commercial use.

3 The Rasdaman Raster Server

The rasdaman¹ system has evolved from a series of research projects. Based on an algebraic foundation inspired by the AFATL Image Algebra [18] it provides a raster query language based on SQL92; storage [22, 8] and query optimization likewise are grounded algebraically [2].

Conceptual Model. The conceptual model of rasdaman centers around the notion of an n-D array (in the programming language sense) which can be of any dimension, spatial extent, and array cell type. Following the relational database paradigm, rasdaman also supports sets of arrays. Hence, a rasdaman database can be conceived as a set of tables where each table contains a single array-valued attribute, augmented with an OID system attribute. As rasdaman is domain-neutral, its semantics does not include geo coordinates; this is to be maintained by an additional layer on top of rasdaman.

Arrays can be built upon any valid C/C++ type, be it atomic or composed, based on the type definition language of the ODMG standard [4]. Arrays are defined through a template `marray<b,d>` which is instantiated with the array base type `b` and the array extent (*spatial domain*) `d`, specified by the lower and upper bound for each dimension (which can be left open for variable arrays). Thus, an unbounded colour ortho image can be defined by

```
typedef marray
< struct{ char red, green, blue; }, [ *:* , *:* ]
> RGBOrthoImg;
```

Type definitions serve for both semantic checks and optimization during query evaluation [17].

Array Retrieval. The rasdaman query language, `rasql`, adds n-D raster expressions to ISO SQL92. Like SQL, a `rasql` query returns a set of items (in this case, raster objects or metadata information). *Trimming* produces rectangular cut-outs, specified through the corner coordinates; a *section* produces a slice with reduced dimensionality.

Example 1: “A slice at time t through $x/y/t$ cube `SatTimeSeries`, cutting out the area between (x_0,y_0) and (x_1,y_1) ”:

```
select SatTimeSeries[x0:x1,y0:y1,t]
from   SatTimeSeries
```

For each operation available on the cell (i.e., pixel) type, a corresponding *induce operation* is provided which simultaneously applies the base operation to all raster cells. Both unary (e.g., record access) and binary operations (e.g., masking and overlaying) can be induced.

Example 2: “Color ortho image `Ortho`, overlaid with bit layer 3 of thematic map `TMap` coloured in red”:

```
select Ortho overlay bit( TMap, 3 ) * {255c,0c,0c}
from   TMap, Ortho
```

¹ “raster data manager”, www.rasdaman.com

In general, raster expressions can be used in the `select` part of a query and, if the outermost expression is of type Boolean, also in the `where` part [2].

Condense operations derive summary data. The general condenser is of the form

```
condense op
over      x in dom
using     expr
```

This expression will iterate over domain `dom`, binding variable `x` to each location in turn and evaluating `expr`, which may contain occurrences of `x`; all evaluation results are combined through binary operation `op` which must be commutative and associative so that evaluation sequence can be chosen by the server. Shorthands are available for the frequently used special cases `sum`, `max`, `min`, `average`, and for boolean quantifiers, similar to the SQL aggregate functions.

Example 3: "For all ortho images Ortho, the ratio between min and max in the near infrared component":

```
select min_cells(Ortho.nir) / max_cells(Ortho)
from   Ortho
```

Finally, the *marray* constructor allows to establish a new raster structure, possibly derived from an existing one. Following the syntax

```
marray x in dom
values expr
```

the query engine creates a new array of domain `dom` and cell type identical to the cell type of expression `expr`; then, it iterates over domain binding variable `x` to each location in turn, evaluates expression `expr` (which may contain occurrences of `x`), and finally assigns the result to the resp. cell in the result.

Example 4: "Histogram over the red channel of Ortho ":

```
select marray bucket in [0..255]
      values add_cells( Ortho.red = bucket )
from   Ortho
```

The result raster object is a 1-D array containing 256 entries for all possible 8-bit intensity values. The induced comparison of the red channel with scalar value `bucket` yields a boolean result which is interpreted as 0 or 1, resp., and then summed up for each bucket.

Actually, general condenser plus operator are sufficient to describe all rasql raster operations; induce, trim, section operations etc. are just shorthands for convenience. This narrow basis significantly eases both conceptual treatment and implementation.

The expressiveness of rasql enables a wide range of signal processing, imaging, and statistical operations up to, e.g., filter kernels. The expressive power tentatively has been limited to non-recursive operations, thereby guaranteeing termination of any well-formed query; this principle is known as "safe evaluation" in query languages.

Array Storage. Raster objects are maintained in a standard relational database, based on the partitioning of an raster object into *tiles* [8]. Aside from a regular subdivision, any user or system generated partitioning is possible (Fig. 1); several different

strategies are available. A geo index is employed to quickly determine the tiles affected by a query. Optionally tiles are compressed using one of various choices, using lossless or lossy (wavelet) algorithms; independently, query results can be compressed for transfer to the client. Both tiling strategy and compression comprise database tuning parameters.

Tiles and index are stored as BLOBs in a relational database which also hold the data dictionary needed by rasdaman's dynamic type system. Adaptors are available for several relational systems, among them open-source PostgreSQL. Interestingly, PostgreSQL has shown adequate to serve multi-Terabyte ortho imagery under industrial conditions.

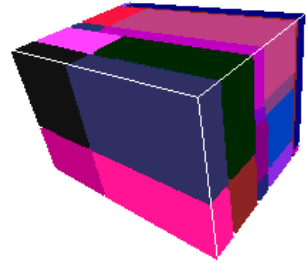


Fig. 1. Tiled 3-D raster object

Query Evaluation. Queries are parsed, optimised, and executed in the rasdaman server. The parser receives the query string and generates the operation tree. Further, it applies algebraic optimisation rules to the query tree where applicable [17]; of the 150 algebraic rewriting rules, 110 are actually optimising while the other 40 serve to transform the query into canonical form. Parsing and optimization together take less than a millisecond on a PC.

Query execution is parallelised. First, rasdaman offers inter-query parallelism: A dispatcher schedules requests into a pool of server processes on a per-transaction basis. Ongoing research work involves intra-query parallelism where the query tree transparently is distributed across available CPUs or computers in the network so that each CPU evaluates a subset of the tiles [9]. Preliminary performance results are promising, showing speed-up / #CPU ratios of 95.5%.

For arrays larger than disk space, hierarchical storage management (HSM) support has been developed [16].

Performance. As benchmarks show [3], databases can not only compete with, but outperform file-based raster services. Several factors contribute to this:

- Optimized tiling makes the server fetch much less data from disk. E.g., 4-D climate models are generated and stored in x/y slices, while evaluation often is along the time axis. Similar effects occur with remote sensing time series sliced along t .
- Enabling the client to send one complex request instead of a long sequence of atomic operations opens up space for the optimizer. Query optimization particularly pays off then. Even in a “simple” WMS request containing overlays and layer coloring the rasdaman optimizer can achieve high gain.
- Further, by offering complex operations to the client there is much less communication overhead: no intermediate results have to be transferred back and forth, and the final result is exactly what the client needs, hence the minimum amount of data that is required to answer the client's needs (cf. Fig. 2 right).

In summary, our observation is that databases can outperform file-based approaches whenever flexibility for complex ad-hoc requests and high scalability is required.

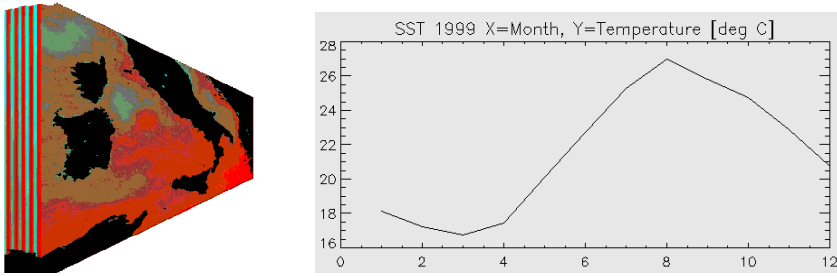


Fig. 2. DLR AVHRR SST time series service: 3-D trim around Italy, Sicily, and Corsica (left) and 1-D section obtaining the temperature curve for a chosen location. Data obtained from the rasdaman/Oracle server through WCS.

Status. The rasdaman system is in operational use since many years both in mapping agencies, in (mining) industry, and in research. Fig. 2 shows a cutout from a rasdaman/Oracle database where about 10,000 images have been merged into a seamless 3-D $x/y/t$ cube of AVHRR sea-surface temperature imagery. Implemented in 2002, it uses an early-version WCS interface. The experience gained there now is being exploited in OGC for raster geo service standardization.

4 Geo Raster Service Standardization

The main driver for interoperable geo service standards is the Open GeoSpatial Consortium (OGC). The Web Coverage Service (WCS) which specifies retrieval of all or part of a coverage object; although coverages have a more general definition, WCS currently supports only regularly gridded rasters. Based on the WCS model, the Web Coverage Processing Service (WCPS) extends WCS with a coverage expression language allowing requests of unlimited nesting complexity. Having two different standards, WCS and WCPS, increases modularity: implementers can choose to implement only the – relatively simple – WCS, or to undertake a WCPS implementation, which is more challenging.

4.1 WCS

Based on the coverage definitions of ISO 19123 [10], WCS specializes on retrieval from coverages, with emphasis on providing available data together with their detailed (metadata) descriptions and returning data with its original semantics (instead of pictures, like, e.g., WMS).

In WCS 1.1, a coverage is seen as a 2-D, 3-D, or 4-D spatio-temporal matrix (aka tensor). The coverage extent is called its *domain*, the coverage value data type comprises the coverage *range*. A range definition consists of a list of fields; a field is either atomic (such as temperature), or compound, in which case it consists of an n -D tensor which can be addressed along the axes (in WCS speak: keys chosen from key-lists), very much like array indexing.

To accommodate orthorectified/georeferenced, georeferenced, and non-georeferenced imagery, a coverage can bear either a ground Coordinate Reference System (CRS), or an image CRS, or both. Addressing is possible via either CRS. A bounding box is associated with each coverage, expressed in the resp. CRS(s).

Operationally, a WCS client first obtains information about the service and coverage offerings; Subsequently, coverages or subsets thereof can be retrieved using a `GetCoverage` request. Requests and responses are exchanged via http, using either key/value pair (KVP) or XML encoding; the XML structures are laid down using XML Schema. `GetCoverage` offers a set of six operations to be executed on a coverage, controlled by the request parameters: spatio-temporal domain subsetting; range subsetting (aka “band” selection, addressing via fields and their keys, if defined); re-sampling (e.g., scaling); reprojection; data format encoding; and result file(s) delivery; results can optionally be delivered directly in the `GetCoverage` response, or stored by the server for later download by the client.

4.2 Web Coverage Processing Service (WCPS)

The Web Coverage Processing Service (WCPS) Implementation Specification has been designed to support non-trivial navigation and analysis on large-scale, multi-dimensional sensor and image data. In the sequel we outline the WCPS concepts, see [14] for details.

Coverage Model. WCPS grounds in its coverage model on WCS, with one extension: coverages can have not only spatio-temporal, but also abstract axes². Examples of such abstract axes are simulation time for climate model computations, input parameter spaces for climate models, or statistical feature spaces. This semantics is known to the server; for example, operations that refer to geo coordinates – such as reprojection – can only be applied to x and y axes. All axes are treated equally in the operations; for example, subsetting and slicing operations can be performed on every axis.

The coverage locations containing values are referred to as *cells*; the (atomic or composite) values associated with a particular cell are called its *cell values*.

Operational Model. The WCPS request structure corresponds to WCS, with only some technically motivated extensions. In place of the WCS `GetCoverage` request, WCPS offers `ProcessCoverage` where the retrieval expression is passed to the server. The WCPS language is defined as an abstract language, with mappings to both key-value pair (KVP) and XML encodings.

A WCPS `ProcessCoverage` request consists of a central request loop where each of a list of coverages in turn is visited to instantiate the associated processing expression if an optionally provided predicate is fulfilled. Coverage names refer to the the list advertised by the service. WCPS Abstract Syntax for the request loop is

² This feature is being discussed in the WCS group for possible inclusion in WCS 1.2.

```

for c in ( coverageList )
[ where cond(c) ]
return pExpr(c)

```

Variable *c* iterates over the coverages enumerated in *coverageList*, considering only those where the predicate *cond(c)* is fulfilled. The processing expression *pExpr(c)* consists of either a metadata accessor operation (such as *tDom(c)* returning *c*'s temporal domain extent), or of an encoding expression

```

encode(e, f)

```

where *e* is a coverage-valued expression and *f* is the name of a supported format.

Example: “Coverages A, B, and C, TIFF-encoded”:

```

for c in ( A, B, C )
return encode( c, "tiff" )

```

WCPS operations are very similar in nature to *rasql*, extended with geo-specific functionality like mapping of spatial and temporal coordinates into cell coordinates and reprojection. For example, if a client wishes to express slicing in spatio-temporal coordinates, it needs to explicitly apply the coordinate mapping function *ttransform()*.

Example: “Slice through A at time ‘Thu Nov 24 01:33:27 CET 2005’”:

```

for c in ( A )
return encode( sect(c,3, ttransform(c,
                                "Thu Nov 24 01:33:27 CET 2005")), "tiff" )

```

This may seem complex and unwieldy. However, this allows addressing on both geo and pixel level; further, this language is not intended for humans – rather, some GUI client will offer click-and-point request composition and then internally generate and ship the corresponding XML request.

Like in *rasql*, *induce expressions* allows to simultaneously apply a cell operation to a coverage as a whole.

Example: “Sum of red and near infrared band from coverage A, as 8-bit integer”:

```

for c in ( A )
return encode( (char) c.red+c.nir, "tiff" )

```

Operations involving interpolation allow to optionally indicate an interpolation method.

Example: “Coverage A, scaled to (a,b) along the time axis”; this assumes that the server offers cubic time interpolation on this coverage:

```

for c in ( A )
return encode( scale(c,t,a,b,"cubic"), "tiff" )

```

The encoded result coverage(s) are, at the client’s discretion, either shipped back immediately as XML response or stored at the server for some time and only the URL is returned. Scalar results (like condensers and metadata) are returned immediately.

The reference implementation currently under [5, 6] way maps WCPS requests received to rasdaman queries, lets rasdaman compose the result coverage(s), and finally adds the XML response metadata. Implementation language is Java.

5 Conclusion

All across the earth sciences the vaults of huge raster data archives need to be opened for scientific exploitation. Retrieval technology for large raster repositories, however, is lagging behind. Raster archives today commonly are implemented in a file-based manner; databases serve only for meta data search, but not for image retrieval itself. Hence, storage usually is driven by the data acquisition process rather than by user access patterns, resulting in inefficient access. Scalability often is achieved only through massive hardware support. Moreover, versatile retrieval as known from database query languages for alphanumeric data is impossible, not to speak of other services like transactions.

Aside from flexibility in task definition, there are several more arguments which advocate the use of database systems. Query languages allow to define complex tasks to the server rather than a small set of atomic steps as in procedural APIs; the consequence is that the query optimiser gains a lot of freedom to rephrase the query optimally for the particular situation. Further, application integration is much higher because there is one central instance in charge of data integration and consistency. All in all, file-based solutions frequently re-invent all the features which have been developed by database technology over decades; using existing mature technology obviously is preferable.

The new generation of geo service standards taps into experience from different domains, among them databases. WCPS is an example where a coverage expression language is defined in a way such as to allow open-ended functionality through few, well-defined concepts which are abstract enough to allow for many transparent server-side optimizations.

WCS 1.1 is expected to be released in Fall 2006. WCPS currently is OGC approved Best Practices Paper and is expected to become Draft Standard in Fall/Winter 2006.

Current work encompasses upgrading of the rasdaman-based WCS 1.0 implementation to the forthcoming version 1.1, and finalizing the WCPS reference implementation, including the server [5], a visual programming client [6] and a compliance test suite.

Acknowledgements

The author is indebted to John Evans (NASA; WCS co-editor) and Arliss Whiteside (BEA Systems; WCS co-editor); Steven Keens (PCI Geomatics); Luc Donea (Ionic); Peter Vretanos (Cubewerx); Wen-Li Yang (NASA); and all the other WCS Revision Working Group members; particular thanks goes to Sean Forde (Lizardtech; Coverages WG speaker).

Further, many thanks to Georgi Chulkov and Ivan Delchev for their great work implementing WCS/WCPS server and clients, and for patiently exploring all the conceptual pitfalls.

References

1. Baumann, P.: Database Support for Multidimensional Discrete Data. Proc. 3rd Intl Symp. on Large Spatial Databases, LNCS 692, Springer 1993, pp. 191-206
2. Baumann, P.: A Database Array Algebra for Spatio-Temporal Data and Beyond. Proc. Next Generation IT and Systems (NGITS), Zikhron Yaakov, Israel, 1999, pp. 76 – 93
3. Baumann, P.: Web-enabled Raster GIS Services for Large Image and Map Databases. Special Track on Image-Based Geospatial Databases, 5th Int'l Workshop on Query Processing and Multimedia Issues in Distributed Systems (QPMIDS'2001), September 3-4, 2001, Munich, Germany
4. Cattell, R.G.G.: The Object Database Standard: ODMG 2.0. Morgan Kaufmann Publishers, California, 1997
5. Chulkov, G.: Architecture and Implementation of a Web Coverage Processing Service Using a Database Back-End. IUB Bachelor's Thesis, May 2006
6. Delchev, I.: Graphical Client for a Multidimensional Geo Raster Service. IUB seminar report, May 2006
7. n.n.: Raster Data in ArcSDE 9.1. ESRI White Paper, September 2005
8. Furtado, P. , Baumann, P.: Storage of Multidimensional Arrays Based on Arbitrary Tiling. Proc. ICDE, Sydney, Australia, 1999, pp. 480-489
9. Hahn, K., Reiner, B., Höfling, G., Baumann, P.: Parallel Query Support for Multidimensional Data: Inter-object Parallelism. Proc. DEXA, Aix-en-Provence, France, 2002.
10. n.n.: FDIS 19123 Geographic information - Schema for coverage geometry and functions. ISO/TC 211, document 1740, October 2004
11. Libkin, L., Machlin, R., Wong, L.: A Query Language for Multidimensional Arrays: Design, Implementation, and Optimization Techniques, Proc. ACM SIGMOD, Montreal, Canada, 1996, pp. 228 – 239
12. Lorie, R.A.: Issues in Databases for Design Transactions. in: Encarnação, J., Krause, F.L. (eds.): File Structures and Databases for CAD, North-Holland Publishing, 1982
13. Marathe, A.P., Salem, K., Query Processing Techniques for Arrays. Proc. ACM SIGMOD '99, Philadelphia, USA, 1999, pp. 323-334
14. OGC: Web Coverage Processing Service. OGC Best Practices Paper, artifact 06-102, August 2006, available from OGC Portal (www.opengis.org)
15. n.n.: Oracle Database 10g: Managing Spatial Raster Data using GeoRaster. Oracle Technical Whitepaper, May 2005
16. Reiner, B., Hahn, K., Höfling, G., Baumann, P.: Hierarchical Storage Support and Management for Large-Scale Multidimensional Array Database Management Systems. Proc. DEXA, Aix en Provence, France, 2002
17. Ritsch, R.: Optimization and Evaluation of Array Queries in Database Management Systems. PhD Thesis, Technische Universität München, 1999.
18. Ritter, G., Wilson, J., Davidson, J.: Image algebra: An Overview. Computer Vision, Graphics, and Image Processing, 49(1):297-331; 1990
19. Sarawagi, S., Stonebraker, M.: Efficient Organization of Large Multidimensional Arrays.. Proc. ICDE'94, Houston, USA, 1994, pp. 328-336
20. Stonebraker, M., Moore, D., Brown, P.: Object-Relational DBMSs: Tracking the Next Great Wave (2nd edition). Morgan Kaufmann, September 1998
21. Varsandan, I.: State of the Art: Arrays in Databases. IUB Bachelor's Thesis, May 2006
22. Widmann, N.: Efficient Operation Execution on Multidimensional Array Data. PhD Thesis, Technische Universität München, 2000

Time-Aggregated Graphs for Modeling Spatio-temporal Networks*

An Extended Abstract

Betsy George** and Shashi Shekhar

Department of Computer Science and Engineering,
University of Minnesota,
200 Union St SE, Minneapolis, MN 55455, USA
{bgeorge, shekhar}@cs.umn.edu
<http://www.cs.umn.edu/research/shashi-group/>

Abstract. Given applications such as location based services and the spatio-temporal queries they may pose on a spatial network (eg. road networks), the goal is to develop a simple and expressive model that honors the time dependence of the road network. The model must support the design of efficient algorithms for computing the frequent queries on the network. This problem is challenging due to potentially conflicting requirements of model simplicity and support for efficient algorithms. Time expanded networks which have been used to model dynamic networks employ replication of the network across time instants, resulting in high storage overhead and algorithms that are computationally expensive. In contrast, the proposed time-aggregated graphs do not replicate nodes and edges across time; rather they allow the properties of edges and nodes to be modeled as a time series. Since the model does not replicate the entire graph for every instant of time, it uses less memory and the algorithms for common operations (e.g. connectivity, shortest path) are computationally more efficient than the time expanded networks.

Keywords: Time-aggregated graphs, shortest paths, spatio-temporal data-bases, location based services.

1 Introduction

Growing importance of application domains such as location-based services and evacuation planning highlights the need for efficient modeling of spatio-temporal networks (e.g. road networks) that takes into account changes to the network over time. The model should provide the necessary framework for developing

* This work was supported by the NSF/SEI grant 0431141, Oak Ridge National Laboratory grant and US Army Corps of Engineers (Topographic Engineering Center) grant. The content does not necessarily reflect the position or policy of the government and no official endorsement should be inferred.

** Corresponding author.

efficient algorithms that implement frequent operations posed on such networks. A frequent query that is posed on such networks is to find the shortest route from one place to another or a search for the nearest neighbor. The shortest route would depend on the time dependent properties of the network such as congestion on certain road segments, which would increase the travel time on that segment. The result of nearest neighbor search could also be time sensitive if it is based on a road network.

Modeling such a network poses many challenges. Not only should the model be able to accomodate changes and compute the results consistent with the existing conditions, it should do so accurately and simply. In addition, the need to answer frequent queries quickly means fast algorithms are required for computing the query results. The model should thus provide sufficient support for the design of correct and efficient algorithms for the frequent computations.

Often dynamic networks have been modeled as time expanded networks, where the entire network is replicated for every time instant. The changes in the network, especially the travel time variations, can be very frequent and for modeling such frequent changes, the time expanded networks would require a large number of copies of the original network, thus leading to network sizes that are too memory expensive. For example, traffic sensors on highway networks send measurement data every 30 seconds. A one-year dataset may need over one million copies of the road network, which itself may have a million nodes and edges for each time instant. Such large sized networks would also result in computationally expensive algorithms.

The proposed model, a time-aggregated graph, models the changes in a spatio-temporal network by collecting the node/edge attributes into a set of time series. The model can also account for the changes in the topology of the network. The edges and nodes can disappear from the network during certain instants of time and new nodes and edges can be added. The time-aggregated graph keeps track of these changes through a time series attached to each node and edge that indicates their presence at various instants of time. Our analysis shows that this model is less memory expensive and leads to algorithms that are computationally more efficient than those for the time expanded networks.

1.1 An Illustrative Application Domain

Location based services find the geographical location of a mobile device and then provide services based on that location [11]. Most of these services rely heavily on road maps, spatial networks that can change with time. For example, the travel times associated with road segments can change over time. One of the most frequent computations performed on a road network is to identify the shortest route from one point in the network to another. The result of this query will depend on the availability of road segments and the time taken to traverse them; these parameters are time-dependent and hence are the results of the shortest route queries. The results to another frequent query, the nearest neighbor query, can also be time dependent, if computed on a road network; the accessibility of various points in a road network can vary with time, depending on

the connectivity of the network at different instants of time. The need to answer such queries in location based services on a spatial network that varies with time makes a simple, efficient model for spatio-temporal networks a necessity.

Such a model is even more critical to applications related to evacuation planning. Route finding here involves identifying paths in a transportation network to minimize the time needed to move people from disaster-impacted areas to safe locations [6]. One key step in this operation is finding the fastest possible evacuation routes in the network. In computing these routes, it is critical to honor the time dependence of the parameters like travel time (which would change with congestion on roads) and the road capacities. Failure to do so could affect the quality of the solution and even create chaos in an emergency situation.

1.2 Problem Formulation

Spatial networks that show time-dependence serve as the underlying networks for most location based services. Models of these networks need to capture the possible changes in topology and values of network parameters with time and provide the basis for the formulation of computationally efficient and correct algorithms for the frequent computations like shortest paths. We formulate this as the following problem:

Given: The set of frequent queries posed by an application on a spatial network, the pattern of variations of the spatial network with time.

Output: A model which supports efficient and correct algorithms for computing the query results.

Objective: Minimize the storage and computational cost of computation.

Constraints: (1) Edge travel times are positive integers. (2) Edge travel time preserves the FIFO (First-In First-Out) property.

Example: The figures 1(a),(b),(c) show a network at three instants of time. The travel times on the edges (the number shown on the edges) change with time. For example the edge N2-N1 has a travel time of 1 at the instant $t = 1$ and 5 at $t = 2$. It can be seen that the topology of the network is also time-dependent. Though the edge N2-N1 is present at $t = 1$ and at $t = 2$, it is absent at $t = 3$. The task is to develop a model that captures the network across time. The time-aggregated graph for this series of graphs is shown in Figure 1(d).

1.3 Related Work and Our Contribution

Time expanded networks have been widely used to model time dependency of parameters in networks [5,4,9]. This method duplicates the original network for each discrete time unit $t = 0, 1, \dots, T$ where T represents the extent of the time horizon. The expanded network has edges connecting a node and its copy at the next instant in addition to the edges in the original network, replicated for every time instant. Figure 2 shows an illustration of a time expanded graph. It significantly increases the network size and is very expensive with respect to

memory. Because of the increased problem size due to replication of the network, the computations become expensive.

Stochastic models which use probability distribution functions to describe travel time [4,8,7,3] have been used to study time-dependence of transportation networks. Though they can give valuable insights into the traffic flow analysis, the computational cost to compute the least expected travel times in these networks is prohibitively large to adapt to real life scenarios [8].

Ding [2] proposed a model that addresses the time-dependency by associating a temporal attribute to every edge and node of the network so that its state at any instant of time can be retrieved. This model performs path computations over a snapshot of the network. Since the network can change over the time taken to traverse these paths, this computation might not give realistic solutions.

Our Contribution: In this paper, we propose a model called time-aggregated graphs that represents the time dependence of the network and its parameters. We aggregate the travel times of each edge over the time instants into a time series and keep track of the edges that are present at every instant. We show that this model has less storage requirements than time expanded networks since it does not rely on replication of the entire network across time instants. We also propose algorithms for computing the connectivity and the shortest route from one node to another based on this model. We assume that the travel times of the edges vary with time in a predictable fashion.

1.4 Scope and Outline of the Paper

The main focus of the paper is our proposed use of time-aggregated graphs to represent spatial networks and account for the changes that can take place in the network over a period of time. The model requires that the spatial network be a discrete time dynamic network. The paper proposes algorithms to compute the connectivity and shortest paths in the time-varying network modeled as a time-aggregated graph.

The rest of the paper is organized as follows. Section 2 discusses the basic concepts of the proposed model and provides the relevant definitions of various terms used. Section 3 proposes algorithms for connectivity and shortest path computation based on this model. It also proposes the cost models for these algorithms. In section 4, we conclude and describe the direction of future work.

2 Basic Concepts

Traditionally graphs have been extensively used to model spatial networks [10]; weights assigned to nodes and edges are used to encode additional information. For example, the capacity of a transit station can be represented using the weight assigned to the node that represents the station and the travel time between two stations can be represented by the weight of the edge connecting the nodes. In a real world scenario, it is not uncommon for these network parameters to be time-dependent. This section discusses a graph based model that can capture

the time-dependence of network parameters. In addition, the model captures the possibility of edges and nodes being absent during certain instants of time.

2.1 A Conceptual Model

A graph $G = (N, E)$ consists of a finite set of nodes N and edges E between the nodes in N . If the pair of nodes that determine the edge is ordered, the graph is directed; if it is not, the graph is undirected. In most cases, additional information is attached to the nodes and the edges. In this section, we discuss how the time dependence of these edge/node parameters are handled in the proposed model, the time-aggregated graph.

We define the time-aggregated graph as follows.

$taG = (N, E, TF, f_1 \dots f_k, g_1 \dots g_l, w_1 \dots w_p | f_i : N \rightarrow \mathbb{R}^{TF}; g_i : E \rightarrow \mathbb{R}^{TF}; w_i : E \rightarrow \mathbb{R}^{TF})$ where

N is the set of nodes, E is the set of edges, TF is the length of the entire time interval, $f_1 \dots f_k$ are the mappings from nodes to the time-series associated with the nodes (for example, the time instants at which the node is present), $g_1 \dots g_l$ are mappings from edges to the time series associated with the edges and $w_1 \dots w_p$ indicate the time dependent weights on the edges.

Definition: Edge/Node Frequency. The number of time steps for which an edge e is present, denoted by $f_E(e)$, is called the frequency of edge e . The edge frequency of a time-aggregated graph f_E is defined as,

$$f_E = \max_{e \in E_{taG}} (f_E(e)).$$

In figure 1, $f_E(N1 - N2) = 2$ and f_E of the time-aggregated graph = 3.

Similarly, we can define $f_N(v)$ to be the number of time steps for which a node v is present and the degree of node presence of a time-aggregated graph is defined as $f_N = \max_{v \in N_{taG}} (f_N(v))$

The term 'frequency' used here does not imply periodicity in the edge/node time series.

We assume that each edge travel time has a positive minimum and the presence of an edge at time instant t is valid for the closed interval $[t, t + \sigma]$

Example: Figure 1 shows a time-aggregated graph and the network at each time step. Figures (a), (b) and (c) show the graphs at time instants 1, 2, 3. (d) shows the time-aggregated graph; each edge has a travel time series (enclosed in square brackets) and the edge time series associated with it. For example, the edge from node N1 to node N2 is present at time instants 1, 2 and disappears at the time instant $t = 3$. This is encoded in the time-aggregated graph using the edge time series of N1-N2 which is (1, 2); the travel times of this edge for all instants within the time interval under consideration are aggregated into a time series [1, 1, -]; the entry '-' indicates that the edge is absent at the time instant $t = 3$.

Figure 2 shows the time aggregated graph (corresponding to Figure 1(a), (b), (c)) and the time expanded graph that represent the same scenario. The time expansion for the example network needs to go through 7 steps since the latest time instant would end in the network is at $t = 7$. For example, the traversal

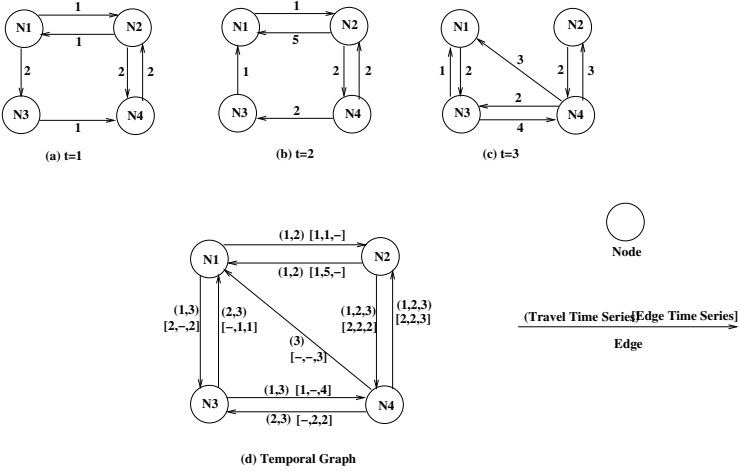


Fig. 1. Time-aggregated Graph, Network at various instants

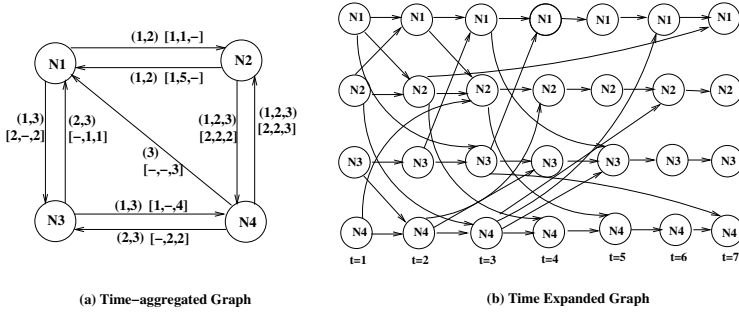


Fig. 2. Time-aggregated Graph vs. Time Expanded Graph

of the edge N3-N4 that starts at $t = 3$ ends at $t = 7$, the travel time of the edge being 4 units. The number of nodes is larger by a factor of T , where T is the number of time instants and the number of edges is also larger in number compared to the time-aggregated graph. Typically the value of T is very large in a spatial network such as a road map since the changes in the network are quite frequent. This would result in time expanded networks that are enormously large and would make the computations slow.

2.2 A Logical Data Model

Basic Graph Operations

We extend the logical data model described in [10] to incorporate the time dependence of the graph model. The three fundamental classes in the model

are **Graph**, **Node** and **Edge**. The common operations that are associated with each class are listed.

```
public class Graph {
public void add(Object label, timestamp t);
// node with the given label is added at the time instant t.

public void addEdge(Object n1, Object n2, Object label,
timestamp t, timestamp t_time)
// an edge is added with start node n1 and end node n2 at
// time instant t and travel time, t_time.

public Object delete(Object label, timestamp t)
// removes a node at time t and returns its label.

public Object deleteEdge(Object n1, Object n2, timestamp t)
// deletes the edge from node n1 to node n2 at t.

public Object get(Object label, timestamp t)
// returns the label of the node if it exists at time t.

public Iterator get_node_Presence_Series(Object n1)
// the presence series of node n1 is returned.

public Object getEdge(Object n1, Object n2, timestamp t)
// returns the edge from node n1 to node 2 at time instant t.

public Iterator get_edge_Presence_Series(Object n1, Object n2)
// the presence series of edge from node n1 to node n2
// is returned.

public Object get_a_Successor_node(Object label, timestamp t)
// an adjacent node of the vertex is returned if an edge exists
// to this node at a time instant at or after t.

public Iterator get_all_Successor_nodes(Object label, timestamp t)
// all adjacent nodes are returned if edges exist to them
// at time instants at or after t.

public Object get_an_earliest_Successor_node(Object label,timestamp t)
// the adjacent node which is connected to the given node with
// the earliest time stamp after t is returned.

public timestamp get_node_earliest_Presence(Object n1,
timestamp t)
// the earliest time stamp after t at which the node n1
// is available is returned.

public timestamp get_node_Presence_after_t(Object n1,
timestamp t)
```

```
// Part of the presence time series of node n1 after time t
// is returned.

public timestamp get_edge_earliest_Presence(Object n1, Object n2,
                                           timestamp t)
// the earliest time stamp after t at which the edge from
// node n1 to node n2 is available is returned.

public timestamp get_edge_Presence_after_t(Object n1, Object n2,
                                           timestamp t)
// Part of the presence time series of edge(n1-n2) after time t
// is returned.

}
```

A few important operations associated with the classes **Nodes** and **Edges** are provided below.

```
public class Node {
public Node(Object label, timestamp t)
// the constructor for the class. A node with the appropriate
// label is created at the time t.

public Object label()
// returns the label associated with the node if it exists at t.
}

public class Edge {
public Edge(Object n1, Object n2, Object label,
timestamp t_inst, timestamp t)
// the constructor for the class. an edge is added with start
// node n1 and end node n2 at time instant t and
// travel time, t_time.

public Object start()
// returns the start node of the edge.

public Object end()
// returns the end node of the edge.
}
```

The table 1 shows the difference in the behavior of the logical operators when the temporal dimension is removed.

We also define two predicates on the time-aggregated graph.

exists_at_time.t: This predicate checks whether the entity exists at the start time instant t .

exists_after_time.t: This predicate checks whether the entity exists at a time instant after t .

Table 1. Logical operators with and without 'time' dimension

Operator	with Time	without Time
delete	deletes the node for the specified time instant	deletes the node for the entire time period
deleteEdge	deletes the edge for the specified time instant	deletes the edge for the entire time period
get	get(node,time)	get_node_Presence_series(node)
getEdge	getEdge(node1,node2,time)	get_edge_Presence_series(node)
get_node_Presence	get_node_earliest_Presence(node, time)	get_node_Presence_series(node)
get_edge_Presence	get_edge_earliest_Presence(node1, node2,time)	get_edge_Presence_series(node, node2)

These predicates are used in conjunction with the entities node, edge and route in a graph to extend the classical graph theory concepts of adjacency, route and connectivity to time-aggregated graphs. Table 2 illustrates these concepts. For example, node v is adjacent to node u at any time t if and only if the edge (u, v) exists at time t as shown in the table. Similarly, a valid route exists from node u to node v if a path exists from node u to node v for start time t at node u and in accordance with the presence of edges along the route.

Table 2. Adjacency, Route and Connectivity in Time-aggregated Graphs

	exists_at_time_t	exists_after_time_t
Node	exists(node u,at_time_t)	exists(node u,after_time_t)
Edge	adjacent(node u,node v, at_time_t)	adjacent(node u,node v, after_time_t)
Route	route(node u,node v,a_route r at_time_t)	route(node u,node v,a_route r, after_time_t)

2.3 Physical Data Model

The adjacency-list and the adjacency-matrix representations are the most common main-memory data structures used in the implementation of graphs. To implement our model, the time-aggregated graphs, a modified version of adjacency list representation is used. This data structure uses an array of pointers, one pointer for each node. The pointer for each node points to a list of immediate neighbors in the time-aggregated graph. At each neighbor node, the edge presence series and travel times for the edge starting from the first node to this neighbor are also stored.

Comparison of Storage Costs with Time Expanded Networks

According to the analysis in [12], the memory requirement for time expanded network is $O(nT) + O(n + mT)$, where n is the number of nodes and m is the

number of edges in the original graph. The memory requirement for the time-aggregated graphs would be $O(m.f_E + n.f_N) + O(n + m)T$, where f_E is the edge frequency and f_N is the node frequency of the time-aggregated graph. This can be simplified to $O(m + n)T$ since T is always greater than f_E and f_N . This comparison shows that the memory usage of time-aggregated graphs is less than time expanded graphs by a factor of $O(nT)$.

The algorithms for connectivity and shortest path computation in time-aggregated network will be discussed in Section 3.

3 Algorithms for Network Computations

The critical step in most queries on a spatio-temporal network is the shortest path computation. Thus the proposed model must include an efficient shortest path algorithm. Here, the shortest path is the route that can be traversed in the shortest time, given the start time at the start node. We assume that there is no cost for waiting at the nodes other than the wait time, and that edge presence is closed for $[t, t + \sigma]$ where σ is the travel time of the edge at the time instant t .

3.1 Algorithm for Shortest Path Computation

As noted earlier, any route computation in a spatio-temporal network must be consistent with the edge presence. Here, the application of a greedy strategy (which is a popular choice in most of the optimization problems) faces a challenge. Not all shortest paths display the optimal sub-structure, which is an essential condition for greedy algorithms to generate an optimal solution. This is clearly illustrated in Figure 3. Although it can be verified that the route, $s - N1 - N2 - N4 - u - d$ is an optimal path from s to d , the route does not display optimal sub-structure since the route from s to u following the above path is not optimal (shortest path being, $s - N1 - N3 - N5 - u$). Though such paths that do not display optimal sub-structure could exist, it can be proved that there is at least one optimal path which satisfies the optimal sub-structure property.

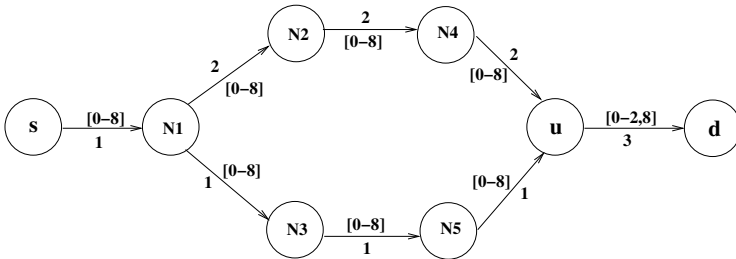


Fig. 3. Illustration of Shortest Paths

Lemma 1. If there is an optimal route from s to d , then there is at least one optimal route from s to d that shows optimal sub-structure.

Proof. As Figure 3 illustrates, the failure of optimal structure of the shortest path occurs due to a potential wait at the intermediate node (u), after reaching this node traversing the optimal path from s to u . Consider the optimal path from s to u . Append this path to the path $u-d$ (allowing wait at the intermediate node u) from the optimal path. This would be still the shortest path from s to d . Otherwise, it would contradict the optimality of the original shortest path.

Lemma 1 enables us to use a greedy approach to compute the shortest path. The algorithm stores the current shortest path travel times to reach every node from the source node. It closes the node with the minimum travel time and updates the travel times of its neighbors. At this step, it uses the operator *get_edge_earliest_Presence* to find the earliest time instant after the arrival at which this edge can be traversed. The updated travel time thus checks for the edge presence and adds the wait time to the travel time, if necessary. The algorithm is similar to Dijkstra's shortest path algorithm, the key difference being the step that looks for the earliest availability of the edges to the adjacent nodes.

Computational Complexity: The cost model analysis assumes an adjacency list representation of the graph with two significant modifications. The edge time series is stored in the sorted order. Attached to every adjacent node in the linked list are the edge time series and the travel time series.

For every node extracted from the priority queue Q , there is one edge time series look up and a priority queue update for each of its adjacent nodes. The time complexity of this step is $O(\log f_E + \log n)$. The asymptotic complexity of the algorithm would be

$$O(\sum_{v \in N} [degree(v) \cdot (\log f_E + \log n)]) = O(m(\log f_E + \log n)).$$

The time complexity of the shortest path algorithm based on a time expanded network is $O(nT \log T + mT)$ [1]. It can be seen that the algorithm based on a time-aggregated graph is faster if $\log n < T \log T$.

3.2 Algorithm for Connectivity

The existence of a valid route from one node to another in a time-aggregated graph is a non-trivial issue since a path in the time-aggregated graph does not always guarantee the existence of a path that is consistent with the edge time series and edge travel times. Figure 4 illustrates this; the node N2 is connected to node N4 for starting time instants 1, 2, 3, 4 (one route being N2 - N5 - N4), and N4 is not accessible from N2 for all time instants after $T = 4$.

Computational Complexity: The cost model analysis assumes an adjacency list representation of the graph with two significant modifications. For every node, the node presence time series is stored in the sorted order. Attached to every adjacent node in the linked list are the edge presence series and the travel time series.

Algorithm 1. Computation of Shortest Path

Input:

- 1) $G(N, E)$: a graph G with a set of nodes N and a set of edges E ;
 define type p positive integer
 Each node $n \in N$ has two properties:
 NodePresenceTimeSeries : series of p
 Each edge $e \in E$ has two properties:
 EdgePresenceTimeSeries, TravelTime(e)series : series of p
 $\sigma_{u,v}(t)$ is the travel time of edge (u, v) at time t
- 2) s : Source node, $s \subseteq N_G$;
- 3) d : Destination node, $d \subseteq N_G$;

Output: Shortest Route from s to d **Method:**

```

     $c[s] = 0; \forall v(\neq s), c[v] = \infty$ ;
    Insert  $s$  in priority queue  $Q$ .
  while  $Q$  is not empty do {
     $u = \text{extract\_min}(Q)$ ;
     $v = \text{get\_all\_Successor\_nodes}(u, c[u])$ 
     $t = \text{get\_earliest\_Presence}(u, v, c[u])$ ;
    if  $t + \sigma_{u,v}(t) < c[v]$  {
      update  $c[v]$ .
       $\text{parent}[v] = u$ ;
      if  $v$  is not in  $Q$ 
        insert  $v$  in  $Q$ ;
    }
    update  $Q$ ;
  }
  Output the route from  $s$  to  $d$ .

```

For each node dequeued from the queue Q , there is one edge series look up an enqueue operation for each of its adjacent node. The queue operations are $O(1)$ operations. The time complexity of this step is $O(\log f_E)$. The asymptotic complexity of the algorithm would be $O(\sum_{v \in N} [\text{degree}(v) \cdot (\log f_E)]) = O(m(\log f_E))$.

The time dependency of the network parameters affects the connectivity and the shortest paths between nodes in the network. Figure 5 depicts the connectivity and shortest path travel times for different start time instants at the source node for the example network shown in figure 4. Figure 5(a) illustrates the connectivity of the node N2 to node N4 at instants 1, 2, 3, 4, 5, 6 (these time instants denote the starting times at the node N2). It can be seen that valid routes exist from node N2 to node N4 if the traversal starts at time instants 1, 2, 3, 4 and that the node N4 is unreachable from N2 for time instants 5, 6. It might also be interesting to note that the routes that connect the nodes also change with time. For example, at time instant 1, routes N2-N3-N4, N2-N5-N4 and N2-N3-N5-N4 connect N2 to node N4; at starting time, $t = 4$, only N2-N5-N4 is available.

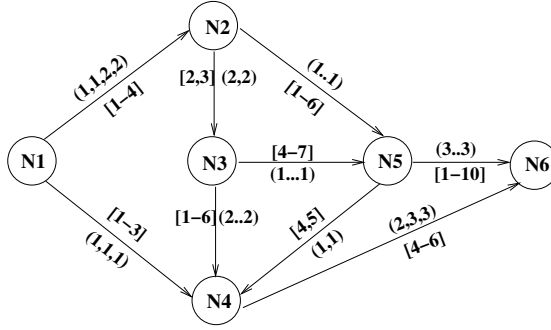


Fig. 4. Illustration of Connectivity

Algorithm 2. Connectivity Algorithm**Input:**

- 1) $G(N, E)$: a graph G with a set of nodes N and a set of edges E ;
define type p positive integer
Each node $n \in N$ has two properties:
 NodePresenceTimeSeries : series of p
Each edge $e \in E$ has two properties:
 EdgePresenceTimeSeries, TravelTime(e)series : series of p
 $\sigma_{u,v}(t)$ is the travel time of edge (u,v) at time t
- 2) s : Source node, $s \subseteq N_G$;
- 3) d : Destination node, $d \subseteq N_G$;

Output: A route from s to d , if exists; else returns FALSE.**Method:**

```

Initialization ;
    Add  $s$  to  $Q$ ;  $found = false$ ;
    for each node  $v \in N_G$  do {
         $arr\_time[v] = 0$ ;
    }
while  $found = FALSE$  or  $Q$  not empty do {
     $u = dequeue(Q)$ ;
     $v = get\_all\_Successor\_nodes(u, arr\_time[v])$ 
    Add  $v$  to the  $Q$ ;
     $t = get\_earliest\_presence(u, v, c[u])$ ;
    if  $t \neq \infty$  {
         $arr\_time[v] = t + \sigma_{u,v}(t)$ 
         $parent[v] = u$ 
    }
    if  $v = d$ ,  $FOUND = TRUE$ ;
}
}

```

Output the route from s to d .

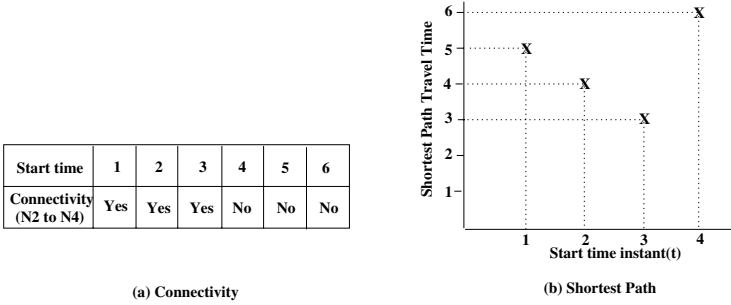


Fig. 5. Time Dependency of Connectivity and Shortest Paths

As shown in figure 5(b), the shortest path routes and the travel times are also dependent on time. Consider the shortest path from node N1 to node N6. The shortest path from node N1 to N6, for starting time $t = 1$ is N1-N4-N6 and travel takes 5 units of time (reaches the destination node at $t = 6$). The route remains the same for start times $t = 2, 3$, but the travel time changes to 4 units and 3 units respectively. At time $t = 4$, the route N1-N4-N6 is no longer available and the shortest route changes to N1-N2-N5-N6 with a total travel time of 6 units. This shows that the shortest paths in a time-dependent network vary with time.

4 Conclusions and Future Work

In this paper, we proposed a new model, time-aggregated graphs to model spatio-temporal networks which accounts for the changes in the network topology and parameters with time. Existing approaches rely on time-expanded networks which lead to high storage overhead and computationally expensive algorithms. Time-aggregated graph which models the time dependence using an aggregation of network parameters across the time horizon without the need to replicate the entire graph. We provided algorithms to compute connectivity and fastest route in the network, two frequent types of queries posed on a spatio-temporal network. Our analysis show that this model is less memory expensive compared to time expanded networks and leads to computationally efficient algorithms.

We plan to implement the work and test its performance on road networks. We are working towards extending the model to incorporate turn restrictions which are mostly time dependent and can significantly influence the fastest route computation. We understand that the model should accomodate time-varying capacities of the road networks while performing path computations, especially in applications like evacuation planning where capacity constraints in the network are the key challenge. Though the presence/absence of nodes has not been exploited in the algorithms presented in this paper, we expect that this feature would be used in applications like evacuation planning where nodes might

become unavailable for certain periods of time due to capacity constraints. Also, we need to formulate an algorithm to compute the fastest path in the network over the entire time period or for a user-defined time interval.

Acknowledgment

We are particularly grateful to the members of the Spatial Database Research Group at the University of Minnesota for their helpful comments and valuable discussions. We would also like to express our thanks to Kim Koffolt for improving the readability of this paper.

This work was supported by the NSF/SEI grant 0431141, Oak Ridge National Laboratory grant and US Army Corps of Engineers (Topographic Engineering Center) grant. The content does not necessarily reflect the position or policy of the government and no official endorsement should be inferred.

References

1. B.C. Dean. Algorithms for minimum-cost paths in time-dependent networks. *Networks*, 44, August 2004.
2. Z. Ding and R.H. Gutting. Modeling temporally variable transportation networks. *Proc. 16th Intl. Conf. on Database Systems for Advanced Applications*, pages 154–168, 2004.
3. R. Hall. The fastest path through a network with random time-dependent travel times. *Transportation Science*, 20:182–188, 1986.
4. D.E. Kaufman and R.L. Smith. Fastest paths in time-dependent networks for intelligent vehicle highway systems applications. *IVHS Journal*, 1(1):1–11, 1993.
5. E. Kohler, K. Langtau, and Skutella M. Time-expanded graphs for flow-dependent transit times. *Proc. 10th Annual European Symposium on Algorithms*, pages 599–611, 2002.
6. Q. Lu, B. George, and S. Shekhar. Capacity Constrained Routing Algorithms for Evacuation Planning: A Summary of Results. *Proc. of 9th International Symposium on Spatial and Temporal Databases (SSTD'05)*, August 2005.
7. E. Miller-Hooks and H.S. Mahmassani. Least possible time paths in stochastic time-varying networks. *Computers and Operations Research*, 25(12):1107–1125, 1998.
8. E. Miller-Hooks and H.S. Mahmassani. Path comparisons for a priori and time-adaptive decisions in stochastic, time-varying networks. *European Journal of Operational Research*, 146:67–82, 2003.
9. S. Pallottino and M. G. Scutella. Shortest path algorithms in transportation models: Classical and innovative aspects. *Equilibrium and Advanced transportation Modelling (Kluwer)*, pages 245–281, 1998.
10. Shekhar S. and Chawla S. *Spatial Databases: Tour*. Prentice Hall, 2003.
11. Shekhar S., Chawla S., R Vatsavai, X. Ma, and Yoo J.S. *Location Based Services, Editors:Schiller, J. and Voisard, A. Morgan Kaufmann*, 2004.
12. D. Sawitzki. Implicit Maximization of Flows over Time . *Technical report, University of Dortmund*, 2004.

An ISO TC 211 Conformant Approach to Model Spatial Integrity Constraints in the Conceptual Design of Geographical Databases

Alberto Belussi¹, Mauro Negri², and Giuseppe Pelagatti²

¹ Dipartimento di Informatica, University of Verona, Strada le Grazie, 15,
37100 Verona, Italy
alberto.belussi@univr.it

² Dipartimento di Elettronica e Informazione, Politecnico di Milano, Via Ponzio, 34/5,
20100 Milano, Italy
{mauro.negri, giuseppe.pelagatti}@polimi.it

Abstract. The ISO TC 211 standards have defined a set of formal models for the conceptual modeling of spatial data using the Unified Modeling Language (UML) and the geometry approach adopted by the ISO spatial data model and by the Geographic Mark-up Language (GML). This approach aims to define a conceptual model for the design of geographic databases and for the geospatial interoperability of heterogeneous spatial databases. The ISO standards are however complex and counterintuitive in dealing with spatial integrity constraints, which are fundamental for the expressiveness of a conceptual model in the geographic application domain. This paper improves the ISO approach by proposing a framework which allows the definition of powerful, easy to use, and ISO conformant modeling abstractions for topological spatial constraints. These modeling abstractions have been incorporated in the definition of the GeoUML conceptual model used in the Italian IntesaGIS project for the definition of the “core” database of the Italian Spatial Data Infrastructure.

1 Introduction

Several conceptual models or design patterns approaches have been proposed to deal with the design of geographical data for spatial databases [1], [3], [4], [5], [6], [10], [15], [16], [17]. In general, these approaches enrich well known data models for traditional databases with specific data types for the management of space, time and spatial relationships, but most of them do not deal enough with the problem of expressing spatial integrity constraints, although integrity constraints are recognized as an important issue in spatial database design [7], [9]. Spatial integrity constraints are often described through additional constraint formulas, written in some constraint language and attached to the elements of the conceptual model. In this way a given spatial constraint can be formulated through different formal expressions and this is particularly ineffective when the constraint represents common situations of the modeled world, since it becomes hard to recognize the constraint. Only few approaches support

specific kinds of spatial constraints through explicit modeling abstractions [3], [4], [5], [10], [16], [17]. Moreover, all the above listed approaches do not consider the ISO TC211 approach and therefore they are not conformant to these standards.

The ISO TC211 provides the 19100 series of standards for the specification of several aspects of geographic data and applications. In particular, the specification of a vector based geometry data model is given in the standard 19107 (called “Spatial Schema “ in the sequel) [12] and an object oriented conceptual model for the design of spatial databases using the Spatial Schema as geometry model is given in the standard 19109 (called “Rules for Applications” in the sequel) [13]. Notice that, these standards also aim to support the interoperability among heterogeneous and autonomous spatial databases. These standards deserve consideration since they have been adopted by the OpenGeospatial consortium and restricted profiles of them [14] are already supported by available GIS and database technologies. These ISO standards however supply only a weak support to the modeling of spatial integrity constraints, since the ISO approach requires explicit expressions written in OCL, providing only little structural support (e.g., the Complex and the Contains associations of [12]). Therefore, also the ISO approach suffers of the above limitations, as discussed in [3].

This paper proposes an approach to overcome the complexity and the difficulties the management of spatial integrity constraints through constraint languages. It allows the definition of ISO conformant modeling abstractions for the constraints specification, which are useful for database designers. The modeling abstractions are then formalized in terms of OCL constraint formulas or topological functions attached to the conceptual elements of the ISO standards. This approach has been used as the basis for supporting spatial integrity constraints in the object oriented “GeoUML” conceptual data model [2], developed in the context of the Italian IntesaGIS project [11]. The IntesaGIS project has been launched with the aim of defining the general structure and the content of a “core” geographic database in Italy according to the INSPIRE directive. The definition of a complete and minimal set of modeling abstractions and the design of graphical symbols for these abstractions is out of the scope of the paper.

The paper is organized as follows: Section 2 provides some specializations of the spatial data types of the ISO spatial data model which are used for the formal and correct definition of the constraints. Section 3 and 4 present two classes of spatial integrity constraints: the *Structural Constraints* defined on spatial objects with a common geometry structure and the *Topological Constraints* based on a reference set of topological relations. Section 5 outlines conclusions and future work.

2 Specializations of the Spatial Schema

The Spatial Schema defines a set of UML classes for the representation and manipulation of geometrical data; the objects of these classes are embedded in a coordinate space of up to 3 dimensions. The classes are organized in an inheritance hierarchy as shown in Figure 1; this figure is simplified with respect to the hierarchy of the Spatial Schema, since in this paper we describe only those features of the Spatial Schema which are relevant for the GeoUML model.

A *GM_Primitive* object represents a simple geometry like a point, a curve or a surface, as shown by the *GM_Point*, *GM_Curve* and *GM_Surface* subclasses of *GM_Primitive*. A *GM_Complex* object is a collection of primitives satisfying the following properties: (1) primitives do not overlap, and (2) the boundary of every primitive of the complex consists of a set of other primitives which are also contained in the complex. The concept of *complex*, (i.e. an object of class *GM_Complex*) is fundamental for the GeoUML model. The *GM_Aggregate* is a collection of objects of all the classes without any constraint on the members of the collection.

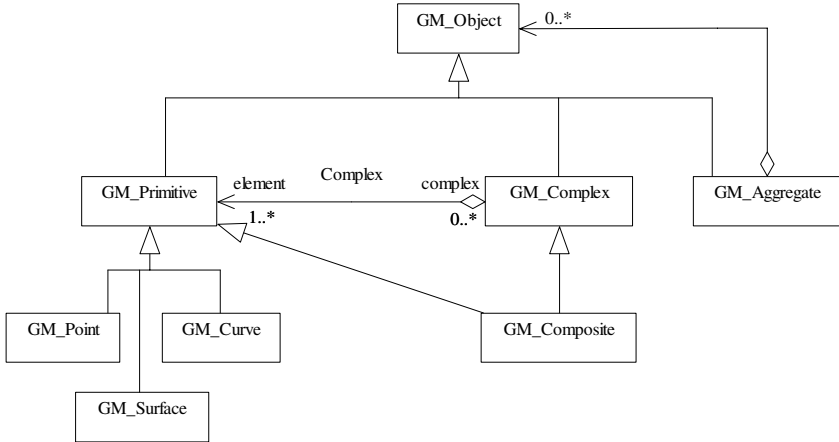


Fig. 1. The main classes of the ISO spatial schema

The *GM_Composite* class, a subclass of *GM_Complex*, has 2 subclasses *GM_CompositeCurve* and *GM_CompositeSurface*, which contain complexes which are homeomorphic to a primitive; for example a *CompositeCurve* is a *Complex* (collection of point and curve primitives), but it is isomorphic to a curve (the connected sequence of curves has exactly two endpoints as a primitive curve). For this reason a *GM_Composite* inherits from both the *GM_Complex* and the *GM_Primitive* classes.

In GeoUML the classes for representing geometric attributes are subclasses of the *GM_Complex* class; this approach has been chosen for the following reasons:

- it permits to represent explicitly the topological structure of the geometric value;
- it allows a geometric value to share geometry (in terms of primitives) with other geometric values;
- it allows to represent topological layers, i.e. sets of geometric values with a structural representation of their topological relations;
- it allows objects to be split in update (e.g., for an intersection), since complexes can be split, while objects of class *GM_primitive* cannot.

In addition to the subclasses of *GM_Complex* provided by the Spatial Schema (the *GM_Composite* classes) in GeoUML it was necessary to define two new classes, called *GU_CXCurve* and *GU_CXSurface*, which are subclasses of *GM_Complex* that

are homogeneous in dimension without being homeomorphic to primitives; the main reason for defining these new classes was the following:

- on one hand, the IntesaGIS project required the definition of complex geographical objects which are not homeomorphic to primitives (e.g., a road path can be composed by more disconnected parts and therefore it cannot be a composite curve);
- on the other hand, defining these objects as generic (i.e., a collection of 0,1,2 dimensional primitives) complex objects has the strong drawback that the boundary is not defined, since ISO supports the topological notion of boundary only for primitives and composites, but not for generic complex objects.

Since the boundary operation is fundamental in expressing many of the integrity constraints of GeoUML, the 2 new classes have been introduced, and the boundary operation has been defined for them; this was possible thanks to the homogeneity in dimension, which allows a safe definition of boundary.

3 Structural Constraints on Complexes

In order to explain the necessity of this category of constraints, we have to analyze briefly the class *GM_Complex* and its properties. As shown in Figure 2, each complex can participate to the association *Contains* (super/subComplex). This implies that each complex C has two properties among others: the roles *superComplex* and *subComplex*, that return the set of superComplexes of C and the set of subComplexes of C respectively. The association *Contains* is not declared as derivable from the association *Complex*. This means that a complex C_1 can be composed of a subset of the primitives that compose another complex C_2 without being one of its subComplexes. However, if a complex C_1 is subComplex of a complex C_2 then $C_1.element \subseteq C_2.element$. Therefore, the association *Contains* can be seen as a relation based on primitives, but not implied by the subset relation on primitives. Moreover, given two complexes C_1 and C_2 and considering their representation as pointsets in the reference space $Pset(C_1)$ and $Pset(C_2)$, $Pset(C_1)$ can be a subset of $Pset(C_2)$, without being C_1 subComplex of C_2 . Viceversa the existence of a *Contains* association between two complexes always implies the containment on the corresponding pointsets.

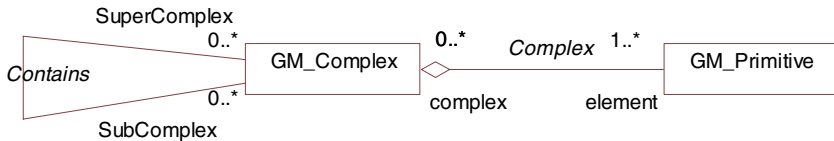


Fig. 2. The Main associations of the class *GM_Complex* of the Spatial Schema

Considering the design of a conceptual schema (according to the standard “Rules for Applications”), another issue has to be analyzed. A conceptual schema might contain two or more classes representing feature types having spatial attributes of type *GM_Complex* and there might be a conceptual relation between these two classes that we want to represent as a structural relation on geometry by using the *Contains*

association. For example an application schema could contain four classes: Road, Road Element, Railway and Railway Element, where the conceptual relations between Road and Road Element and between Railway and Railway Element are represented by the association Contains of GM_Complex. This schema accepts as possible states of the database the situation where a road element is not a subComplex of a road or where it is a subComplex of a railway or where it is not subComplex of any complex. This example shows that, in an application schema having many classes with spatial attributes of type GM_Complex, constraints on the association Contains are always required. Moreover, considering the implementation of the schema in a database system, it is useful to know which complexes may share common primitives, since it is necessary to indicate which maximal complexes we need to create.

These arguments lead us to the introduction of a category of constraints regarding the Contains association that we call *Structural Constraints*, with two objectives: a) the correct definition of the consistent states of the database and b) the explicit specification of the structural bindings among the classes having spatial attributes that belong to the same maximal complex. The basic structural constraints are of three categories [16]:

1. the structural constraint *BelongsTo*, which requires that objects of one class must belong to an object of another class;
2. the structural constraint *CoveredBy*, which requires that each object of one class must be covered by a set of objects of another class;
3. the structural constraint *UnionOf*, which requires that an object of one class coincides with the union of some objects of another class.

3.1 The Structural Constraint Belongsto

This structural constraint defines a subcomplex constraint on a geometric attribute g of each object belonging to a class X with respect to the geometric attribute f of at least one object belonging to a class Y . The constraint *BelongsTo* involves the closure (interior and boundary) of all the complex objects of both the classes considered in the constraint.

Table 1 shows the formal definition of the constraint in OCL; this kind of definitions is intended as a formal specification of GeoUML, but the user does not need to understand it, since the meaning of the constraint is rather intuitive.

Table 1. Formal definition of the structural constraint *BelongsTo*

<i>Structural Constraint BelongsTo</i>	<i>Definition of the constraint in OCL</i>
<i>BelongsTo</i> ($X, g: GM\_Complex, Y, f: GM\_Complex$)	Context X inv: $Y.allinstances \rightarrow exists(a: Y \mid self.g.supercomplex \rightarrow includes(a.f))$

Different variants of this constraint allow to refer to the boundary or to the closure of the involved complex geometric objects. The *BelongsTo*^{B-} constraint considers the boundary of the complex objects of the attribute $X.g$ and the closure of the objects of

the attribute $Y.f$. The constraint $BelongsTo^{-B}$ considers the closure of the complex objects of the attribute $X.g$ and the boundary of the objects of the attribute $Y.f$; of course, the constraints involving boundaries are admitted only for the geometric types supporting a precise boundary definition. Notice that, a constraint involving the boundaries of both the classes could be easily derived.

The basic constraint of Table 1 and its variants can be further specialized in order to support additional secondary properties as discussed in [17]; one of the more important additional properties is the disjointness required among the complex objects of the attribute $X.g$ that are subcomplexes of a same complex object of the attribute $Y.f$.

Table 2 shows how the constraint formulation of Table 1 changes when disjointness is required. Notice that the semantics of the topological relationships (e.g., the meaning of the disjoint notion with curves and surfaces) is strongly influenced by the topological dimension of the involved objects, therefore the disjointness property could require the definition of different abstractions. The disjointness considered in Table 2 regards spatial objects of the attribute $X.g$:

- which are of the same topological dimension and;
- having no common interior points, i.e. they can share only primitives of their boundary.

Table 2. Formal definition of the structural constraint $dj\text{-}BelongsTo$

<i>Structural Constraint</i> <i>dj-BelongsTo</i>	<i>Definition of the constraint in OCL</i>
$Dj\text{-}BelongsTo(X, g:GM\_Complex, Y, f: GM\_Complex)$	context X inv: $Y.allinstances \rightarrow exists(a: Y \mid$ $self.g.supercomplex \rightarrow includes(a.f)) \text{ and}$ $(X.allinstances \rightarrow forall(x: X \mid brother(x.g, self.g, Y, f) \text{ implies}$ $(DJ\text{-}str(x.g, self.g)))$ where: $brother(x, y, Z, c) \equiv Z.allinstances \rightarrow$ $exists(z: Z \mid x.supercomplex \rightarrow includes(z.c) \text{ and}$ $(y.supercomplex \rightarrow includes(z.c)))$ $DJ\text{-}str(x, y) \equiv x.element \rightarrow forall(z: GM\_Primitive \mid$ $not (y.element \rightarrow includes(z)) \text{ or}$ $(x.boundary().element \rightarrow includes(z) \text{ or}$ $(y.boundary().element \rightarrow includes(z)))$

Two variants $dj\text{-}BelongsTo^{B-}$ and $dj\text{-}BelongsTo^{-B}$ can be defined in the same way as for the $BelongsTo$ constraints; they can be applied only to boundary attributes $g(f)$ of type: $GU\_CXSurface2D$, $GU\_CXCurve$, and that the constraint $dj\text{-}BelongsTo^{BB}$ involving only the boundaries of both the attributes $g(f)$ is derivable as conjunction of the two constraints $dj\text{-}BelongsTo^{B-}$ and $dj\text{-}BelongsTo^{-B}$.

3.2 The Structural Constraints CoveredBy and UnionOf

The structural constraint *CoveredBy* (*UnionOf*) defines a composition constraint on a geometric attribute f of the objects belonging to a class Y . This constraint requires that f must be obtained by building the union of a subset of the primitives (or all the primitives in the case of *UnionOf*) of the geometric attributes g of one or more objects of the class X . Following the same approach applied to *BelongsTo*, other specializations of the basic structural constraint are defined in order to apply the constraint to the boundary of the involved complexes. Table 3 and 4 show the definitions of the structural constraints *CoveredBy* and *UnionOf* respectively. In the tables the constraint is referred to a constrained class Y containing a geometric attribute f and to a constraining class X with a geometric attribute g . The formal definition of *CoveredBy*($Y, f: GM\_Complex, X, g: GM\_Complex$) is based on the condition that each primitive p that is element of the complex f ($p \in self.f.element$) must be also element of the complex that represents the geometric attribute g of at least one object of X . It is important to notice that g could also not be a subComplex of f . This additional condition can be expressed by specifying a constraint of the category *BelongsTo*.

Also in this case two variants *CoveredBy*^B and *CoveredBy*^B can be defined; they can be applied only to geometric attributes $f(g)$ of type: *GU_CXSurface*, *GU_CXCurve* and the constraint *CoveredBy*^{BB} is derivable as conjunction of the two constraints *CoveredBy*^B e *CoveredBy*^B. Notice that the disjointness of components can also be applied to these abstractions (e.g., the *UnionOf* together the disjointness constraint introduced for the *BelongsTo* will generate the notion of *geometric partition*).

Table 3. *Formal definition* of the structural constraint *CoveredBy*

Structural Constraint <i>CoveredBy</i>	OCLE definition of the constraint
<i>CoveredBy</i> ($Y, f: GM\_Complex, X, g: GM\_Complex$)	context Y inv: $self.f.element \rightarrow forall(e: GM\_Primitive \mid X.allinstances \rightarrow exists(a: X \mid a.g.element \rightarrow includes(e)))$

Table 4. *Formal definition* of the structural constraint *UnionOf*

Structural Constraint <i>UnionOf</i>	OCLE definition of the constraint
<i>UnionOf</i> ($Y, f: GM\_Complex, X, g: GM\_Complex$)	context Y inv: $self.f.element = X.allinstances.g \rightarrow select(a: GM\_Complex \mid a.superComplex \rightarrow includes(self.f)) \rightarrow collect(element) \rightarrow asSet$

4 Topological Constraints

In a conceptual schema containing geographic data, it is often necessary to specify integrity constraints based on spatial properties that are based on topological relations among objects. Topological relations are all those spatial relations that are invariant

with respect to topological transformation or homeomorphism, i.e. “rubber sheet” transformations of the space (deformations obtained by bending, stretching, twisting, and the like, but not by tearing or cutting the space.). Topological relations are interesting in geographic applications since the natural perceptions of the content of a given map by the human being is based on these kinds of relations.

Also the structural constraints of Section 3 are based on topological relations, however they always require that the geometric objects involved in the constraints are sub-complexes of a common complex. This precondition is not always satisfied, moreover the structural constraints allow to express only a restricted subset of topological relations.

The topological constraints are defined by combining a logical structure with a topological relation. Two types of logical structure are considered: the existential (the most used) and the universal structures. An *existential topological constraint* requires that, given an object c belonging to the constrained class C , there exists an object c' of the constraining class C' such that a given topological relation between the two objects is satisfied. The universal logical structure replaces the existential quantification with a universal quantification.

The topological relations chosen for constraint specification in GeoUML are a specialization of those proposed in Clementini et al. [8], but a different set of topological relations could be used in the same manner.

Notice that some topological relations require to distinguish the interior from the boundary of an object and therefore they cannot be applied to objects of the class $GM\_Complex$, but only to objects of its specialization classes (e.g., GU_CXC_Curve).

The basic form of topological constraint can be varied, thus obtaining several, more expressive constraints, in the following way:

Given a constrained class X with a geometric attribute g , a constraining class Y with a geometric attribute f and a disjunction of topological relations $rel_1 | \dots | rel_n$, the following kinds of topological constraints can be defined in GeoUML:

- Basic existential topological constraint: it requires for each object x of X the existence of an object y of Y such that one of the relations $rel_1 | \dots | rel_n$ is satisfied between $x.g$ and $y.f$.
- Existential topological constraint with selections: this version allows to express selection conditions for X or Y or both. The condition regarding Y can contain attributes of Y and also attributes of X , while the condition on X can contain only attribute of X .
- Existential topological constraint on boundaries: this version allows to apply the topological relation to the boundary of the object instead of applying it directly to the object. In this way for example the containment of an object in the boundary of another object can be specified.
- Basic universal topological constraint: this constraint is more restrictive than the existential one, since it requires that the topological relation (or disjunction of topological relations) exists between the constrained object and all the objects of the constraining class (this constraint is meaningful only for some kinds of topological relations, for instance “disjoint” or “touch”).
- Existential topological constraint with union: this form takes the point-set union of the geometric attributes f of the objects belonging to the constraining class Y . This means that for each x of the constrained class X , the topological relation is tested between $x.g$ and the geometric object obtained by building the union of the objects $y.f$, y being an object of Y .

These topological constraints can be combined together in order to obtain more complex constraints, with the exclusion of the combination between universal and existential versions.

Table 5 contains the formal definition in OCL of the basic existential topological constraint; similar definitions for the other kinds of constraints can be found in [2]. In order to simplify the formulation of the OCL expressions the disjunctions of topological relations are represented by a general function: *check* (*g*: *GM_Object*, *R*: *REL_{topo}*, *f*: *GM_Object*): *Boolean*. In [2] the translation of this function in expressions on the Spatial Schema classes is given.

Table 5. Formal definition of the basic existential constraint

<i>Existential topological constraint</i>	<i>Formal definition of the OCL constraint</i>
<u>Basic existential constraint:</u> $Topo \exists X, g, \{rel_1, ..., rel_k\}, Y, f)$	context <i>X</i> inv: <i>Y.allinstances</i> $\rightarrow exists(a: Y \mid check(self.g, \{rel_1, ..., rel_k\}, a.f))$

5 Conclusions and Future Work

The paper has proposed a framework for the definition of easy to use and ISO compliant modelling abstractions for modeling common spatial integrity constraints with a same modeling pattern. The effectiveness of the proposed approach has been successfully proved in the conceptual design of the core SDI in the Italian IntesaGIS project. Future researches have to address the definition of a taxonomy of the required spatial integrity constraints for spatial database design and of a minimal and complete set of modeling abstractions covering the constraints classified in the taxonomy.

References

1. Bédard, Y.: Visual Modelling of Spatial Databases: towards spatial PVL and UML, GEOMATICA, Vol. 53 (1999) 169-186
2. Belussi, A., Negri, M., Pelagatti, G.: GeoUML: an ISO TC 211 compatible data model for the conceptual design of geographical databases. D.E.I. - Politecnico di Milano, n.21, (2004)
3. Belussi, A., Negri, M., Pelagatti, G.: Modelling Spatial Whole-Part Relationships using an ISO-TC211 conformant approach, will appear in Information and Software Technology (2006)
4. Borges, K.A.V., Laender, A.H.F., Davis, C.A.: Spatial Data Integrity Constraints in Object Oriented Geographic Data Modeling, 7th ACM-GIS (1999) 1-6
5. Borges, K.A.V., Davis, C.A. Laender, A.H.F.: OMT-G: An Object Oriented Data Model for Geographic Applications, GeoInformatica, Vol. 5 (2001) 221-260
6. Brodeur, J., Bédard, Y., Proulx, M.J.: Modelling geospatial application databases using UML based repositories aligned with international standards in geomatics, 8th ACM-GIS (2000) 39-46

7. Christensen, A.F., Tryfona, N., Jensen, C.S.: Requirements and Research Issues in Geographic Data modeling, ACM Int. Symp. on Advances in Geographic Information Systems, Atlanta, USA (2001) 2-8
8. Clementini, E., Di Felice, P., Van Oosterom, P.: A Small Set of Formal Topological Relationships Suitable for End-User Interaction. Advances in Spatial Databases, 3rd Int. Symp., SSD'93, Singapore, Lecture Notes in Computer Science, Vol. 692, (1993) 277-295
9. Cockcroft, S.: The design and Implementation of as repository for the management of spatial data integrity constraints, GeoInformatica, Vol. 8 (2004) 49-69
10. Hadzilacos, T., Tryfona, N.: An Extended Entity-Relationship Model for Geographic Applications, SIGMOD Record, Vol. 26 (1997) 24-29
11. http://www.intesagis.it/specifiche/Doc_wg01/1n1007_4.pdf
12. ISO/TC 211 Geographic information/Geomatics, 19107 Geographic information - Spatial schema, text for FDIS, N. 1324 (2002)
13. ISO/TC 211, Geographic information/Geomatics, 19109, Geographic Information - Rules for application schema, text for FDIS, N. 1538 (2003)
14. ISO/TC 211, Geographic information/Geomatics, 19125, Geographic Information - Simple feature access, text for DIS, N. 1003 (2000)
15. Parent, C., Spaccapietra, S., Zimanyi, E., Donini, P., Plazanet, C., Vangenot, C.: Modeling spatial data in the MADS conceptual model, 8th Int. Symp.on Spatial Data Handling, Vancouver, Canada (1998) 138-150
16. Price, R., Tryfona, N., Jensen, C.S.: Modeling Part-Whole Relationships for Spatial data, 8th ACM GIS, Washington D.C., USA (2000) 1-8
17. Price, R., Tryfona, N., Jensen, C.S.:Modelling Topological Constraints in Spatial Part-Whole Relationships, ER 2001, Lecture Notes in Computer Science, Vol.LNCS N. 2224 (2001) 27-40.

Access Control in Geographic Databases

Liliana Kasumi Sasaoka¹ and Claudia Bauzer Medeiros²

¹ IBM Silicon Valley Lab
555 Bailey Ave, San Jose, CA 95141, USA
lilianas@us.ibm.com

² Institute of Computing, UNICAMP
13081-970 Campinas, SP Brazil
cmbm@ic.unicamp.br

Abstract. The problem of access control in databases consists of determining when (and if) users or applications can access stored data, and what kind of access they are allowed. This paper discusses this problem for geographic databases, where constraints imposed on access control management must consider the spatial location context. The model and solution provided are motivated by problems found in AM/FM applications developed in the management of telephone infrastructure in Brazil, in a real life situation.

1 Introduction

Security and trust in databases are intimately associated with access control [AJS⁺96]. They determine *who* can access *what* data and *how*. In most cases, security models and mechanisms concentrate on low level system details, and do not consider semantics associated with the data. In particular, spatial applications present challenges not met by standard access control proposals.

Security issues are considered only at the implementation level, and not usually integrated into the modeling stage. Several access control models have been defined for relational or object-oriented databases. Specific models have also appeared – e.g., in the case of temporal [BJS95, BBF01] or video databases [BHA00]. However, none of these mechanisms can be directly applied to geographic applications, because of their particular characteristics. Indeed, when attribute semantics are associated with the spatial localization, data management demands distinct types of control, which has to be defined in terms of geographic region. In other words, access control becomes spatially sensitive.

Consider the following scenario, which will be used throughout the paper to motivate our solution. A utility (telephone) company wants to develop a GIS project that concerns infrastructure expansion in a city, for a specific geographic region R . Several engineers and experts will be concerned – they work cooperatively in the expansion planning for R , having distinct needs and authorizations for data access. At the same time, normal operations proceed (e.g., repairs and maintenance) and other people will have access to data on the same region, again with distinct permissions. Whereas standard access control proposals concern only thematic data, spatial access control involves issues such as “John

can only update data concerning the area within blocks A and B”, or “Repairs recorded for an area X will override any other operations being requested for this area”. It must furthermore be possible to grant access only for one spatial object (a pole), a set of objects (e.g., poles in a street), or a neighborhood.

A specific system that demands this kind of geographic access control is the Brazilian CPqD Outside Plant Management System, formerly known as the SAGRE System [Mag97]. It is an integrated set of GIS-based software applications to manage the expansion, modernization and operation of an outside telephone plant. Used throughout Brazil by major telephone companies, it has very large geographic databases for most of Brazil’s major cities, and hundreds of thousands of lines of code.

SAGRE has been in operation and continuous evolution since the beginning of the nineties. It is used in several sectors of telecom companies, by people with different roles. This gave rise to the need to control access to the operations that use its database taking spatial information into account.

Our paper shows how to solve this problem by extending classical models and mechanisms to the spatial context. Though our solution is general, it was motivated by the needs of the CPqD Outside Plant Management System.

The rest of this paper is organized as follows. Section 2 introduces related work. Sections 3, 4 and 5 describe our model and access control mechanism. Section 6 presents the access control problems in SAGRE and discusses the use of the proposed mechanism in this context. Finally, section 7 presents conclusions and possible extensions.

2 Basic Concepts and Related Work

2.1 Authorization Models

All access control mechanisms are based on some authorization model, which defines how a database management system must implement access control. It is generally composed by: (i) access granularity indication; (ii) structures to represent the authorization (formal semantics of representation); (iii) a set of policies to manage and to grant authorizations; and (iv) algorithms to analyze access requests based on the existing authorizations.

Access *granularity* defines the storage unit to control data access – e.g., at the tuple, tables or databases levels. The most common *authorization structure* is represented by the triple $\langle s, o, m \rangle$, where: s is the subject who receives the authorization, o the object which is authorized and m the access mode.

Objects o are the passive entities storing information, such as tables, tuples, or even elements of a tuple. Subjects are active entities that access the objects and can be users, user groups or processes operating on behalf of users. The subject can also be defined in terms of roles.

The m in $\langle s, o, m \rangle$ corresponds to the access mode – i.e., the type of operation that the subject has permission to execute on the object. [BDPSN96] defined the basic set of operations as: read, write, delete, execute and create. Authorizations can be further refined into positive or negative (forbidden).

The set of *policies* to manage authorizations are rules that define: who will grant and revoke permissions (e.g., owner, administrator, any user), operations authorized (e.g., read, write), and how these will be executed. Policies also define factors such as negative authorizations and authorization derivation.

Finally, in order to have a complete authorization model, one must also define *mechanisms* or *algorithms* to validate an access request based on the stored authorizations. As will be seen, the mechanism we propose specifies all the required model components: granularity, structure, policies and algorithms.

2.2 Access Control Mechanisms

Current research efforts on access control can be classified in three main directions [BDPSN96]: Discretionary Access Control (DAC), Mandatory Access Control (MAC) and the combination of both, the Role Based Access Control (RBAC). Efforts normally are defined in terms of the $\langle s, o, m \rangle$ structure.

DAC is based on granting and revoking privileges [GW76]. Discretionary protection policies govern the access of users (the subjects) to the information, on the basis of the users identity and the rules that specify, for any user and any object in the system, the types of accesses allowed. A subject's request to access an object is checked against the specified authorizations; if there exists an authorization stating that the subject can access the object in the specific mode, the access is granted; otherwise, it is denied. Policies are discretionary: they allow subjects to grant other subjects authorizations to access the objects.

MAC is based on classifying subjects and objects of the system in hierarchical levels, satisfying the requirements of military, governmental and commercial organizations [BJS95]. This hierarchical organization assures that classified information does not flow to lower levels. It is based on two principles formulated by Bell and LaPadula [BP76]. The first states that no subject can read an object of an upper level. The second does not allow a subject to write in an object of a lower level, ensuring that no information will flow from upper to lower levels.

Access decisions on the Role Based Access Control (RBAC) [FK92] are based in the roles that a user can perform inside an organization. This adds flexibility to access grants, which become context-sensitive.

New devices and applications have given rise to other kinds of concerns. The Web has motivated research on adaptations of RBAC to this new environment (e.g. [PSA01]), and studies on distinct granularity levels for protection of XML documents [BCFM00]. The field of sensor networks has prompted studies on coordination and fusion of sensor data, and protocols for access control to save energy (e.g., [WHE04]).

Few authors are concerned with the special needs of spatial access control. The work of [BBC⁺04] proposes a discretionary model that considers, among others, derivation of authorization rules, privilege propagation and negative authorizations over vector data. This work is extended to a model called GEO-RBAC, which considers RBAC in the spatial context [BCDP05]. This model is motivated by the needs of location-based services and mobile applications. It provides flexibility in access specification, associating roles with a spatial context and changing

authorizations according to spatial granularity. Roles are instances of a role schema; authorizations can be globally assigned to all roles in a schema, or be refined for a specific role. Roles are “activated” according to a subject’s location.

As will be seen, the main differences between these two proposals and our model are the fact that we were motivated by the needs of cooperative work in spatial applications, for a very large real GIS application. As a consequence, some aspects of our solution are concerned with simplifications for performance reasons, and specific user needs. Roles are defined by user groups.

3 Authorization Model for Geographic Data

This section presents the main components of our model: granularity, subject, object, access mode, adopted authorization rules, policies and algorithms.

Definition - Spatial authorization rule. A spatial authorization rule is defined by the triple $\langle s, o, m \rangle$, where s is the authorized subject; o the set of authorized objects and m , the access mode. The object o can be represented by identifiers (explicit enumeration) or by a spatial query (implicit specification). Queries are discussed in section 5. The access mode can be read or write.

3.1 Stating and Storing an Authorization Rule: s , o , m

We assume that all spatial data are stored in a spatial database, accessed by a GIS. Moreover, this database also contains a special repository with the authorization rules (referred to as “rule database”), which specify spatially-dependent access control. We use a simplified spatial data model, based on OGC’s, which is sufficient for the purposes of our explanation. We consider that data in geographic databases can be characterized as having two types of attributes: descriptive and spatial features. This research is limited to vector data, geometries being classified into three types: point (e. g., a pole), line (e. g., a street), or polygon (e. g., a parcel).

From a high abstraction level, an authorization process can be understood as being defined according to the following sequence of stages: (1) definition of authorization rules, (2) mapping of these rules into some set of database structures and (3) definition of a rule management mechanism.

In our context, the first stage – definition of authorization rules – is specified as:

[Define $\langle s, m \rangle$ on $\langle o \rangle$], where $\langle o \rangle$ is a result of a spatial query.

An authorization can be granted to an individual user, groups of users or user roles associated with different operations. Object $\langle o \rangle$ defines a data partition within the database for which that authorization holds. It can be a spatial component or a set of components, with geometries of type polygon, point or line, and be directly specified (through identifiers), or indirectly, as a query result.

A spatial permission is therefore directly related to the spatial query that it must satisfy. For example, the authorization “Ann has read access to all the rivers in São Paulo state” is nothing more than a read permission to access all

data on rivers resulting from the spatial query "select all the information of river features in São Paulo state" – see Section 5.

Subjects s can be defined in the same way as in conventional databases. The model considers that subjects are end users – engineers and designers within an AM/FM planning environment: their roles are indirectly defined by their login group. This is a compromise between full RBAC and DAC. This can easily be extended to include explicit roles, or software.

3.2 Granularity

Access granularity in our rules is that of the objects they define. This requires considering trade offs between number of objects considered in a rule and increased system complexity – the number of rules in the rule database increases with smaller access granularity. Similar to [BCDP05], we support hierarchical definitions of spatial extent, which is used to infer non-explicit rules.

Our solution considers two authorization rule specifications: $\langle s, o, m \rangle$ and $\langle s, Q, m \rangle$. The first one explicitly references the object identifier (for example, a point related to a specific pole, a line related to a street, a polygon related to a neighborhood). In this case, the authorization is executed in an individual object in the database. The second specification contains a spatial query, which defines the objects under control (see section 5). This solution is a compromise between management of specific objects ($\langle s, o, m \rangle$ rules) versus flexibility in defining authorizations ($\langle s, Q, m \rangle$ rules).

Consider the following rule: "Ann has read access to Jardim Paulista neighborhood", where object "Jardim Paulista" is a polygon identified by [id 501] – its geometry defines access granularity. The rule which is going to be stored is $\langle Ann, 501, read \rangle$ – Ann is allowed access to object 501 and all the objects inside 501 (see section 5).

A rule example using a query and with point granularity is "John can access just the subway stations in Vergueiro Street", where subway stations are points in the street. In this case, the rule is (John, all the subway stations in Vergueiro Street, read), where "all the subway stations in Vergueiro Street" can be specified as a spatial SQL query.

3.3 Set of Policies to Manage and Administer Authorizations

The model proposes a centralized administration of authorizations: just the administrator can grant and revoke permissions. Thus, it is not necessary to worry about the cascade and non-cascade revocation of authorizations, as in the DAC model [GW76].

Our model does not consider negative authorizations. These must be analyzed according to the application, and introduce a major complexity in the algorithms that evaluate an access request. If the mechanism allows negative authorizations, given an access request, it is necessary to verify if there are negative authorizations denying the use of an object, before allowing the access.

3.4 Algorithms to Analyze Access Requests

As mentioned before, our access control mechanism assumes that authorization rules are stored in a special repository within the database, and checked at access request. This request can be per transaction, or apply to an entire user session, and assumes that all the rules stored in the database are consistent according to the policies defined by the administrator. Access right is only granted if there is an explicit rule authorizing the subject to access that object with that access mode, or if the user access grant can be inferred using spatial containment properties.

Algorithm Access request validation

Input:

[1] access request (S, Q_a, M) .

[2] set of database authorization rules (s, o, m) and (s, Q, m) , stored in the rule repository

Output: [1] AUTHORIZED or [2] DENIED

1. Given an access request $AR = \langle S, Q_a, M \rangle$ where the query statement Q_a defines objects to be accessed, select all authorization rules $r_i = \langle s, o, m \rangle$ and $r_j = \langle s, Q_j, m \rangle$ from the rule database, where $s = S$ and $m = M$. The result of this step is a set of rules $RA = \langle S, o_i, M \rangle \cup \langle S, Q_i, M \rangle$.

2. Process the queries Q_i in $\langle S, Q_i, M \rangle$ in order to determine the referenced objects, obtaining the final set of rules $RF = \langle S, o_k, M \rangle$, where o_k are the objects returned by the execution of all Q_i queries.

3. Process the query Q_a , getting $AR = \langle S, o_a, M \rangle$, which determines the objects involved in the access request.

4. Detect conflicts between AR and objects in RF , according to section 4.

5. Resolve the conflicts using the policies defined in section 4.

Details of steps 4 and 5 can be found in [Sas02].

4 Managing Conflicts for Geographic Access Control

Access conflicts require checking spatial relationships between access requests and rules in the database. Generally speaking, conflicts fall into two cases: (i) objects totally or (ii) partially contained in another.

In case of total containment, access is granted, according to inference rules for hierarchies of objects subject to total containment. The existence of authorization $\langle s_1, o_2, m_1 \rangle$ allows to infer $\langle s_1, o_1, m_1 \rangle$, if o_1 is totally contained in o_2 .

Partial containment, however, introduces conflicts. Again, suppose s_1 has access to o_2 , and that object o_1 is partially contained in o_2 . Should s_1 be granted access to o_1 ? In this case, there are the following alternatives, which are considered at step 4 with possible user disambiguation:

1. yes, s_1 can access object o_1 , even if it is partially contained in o_2 ;
2. s_1 can access the part of the object o_1 contained in o_2 . This requires cutting the object in parts;

3. yes, only if there is also an authorization rule $\langle s_1, o_1, m_1 \rangle$, which authorizes s_1 to access the object o_1 explicitly;
4. yes, only if there is also an authorization rule $\langle s_1, o_3, m_1 \rangle$ in the database, where o_3 contains the rest of o_1 not contained in o_2 ;
5. yes, s_1 can access o_1 , if there is no negative authorization $\langle s_1, o_1, m_1, - \rangle$;
6. no, the situation does not occur because objects partially contained in another do not exist in the application domain.
7. no, the access to objects partially contained in another is denied.

5 Spatial Queries for Access Control

The spatial attributes considered in this research for access control are of type point (e.g., poles, trees), lines (e.g., street segments) and polygon (e.g., neighborhoods). The type depends on the scale. For example, in a 1:1.000.000 scale, cities, small woods and many types of surfaces can be represented by points.

Queries for access control involve different relationships between spatial object types (e.g., Point x Point, or Line x Line). They return a result set, which is the target of access control, the object o of the $\langle s, o, m \rangle$ triple. Different types of permission can be associated with each query result. We consider topologic and metric spatial query predicates and adopt the five topological operators defined by Clementini et. al. [CdFvO93] – in, overlap, touch, cross and disjoint – as sufficient to cover binary topological relationships.

The study of the objects in the database for access control must take two factors into account: (1) the result - spatial object, non-spatial object or part of an object; and (2) the predicate - spatial, non-spatial or both. Queries can produce descriptive or spatial attributes, or both. Query Q_x “Who are the subscribers recorded in the database” returns non-spatial objects (subscribers). The query Q_y “Which are the types of the cables installed in the Cambuí neighborhood” returns descriptive attributes (types of cable) for a spatial object (cables). The query Q_z “Supermarkets with more than 5 telephones installed” returns spatial objects (supermarkets), assuming that they have a spatial component. Query Q_x uses non-spatial predicates, while query Q_y uses a spatial predicate.

Consider query Q_z “Supermarkets with more than 5 telephones installed”. An example of an authorization rule involving Q_z might be “Ann can update the account charges data of the supermarkets with more than 5 telephones installed”, where s : Ann; o : points (supermarkets); m : write; the predicate is defined on descriptive attributes (number of telephones in a supermarket).

This type of reasoning, separating the definition of the permission from that of objects subject to access control, can be repeated for combinations of spatial objects and distinct predicates, and involve distinct kinds of geometric features.

6 Access Control in SAGRE

As mentioned in section 1, our work was motivated by the need for spatially sensitive access control for cooperative work in the CPqD Outside Plant Management System. This system will be referred to in the rest of this section by its

ancient name – SAGRE – to disambiguate references to the system and to its modules (see [Sag] for a description of the main functionalities of the system). It is a GIS-based system composed by a set of applications which automate processes related to outside telephone plant management. Two of its applications are relevant to access control issues: *Adm* and *Cad*.

The *Adm* application is geared towards system administrators in telecom companies. It allows managing the system users, and groups, inserting and deleting users, granting and revoking role permissions for users/groups.

The *Cad* application maintains the basic urban map and the telephone outside plant. The basic urban map [Mag97] is composed by the urban planning basic elements, such as: streets, street segments, monuments. The outside plant corresponds to the infrastructure information used by telecommunications services such as poles, terminal boxes, cables. The *Cad* application supports the management of projects, where a “project” involves infrastructure maintenance or expansion planning for a given region, usually within some urban area. When creating projects, it is necessary to indicate a manager and the manager’s area using geographic coordinates, defined as a polygon.

Our first modification concerns the *Adm* application, changing the internal tables that store user roles. They must contain insert, update and delete authorization rules that indicates the spatial element o , which will be authorized. User authentication must also be changed, since it will pass through more verification stages. Project managers can also intervene here.

Figure 1 presents a screen copy with a project developed using *Cad*. In normal system usage, a user has to define the geographic limits of a project (a polygon). Notice the area covers parts of features (e.g., lines), which complicates access control. The polygon is only used for visualization and does not impose any restrictions on objects to be modified by this project. The present version of *Cad* contains special code that verifies some spatial access control, but it is not flexible enough to consider different situations. An example of such a problem is the case of update cascades, where an update in a given object may propagate to objects outside the visible polygon. Thus, a person within a project confined to this polygon can change objects even when they are outside the project polygon.

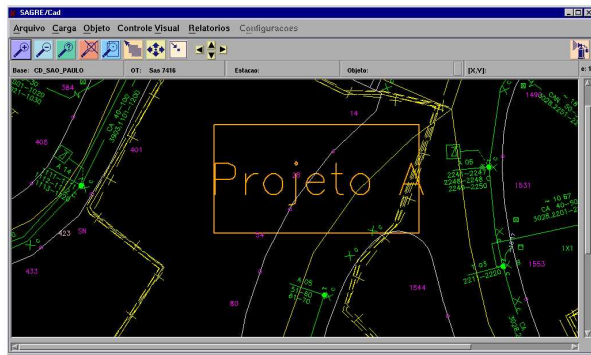


Fig. 1. Project designed in SAGRE/Cad

This means that changes must be made to allow preprocessing access requests. Even though some of our solutions have been considered in SAGRE, their full-fledged implementation would require a new module - Geographic Access Manager - to be created to check and manage spatial access rules [Sas02]. The generic solution, using authorization rules in a database and distinct kinds of access modes, still needs to be taken into account. Visualization must also be restricted to prevent users from seeing certain objects.

7 Conclusions and Extensions

This paper presented a generic access control model for GIS applications. The proposal is based on the definition of authorization rules $\langle s, o, m \rangle$, where objects o are characterized as a result of a geographic query. The main contributions of this paper are: survey of requirements for access control in geographic databases; definition of an authorization model based on the spatial characterization; discussion of implementation aspects of this model; brief presentation of application of the proposed mechanism for a real GIS system.

Spatially-sensitive access control is a research area that presents several challenges, with relatively few papers on the subject – e.g., [BCDP05, BBC⁺04]. The main differences with our proposal were that we were forced to simplify some of the issues, given the size and scope of SAGRE and its multiple user roles – e.g., we do not consider negative permissions, and roles are defined by login groups. Moreover, our proposal is geared towards solving problems that arise in cooperative planning activities using a GIS, while at the same time allowing normal operation for the same region.

Many extensions can be proposed. One concerns spatio-temporal access control. Another possibility is the incorporation of nested permissions. Also, conflicts among our rules must be studied, to maintain rule consistency. We have made a preliminary study concerning performance impact of our rule checking algorithms. Further work must be conducted along these lines.

Acknowledgements. This work was partially financed by CPqD Telecom & IT Solutions, CNPq, FAPESP, CNPq SAI, Agroflood and Web-MAPS Projects.

References

- [AJS⁺96] V. Ashby, S. Jajodia, G. Smith, S. Wisseman, and D. Wichers. Trusted Database Management Systems - Interpretation of the Trusted Computer System Evaluation Criteria. Technical Report 001-005, National Computer Security Center, 1996. 75 pages.
- [BBC⁺04] A. Belussi, E. Bertino, B. Catania, M. Damiani, and A. Nucita. An Authorization Model for Geographical Maps. In *Proc. 14th ACM GIS*, pages 82–91, november 2004.
- [BBF01] E. Bertino, P. Bonatti, and E. Ferrari. TRBAC: Temporal Role-Based Access Control Model. *ACM Transactions on Information and System Security*, 4(3):191–223, 2001.

- [BCDP05] E. Bertino, B. Catania, M. Damiani, and P. Perlasca. GEO-RBAC: a spatially aware RBAC. In *Proc, 10th ACM Symposium on Access Control*, pages 29–37, June 2005.
- [BCFM00] E. Bertino, S. Castano, E. Ferrari, and M. Mesiti. Specifying and enforcing access control policies for XML document sources. *World Wide Web*, 3(3):139–151, 2000.
- [BDPSN96] A. Baraani-Dastjerdi, J. Pieprzyk, and R. Safavi-Naini. Security in Databases: A Survey Study. February:1–39, 1996. <http://citeseer.nj.nec.com/baraani-dastjerdi96security.html>.
- [BHA00] E. Bertino, M. A. Hammad, W. G. Aref, and A. K. Elmagarmid. An access control model for video database systems. In *CIKM*, pages 336–343, 2000.
- [BJS95] E. Bertino, S. Jajodia, and P. Samarati. Database Security - Research and Practice. *Information Systems*, 20(7):537–556, 1995.
- [BP76] D. E. Bell and L. J. La Padula. Secure Computer Systems: Unified exposition and Multics interpretation. Technical report, The Mitre Corp., 1976.
- [CdFvO93] E. Clementini, P. di Felice, and P. van Oosterom. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. *Proceedings of the 3rd Symposium Spatial Database Systems*, pages 277–295, 1993.
- [FK92] D. Ferraiolo and Richard Kuhn. Role-Based Access Control. *Proceedings of 15th National Computer Security Conference*, 1992.
- [GW76] P. G. Griffiths and B. Wade. An authorization mechanism for a relational database system. *ACM TODS*, 1(3):243–255, 1976.
- [Mag97] G. C. Magalhaes. Telecommunications outside plant management throughout Brazil. In *Proc GITA 1997*, 1997.
- [PSA01] J. Park, R. Sandhu, and G. Ahn. Role-Based Access Control on the Web. *ACM Transactions on Information and System Security*, 4(1):37–71, 2001.
- [Sag] Sagre. <http://www.cpqdusa.com/solutions/outside.html>, accessed on April 2006.
- [Sas02] L. K. Sasaoka. Access Control in Geographic Databases. Master's thesis, Universidade Estadual de Campinas, June 2002. In Portuguese.
- [WHE04] W. Ye, J. Heidemann, and D. Estrin. Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks. *IEEE/ACM Transactions on Networking*, 12(3):493–506, 2004.

VTPR-Tree: An Efficient Indexing Method for Moving Objects with Frequent Updates

Wei Liao, Guifen Tang, Ning Jing, and Zhinong Zhong

School of Electronic Science and Engineering,
National University of Defense Technology, China

Abstract. Moving object databases are required to support queries on a large number of continuous moving objects. Indexes for moving objects must support both query and update operations efficiently. In previous work TPR-tree is the most popular indexing method for the future predicted position, but its frequent updates performance is very poor. In this paper we propose a novel indexing method, called VTPR-tree, for predicted trajectory of moving objects. VTPR-tree takes into account both the velocity and space distribution of moving objects. First the velocity domain is split, and moving objects are classified into different velocity buckets by their velocities, thus objects in one bucket have similar velocities. Then we use an improved TPR-tree structure to index objects in each bucket. VTPR-tree is supplemented by a hash index on IDs of moving objects to support frequent updates. Also an extended bottom-up update algorithm is developed for VTPR-tree, thus having a good dynamic update performance and concurrency. Experimental results show that the update and query performance of VTPR-tree outperforms the TPR*-tree.

1 Introduction

A number of emerging applications of data management technology involve the monitoring and querying of large quantities of continuous moving objects (e.g., the position of mobile service users, traffic control, etc.). Further, a wide range of other applications beyond moving-object applications also rely on the sampling of continuous, multidimensional variables. The provision of high performance and scalable data management support for such applications presents new changes. One key challenge derives from the need to accommodate frequent updates while simultaneously allowing for efficient query processing [1] [2] [3].

With the exception of few structures that are either purely theoretical, or applicable only in certain environment, the TPR*-tree [4] is the sole practical spatio-temporal index for predicted queries. TPR*-tree processes update as combinations of separate deletion and insertion operations that operate in a top-down manner. For frequent updates this technique often leads large mount of index node accesses and does not support concurrent operations well. On the other hand, TPR*-tree is constructed merely on the space domain, not considering the velocity distribution of moving objects. Thus its query and update performances will degrade greatly with time. Motivated by the above observations, in this paper we develop a novel index structure to solve the query and update problems in management of moving objects with frequent updates.

Our first contribution is a velocity-based time-parameterized R-tree (VTPR-tree), which takes into account the velocity distribution of moving objects, splits the velocity domain regularly into different velocity buckets, and then indexes each bucket with an improved TPR-tree structure. Also the construction, dynamic maintenance and query methods of VTPR-tree are discussed.

Second, we present an enhanced bottom-up update (EBUU) approach, referencing the R-tree update technique in [5], to support frequent update operations. The cost model and performance analysis of EBUU algorithm are also given.

Finally, we experimentally evaluate the proposed technique under a simulation environment. We study the effect of velocity buckets number, and compare the performance between VTPR-tree and TPR*-tree thoroughly. Experimental results show that the query and update performances of VTPR-tree outperform TPR*-tree.

2 Related Works

The TPR-tree [2] is an extension of the R-tree that can answer predicted queries on dynamic objects. The index structure is very similar to R-tree, and the difference is that the index stores velocities of elements along with their positions in nodes. The leaf node entry contains not only the position of moving objects but also their velocities. Similarly, an intermediate node entry also stores an MBR and its velocity vector VBR. As in traditional R-tree, the extents of MBR are such that tightly encloses all entries in the node at construction time. The velocity vector of the intermediate node MBR is determined as follows: (i) the velocity of the upper edge is the maximum of all velocities on this dimension in the sub-tree; (ii) the velocity of the lower edge is the minimum of all velocities on this dimension. This ensures that the MBR always encloses the underlying objects, but it is not necessarily tight all the time.

With the observation that top-down update is inherently inefficient because objects are stored in the leaf nodes, whereas the starting point for updates is the root, Lee presents a localized bottom-up update (LBUU) algorithm for R-tree with frequent updates[5]. To access an object entry in leaf node directly, a secondary index on object IDs is added to the traditional R-tree. When an update is issued, the object entry can be located directly by the secondary hash index. The LBUU algorithm gains the most when updates preserve locality, so that the majority of updates are concentrated on the leaf level and its parent level. However, this approach results in a dip in query performance due to the enlargement of leaf MBRs. Further, the need to maintain parent pointers at the leaf level reduces fanout and increases the maintenance costs during node splits

3 Problem Definition and Overview

The TPR-tree is constructed mainly in the space domain, not considering the distribution of moving objects in the velocity domain. At the construction time, TPR-tree index clusters the moving objects according to their spatial proximity into different nodes; however, the velocities of objects in the same page are always discrepant greatly. The minority of moving objects with higher velocity make the VBR relatively

larger. Thus the MBR will increase extremely with time, causing deterioration of the query and update performance. Actually, in most applications the predicated queries visit many unnecessary intermediate TPR-tree nodes due to overlaps between area of MBRs and the massive dead space.

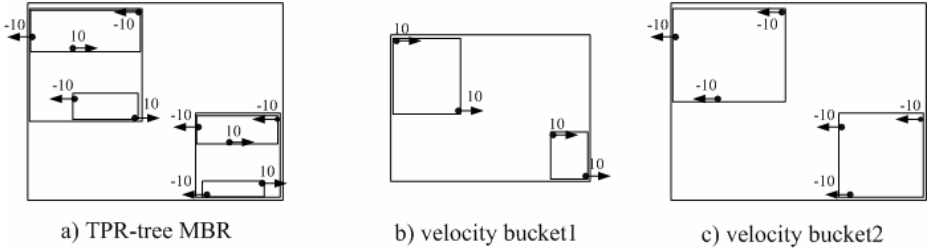


Fig. 2 velocity buckets partition of VTPR-tree

Figure 2 shows an example, where the moving objects in two dimensions have velocity vectors $(10, 0)$ and $(-10, 0)$ respectively. Figure 2 a) illustrates the MBRs of TPR-tree which is constructed merely in space domain. Obviously the MBRs of intermediate TPR-tree nodes extend extremely along the horizon direction, thus incurring the degradation of query performance. So an effective indexing technique needs to consider the distribution of moving objects in both velocity and space domain to avoid the deterioration of query and update performance.

Motivated by this, we propose a velocity distribution based time-parameterized R-tree (VTPR-tree) index structure. First the velocity domain is partitioned regularly into buckets with the same velocity window. Then moving objects are mapped into different velocity buckets by their velocities, and objects with similar velocities are clustered into one bucket. Finally, an improved TPR-tree structure is presented for indexing objects in each bucket.

Figure 2 b) and c) shows the MBRs of VTPR-tree that considers the distribution of velocity domain. The moving objects are classified into two velocity buckets. Bucket 1 (as shown in Figure 2 b)) contains the moving objects with the velocity vector $(10, 0)$, while the bucket 2 (as shown in Figure 2 c)) contains the moving objects with the velocity vector $(-10, 0)$. As seen, the MBR of each bucket moves with time, but the shapes of MBR keep unchanged. So VTPR-tree can hold a good query performance during all the future time.

On the other hand, existing TPR-tree update algorithms work in a top-down manner. For each update, one index traversal to locate the item for deletion and another traversal to insert a new item are needed. The deletion operation effects the disk I/O cost (one deletion searching, and one insertion searching) absolutely. For the MBRs of TPR-tree become looser with time, thus incurring lots of area overlaps between MBRs, and the searching operation for deletion needs to visit all the TPR-tree nodes at worst case. Although an active tightening technique is presented in TPR*-tree update algorithm to decrease area overlaps between MBRs, it is not effective in frequent update applications.

The top-down approach for visiting the hierarch tree-like structure is very simple, and easy for dynamic maintenance. But for TPR-tree like index structures, the area between intermediate node MBRs inevitably overlaps each other (which is not

occurred in other traditional indexes such B-tree), so that the top-down update strategy is inefficient in nature. This manner results in the deterioration of TPR-tree, and is not suitable in frequent update applications. Motivated by this, we exploit the bottom-up strategy in [5] to improve the update performance of VTPR-tree.

4 The VTPR-Tree

Section 4.1 discusses the structure of VTPR-tree, while Section 4.2 provides the construction algorithm, Section 4.3 gives the insertion and deletion algorithm, Section 4.4 describes the query algorithm, and Section 4.5 evaluates the effects of index parameters.

4.1 The VTPR-Tree Structure

In VTPR-tree index, the basic TPR-tree structure is kept intact, and another linear velocity bucket queue structure is introduced. Each item in the velocity bucket queue holds summary about the velocity range and space extent of moving objects contained in this bucket. In addition, each bucket item points to a TPR-tree structure that constructed on the moving objects in this bucket. Since the velocity bucket queue is often visited when an update or query is presented, we organize this structure into main memory to reduce frequent disk I/Os.

To support bottom-up update strategy, VTPR-tree index structure supplements a secondary hash index on IDs of moving objects to access object entries at the leaf level directly and thus avoid a top-down search from the root point. When an update from moving object is issued, the update algorithm first locates the entry in leaf node with hash index, and then modifies the VTPR-tree from leaf level until the root to reflect the update.

Specifically, in VTPR-tree structure we keep the basic TPR-tree index structure, and a main-memory linear summary queue is added to maintain the sketch of each bucket. The items in the queue are organized as vector $\langle MBR, VBR, ptr \rangle$, where MBR , VBR , ptr denote the space extent, velocity range of this bucket and pointer to its corresponding TPR-tree respectively. To improve the update performance of VTPR-tree, we introduce a disk-based hash index structure to access the leaf node of VTPR-tree directly. Further, we modified the original TPR-tree node item as vector $\langle entry, \dots, entry, parentptr \rangle$, where $entry$ denotes the child nodes contained in this node, and $parentptr$ denotes the physical address of its parent node. Compared with node page size, the space consumption by pointer $parentptr$ is trivial, and its effect on fanout of VTPR-tree is ignorable. The form of $entry$ is vector $\langle MBR, VBR, ptr \rangle$, where MBR , VBR , ptr denote the bounding rectangle, velocity range of this node and pointer to its subtree respectively.

Figure 3 illustrates the VTPR-tree structure. The top right corner is a velocity bucket queue, in which each item describes MBR and VBR of its corresponding TPR-tree; the bottom right corner is a hash index constructed on IDs of moving objects, the item is defined as vector $\langle oid, ptr \rangle$, where oid denotes the identifier of moving objects,

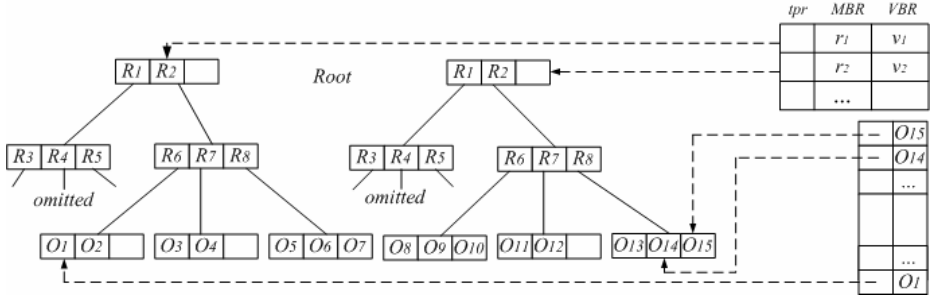


Fig.3. Structure of VTPR-tree

and ptr denotes physical offset of the object entry in leaf node; the left of Figure 3 is the TPR-tree structures pointed to by the velocity bucket; In this figure pointer to parent node is not clearly depicted for concision.

4.2 Construction

We use spatio-temporal histograms as mentioned in [6] on two velocity dimensions to compute the velocity buckets number and their velocity range. The main idea is to divide the moving objects set into different partitions with about the same objects number. Then the algorithm scans the moving objects in sequence, and inserts objects into the corresponding TPR-tree pointed to by a bucket according to their velocities. To avoid redundant disk I/Os caused by frequent insertion one by one, construction algorithm exploits the bulk loading technique [6] to construct the TPR-tree.

4.3 Insertion and Deletion

The insertion algorithm is straightforward. When inserting a new moving object into VTPR-tree index, the algorithm first scans the main-memory velocity buckets queue to find the bucket that contains this object, and the TPR-tree related to this bucket can be obtained by the pointer in this bucket item. Then a new object entry is produced and inserted into a suitable leaf node using standard TPR-tree insertion algorithm. Finally, a new item is produced and inserted into the hash index. If overflow occurs while insertion, the TPR-tree node must be split with the method mentioned in [4].

To delete an object entry from the VTPR-tree, deletion algorithm first locates the leaf node that holds the object entry with hash index, and then deletes the entry directly. Then the algorithm ascends the branches of VTPR-tree by the pointer $parentptr$ until the root node, and modifies the MBR and VBR of intermediate nodes along the path meanwhile. Finally, the corresponding object item in hash index must be deleted to reflect the change.

4.4 Query Algorithm

Specifically, given a predicted widow query $q(q_R, q_T)$, where q_R denotes the query space area, and q_T denotes the query time range. The algorithm first scans the velocity

bucket queue, and for each bucket we can get whether any object in this bucket lies in the query area q_R during q_T according to the bucket's MBR and VBR. If q_R does not intersect the space area covered by this bucket during future time q_T , the TPR-tree pointed by this bucket need not to be visited; or else the corresponding TPR-tree must be searched with the standard TPR-tree query algorithm.

4.5 Effects on Index Performance

VTPR-tree partitions moving objects into velocity buckets that don't overlap each other in velocity domain. So the index can perform a good concurrency when large amount of concurrent dynamic operations occur. Obviously, the concurrency improves with the number of velocity buckets. But with too large the bucket number the index structure is prone to instability. That is to say, because the velocity range of each bucket is too small, when an object is updated, the likely object's shift from one bucket to another bucket may cause excessive disk I/Os. In addition, limited by the system memory, and considering the space and velocity distribution uncertainty of moving objects, too many velocity buckets don't reduce node accesses for processing predicted query. So that VTPR-tree with a proper velocity buckets number can obtain the optimal query and update performance.

The cost of maintain the velocity bucket queue is inexpensive. The queue is constructed along with VTPR-tree, when dynamic operations such as insertion and deletion occurred, after the modification of TPR-tree pointed to by the corresponding bucket item, the summary in which must also be updated. The queue is pinned in main memory, thus decrease the visit cost greatly. And the space consumed by velocity queue is rather small and ignorable.

5 Update Algorithm

The main idea of EBUU algorithm is described as follows: when an update from moving object is presented, the algorithm first judges whether the object locates in the MBR and VBR of current leaf node; if true, the algorithm updates the leaf node directly; or else, the algorithm deletes old object entry in current leaf node, and ascends the TPR-tree branches to find a minimum subtree which contains the updated object, then inserts a new entry into this subtree with standard TPR-tree insertion algorithm. Otherwise, if the velocity of this object lies outside current bucket, the algorithm deletes the object entry from the TPR-tree related to this bucket and inserts the object into a suitable bucket.

Specifically, the EBUU algorithm processes the following three cases during an update operation:

- 1) The new position and velocity of moving object lie in the MBR and VBR of current leaf node. As shown in Figure 1 object b moves from along the positive direction of horizon axis towards the negative direction of horizon axis and the value remains unchanged, thus the updated velocity still lies in the bounding rectangle $[-1,1]$. So the algorithm only needs to modify the object entry in leaf node and write it back into disk.

2) The new position and velocity of moving object lie in the MBR and VBR of a subtree (intermediate node). As in Figure 1 object d moves from along the positive direction of horizon axis towards the positive direction of vertical axis, thus the velocity lies outside the VBR of rectangle E_l . Then the algorithm ascends the TPR-tree branches to find a local subtree (root node in this case) and performs a standard top-down update under this subtree to insert object d into a suitable leaf node.

3) The new velocity of moving object lies in some other velocity bucket. The algorithm removes the old object entry from TPR-tree leaf node, and inserts this updated object into the corresponding bucket.

The EBUU algorithm is detailed as follows.

Algorithm EBUU

Input: oid , $newMBR$, $newVBR$

Output: updated $VTPR$ -tree

BEGIN

1. For all $h \in H$ Do
2. If $h.oid = oid$ Then
3. $node \leftarrow h.node$; break;
4. End If
5. End For
6. If $newMBR \subset node.VBR$ and $newVBR \subset node.VBR$
7. Then write out $node$; return;
8. Else
9. Delete old entry in $node$; continue;
10. While $node.parent \neq NULL$
11. $node \leftarrow node.parent$;
12. For all $t \in node$
13. If $newMBR \subset t.MBR$ and $newVBR \subset t.VBR$
14. Then $Insert(t, oid, newMBR, newVBR)$;return;
15. End If
16. End For
17. End While
18. For all $l \in L$ Do
19. If $newMBR \subset l.VBR$ and $newVBR \subset l.VBR$ Then
20. $Insert(l.tpr, oid, newMBR \sqcup newVBR)$; return;
21. End For

END

6 Experimental Results and Performance Analysis

6.1 Experimental Setting and Details

In this section, we evaluate the query and update performance of VTPR-tree with TPR-tree and TPR*-tree. We use the Network-based Generator of Moving Objects [12] to generate 100k moving objects. The input to the generator is the road map of Oldenburg (a city in Germany). An object appears on a network node, and randomly

chooses a destination. When the object reaches its destination, an update is reported by randomly selecting the next destination. When normalize the data space to 10000×10000 , the default velocity of objects is equal to 30 per timestamp.

The predicted window queries are generated similarly, i.e., they are objects moving on the same network. The query parameters are as follows: i) the query spatial extent $qRlen$ is set as $100 \times 100, 400 \times 400, 800 \times 800, 1200 \times 1200, 1600 \times 1600$ respectively, and the starting point of its extent randomly distributes in $(10000 - qRlen) \times (10000 - qRlen)$. ii) the query time window is $[Tisu, Tisu + qTlen]$ ($Tisu$ is the time when the query is presented), where $qTlen = 20, 40, 60, 80, 100$ respectively. The query performance is measured as the average number of node accesses in processing ten predicted window queries with the same parameters. The update performance is measured as the average node accesses in executing one hundred moving objects updates. For all simulations we use a Celeron 2.4GHz CPU with 256MByte memory.

6.2 Performance Analysis

We compare the query and update performance of VTPR-tree, TPR*-tree and TPR-tree by node accesses. In order to study the deterioration of the indexes with time, we measure the performance of VTPR-tree, TPR-tree and TPR*-tree, using the same query workload, after every 5k updates.

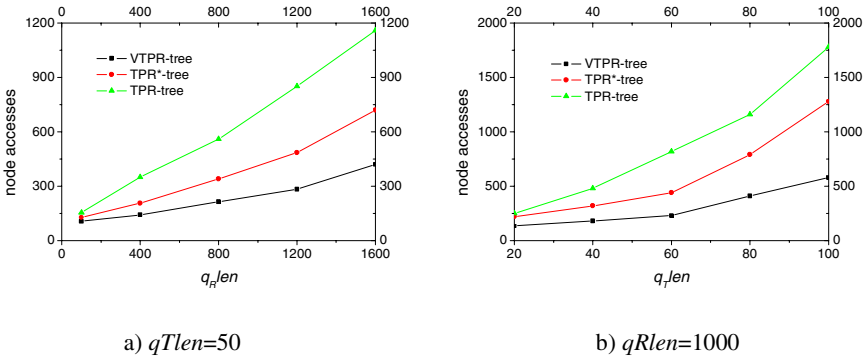


Fig. 4. Predicted query cost comparison

Figure 4 a) and b) shows the query cost as a function of $qRlen$ and $qTlen$ respectively. In Figure 4 a) we fix the parameter $qTlen=50$ and in Figure 4 b) we fix the parameter $qRlen=1000$. As seen, the query performance of VTPR-tree works best and TPR-tree exhibits a worst performance. This is because VTPR-tree is constructed both in the space and velocity domain, the overlaps between area of MBRs are relatively less than those of TPR-tree and TPR*-tree, thus having a good query performance. While TPR*-tree and TPR-tree may visit many unnecessary nodes for the massive overlaps between area of MBRs with time, causing a worse query performance.

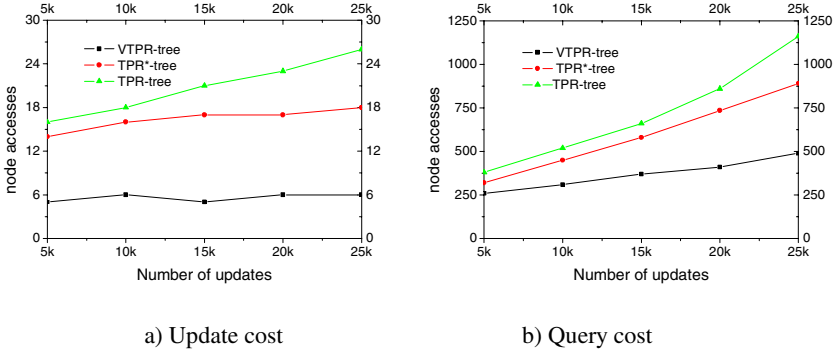


Fig.5. Update and query performance comparison

Figure 5 compares the average query and update cost as a function of the number of updates. As shown in figure 5 a), the node accesses needed in VTPR-tree update operation are far less than TPR*-tree and TPR-tree. And VTPR-tree and TPR*-tree have nearly constant update cost. This is because VTPR-tree exploits the bottom-up update strategy to avoid the excessive node accesses for deletion search, while TPR*-tree and TPR-tree process update in top-down manner, needing more node accesses.

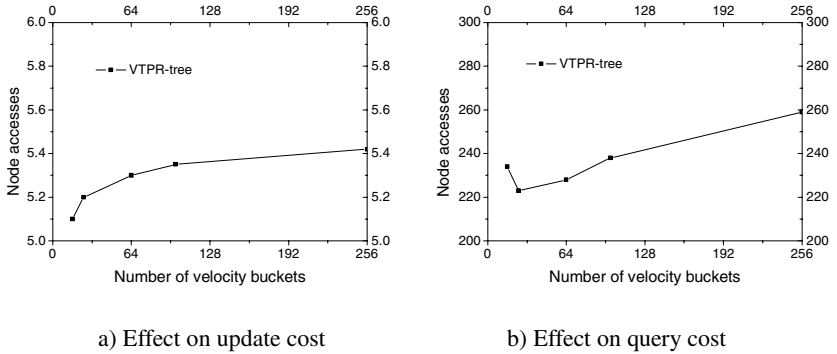


Fig. 6. Bucket number effect on query and update performance

Figure 5 b) shows the node accesses in processing one predicted query as function of an interval of 5k updates when fixing the parameters $qTlen=50$ and $qRlen=1000$. It is clear that the query cost increases with the number of updates. The VTPR-tree has a slow increasing query cost, while the cost of TPR*-tree and TPR-tree increase significantly. This is because the MBRs of VTPR-tree extend less than those of TPR*-tree and TPR-tree, so the degrading is not much expensive.

Figure 6 shows the effect of velocity buckets number on query and update performance of VTPR-tree. We fix the parameters $qTlen=50$ and $qRlen=1000$. As shown in Figure 6 a), the update cost of VTPR-tree increase slightly with the num-

ber of velocity buckets, because the small velocity bucket window causes the shift of moving objects between buckets, thus incurs excessive node accesses. From Figure 6 b) we can see, the query performance will degrade with too many buckets, this is because of, as mentioned earlier, the uncertainty of moving objects distribution in velocity and space, the probability of overlaps between the area covered by velocity buckets and query region don't decrease linearly as expected. However, it may bring more TPR-tree node accesses. In this experiment, we set the number of velocity bucket equal to 25.

7 Conclusion

This paper investigates the problem of indexing predicted moving objects trajectory. Our contribution is a novel indexing method, referred to as velocity based time-parameterized R-tree (VTPR-tree). VTPR-tree considers the distribution of moving objects both in velocity and space domain, and supplemented by a hash index on IDs of moving objects to support frequent updates. Also an extended bottom-up update algorithm is developed for VTPR-tree, thus having a good dynamic update performance and concurrency. Experimental results show that VTPR-tree's update and query performance outperforms the conventional TPR-tree and TPR*-tree under all conditions.

The work in this paper can be extended in several directions. First, we would like to investigate alternative predictive queries using the VTPR-tree, in particular, nearest neighbors and joins. Another avenue of research is integrating the approach developed here with techniques to solve the phantom problem.

References

1. Christian S. Jensen, Dan Lin, Beng Chin Ooi.. Query and Update Efficient B+-tree Based Indexing of Moving Objects. Proc. VLDB Conf. (2004)
2. Simonas Saltenis, Christian S. Jensen, et al. Indexing the Positions of Continuously Moving Objects. Proc. SIGMOD Conf. (2000) 331-342
3. Mohamed F. Mokbel Thanaa M. Ghanem. Spatio-temporal Access Methods. IEEE Data Engineering Bulletin (2003)
4. Tao, Y., Papadias, D. and Sun, J.. The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries. Proc. VLDB Conf. (2003) 790-801
5. M. Lee, W. Hsu, C. Jensen, B. Cui, and K. Teo. Supporting Frequent Updates in R-Trees: A Bottom-Up Approach. Proc. VLDB Conf. (2003)
6. Bin Lin and Jianwen Su. On bulk loading TPR-tree. Proc. IEEE Conf. Mobile Data Management (2004) 114-124
7. Brinkhoff, T. A Framework for Generating Network-Based Moving Objects. GeoInformatica Vol.(6)2 (2002) 153-180

New Query Processing Algorithms for Range and k-NN Search in Spatial Network Databases*

Jae-Woo Chang, Yong-Ki Kim, Sang-Mi Kim, and Young-Chang Kim

Dept. of Computer Engineering, Chonbuk National Univ.,
Chonju, Chonbuk 561-756, South Korea

jwchang@chonbuk.ac.kr, {ykkim, smkim, yckim}@dblab.chonbuk.ac.kr

Abstract. In this paper, we design the architecture of disk-based data structures for spatial network databases (SNDB). Based on this architecture, we propose new query processing algorithms for range search and k nearest neighbors (k-NN) search, depending on the density of point of interests (POIs) in the spatial network. For this, we effectively combine Euclidean restriction and the network expansion techniques according to the density of POIs. In addition, our two query processing algorithms can reduce the computation time of network distance between a pair of nodes and the number of disk I/Os required for accessing nodes by using maintaining the shortest network distances of all the nodes in the spatial network. It is shown that our range query processing algorithm achieves about up to one order of magnitude better performance than the existing range query processing algorithm, such as RER and RNE [1]. In addition, our k-NN query processing algorithm achieves about up to 170~400% performance improvements over the existing network expansion k-NN algorithm, called INE, while it shows about up to one order of magnitude better performance than the existing Euclidean restriction k-NN algorithm, called IER [1].

1 Introduction

Most of existing work in spatial databases considers Euclidean spaces, where the distance between two objects is determined by the ideal shortest path connecting them [2]. In practice, point of interests (POIs) and moving objects usually lie on a spatial network, e.g. road network, where the network distance is determined by the length of the practical shortest path connecting POIs and objects on the network. For example, a gas station nearest to a query point in Euclidean space may be more distant from the query point in a given network space than any other gas stations. Therefore, the network distance, rather than the Euclidean one, is an importance measure in spatial network databases (SNDB). Recently, the spatial network databases have been studied for emerging applications such as location-based service (LBS) and Telematics [1,3,4]. Studies on SNDB can be divided into three research categories, that is, data model, query processing techniques, and index structures. First, Speicys dealt with a

* This work is financially supported by the Ministry of Education and Human Resources Development(MOE), the Ministry of Commerce, Industry and Energy(MOCIE) and the Ministry of Labor(MOLAB) through the fostering project of the Lab of Excellency.

computational data model for spatial network [3]. Secondly, Papadias et al. proposed query processing algorithms for range search, spatial joins, closest pairs as well as k-NN [1]. Finally, Pfoser and Jensen designed a novel index structure for SNDB [4]. In this paper, we design the architecture of disk-based data structures for SNDB. Based on this architecture, we also propose new query processing algorithms for range search and k-NN search, depending on the density of POIs in the spatial network. In addition, our query processing algorithms can reduce the computation time of network distance between a pair of nodes as well as the number of disk I/Os accesses for visiting nodes by maintaining the shortest network distances of all the nodes in the spatial network. Thus, our query processing algorithms can improve the existing efficient k-NN and range query processing algorithms [1]. This paper is organized as follows. In Section 2, we analyze related work on query processing algorithms for SNDB. In Section 3, we first present our architecture of disk-based data structures for SNDB, and then we propose new range and k-NN query processing algorithms. In Section 4, we provide the performance analysis of our range and k-NN query processing algorithms. Finally, we draw our conclusions and suggest future work in Section 5.

2 Related Work

In this section, we overview related work on query processing algorithms for spatial network databases (SNDB). First, Jensen et al. described a general framework for k-NN queries on moving objects in road networks [5]. The framework includes a data model and a set of concrete algorithms needed for dealing with k-NN queries. The algorithms for k-NN queries employ a client-server architecture that partitions the NN search. A preliminary best-fit search for a nearest-neighbor candidate (NNC) set in a graph is performed on the server. Then, the maintenance of the query result is done on the client, which re-computes distances between data points in the NNC set and the query point, sorts the distances, and refreshes the NNC set periodically to avoid significant imprecision. The combination of NNC search with the maintenance of an active result provides the user with an up-to-date query result. Secondly, Papadias et al. proposed a flexible architecture for SNDB by separating the network from the entity datasets [1]. That is, they employ a disk-based network representation that preserves connectivity and location, while spatial entities are indexed by respective spatial access methods for supporting Euclidean queries and dynamic updates. Using the architecture, they also developed two frameworks, i.e., Euclidean restriction and network expansion, for each of the most common spatial queries, i.e., nearest neighbors, range search, closest pairs, and distance joins. The proposed algorithms expand conventional query processing techniques by integrating connectivity and location information for efficient pruning of the search space. Specifically, the Euclidean restriction algorithms take advantages of the Euclidean low-bound property to prune the search space while the network expansion algorithms perform query processing directly in the network. Thirdly, Kolahdouzan and Shahabi proposed a novel approach to efficiently evaluate k-NN queries in SNDB by using first order Voronoi diagram [6]. This approach can overcome the problem of expensive network distance computation because it is based on partitioning a large network to small

Voronoi regions, and then pre-computing distances within the regions. Finally, Haung et al. proposed a versatile approach to k-NN computation in spatial networks, called island approach [7]. The island approach computes the k-NNs along with the distance to each one, but does not compute the corresponding shortest paths. The rationale for the design decision is that a mobile use is expected to only be interested in the actual path to one nearest neighbor selected from the k-NN result, and so the path computation is better left to a subsequent processing step. Thus the island approach focuses on managing the trade-off between query and update performance by offering flexible means of balancing re-computation and pre-computation.

3 New Query Processing Algorithms for SNDB

In this section, we first design the architecture of disk-based data structure for spatial network data in SNDB. Based on the architecture, we propose new query processing algorithms for range and k-NN search in SNDB, respectively.

3.1 Architecture for Disk-Based Data Structure

In SNDB, a road network is defined as a graph $G = (V, E)$ where V is a set of vertexes (nodes) and E is a set of edges. A node denotes a road junction and the starting/ending point of a road. An edge denotes the road between two junctions and the weighted edge denotes the network distance. That is, each edge connecting node n_i and n_j includes a network distance $d_N(n_i, n_j)$ which equals the length of the shortest path from n_i to n_j in the network. For the simplicity of our definition for a road network, an edge is assumed to be undirected edge, thus meaning that moving objects can move on a road in both directions. Most of the existing work on storage structures for SNDB focuses on disk-based data structures representing spatial network, especially, storing both the nodes and the edges of the spatial network into a secondary storage. The objects on the spatial network can be divided into two types according to their mobility, such as points of interest (POIs), like restaurants, hotels, and gas stations, and moving objects, like cars, persons, motorcycles, etc. To design our architecture for disk-based data structures for SNDB, we differentiate the underlying network from POIs and moving objects. For the spatial network data, we maintain the disk-based data structure for both nodes and edges. For nodes, the node-node matrix file is used to store all the network distance $d_N(n_i, n_j)$ between node n_i and node n_j in the spatial network and the node adjacent information file is used to maintain the connectivity between nodes. Both the node ID table and the hash table are used to gain fast accesses to the information of a specific node. For edges, the edge information file is used to store the edge information as well as to maintain POIs residing on an edge. The edge R-tree is used to locate edges rapidly for answering spatial queries. The architecture of disk-based data structures for spatial network data is shown in Figure 1. The architecture supports the following main primitive operations for dealing with SNDB. (i) `find_edge(p)` outputs a set of edges that covers a point p by performing a point location query on the edge R-tree. (ii) `find_points(e)` returns a set of POI points covered by the edge e . (iii) `compute_ND(p1, p2)` returns the network distance $d_N(p1, p2)$ of two points $p1, p2$ in the network. This can be achieved in an effective way by accessing the node-node matrix file incorporated into our architecture via the hash table.

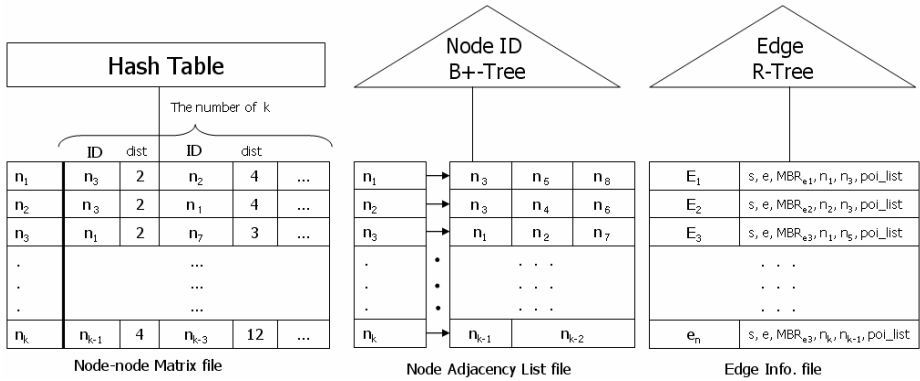


Fig. 1. Architecture of disk-based data structures for SNDB

3.2 Range Query Processing Algorithm

Because all of POIs and moving objects lie only on the spatial network, a range query processing algorithm for SNDB is quite different from the conventional ones proposed for the ideal Euclidean space [8]. For instance, suppose a query to find gas stations within 10Km from a query point (q) in Figure 2. The results to satisfy the query in the Euclidean space are p1, p2, and p3 while only p2 can satisfy the query in the network space.

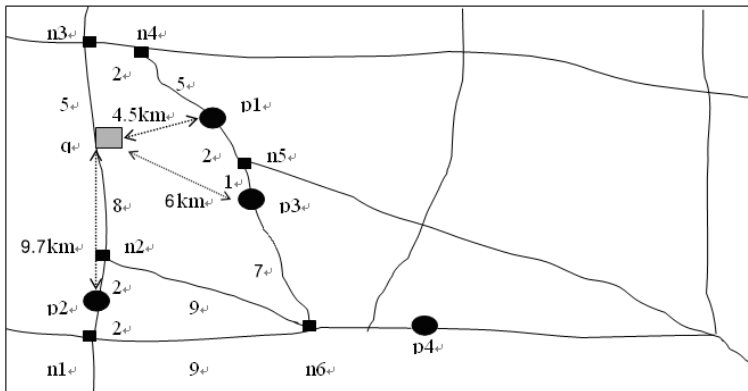


Fig. 2. Range search in Euclidean spaces and spatial networks

To design an efficient range query processing algorithm in SNDB, the Euclidean restriction algorithm, called RER, was proposed [1] by simply applying into the spatial network the conventional algorithms being proposed in Euclidean space. But, the RER generally requires a large number of disk I/O accesses to answer a range query in the underlying network. To remedy this problem, the network expansion algorithm, called RNE, was proposed [1], where it performs network expansion starting from an

edge covering a query and determines if objects encountered are within a given range. However, both the RER and the RNE are inefficient where there are lots of roads, being represented as lines, and lots of intersections cross them, being represented as nodes, in spatial networks. This is because they require a lot of the computation time of network distance between a pair of nodes and the number of disk I/Os accesses for visiting nodes. Moreover, the RER and the RNE do not always outperform each other, and their performances highly depend on the density of POIs in the spatial network. Therefore, we need to consider the following for designing an efficient range query processing algorithm.

Consideration 1: To efficiently perform range (or k-NN) query processing in SNDB, the pre-processing technique for network distance computation should be used because the computation cost is very expensive.

Consideration 2: To efficiently perform range (or k-NN) query processing in SNDB regardless of the density of POIs in the spatial network, an approach to combine both Euclidean restriction and network expansion techniques should be considered because their performances highly depend on the density of POIs.

To satisfy the first consideration, we perform the materialization of pre-computed results one time in order that the network distance computation can be facilitated [9]. A critical disadvantage of maintaining all the network distances is to require a huge storage space. For example, we assume that the number of nodes in the network is 200,000 and one network distance is stored with four bytes, we require 160GB to store all the network distances. Because Maxter and Segate Inc. offer the hard disk drivers (HDDs) of 500GB in capacity, it is possible to maintain all the network distances requiring a huge storage capacity in a disk. A record RM_i for a node N_i in the node-node matrix file, as shown in Figure 1, is $RM_i = \langle \text{dist}(N_i, N_1), \text{dist}(N_i, N_j) \dots \text{dist}(N_i, N_n) \rangle$ where $\text{dist}(N_i, N_j)$ is the shortest network distance between N_i and N_j . There are little updates on node-to-node matrix file because nodes meaning the intersections of a road are seldom changed. To satisfy the second consideration, we combine both Euclidean restriction and the network expansion techniques in an effective manner, depending on the density of POIs. In general, Euclidean restriction technique is efficient when the density of POIs is very high, i.e., in a city center, but does not perform well in a situations when POIs are sparse in spatial network. On the other hand, the network expansion technique excels in query performance when the density of POIs is low, but is not efficient in a situation when many POIs are located in a small area and the size of the radius in a range query is relatively large. Thus, based on our two considerations, we propose a new range query processing algorithm which performs the pre-computation of the shortest paths between all the nodes in the spatial network and combines the Euclidean restriction and network expansion techniques according to the density of POIs in the network, as shown in Figure 3. Here a density (D) can be calculated the number of POIs in a circle made from by the radius (r) in a range query. If D is greater than a thresh-hold value given through the experiment, our range query processing algorithm adopts the Euclidean restriction technique. Otherwise our algorithm adopts the network expansion technique.

Algorithm Range(q, r) /* q is the query point and r is the network distance */

1. Estimate D , the density of POIS in a circle made by r from q
2. $result = \emptyset$
3. if($D > threshold\_value$) {
4. $PE = \text{Euclidean-range}(q, e)$
5. for each point p in PE {
6. $dN(q, p) = \text{compute\_network\_dist}(q, e(ni, nj), p)$
7. if($d_N(q, p) \leq r$) $result = result \cup p$ }
8. } else { $e(ni, nj) = \text{find\_edge}(q)$
9. $EN = \text{expand\_network}(e(ni, nj))$ // EN is a set of edges
10. for each edge $e(n, m)$ in EN {
11. $PS = \text{set of POIs covered by } e(n, m)$
12. for each p in PS {
13. $dN(q, p) = \text{compute\_network\_dist}(q, e(ni, nj), p)$
14. if($dN(q, p) \leq r$) $result = result \cup p$ }
15. } } //end of for //end of if else

End Range

Function compute_network_dist($q, e(ni, nj), p$) /* q is a query point, p a target POI, and $e(ni, nj)$ an edge covered by q */

1. $e(nk, nl) = \text{find\_edge}(p)$
2. $return dN(q, p) = \min\{dN(ni, nk) + dN(ni, q) + dN(nk, p),$
 $dN(ni, nl) + dN(ni, q) + dN(nl, p),$
 $dN(nj, nk) + dN(nj, q) + dN(nk, p),$
 $dN(nj, nl) + dN(nj, q) + dN(nl, p)\}$

End compute_network_dist

Fig. 3. Our range query processing algorithm

3.3 k-Nearest Neighbors Query Processing Algorithm

A k-NN query processing algorithm for SNDB is also quite different from the conventional ones proposed under the assumption that POIS and moving object lie on the ideal Euclidean space [10]. As a result, the nearest neighbor of a query in the Euclidean space may be not the case in a spatial network. For example, the nearest neighbor of q is p_1 in the Euclidean space, their distance being 4.5Km in Figure 2. In the spatial network, the nearest neighbor of q is p_2 , not p_1 , and the network distance is 10Km because there is no direct path between q and p_1 . As a k-NN query processing algorithm in SNDB, the Euclidean restriction algorithm, called IER, can apply the conventional algorithms proposed for Euclidean space into the spatial network [1]. However, it generally searches a large number of Euclidean nearest neighbors to find the network nearest neighbors, thus leading to a large number of disk I/O accesses. To remedy this problem, the network expansion algorithm, called INE, can perform network expansion starting from a query and examines objects in the order that they are encountered until finding k-nearest neighbors [1]. It computes the network distance of k-nearest neighbors from q and terminates when the network distance of the first node in a queue is greater than the Euclidean distance. However, the INE can cause the expensive cost of network distance computation. To remedy the problem, the Voronoi-based and the island approaches pre-compute the networks distances within each Voronoi polygon (or cell) for answering k-NN queries. However, both approaches are

inappropriate for medium and dense datasets because both the number of cells and the number of border points increase dramatically as the dataset gets denser. That is, they cause higher precomputation overhead depending on the density of POIs and so they cannot be used for real applications having a high density of POIs. In order to overcome their problem, it is necessary to propose a new k-NN query processing algorithm to satisfy both Consideration 1 and 2 mentioned in the section 3.2. We also need to consider the additional one which is suitable to k-NN search as follows.

Consideration 3: To acquire the actual k-nearest neighbors of q in an effective manner, the initial set of near-optimal candidates k' ($k' \leq k$) should be obtained.

Algorithm K-NN(q, k) /* q is the query point and k is the number of nearest neighbors needed to find */

1. Estimate D , the density of POIs in a circle whose radius is Euclidean distance of the k -th nearest neighbor from q
2. if($D > \text{threshold\_value}$) {
3. Determine k' , the number of initial set of candidates for acquiring k-NN, depending on D
4. $\{p_1, \dots, p_{k'}\} = \text{Euclidean-NN}(q, k')$ // k' is close to k
5. for each p_i
6. $dN(q, p_i) = \text{compute\_network\_dist}(q, e(n_i, n_j), p_i)$
7. Sort $\{p_1, \dots, p_{k'}\}$ in ascending order of $dN(q, p_i)$ and make it as initial candidate result set
8. $\text{max\_range} = dN(q, p_{k'})$
9. $PE = \text{Euclidean-range}(q, \text{max\_range})$
10. for each point p in PE {
11. if (p has not calculated) {
12. $dN(q, p) = \text{compute\_network\_dist}(q, e(n_i, n_j), p)$
13. if ($dN(q, p) < \text{max\_range}$) Update candidate result set
14. } } //end of for
15. } else { $d_{\text{max}} = \infty$
16. $e(n_i, n_j) = \text{find\_edge}(q)$
17. $Q = \langle (n_i, dN(q, n_i)), (n_j, dN(q, n_j)) \rangle$ // Q is queue sorted by distance
18. delete from Q the node n with the smallest $dN(q, n)$
19. while($dN(q, n) < d_{\text{max}}$) {
20. for each non-visited adjacent node n_j of n {
21. $SP = \text{find-point}(e(n_j, n))$
22. for each point p in SP {
23. $dN(q, p) = \text{compute\_network\_dist}(q, e(n_i, n_j), p)$
24. if($dN(q, p) < d_{\text{max}}$)
25. update new $\{p_1, \dots, p_k\}$ by deleting old p_k and repositioning p by distance
26. insert $\langle n_j, dN(q, p_j) \rangle$ into Q
27. $d_{\text{max}} = dN(q, p_k)$ } //end of if //end of for
28. } // end of for each non-visited adjacent node
29. delete from Q the next node n with the smallest $dN(q, n)$
30. } /* end of while

End K-NN

Fig. 4. Our k-NN query processing algorithm

To satisfy the above consideration, we are required to build the initial set of candidates k' for k nearest neighbors in an effective way, in accordance with the density of POIs in the spatial network. A density (D) can be calculated the number of POIs within a circle whose radius is the Euclidean distance of the k -th nearest neighbor. If D is greater than a thresh-hold value given through the experiment, our k -NN query processing algorithm generally builds the initial set of candidates k' being close to k , and then adopts the Euclidean restriction technique. It takes relatively little time to compute the network distance between a query and the candidate POIs made from the Euclidean restriction technique. This is because our node-node matrix file makes it possible to obtain the shortest network distance between any pairs of nodes directly, thus remarkably reducing the network distance computation time and the number of disk I/Os accesses for visiting nodes. Otherwise, our k -NN query processing algorithm adopts the network expansion technique. Because the POIs are not dense in the spatial network, it is more efficient to find k nearest neighbors by expanding nodes from ones incident an edge covering a query. Because we obtain the initial set of candidates k' depending on the density of POIs, it is possible to acquire an initial set of near-optimal candidates, regardless the characteristic of the underlying road network (a city center area or a mountain area). We propose a new k -NN query processing algorithm to satisfy the above three considerations, as shown as Figure 4.

4 Performance Analysis

For our experiment, we make use of a road network consisting of 170,000 nodes and 220,000 edges [11]. We generate 10,846 points of interest (POIs) randomly on the road network by using RunTime21 algorithm [12]. We implement our range and k -NN query processing algorithms under Pentium-IV 2.0GHz CPU with 1GB main memory, running Window 2003. We also use 229G node-node matrix file. For performance analysis, we compare our range query processing algorithm with RER and RNE because RER and RNE are considered as efficient range query processing algorithms [1]. Similarly, we compare our k -NN query processing algorithm with IER and INE because IER and INE are considered as efficient k -NN query processing algorithms [1]. We measure a time for answering a range query whose radius r is between 0.001 and 0.05. Figure 5 shows the range query processing times. The RER, RNE and our range query processing algorithm require about 0.36, 0.17, and 0.10 seconds, respectively, when $r = 0.005$. It is shown that our range algorithm achieves 260% and 70% better performance than RER and RNE, respectively. In addition, the RER, RNE and our range query processing algorithm require about 21, 15, and 0.8 seconds, respectively, when $r = 0.05$. It is shown that our algorithm achieves over one of magnitude better performance than RER and RNE, and the performance improvement of our algorithm over RER and RNE is increased as the range r is increased. This is because our range query processing algorithm can reduce the network distance computation time and the number of disk I/Os required for accessing nodes by using our node-node matrix file.

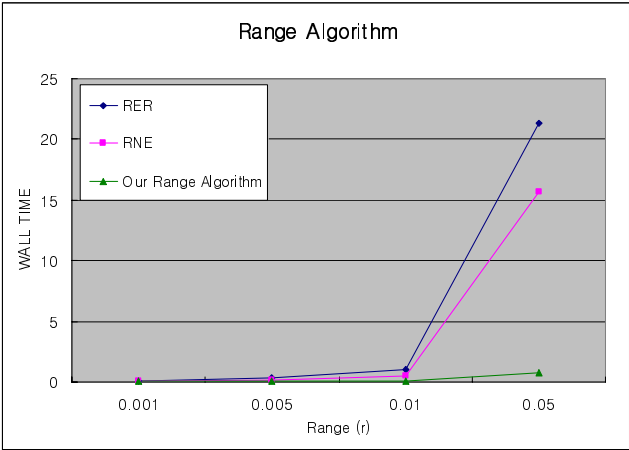


Fig. 5. Range query processing time

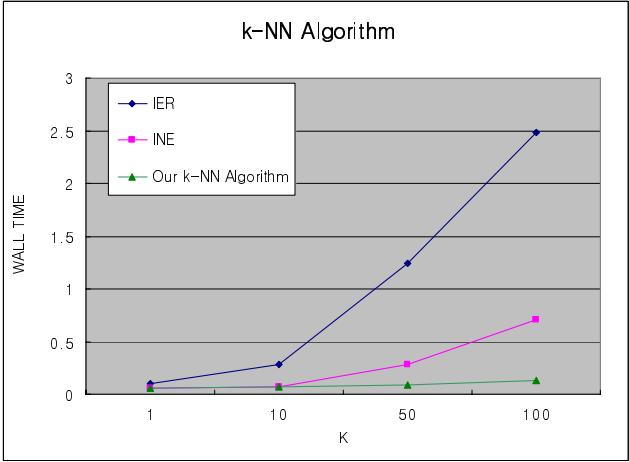


Fig. 6. k-NN query processing time

Figure 6 shows the k-NN query processing times. The IER, INE and our k-NN query processing algorithm require about 0.10, 0.05, and 0.05 seconds, respectively, when $k = 1$. When $k = 10$, the IER, INE and our k-NN algorithm require about 0.27, 0.07 and 0.07 seconds, respectively. It is shown that our k-NN algorithm is nearly the same performance as INE while it shows about 100~300% better performance than IER. When $k=50$, the IER, INE and our k-NN algorithm require about 1.24, 0.27, and 0.10 seconds, respectively. Thus, it is shown that our k-NN algorithm achieves about 170% performance improvements over the INE while it shows over one order of magnitude better performance than IER. This is because our k-NN query processing algorithm can reduce the computation time of network distances between a pair of nodes met during network expansion and the number of disk I/Os required for accessing nodes by using our node-node matrix file.

5 Conclusions and Future Work

In this paper, we designed the architecture of disk-based data structures for SNDB. Based on this architecture, we proposed new query processing algorithms for range and k-NN searches, depending on the density of point of interests in the spatial network. In addition, our query processing algorithms can reduce the computation time of network distances between a pair of nodes and the number of disk I/Os required for accessing nodes by using our node-node matrix file containing the shortest network distances of all the nodes in the spatial network. It was shown that our range query processing algorithm achieved about up to one order of magnitude better performance than RER and RNE. In addition, it was shown that our k-NN query processing algorithm achieved about up to 170~400% performance improvements over the INE while it showed about up to one order of magnitude better performance than IER. As future work, based on our range and k-NN query processing algorithms, it is required to study on e-distance join and closest-pairs query processing algorithms for SNDB.

References

1. D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query Processing in Spatial Network Databases" Proc. of VLDB, pp. 802-813, 2003.
2. S. Shekhar et al., "Spatial Databases – Accomplishments and Research Needs," IEEE Tran. on Knowledge and Data Engineering, Vol. 11, No. 1, pp 45-55, 1999.
3. L. Speicys, C.S. Jensen, and A. Kligys, "Computational Data Modeling for Network-Constrained Moving Objects," Proc. of ACM GIS, pp 118-125, 2003.
4. D. Pfoser and C.S. Jensen, "Indexing of Network Constrained Moving Objects," Proc. of ACM GIS, pp 25-32, 2003.
5. C.S. Jensen, J. Koláf, T.B. Pedersen, and I. Timko, "Nearest Neighbor Queries in Road Networks," Proc. of ACM GIS, pp 1-8, 2003.
6. M. Kolahdouzan and C. Shahabi, "Voronoi-Based K Nearest Neighbor Querying in Spatial Networks," Proc. of VLDB, pp. 840-851, 2004.
7. X. Huang, C.S. Jensen, and S. Saltenis, "The islands Approach to Nearest Neighbors Search for Spatial Network Databases,," Proc. of SSTD, pp 73-90, 2005.
8. T. Seidl, N. Roussopoulos, and C. Faloutsos, "The R+-tree: a Dynamic Index for Multi-Dimensional Objects, Proc. of VLDB, 1987.
9. N. Jing, Y.-W. Huang, and E.A. Rundensteiner, "Hierarchical Encoded Path Views for Path Query Processing: An Optimal Model and Its Performance Evaluation," IEEE Tran. on Knowledge and data Engineering, Vol. 10, No. 3, pp 409-432, 1998.
10. T. Seidl and H. Kriegel, "Optimal Multi-step k-Nearest Neighbor Search, Proc. of ACM SIGMOD, 1998.
11. <http://www.fh-oow.de/institute/iapg/personen/brinkhoff/generator/>, San Francisco Bay map
12. T. Brinkhoff, "A Framework for Generating Network-Based Moving Objects," GeoInformatica, Vol. 6, No. 2, pp 153-180, 2002.

A ONCE-Updating Approach on Moving Objects

Hoang Do Thanh Tung and Keun Ho Ryu*

Database and Bioinformatics laboratory, Chungbuk National University
{tunghdt, khryu}@dblab.chungbuk.ac.kr

Abstract. Indexing methods that manage current states of moving objects in a database are confronted with keeping their indexes up to date. Numerous objects that move continuously can invoke many their updates at a time constantly. R-tree based indexing methods promise to be efficient and easy to be implemented and deployed to real applications. Nevertheless, recent updating improvements for R-tree have been not either designed or appropriate to a moving object environment. Therefore, in this paper, we propose a ONCE-Updating approach applied for indexing methods developed from the R-tree. The idea is that instead of incurring the high cost of updating an indexed data on many new states at a timestamp one by one, an indexing method can easily update the data on all the states just once. In our experiments, in addition to much improving update performance, the approach even increases quality of queries extracted from indexed data. This approach can be used to support methods managing data from traffic surveillance systems.

1 Introduction

Nowadays, we are all witnessing marvelous development of various positioning technologies, such as a Global Positioning System (GPS) using a network of satellites to provide excellent receiver's position accuracy, as a sensor network using computer network of many spatially distributed devices using sensors to monitor conditions at different locations. Using these technologies for the ability to pinpoint the location of a mobile caller or vehicle in transit, generally called Moving Objects (MOs), location-based services (LBS) have been being developed to be used for emergency purposes as well as enhanced business applications such as location-sensitive billing, traffic surveillance, digital battlefields, mobile e-commerce, and people tracking, etc. These technologies are expected to become ubiquitous.

Along with growing positioning technologies and their applications/services, developing database systems capable of managing MO's position data is inevitable and on demand. The data stored in a database should be indexed so well that satisfying requirements of accessing and querying from ad hoc applications or specific users. However, much more difficult than an indexing method supporting static data, such a moving object indexing method must not only well arrange constantly changing data but also keep up with the latest states of numerous MOs. Although there are so far several methods proposed to index MO data, many of them are possibly challenged

* Corresponding author.

by their low update performance due to the fact that they are almost developed from spatial approach that inherently supporting static object data.

Perhaps the most desirous of accelerating update performance, the methods are developed from R-tree in order to manage current states of MOs. Those methods are very promising because easily implemented and flexible to naturally random distribution of MOs. Despite that, they still update an index in the manner of each object at a time. Intuitively, if current information is not up to date, the current and future queries will be obviously helpless. From the fact, our goal is to develop a new updating approach that is practically efficient to moving object management.

We found out an update's principle deriving from Dirichlet's Box Principle applied to moving objects as follows.

Update's principle:

Assume that during a time period P composed of k timestamps $[p_1, p_2, \dots, p_k]$, we manage n moving objects each of which must invoke at least one update. Thus, at least one timestamp we receive at least $\lceil n/k \rceil$ updates.

For example, 1000 cars are running in 30 minutes in a city. If each of them generates an update, at least one minute, there exists at least 160 cars invoke updates concurrently. In fact, the number of updates at a timestamp can be very large. Referring to a real survey for Seoul city, the daily average number of vehicles passing through downtown is about 1.13 million, and that of vehicles using main roads in the city is about 2.71 million in 2003. Motivated from the principle, in this paper, we propose a ONCE-Updating approach applied for indexing methods developed from the R-tree. The idea is that instead of incurring the high cost of updating an index on many new states at a timestamp one by one, an indexing method can easily update the index on all the states just once. In our experiments, addition to dramatically improving update performance, the approach even increases quality of queries extracted from indexed data.

The rest of the paper is organized as follows. Section 2 reviews important methods proposed to improve update performance. Section 3 gives the idea of ONCE-updating approach. Section 4 reports on performance experiments. We conclude in Section 5 by summarizing our work and suggesting further direction.

2 Related Works

As we all know, the R-tree is a hierarchical, height-balanced external memory data structure proposed by Gutman in [2]. It is a generalization of the B-tree for multidimensional spaces. R-tree is a method of data driven approach so that is very flexible to various data distributions. As a result, many well-known MO indexing methods were developed from R-tree. However, the weakness of the R-tree is update performance because each R-tree's update requires a process of a deletion for the obsolete state and then an insertion for the new state in top-down manner. Despite that, only some group-insertion approaches were proposed for pure R-tree for static objects as follows.

First, the GBI approach was proposed by Rupesh Choubey et al. [4]. This approach only improves insertion of R-tree. Instead of inserting all objects on-by-one into an

R-tree, the approach first uses the K-Mean method to cluster new (skewed) dataset, and then from each cluster builds a new R-tree (small tree), finally inserts every the new tree as a data object into an already existing R-tree (large tree). Although this approach substantially improves insertion, the quality of tree highly depends on new datasets supposed to be skewed. Practically, a new dataset is mostly much less than the main dataset so that it is possibly sparse rather than skewed. Moreover, the small tree actually has no connection to the large tree so that the combination of both trees must be worse than the large tree.

Secondly, similar to Rupesh Choubey's, an approach was proposed by Taewon Lee et al. [5]. The difference is that instead of using any clustering algorithm, this approach uses a seed tree for clustering a new data set. A seed tree is constructed by copying the top k levels of the target R-tree into main memory. If any object lies outside the leaves of the seed tree, it will be inserted individually by tradition R-tree insertion. The approach improves loose relationship problem between small trees and a large tree. Nevertheless, it is really difficult to determine value of K to build a seed tree in memory. If K is small, this approach will be nearly the same as Rupesh Choubey's or even worse because the new dataset is no carefully clustered. If K is large, cost of memory will be high and additionally many objects possibly fall outside top- K -level so that many individual insertions occur.

As a conclusion, above approaches significantly depend on a new dataset distribution that is not appropriate to various distributions of moving objects. Furthermore, to improve update performance, they possibly deteriorate the quality of R-tree because of loose relationship between the small tree and the large tree. In other words, entire nodes of both these trees do not guarantee sum of either MBR sizes or MBR overlaps to be minimal. In spite of that, they are not actually designed for moving object methods. In the next section, we are showing our proposed approach that improves both update performance and quality of the tree of MOs.

3 ONCE-Updating Approach

Our goal in this section is to design a new approach flexible and slightly dependent on assumptions. The approach approves requirements as follows

- It supports both deletion and insertion process.
- It generates just one update process for all new states at a timestamp. In other words, it tries to access a disk block at most once in a process.
- Besides improving update performance, it must not deteriorate the quality of a tree.
- It is not dependent on a specific type of new data distribution.
- The capacity of main memory used in algorithm is not significant and able to be estimated.

We assume that the number states of the new dataset are much less than the total number of objects. They are concurrently generated at a timestamp. Therefore, the entire new dataset is feasible to be kept in main memory. Each update normally requires deleting the obsolete state and then inserting the new one. Therefore, an update process is composed of two sub processes, deleting all obsolete states once and inserting all new states once.

3.1 Deletion Process

In deletion process, we also do not generate deletion from a root. All obsolete states will be deleted from the leaf nodes containing them directly. To access to leaf nodes directly, we add a leaf block address property to an *object information table* in a database that normally keep only the latest states of all objects, see fig.1. Practically, most indexing methods must use the table to look for any state being obsolete when it's necessary. We also use a main memory *parent_of* array containing block addresses of internal nodes. The declaration of this array in C++ is

```
int *parent_of = new int[num_of_nodes];
```

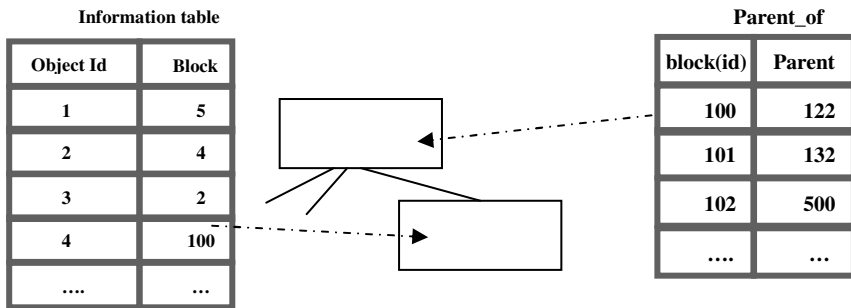


Fig. 1. Using information table to seek a leaf, and Parent_of to seek the parent

In fig.1, block(id) property of Parent_of table is array id property. In fact, this array's space is much smaller than that of the associated R-tree. Given a 4 KB page with a fanout of 204, the parent_of's space is just about 0.00097 of that of the associated R-tree in case that integer's space is 4 bytes.

This deletion process will delete all obsolete states from leaves and adjust tree by using parent_of array in case that the deletion makes the leaf node's size change. If a leaf node is underflow, the process will not reinsert its entries immediately. Instead, it moves them into a stack in main memory for being inserted together with Insertion process. If an internal node is underflow, the process will reinsert immediately by traditional insertion. Experimentally, reinsertion of internal nodes was rare. This process is considered as a preparation step for insertion process. Admittedly, a group insertion finds it hard to make a tree as good as one-by-one traditional insertion of R-tree do in general. Hence, the role of this process is technically to make the tree better at the end of process because of bottom-up adjustments. Bottom-up adjustments possibly make a tree better by either reinserting its entries or simply re-compacting all MBRs that this process went over.

3.2 Insertion Process

Insertion process is invoked after the deletion process. This process is composed of two main steps, choosing subtrees for all new states and building new subtrees.

Assuming that the dataset of new states is possibly kept in main memory, we store them by a linked list, called *input_list*, which is very useful for sharing data items in a main memory. The code is as follows.

```

Bulk_Insert(new_dataset, node N, subtree st)
Input: input_list to be inserted
Output: A tree fully inserted

to_entries  $\leftarrow$  chooseSubtrees(new_dataset)
If (level > bulk_level)
    For each entry e such that entries[e] not empty
        Bulk_insert(to_entries[i], e.Child_node, st)
    If st.level < level then
        st = build_subtree(e, st)
If (level == bulk_level)
    For each entry e such that entries[e] not empty
        st = build_subtree(e, entries[e])

```

to_entries is a linked list array. Each element of *to_entries* is a sublist of new states. We should keep in mind that because *new_dataset* is a kind of pointer list, moving new states to sublists or nodes in *build_subtree* is not to increase memory occupation.

choosesubtrees

As we want to insert all new states just once, the former step uses a *choosesubtrees* algorithm. Not choosing only one subtree as in the traditional algorithm, the *choose-subtrees* chooses a number of subtrees and is responsible of splitting *input_list* into sublists each of which subsequently will be inserted into a chosen subtree. However, the algorithm to pick a new state to a sublist makes use of the traditional *choosesubtree* algorithm, see below *chooseSubtrees* algorithm. This step is recursively applied at each node selected top down and stops at a level, called *bulk_level*. In this paper, we assume that the *bulk_level* is 2. Although buffering higher level subtree as a based subtree gains better update performance, it possibly spends M^{k-1} node buffers in a main memory in the worst case, k is a *bulk_level*. The problem to manage using such a large memory will be solved in our other paper. At the node with level equal to *bulk_level*, the later step invokes a *build_subtree* algorithm to build one new subtree for every sublist based on a node's subtree as a based one. The new subtree can be inserted from current node in bottom up manner if generated.

build_subtree

To avoid the new subtree loosely relevant to the main tree, the algorithm builds it on the foundation of a subtree pointed by an entry in the node at *bulk_level*. The entry is

also selected by the choosesubtrees. A based subtree is buffered in main memory (MM). However, technically, only node accessed need buffered. In case `bulk_level` is 2, the subtree has only two levels so that the maximum number of nodes buffered is `fanout × block_capacity`. In our experiment, the subtree maximally occupies 30k with block's size 1k, and 120k with block's size 4k. Especially, the height of a subtree is independent on height of the main tree, whereas in Taewon Lee's approach, a higher main tree possibly requires a higher seed tree to keep clustering quality.

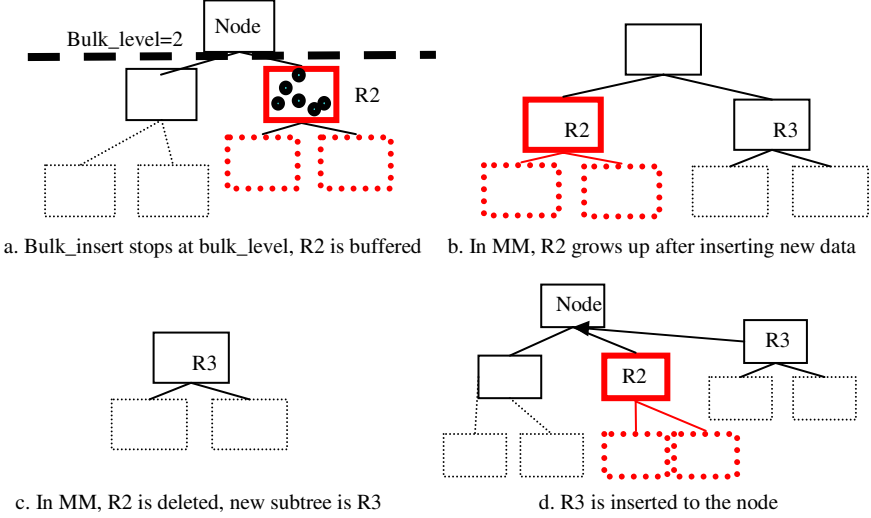


Fig. 2. An example of building a new subtree

Given a based tree, a sublist of new states is inserted into it. If the consequent subtree grows up, the initial based subtree will be split off it. Remaining is the new subtree we need. Finally the new subtree will be inserted to the main tree at current node in bottom up manner. We should keep in mind that a new subtree may not be generated if the consequent subtree is as high as the based tree. The fig.2 is an example of building a new subtree. After inserted new states, in main memory, the consequent subtree grew up. R2 must be deleted. The new subtree remaining is R3. Consequently, the new subtree R3 will be inserted to the tree from Node. Deleting R2 is to ensure not to affect the main tree.

3.3 A Variant of Updating Approach

Until now, there may be the Taewon Lee's group insertion approach [5] most efficient to R-tree. However, as we know, this approach only enhances insertion performance of a pure R-tree. Therefore, to compare with our approach, we combine it with our deletion process. We allows a seed tree to copy top-K levels of the main tree into main memory such that $K = \text{height (of main tree)} - 2$, namely a new dataset will be clustered by nodes at level 1 of the main tree. The value of K is fixed for any size of a

dataset. Because our datasets will be not too large, the occupation of a seed tree is acceptable in main memory. At level 1, new dataset will be cluster most effectively.

4 Experiments

We implemented our approach on the TPR-tree [3] that extends the R*-tree [11]. TPR-tree employs motion function rectangles replacing static rectangles of the R*-tree. The experiments were performed on a Pentium IV 2.8G machine with 512MB memory, running Windows XP. For all experiments, the disk page size is set to 1k bytes, and the maximum number of entries in a node is 27 for all indexes. Our goal is to compare update and query performance among a ONCE updating approach using bulk_insertion, named DS (dependent subtree) update, a variant updating approach using Taewon Lee's group insertion, named IS (independent subtree) update, and the original TPR-tree update. The IS update clustered new dataset at level 1, parent's level of leaves. Without real spatio-temporal dataset, our dataset simulates a traffic area of moving cars. We use a real spatial dataset LA or LB [7] to initiate positions and directions of MOs at timestamp 0. Thus, we assume that moving cars/objects randomly distributed in 2d area where each axis of the area is normalized to [0, 10000 meters]. Then, every object is associated with a VBR such that (a) object does not change spatial extents during its movement, (b) every car can run at most of about 90km/h, velocity can be positive or negative as two opposite directions with equal probability. A time unit is 3 seconds. At each time unit, every object can randomly change its speed, direction if it reaches an intersection, or even nothing.

Each query q has three parameters: $q_R.len$, $q_V.len$, and $q_T.len$, such that (a) its MBR q_R is a spare, with length $q_R.len$, uniformly generated in the data space, (b) its VBR is $q_V = \{-q_V.len, q_V.len, -q_V.len, q_V.len\}$, (c) its query interval is $q_T = [0, q_T.len]$. Query cost is measured as the average number of node accesses in executing a workload of 200 different queries with the same parameters. All of queries are moving windows.

4.1 Comparing Performance at Different Timestamps

In this experiment, we examined performance on the dataset 10,000 cars. We randomly generated updates at timestamps from 1 to 4 with two update rates 1% and 5%.

We forced MOs in new dataset to randomly change their speeds or directions if they reach intersections. The results in fig.3 show that update performance of DS and IS significantly outperforms than that of TPR update. However, the DS update is always better than IS update because IS update might incur the cost of reinserting outside-falling new states. Query performance of DS is always best of all kinds of update. Moreover, the performance is not susceptible to rates of update. Meanwhile, Query performance of IS becomes worst in (d) clearly, even the worse when the later timestamp. The reason is that though the IS clustered a new dataset at the deepest internal nodes, relationship between small trees and the large tree is still loose that makes new nodes weakly relational to nodes in the main tree. Obviously, if the number of update processes increase and the number of new states for every process are large enough, the number of weakly relational nodes will also increase. Consequently, that deteriorates query performance more and more.

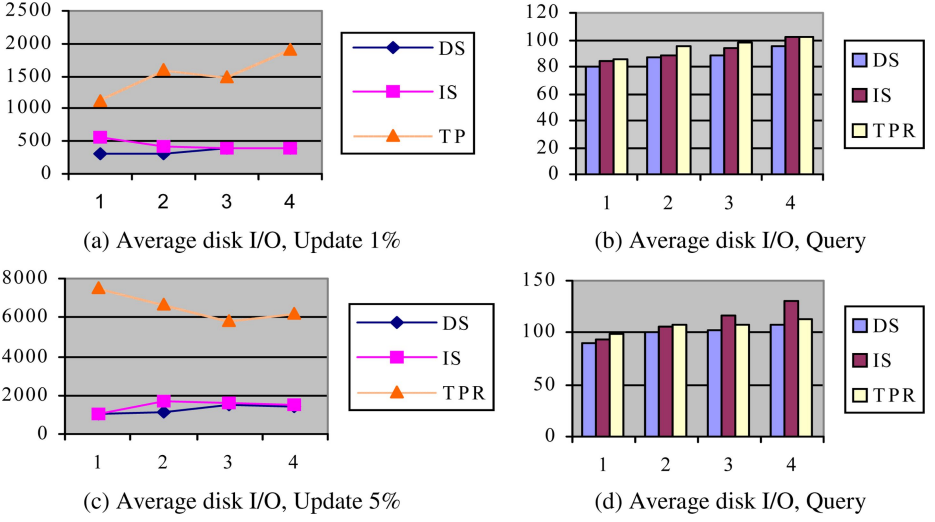


Fig. 3. Experiment on dataset 10k at timestamp 1-4 with two update rates 1%, 5%

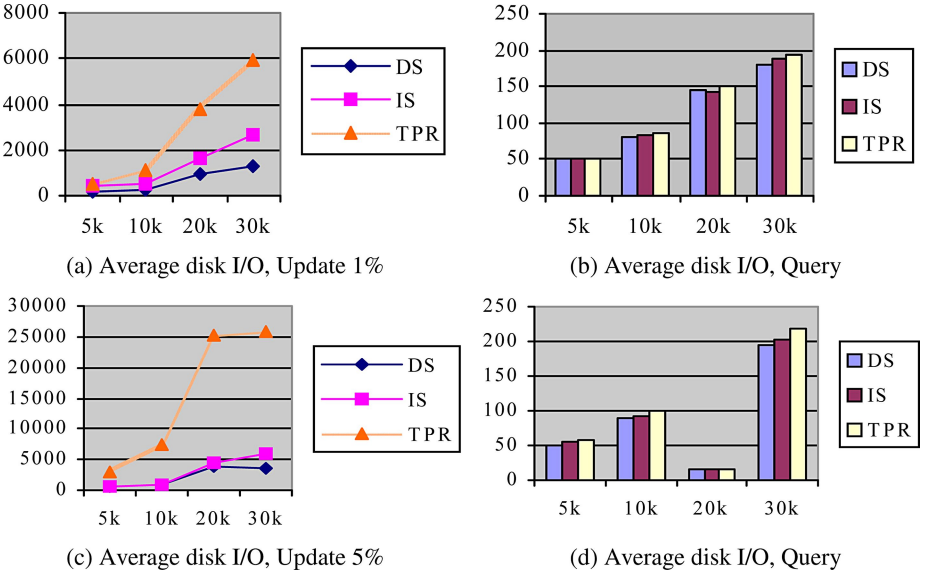


Fig. 4. Experiment on dataset 5k, 10k, 20k, 30k at timestamp 1 with two update rates 1%, 5%

4.2 Comparing Performance with Different Dataset Sizes

In this experiment, we examined performance on various dataset sizes 5k, 10k, 20k, 30k cars. We randomly generated updates with two update rates 1%, 5% at timestamp

1, shown in fig.4. See (a) and (c), that the update performance of IS is worse than that of DS is more clear with larger sizes of datasets. The reason is that IS is a kind of group update approach, but it still incurs cost of some top-down insertions of new subtrees as well as possible cost of individual top-down insertion of outliers of clusters. As a result, the larger dataset's size, higher cost of top-down insertion. The difference between DS and IS in (c) is not clear because TPR-tree becomes much too bad. See (b) and (d), DS's are better than TPR's. IS's queries are too because at timestamp 1, the number of weakly relational nodes are still few. However, see a result (b) of dataset's size 20k, it shows that query performance of IS is slightly better than that of DS. The reason is that perhaps in this case, the small trees must have been incidentally fit to the large tree.

In above experiments, we recognize that the update and query performance of DS or ONCE updating approach is best and most stable. The reason is that this approach keeps strong relationship between the main tree and subtrees in the insertion process to help maintain quality of the tree, and makes good preparation in the deletion process to help improve query performance. Update performance of IS including Taewon Lee's approach also gain good results but query performance becomes worse quickly after some update processes. This states that Taewon Lee's approach is not feasible to be applied to managing moving objects that are large amounts and change continuously over time.

5 Conclusions

Motivated from the demand of improving update performance for moving object indexing methods, in this paper we proposed an updating approach supporting R-tree based indexing methods that managing current states of moving objects. This approach satisfied that (1) it supports both deletion and insertion process, (2) it generates just one update process for all new states at a timestamp. In other words, it tries to access a disk block at most once in a process, (3) it improves both update performance and the quality of a tree, (4) it is not dependent on a specific type of new data distribution because created subtrees are strong relational to the large tree, (5) the occupation of main memory used in this approach is at most $4 \times M^{\log_m n} + M \times b$ bytes, M is maximal fanout of a node, b is node capacity, n is the number of moving objects, m is minimal fanout of a node, $4 \times M^{\log_m n}$ is of the parent_of array, $M \times b$ is of a subtree. This occupation is small and estimated. In the experiments, this approach proves to outperform TPR-tree's and Taewon Lee's cooperating with our deletion process. We strongly believe that this approach can well support other methods developed from R-tree method.

In future, a method able to manage large subtrees in MM efficiently can be our research. We hope that this approach can replace traditional updating approach of indexing methods managing moving objects generally as well as applications surveying traffic systems specifically.

Acknowledgement

This work was supported by the RRC program of MOCIE and ITEP.

References

1. Lee, E.J., Ryu, K.H., Nam, K.W. Indexing for Efficient Managing Current and Past Trajectory of Moving Object. *Apweb*, Hangzhou (2004) 781-787.
2. Guttman, A. R-Trees: A Dynamic Index Structure for Spatial Searching. *Proc. ACM SIGMOD*, pp. 47-57, June 1984.
3. S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez. Indexing the Positions of Continuously Moving Objects. In *Proceedings of SIGMOD*, 2000.
4. Choubey, R., Chen, L. and Rundensteiner, E. A. GBI: A Generalized R-Tree Bulk-Insertion Strategy. *International Symposium on Spatial Databases (SSD'99)*, Hong Kong, July 1999, pp 91--108
5. Taewon Lee, Bongki Moon and Sukho Lee. Bulk Insertion for R-tree by Seeded Clustering. *Proceedings of the 14th International Conference on Database and Expert Systems Applications (DEXA'2003)*, pages 129-138, Prague, Czech Republic, September 2003.
6. N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An Efficient and Robust Access Method For Points and Rectangles. In *Proc. ACM-SIGMOD International Conference on Management of Data*, pages 322-331, Atlantic City, May 1990.
7. [Http://www.census.gov/geo/www/tiger/](http://www.census.gov/geo/www/tiger/)
8. Hoang Do Thanh Tung, Young Jin Jung, Eung Jae Lee, Keun Ho Ryu. *Moving Point Indexing for Future Location Query*. *Lecture Notes in Computer Science*, Volume 3289, Springer-Verlag, 2004.
9. Jung, Y.J., Lee, E.J., Ryu, K.H. MP-tree: An Index Approach for Moving Objects in Mobile Environment. *ASGIS* (2003) 104-111.

A Progressive Transmission Scheme for Vector Maps in Low-Bandwidth Environments Based on Device Rendering

David Cavassana Costa¹, Anselmo Cardoso de Paiva¹, Mario Meireles Teixeira¹,
Cláudio de Souza Baptista², and Elvis Rodrigues da Silva²

¹ Departamento de Informática, Universidade Federal do Maranhão,
Avenida dos Portugueses SN, 65085-580 São Luis, Ma, Brasil
davidcavassana@yahoo.com.br, {paiva, mario}@deinf.ufma.br

² Computer Science Department, University of Campina Grande
Rua Aprigio Veloso, 882, 58107-970 Campina Grande, PB, Brazil
{baptista, elvis}@dsc.ufcg.edu.br

Abstract. The Internet has created an interesting environment for geospatial data sharing, so that users may transfer, visualize, manipulate and interact the data sets. This environment not only provides new opportunities to geospatial data usage, but also introduces new problems that should be addressed in order to provide an efficient and effective use of these datasets. One of such problems is related to the use of these spatial datasets in a low-bandwidth environment, such as those involving mobile computing. This paper presents a progressive transmission method for vector maps on the Web. The proposed method anticipates the map rasterization process to be performed at the server side, so that the amount of information transmitted may be reduced. We combine scale-dependent transmission techniques with simplification and progressive ones, in order to maximize the overall performance of a Web GIS environment.

1 Introduction

Visualization plays an important role in geographical phenomena understanding and analysis. For a long time, maps have been used for representing spatial data, aiming to clarify and understand their relationships.

Currently, the Internet is becoming the main vehicle for publishing geographical information, which enables data interchange, analysis and geographical data visualization [1], [2]. The rapid evolution on Web technology has lead to an improvement of the geographical information utilization and availability [3], [4], [5], [6], [7].

The Internet has created an interesting environment for geospatial data sharing, in which data providers make their databases available through the Web, and users may transfer, visualize, manipulate and interact with them [8]. This environment introduces new problems that must be addressed to make possible an efficient and effective use of these datasets. One of such problems is related to the availability of huge geospatial data amount in repositories with limited connection bandwidth [9].

Even though the access to large network bandwidths is becoming more popular, there are situations where the transmission is a critical factor, as in wireless networks.

These requirements have lead to a set of techniques devoted to the minimization of bandwidth requirements and better utilization of available resources, so that it may enable the publication of vector geospatial data in low bandwidth environments.

There are some classes of techniques employed to address the problem of efficient transmission of huge amount of geographical data, which include multiresolution, generalization and progressive transmission.

This paper presents a scheme based on the combination of generalization, content adaptation and progressive transmission techniques to make more efficient the map transmission process in a client-server environment. Our approach aims both to minimize the impacts in data consistency and to improve the application usability through a better user response time. Furthermore, we seek to deal with large vector maps transmission and visualization over the Internet.

The remainder of this paper is organized as follows: section 2 discusses related work on vector map transmission. Section 3 focuses on architectural issues. Section 4 presents our proposed progressive transmission scheme. Section 5 highlights the results obtained from the tests performed. Finally, section 6 concludes the paper and highlights further work to be undertaken.

2 Related Work

Several methods have been proposed to transmit spatial data in its vector format using a progressive process [1], [8], [10], [11], [12], [13]. Such methods are based on the idea that a typical user consumes the map information gradually from low levels of detail to more detailed scales. Based on this evidence, we can generate the map and transmit it gradually.

In the progressive transmission, the map server divides the map into a low resolution version and a set of incremental versions, that when incorporated to a certain map version will generate a more detailed map version. The client is responsible for receiving a map detail increment at some level n and for integrating it into the current map version which generates a map version at level $n+1$.

Other approaches to this problem explore device restrictions like visualization resolution. This can determine the level of detail that can be visualized in a specific device. In general, the device can represent less detail information than the amount that is sent by the map server. The level of detail that cannot be visualized increases the transmission cost and does not improve the quality of the map visualized by the client. Thus the elimination of such information reduces response time at the client side without compromising map quality [14].

Bertolloto and Engenhofer [15] have proposed an on-the-fly mapping generation for the Internet using a progressive transmission scheme. Their work creates multiple dataset representations corresponding to different levels of detail, which are sequentially transmitted and added to the map visualized on the client.

Harrie et al. [16] address the visualization of geographical data for mobile users. Methods to adapt data stored in the database to small-scale areas of the map in real time are described, using generalization procedures through a XSLT transformation.

Solutions based on map points simplification like the Douglas-Peucker Algorithm [17] and the method proposed by Buttenfield [10], do not take into account the resolution of the device where the map will be visualized. Thus, after the complete simplification is performed, we still maintain in the map the so-called spurious information that will not be visualized by the user in its initial low resolution map consumption. Thus, an approach focused on user device resolution is really needed.

Ferreira [18] suggests a five-step method for map size reduction, based on the idea later revisited by [14]. The simplification for visualization purposes must be done based on the visualization device resolution. In other words, we should transmit only what can be seen. The work presented in [14] discusses a set of simplification and transmission techniques, scale-dependent, which is used to minimize the response time to end users. Additionally, it presents a progressive transmission scheme that avoids the transmission of features that have not changed their details.

The work that is described in this paper follows the idea that we must transmit just what can be seen, as proposed by [14]. However, our solution is based on a simple approach, which regards low level map information, point coordinates and the visualization device resolution.

3 Architecture

In order to validate our vector transmission algorithm, we used iGIS (Internet Geographical Information System) [19]. iGIS is a framework based on a 3-tier architecture, which aims to implement a web-based spatial information system, following the OpenGeoSpatial standard. As a framework, iGIS enables rapid application development of Internet based geographical information systems.

The iGIS architecture was designed according to the Model-View-Controller (MVC) design pattern. iGIS may be deployed in several computers, as follows:

- A spatial database server;
- An application and presentation server;
- Clients running a Web browser.

Fig. 1 presents this architecture. In the presentation layer Java Server Pages –JSP – technology is used to implement dynamic pages. Moreover, SVG and JavaScript are used for map drawings and graphical tools.

The data layer is represented by a spatial database server instance, which is called data source. iGIS can deal with different data sources, as, for example, Oracle, Postgresql, IBM DB2 and ESRI shapefiles. This layer is in charge of database independent geographical data manipulation.

The application layer is responsible for the business logic. This layer handles client requests; it gets the required data from the data layer; then it converts them to the OGC model, and finally it formats them for a specific presentation layer instance. This layer is composed of the following modules: control, data loader, and data formatter.

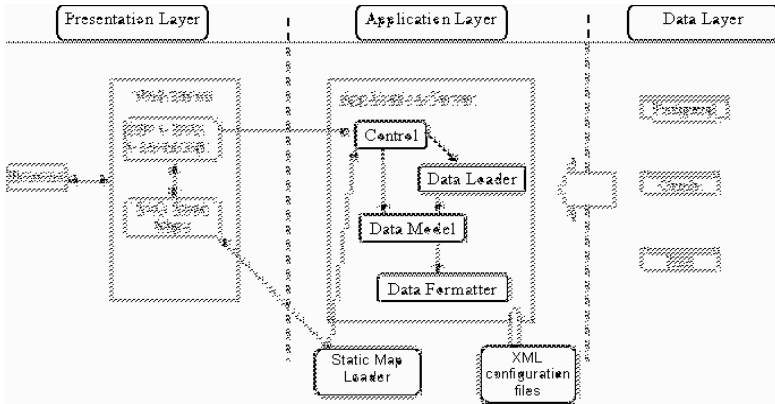


Fig. 1. iGIS Architecture [19]

The data layer is responsible for loading data from different data sources, which are configured by XML configuration files. The control module defines the action sequences which are necessary to satisfy the client request, and it returns to the presentation layer the data adequately formatted.

The formatter is the module that assures the independence between data manipulation and data presentation. The iGIS has already implemented vector and raster formatters.

As a framework, iGIS has some extension points. One of them is the map formatter. This extension point is useful for implementing a progressive transmission scheme, reusing the rest of the framework functionalities.

Thus, we implemented the progressive transmission scheme in the framework package formatter, generating a new sub-package named progressive.

4 The Progressive Transmission Algorithm Proposed

In order to provide, in the iGIS project, an efficient vector map transmission and visualization over the Internet, it is necessary to render maps in different levels of detail. Our approach is based on a progressive transmission of those maps, in which we consider the visualization device characteristics.

By using progressive transmission of maps, the user firstly starts using a low resolution map to locate the area of which he is interested in. Thus, by the time some visualization operations are performed, the user will view the area of interest on larger resolutions incrementally.

The main idea in our proposed algorithm is to take into account the target device resolution. Hence, only the map information that is strictly necessary to generate a correct visualization in that resolution is sent. This approach eliminates points and lines that are not visible.

In the rendering process map points are transformed from map coordinates to device coordinates. In this process some map points are transformed to the same pixel. The proposed algorithm anticipated this process by eliminating in a polygon or

polyline consecutive points that will be mapped to the same pixel when the map is going to be rendered.

In the proposed algorithm, the server initially sends a map generated for a low resolution, and as the user performs zooming operations, the visualization client requires a more detailed map. Then, the server sends only the incremental information which is needed for generating the next resolution level in the client, thus saving network bandwidth. We highlight that only the details that will be visible in the next target resolution will be sent. These details are then composed in the client using Javascript and DOM manipulation.

The dynamic generation of all of these different maps for each level of detail would overload the iGIS server. To avoid this overloading, we propose the static generation of all maps and incremental information.

In short, the main contribution of our work is the proposal of a method that anticipates the map rasterization process to be performed at the server side, so that the amount of information transmitted may be reduced. Indeed, we combine scale-dependent transmission techniques with simplification and progressive ones. This integration of different techniques aims to maximize the overall performance of a Web GIS environment.

The method may be divided into two main phases: the offline phase which consists of an offline preparation of maps and map details; and the interactive phase in which the client demands from the server for more level of detail of the current rendered map.

The offline phase starts by generating the map on the largest possible resolution. Thus we compute the points that are visible in the lowest resolution (level 0), eliminating the other ones. The eliminated point set is traversed to verify which points are visible in the next resolution, generating the details for the next resolution level. This process is repeated until we reach the largest resolution. This phase generates a low resolution map as a SVG file and some increment information made available as text files in a specific data format which is described later. The number of increment levels and the initial resolution are empirically defined for each map.

To generate a map suitable for visualization in a specific resolution we adapt the process proposed by Ferreira [18] using two phases: the coordinates quantization and relative coordinates codification.

Quantization is the process of reducing the size of map coordinates, mapping the coordinates to the space of positive integers Z_+ . At the end of this conversion, some consecutive points will be mapped to the same integer point generating zero-size segments. In this case, we keep just one of such coincident points and discard the others. This process reduces the number of points needed to represent a vector geographic feature in a map.

Relative coordinates express each point position in the polyline or the polygon based on the previous point position. The initial point is always expressed using absolute coordinates. This approach is already supported by the SVG specification.[20].

Hence, our method initially discretizes the map to the largest resolution, performing the coordinates quantization and relative coordinates coding. During this process we eliminate, for each map feature, the consecutive points that are mapped on the same pixel. This is the most detailed map version that is available for the client.

Then, we compute for each map feature its quantization to the lowest resolution, generating two lists of points: F^0 , which represents the points necessary to render the

feature at that resolution; and S , which represents the superfluous points for resolution 0, that we must avoid transmitting. This is done according to the following algorithm:

Map quantization to the lowest resolution, generating F^0 and S

1. Let previous $\leftarrow$ first point of F^n quantized for resolution 0
2. For each point p_j in F^n
 - $p^q \leftarrow$ point p_j quantized for resolution 0
 - if $p^q = \text{previous}$
 - include p_j in S
 - else
 - include p^q in F^0
 - previous = p^q

The algorithm proceeds in a loop that will compute, for each resolution r : the feature version F^r ; the points necessities to add to F^{r-1} to generate F^r included in Inc^r ; and it maintains in S the points from F^n that were removed to generate F^r . This may be expressed as follows:

Computes the increment points of F^{r-1} , necessities to reach the F^r , Inc^r

3. For each resolution r from 1 to n
 - $F^r = F^{r-1}$
4. For each point p_j in S , j from 1 to n_p
 - $p^q =$ point p_j quantized for the resolution r
 - previous = the previous point of p_j got from F^r
 - if $p^q \neq \text{previous}$
 - include p^q in F^r
 - include p^q in Inc^r
 - remove p_j from S

In order to insert the points into the client device incrementally, we add metadata to specify where each new point must be included. This is coded into the Inc^r array by adding a reference point index to indicate that the new point will be included after that reference point. Thus we generate a string like “p0 0 1 1 0 1 1 0 1”, where p0 indicates the reference point 0, and the following pairs of numbers indicate the relative coordinates for the new points that must be inserted.

At the client side, the device receives the array of points Inc^r that contains the points that must be added to the actual map version in order to generate the next one, and include them appropriately in the feature description.

5 Results

The main objective of this work is the minimization of transmission time for geographical map data. To assess the performance of the proposed progressive map transmission scheme we have performed some experiments to quantify the gains

obtained with our proposal. For the experiments we used the maps depicted in Fig. 2, as follows: Brazil1 (Brazilian counties - 18.7 MB); Brazil2 (hydrograph, roads, soil and state limits from Brazil - 32.1MB) , Amazon, (land use and other themes from the Amazon region.- 9.3MB).



Fig. 2. Maps Brazil1, Brazil2 and Amazon region visualized in iGIS

Table 1. Test results using the Brazil1 map

R	TS	QMS	TS/OMS	QMS/OMS	TS/QMS
300x300	1427	1427	7%	7%	100%
600x600	2147	2309	11%	12%	93%
1200x1200	2203	3596	12%	19%	61%
2400x2400	2012	5011	11%	26%	40%
4800x4800	1462	6196	8%	32%	24%
9600x9600	690	6670	4%	35%	10%
19200x19200	309	6728	2%	35%	5%

For each map resolution level (ML) we have the associated resolution (R); the amount of information transmitted to generate this resolution (TS); the size of a map quantized directly for a target resolution R (QMS); and the original map size (OMS) and some compression ratios. Tables 2, 3 and 4 show the parameters and results obtained for each of the used maps. The size measurements are depicted in kilobytes.

Table 2. Test results using the Brazil2 map

R	TS	QMS	TS/OMS	QMS/OMS	TS/QMS
600x600	1400	1400	15%	15%	100%
1200x1200	1353	2131	14%	22%	63%
2400x2400	1125	2797	12%	29%	40%
4800x4800	796	3271	8%	34%	24%
9600x9600	490	3593	5%	38%	14%
19200x19200	249	3722	3%	39%	7%
38400x38400	113	3709	1%	39%	3%

Table 3. Test results using the Amazon map

R	TS	QMS	TS/OMS	QMS/OMS	TS/QMS
800x540	7935	7935	24%	24%	100%
1600x1080	6118	11492	19%	35%	53%
3200x2160	4279	13902	13%	42%	31%
6400x4320	1921	14626	6%	44%	13%
12800x8640	1032	14770	3%	45%	7%
25600x17280	893	14818	3%	45%	6%
51200x34560	839	14699	3%	45%	6%

After evaluating the results obtained we may highlight that the amount of information to be transmitted for each user request consists of only a small percentage of the original map. This statement enables the use of large maps even over a low bandwidth network, which indicates the use of our approach in mobile devices. The amount of information that must be transmitted to the client will decrease as the resolution increases. We also have noticed that the resolution discretization must be performed in small sets for the low resolution, and it may become larger as the maximum resolution is reached.

By comparing the progressive transmission scheme with just sending the completely quantized map for each resolution, we can infer that the compression ratio between the increment and the quantized map is always less than 1. This evidence states that in the worst case both approaches have similar performance, however our proposed progressive scheme tends to achieve better performance. As an example, regarding the results presented in Table 2, if the maps for resolutions 0, 1, 2 e 3 were requested in sequence, we would transmit 7,789 kilobytes with the progressive transmission and 12,343 kilobytes if using only the quantization approach. As the user does not know in which resolution the information he might interested in is available, the exploratory visualization is efficiently handled by the proposed scheme.

Figure 3 presents the curves obtained for the compression ratio of all used maps related to the number of pixels (map display resolution). We have observed that the curves present a similar shape, with a maximum bandwidth need that progressively decreases as the user requests for more resolution. Furthermore, we have observed that the proposed scheme provides in each step the minimum compression ratio of 16%. This result evidences there exists an efficient bandwidth use.

When comparing our work to a similar one proposed by Liang [14], we may state that our simplification algorithm, which is based only in a rasterization process, is similar but simpler. The simplicity derives from discarding the use, at the client side, of explicit data structures to manage map upgrades, since only SVG is used for map representation and map processing is performed using JavaScript. Another important aspect is related to the fact that our proposal can assure that a point information is transmitted only once, conversely to Liang's work that sends a new element fully when it must be upgraded, discarding the low resolution information that had been previously transmitted.

We use an average time of 259 KB of real map size per second to generate the low resolution maps and all of its increments. For the increments integration the average time was approximately 15 KB/s, related to the TS. Concerning the Brazil1 map (RMS = 18,2Mb and TS1 = 1400Kb) the time generation was 72 seconds and the processing time at level 1 was 94 seconds. These results were obtained in a computer with a 2GHz Athlon64 processor and 1GB of RAM.

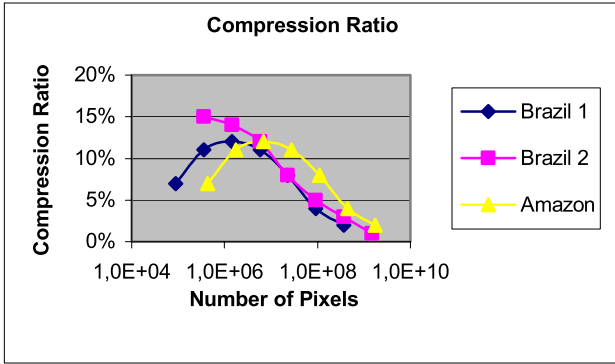


Fig. 3. Compression ratios comparison for the three maps used

6 Conclusion

This paper presents a progressive transmission scheme for the iGIS Web mapping framework. The proposed scheme enables the Internet GIS developer to plan a scale discretization for multiple levels of detail offline maps, providing the user application a more efficient environment to visualize and interact with the geographic data, especially when considering low bandwidth networks.

Our proposed scheme is based on the visualization device resolution. The map features are simplified and a compact map representation is generated aiming to improve response time. To evaluate the impact of our proposed scheme in the visualization task several experiments were performed. The experimental results have demonstrated that the amount of information transmitted, is significantly less in our proposed scheme, being therefore very effective for Web mapping systems.

As further work, we plan to incorporate in the same progressive transmission scheme the ability to work simultaneously with vector and raster maps, and we also intend to implement a GIS Web Service based on this progressive transmission approach.

References

- [1] Cecconi, A.: Integration of Cartographic Generalization and Multi-Scale Databases for Enhanced Web Mapping, PhD Thesis, Zurich University, 2003.
- [2] ESRI White Paper: Map Generalization in GIS: Practical Solutions with Workstation ArcInfo Software, ESRI Inc. July 2000.

- [3] ARCNEWS: Scalable Internet Mapping Solutions from ESRI: Maps on the New Frontier, ArcNews, Fall, 1997.
- [4] Callejas, L. M.: Web-Based GIS: A Brief History, Available on line at: <http://blaze.trentu.ca/~ermc/web-gis.html>.
- [5] Cho, Y.S., Kim, H.Y., Kim, J.H., Bae, H.Y.: The Design and Implementation of Componentized Web-enabled GIS, ICIMU 98, Sept. 1998.
- [6] ESRI White Paper, The Future of GIS on the Internet, ESRI Inc. June 1997.
- [7] Peng, Z.R.: An Assessment of the Development of Internet GIS, Proc. of the 1997 ESRI User Conference, 1997.
- [8] Bertolotto, M., Egenhofer, M.: Progressive Vector Transmission, ACMGIS'99, p.152-157.
- [9] Flewelling, D., Egenhofer, M.: Using digital spatial archives effectively. International Journal of Geographical Information Science, Vol 13(8). P.1-8, 1999.
- [10] Buttenfield, B.: Transmitting Vector Geospatial Data across the Internet. In: M. J. Egenhofer and D. M. Mark (editors), Second International Conference, GIScience, 2002, Lecture Notes in Computer Science, pp. 51-64, Berlin.
- [11] Oh, Y.: Advanced Progressive Transmission for Spatial Data in Web based GIS, The Second AEARU Workshop on Web Technology, Oct. 1999, pp.41-46.
- [12] Han, Q., Bertolotto, M.: A Multi-level Data Structure for Vector Maps, GIS'04, November 12-13, 2004, Washington, DC, USA, ACM 2004.
- [13] Yang, B.S., Purves, R.S., Weibel, R., 2004: Implementation of progressive transmission algorithms for vector map data in web-based visualization, International Archives of Photogrammetry and Remote Sensing, Istanbul, Turkey, July 12-23, 2004.
- [14] Liang, C., Lee, C. H., Lee, J. D., Bae, H. Y.: Scale-Dependent transmission of spatial vector data on the Internet. The 3rd International Conference on Information Integration and Web-based Applications & Services, Austria, 2001.
- [15] Bertolotto, M., Egenhofer, M.: Progressive Transmission of Vector Data over the World Wide Web. GeoInformatica 5(4):345-373, 2001
- [16] Harrie, L., Sarjakoski, T., Lehto, L.: A Variable-Scale Map for Small Display Cartography. In: Proceedings of the Joint International Symposium on Geospatial Theory, Processing and Applications, Ottawa, 2002.
- [17] Douglas, D., Peucker, T.: Algorithms for the reduction of the number of points required to represent a digitalized line or its caricature, The Canadian Cartographer, 10, 2, p. 112-122., 1973.
- [18] Ferreira, C. C. F.: A Study about Vector Files for Map Visualization in the Web. Master Thesis (in Portuguese) PUC, Rio de Janeiro, Brazil, 1998.
- [19] Baptista, C. S., Leite Jr., F. L., Silva, E. R., Paiva, A. C.: Using Open Source GIS in e-Government Applications, Lecture Notes in Computer Science, DEXA e-GOV 2004.
- [20] W3C : Scalable Vector Graphics (SVG) 1.1 Specification, <http://www.w3.org/TR/SVG11>, 2003.

Map2Share – A System Exploiting Metadata to Share Geographical Information

L. Paolino, M. Sebillo, G. Tortora, and G. Vitiello

Dipartimento di Matematica e Informatica,
Università di Salerno
Fisciano (SA) - Italy

{lpaolino, msebillo, tortora, gvitiello}@unisa.it

Abstract. A crucial aspect in GIS project development is the high cost that companies face to collect geographical data. On the other hand, GIS data are abundant and readily available on the Internet for either on-line viewing or direct downloading. We present a software tool which exploits emerging P2P technologies in order to retrieve what is already available on the Internet, thus reducing the data collection cost. In the proposed *Map2Share* system, two aspects are of special interest, namely the platform which manages access and communication over the network and the retrieval methods to discriminate geographical data. Access and communication management is ensured by the JXTA platform, which is a set of protocols allowing to develop interoperable, platform-independent and ubiquitous P2P architectures. As for the retrieval methods, a minimal set of metadata, on which users may perform efficient retrieval tasks, has been identified on the basis of the well-established ISO/FDIS 19115 and Dublin Core standards. The visual environment characterizing *Map2Share* is described showing some of its advanced search functionality for geographical data retrieval and sharing.

1 Introduction

One of the most important issues that have to be faced when a Geographical Information System (GIS) project is started, is to decide whether to collect data from external sources, when possible, or to build data from scratch. In case a GIS designer decides that to build data is of primary importance, he/she should also take into account the impact of this activity on the overall project costs. According to Longley et al., it is definitively the most consuming, tedious and expensive process which may account for 15-50% of a total cost of a GIS project and may reach up to 85% whenever staff costs are assumed to be fixed and are not used in budget accounting [2].

On the other hand, if one decides to obtain data from external sources, the Internet is today providing abundant and freely available GIS data for either on-line viewing or direct downloading. However, despite such availability, many of those data are underutilized because users do not know where to find them.

In this paper we describe a new application, named *Map2Share*, which provides users with an effective, simple and affordable method to search for geographical data

on a virtual network based on a Peer To Peer (P2P) architecture. The genesis of the name *Map2Share* has the twofold aim to recall the kind of data we want to exchange, namely maps, and data sharing activity which is realized by P2P. As a matter of fact, *Map2Share* allows users to retrieve spatial files by posing queries which involve metadata about content, file format, typology, and even administrative information. Moreover, on the basis of the user's knowledge about the domain of interest two different kinds of matching can be invoked. When the user has a deep knowledge he/she may pose very detailed queries in order to select just data which are strictly necessary. Differently, ordinary end users may formulate raw queries in order to obtain a wider view of all data which are reasonably close to their requirements. In both cases, results can be ordered by importance, downloaded and, successively, visualized together.

The *Map2Share* P2P engine has been developed using the JXTA technology [1], which follows P2P protocols as a different approach to discover and download data. The adoption of a P2P architecture for geographical data collection guarantees several properties in terms of robustness, simplicity, high capacity and bandwidth, which are very important features for geographical data management cost, especially for data whose resolution implies considerable sizes, such as satellite images.

The remainder of the paper is organized as follows. In Section 2, a presentation of the network architecture underlying *Map2Share* is depicted together with a description of a *Map2Share* peer. Section 3 explains the query formulation activity including metadata and communication data streams. A presentation of the *Map2Share* interface is illustrated in Section 4. Section 5 concludes the paper.

2 The *Map2Share* Architecture

The aim of this Section is to illustrate the whole architecture underlying *Map2Share*. In particular, the problem of communication is first faced, some network design choices are discussed and communication roles of network participants are also presented. Then, the architecture of a single peer is described, in terms of its modules and repositories, and communication streams occurring during query/reply phases.

2.1 The Network Architecture

In the world of networks there exist a lot of different architectures implementing P2P communications, which generally differ from each other in the number of services provided. However, they may be grouped in three main categories, that is hybrid, pure and super-peer architectures [3]. The architectural choice we have adopted with *Map2Share* is to implement a 2-redundant super-peer network which provides a good trade-off between the network robustness and the number of transmitted messages.

In order to implement the *Map2Share Virtual Network (MVN)* according to the previous choices, the JXTA platform has been used. It consists of a set of open and generalized protocols for P2P architectures which allow devices (computers, PDAs, cell-phones, etc.) connected to the Internet to communicate and collaborate as peers. The JXTA protocols are independent from the programming language and multiple implementations exist for different environments, which are named bindings.

Information about resources are represented and exchanged by *advertisements*, which are implemented as XML documents.

2.2 The Peer Architecture

Figure 1 shows the architecture of a *Map2Share* peer. It is divided into five components, integrated with the *JXTA platform*, namely, the *Query XML Translator*, the *Query Resolver*, the *Downloader/Uploader*, the *Visualizer* and the *Interface*.

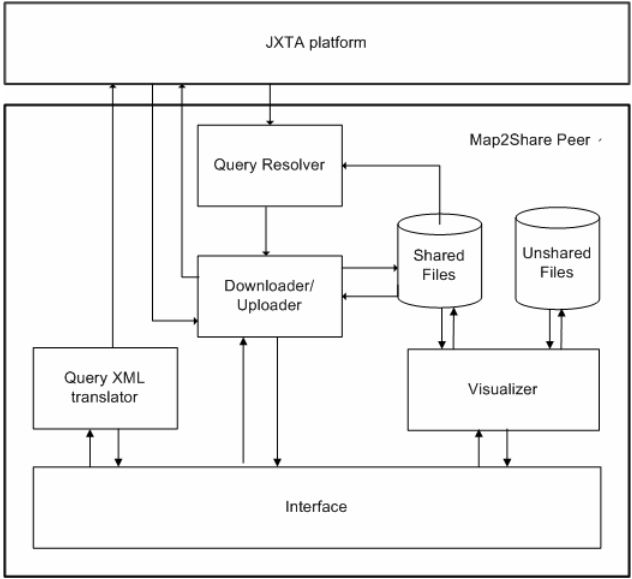


Fig. 1. The Map2Share Peer Architecture

Moreover, there are two repositories, named *Shared Repository* and *Unshared Repository*, respectively, which contain the files managed by the application.

The behavior of each component varies according to the peer working mode which may be either *client* or *server*. In the first case, the user is interested in posing queries and visualizing results. Queries are composed on the *Interface*. Here the user can specify patterns, specific attributes, and the type of query he/she is going to pose.

Once the query is composed, the *XML Query Translator* translates the filled forms into an XML representation which will be fed to the *JXTA platform*. According to the routing algorithm, the representation is forwarded to the Rs-peers of the peer group which the peer belongs to, in order to be broadcasted. Then, the peer goes in a waiting status ready to receive advertisements from other peers. As advertisements arrive, the *JXTA platform* sends them to the *Downloader/Uploader* module which stores and forwards them to the *Interface* in order to present results to the user, who may then decide to download the geographical resources described by the advertisement.

Once files have been downloaded, the user may decide to store them, either into the *Shared Repository* or into the *Unshared Repository*. The former contains all files the user would share with other users, the latter does not allow sharing. Anyway, both can be used to look for files to be visualized by the *Visualizer* module.

In the second case, when the peer works in a server mode, its behavior is only affected by the received request and no user interactions can influence it. Generally, requests are directly sent to the *Query Resolver* module which looks for matches in the *Shared Repository* and returns an XML representation of results that satisfy the request. In the next section, we give a more detailed description of the query management.

3 The Map2Share Queries

The main difference between *Map2Share* and other file sharing systems lies in the capability to perform specific queries on geographical data. Such queries are more flexible than conventional ones, where only a specified pattern matching is allowed. In *Map2Share*, users may exploit a *Basic* or an *Advanced* matching, which may be further divided into *Superficial* and *Deep* levels. The discrimination among those query typologies depends on the advertisement contents. An advertisement may contain a limited amount of information, which typically refers to general characteristics of data, such as filename and date. When more detailed information has to be managed, such as the feature names of a layer or the set of attributes describing them, it is no longer possible to use an advertisement, because too many values should be stored, and the advertisement would become too big.

The solution we have adopted in *Map2Share* is to provide two search levels, *Superficial* and *Deep*, which allow users to perform searches based only on the advertisements or on both the advertisements and the file contents.

Provided the opportunity to search through advertisements and/or file contents, *Map2Share* allows users also to pose queries where a specified pattern could be matched with respect either to a selected or to any attribute in the advertisement. In particular, a *Basic* matching looks for a pattern in any attribute, thus returning a possible superset satisfying the query. An *Advanced* matching aims to solve a correspondence between patterns and specific attributes. Thus, when users are properly informed about metadata structuring data, they can require a matching with implicit data properties, thus improving the effectiveness and the efficiency of the search task. Specifically, in case of a superficial level advanced search, an exact matching is searched with specified attributes in the advertisement only, whereas, in case of a deep level advanced search, the user can also look for matching within the file metadata and data, by specifying a textual pattern.

In order to allow *Map2Share* users to capture all the implicit properties of geographical data, it was necessary to characterize a set of basic attributes on which users may pose queries. To this aim, some standards describing metadata were considered, namely the ISO/FDIS 19115[5] and the Dublin Core[4]. By merging the basic elements of those standards, a set of 17 attributes were selected which correspond to metadata elements answering to the following questions: “Does a dataset on a specific topic exist (*what*)?”, “For a specific place (*where*)?” and “For a specific date or period

(when)?”. Table 1 reports the selected metadata with their description, and the corresponding domains are defined either as a free text or as a set of default values. In order to guarantee users with the capability to perform a *Basic* matching on every kind of available data, a set of such attributes are mandatory and have to be always associated with each file to share.

Table 1. Metadata for geographical data

Attribute	Description	Domain
Title *	Name by which the cited resource is known.	Free text
Keywords*	A topic of the resource content	Free text
Abstract	Brief summary of the content of the resource	Free text
Author*	An entity primarily responsible for making the content of the resource	Free text
Language	Language used within the dataset	Predefined values (ISO 639-2)
Spatial representation type	Method used to spatially represent geographic information (Vector, Raster, ...)	Predefined values
Format*	Description of the format of the data to be distributed (SHP, DWG, TIF, ...)	Predefined values
Geometry	Value indicating the geometric representation	Predefined values
Reference system type	Type of the reference system (Planar or Geographic)	Predefined values
Reference system*	Name of the reference system	Free text
Copyright	Information about rights held in and over the resource.	Free text
Date*	Reference date for the cited resource	YYYY/MM/DD (ISO 8601)
Scale *	Factor which provides a general understanding of the density of spatial data in the dataset.	Free text
Resource location	Location where the resource can be looked up	Free text
Geographic location of dataset	Extended information including the bounding box, bounding polygon, vertical, and temporal extent of the dataset	Free text
Responsabile party	Identification of, and means of communication with, person(s) and organization(s) associated with the resource	Free text
On-line resource	Information about (online) sources from which the resource can be obtained	Free text

In Table 1, attributes marked with a single asterisk represent the minimal set of metadata featuring advertisements, which have to be valued before sharing files through *Map2Share*. Such a task is performed by automatically extracting metadata from files and storing them in their corresponding advertisement using an XML

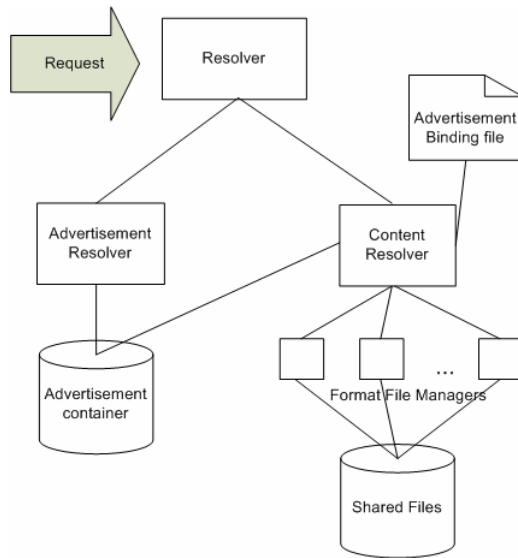


Fig. 2. Components managing requests

schema. When the automatic evaluation is impossible, then users providing data are requested to fill a form in order to be consistent. The modular architecture providing the two search levels is shown in Figure 2.

Based on this predefined framework, a user request is satisfied by comparing it with advertisements (and possibly with file contents) in order to verify whether and which files satisfy requirements. In particular, the *Resolver* verifies if the request asks for an advertisement (*Superficial* level) or also for a file content-based retrieval (*Deep* level).

In the first case, the request is only routed to the advertisement *Resolver* which examines the advertisement *Container* and returns those satisfying the request.

In the second case, the request is also sent to the *Content Resolver* which looks for matches in the *Shared files* repository through the *File Format Managers*. If matches exist, the binding files, i.e. the files containing the binding between files and their advertisements, are queried in order to retrieve the advertisements corresponding to the matched files. Such advertisements are successively sent back to the *Resolver*.

In both cases, the *Resolver* allows specified patterns to be matched with respect to either a set of selected attributes (*Advanced* match) or to any attribute (*Basic* match) and, in case of success, the advertisement file is transmitted to the requester.

Once the requesting peer acquires the query results, it needs to show them. In order to allow the user to derive which results are more significant for his/her aims, a set of ordering methods are provided to lead the download task. Among the others, the ordering by number of exact matches is the most important. It assigns a unit cost to each match and the file having the greatest value is considered the most important one.

In the next section, we present the environment we have implemented, which provides users with the access to the *MVN* services.

4 The Interface

Map2Share is an application whose execution is strongly dependent on its Graphical User Interface (GUI). Here users specify information to establish the *MVN* connection, they can compose queries, look at the data transfer status, visualize downloaded data and perform other tasks to solve their specific issues.

The GUI has been developed using the Swing package of JAVA. This choice allows users to preserve the platform independency, which allows us to easily distribute the software over multiple platforms without need to implement different versions.

Basically, in order to exploit the familiarity that users may have with other data sharing systems, we have organized the *Map2Share* interface following the approach used in the popular *Kazaa* [3] application. It provides four main working areas, which can be accessed through the corresponding *Search*, *Traffic*, *MyMap2Share*, and *Map* buttons, located on the toolbar. The last button, named *Tools* allows users to modify the system configuration about the peer status (FF/Rendezvous), the connection (TCP port), and other things.

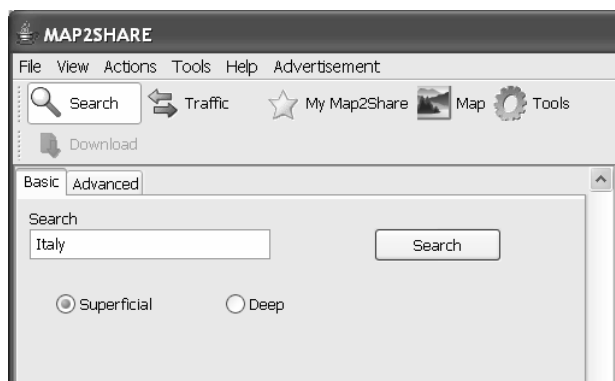


Fig. 3. A detail of the *Basic Search*

The Search Area

When a user aims to retrieve and download data, he/she fills the form characterizing the *Search Area*, according to his/her requirements. As previously stated, *Map2Share* users may pose four different kinds of queries which correspond to all the possible combinations between *Basic/Advanced* matching and *Superficial/Deep* levels. Figure 3 shows this working area where a *Basic* match is invoked involving a *Superficial* level searching on any attribute belonging to the advertisement, namely the *Italy* pattern is specified in the *Search* field and the *Superficial* checkbox is selected.

In this case, there are no tag distinctions and the *Advertisement Resolver* examines the *Advertisement Container* contents by performing the comparison between the specified pattern and the textual attributes of each advertisement. The resulting set will contain files having at least one match. Differently, Figure 4 shows a more detailed query (*Advanced /Deep*). In this case the *File Name* input field pattern (*Italy*) has been

searched within the corresponding field in the advertisement. Analogously, *Author* is set to *IGM* to retrieve resources edited by the *Istituto Geografico Militare* (Military Geographic Institute), which functions as cartographic office for the Italian government.

Fig. 4. An *Advanced Search*

According to the example, a further refinement is given by selecting the *vector* radio button in order to indicate to the *Query Resolver* to select every file representing spatial data based on the SHP format extension. The selection of the vector radio button enables the *File Format* and *Geometry* combo boxes which allow users to select both a particular file extension, such as SHP, DWG, etc., and a geometry such as point, line, polygon, respectively. Finally, the *File Content* pattern *boundaries* has been searched within the descriptive component of geographical data.

When the *Search* button starts the searching task, the *XML Query Translator* module translates the filled fields into an XML format. Successively, the query is routed to the JXTA module which broadcasts it over the MVN.

When results arrive from other peers, they are summarized in a table where each row corresponds to a file that completely/partly matches query requirements. Files belonging to the resulting set may be downloaded by means of a simple mouse action.

In the example, two instances satisfying the requirements are found, namely the *italy.shp* and *regions.shp*. The former matches the file name while the latter matches

some metadata within the database file (DBF). The matching rate of the retrieved data is described by the table *Match* field.

The Traffic Area

The final aim of the *Map2Share* application is to allow users to exchange geographical data files by downloading or uploading files from/to other computers. In order to visualize the current status of the file exchange, a *Traffic Area* has been built. It is divided into two parts, the upper one shows information related to files which have been previously downloaded or are being downloaded. A progress indicator has been included for each file in order to have a visual feedback of the current status.



Fig. 5. A detail of the downloaded files in the Transfer Area

The lower part of the *Traffic Area* manages the file upload, that is it shows uploaded files or files which are being uploaded and their current status. In the example of Figure 5, both the found files have been downloaded.

The MyMap2Share Area

Once files have been downloaded they have to be stored into some repositories. In this case, the user selects the *Shared Repository*, which implies that also other users may examine their content.

The Map Area

Finally, the last area we describe is the *Map Area*, shown in Fig. 6. It communicates with the *Visualizer* module in order to allow the user to visualize downloaded files. Following the approach of most popular GIS applications, *Map2share* allows users to visualize or hide layers by selecting or deselecting some check fields.. Moreover some conventional mapping tools are provided, such as zoom in, zoom out, pan, etc. Figure 6 shows the *Map Area* where both the downloaded files are visualized and overlaid.

The *Map Area* functionality for the map visualization and management have been implemented by using the Geotools API. It is an open source, Java GIS toolkit for

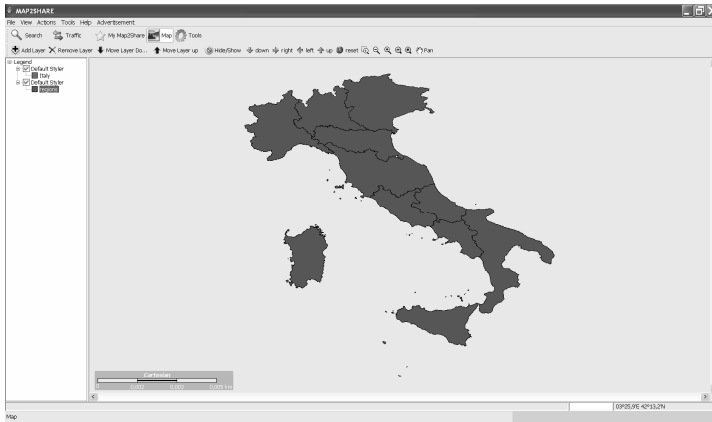


Fig. 6. The Map area

developing standard compliant solutions, implemented in separate modules, each of which realizes a specific requirement. Extra functionality may be easily incorporated thanks to its modular architecture aiming to support the OpenGIS standard [6].

6 Conclusion and Future Work

In this paper we have presented *Map2Share*, an application for geographical data sharing. The approach highly benefits from the use of a P2P transfer model, allowing organizations/individuals to share geographical resources by a direct exchange among single systems in a digital network, with no need to rely on intermediary servers. However, while traditional P2P applications allow data to be retrieved on the basis of file or path names, *Map2Share* has been explicitly conceived to allow users to search for and retrieve geographical data on the basis of several geographical features.

In the future, we plan to enhance the system adding a scalable interface which allows new file format managers to connect for generating advertisements and retrieving data based on those file formats.

References

- [1] D. Brookshier, D. Govoni, N. Krishnan, J.C. Soto, *JXTA: Java P2P Programming*, 1st edition, Sams: USA, 2002.
- [2] P.A. Longley, M. Goodchild, D. J. Maguire, D. W. Rhind, *Geographic Information Systems and Science*, 2nd edition, John Wiley & Sons Inc: USA, 2005.
- [3] M. Miller, *Discovering P2P*, SIBEX: USA, 1999.
- [4] Dublin Core Metadata Element Set, Version 1.1: Reference Description: <http://dublincore.org/>.
- [5] ISO/FDIS 19115: http://www.ncits.org/ref-docs/FDIS_19115.pdf.
- [6] OpenGIS, Simple Features Specification for SQL. version 1.1.0 2005, Open GIS Consortium Inc. <http://www.opengeospatial.org/specs/?page=specs>

Preface for CoSS 2006

Roger Chiang¹ and Vojislav B. Mišić²

¹ University of Cincinnati, USA

² University of Manitoba, Canada

Welcome to the CoSS2006, Workshop on Conceptual modeling of Service-oriented software Systems, held in conjunction with the 25th International Conference on Conceptual Modeling ER2006. The workshop focuses on different facets of conceptual modeling and design of service-oriented systems, a recent development paradigm that has the potential to transform current business software systems and applications from an archipelago of isolated entities that barely talk to one another, into a seamless organism that transcends physical and organizational boundaries, and creates new opportunities for interoperability and integration.

The Call for Papers has resulted in 11 paper submissions, out of which three regular papers and one position paper were selected for presentation at the Workshop. The papers deal with topics such as service choreographies and their relationship to orchestrations, design of services to support user tasks, building of semantic web services, and issues related to methodology reuse. While the number of submitted papers was not high, one must keep in mind that the field of service-oriented systems is still in its infancy and most efforts are devoted to lower level issues such as basic interoperability and standards to support it. As the field matures, emphasis may be expected to shift toward issues at a higher level of abstraction, and those related to conceptual design in particular.

The Workshop owes its success to the effort and support of many individuals. We would like to extend our thanks to the ER2006 conference chairs, Prof. Sudha Ram and Prof. Mohan Tanniru, and to the workshop chair, Prof. John Roddick, for their continuous support and guidance. We would like to thank the members of the Advisory Committee for their support and encouragement, as well as Program Committee members for their efforts, in particular for their professional and timely reviews. Last but not least, we thank all the authors for their submissions.

We hope that you will enjoy the presentations and discussions during the Workshop in Tucson!

Building Semantic Web Services Based on a Model Driven Web Engineering Method*

Victoria Torres, Vicente Pelechano, and Óscar Pastor

Technical University of Valencia,
Department of Information System and Computation
46022 Valencia, Spain
{vtorres, pele, opastor}@dsic.upv.es

Abstract. The first step to turn into reality the vision of the Semantic Web is to provide web contents (data and functionality) in a format that can be processed by software agents. However, this change will not be achieved until users count on methods and techniques that help them in its development. For this reason we provide in this paper a methodological solution based on the Web Engineering discipline. We propose to extend a Web Engineering method with an Ontology model in order to specify, at the modelling level, the system data and functionality that are going to be exposed for the use of external applications. Moreover, by the application of a set of transformation rules we (1) build a draft of this Ontology model from the structural and behavioural models defined in the OOWS approach and (2) generate the equivalent representation of the Ontology model in a particular semantic web language.

1 Introduction

In order to turn into reality the vision of the Semantic web [1], it is necessary to provide developers with guides, methods and tools that encourage them to make use of semantic technologies for the development of real-world applications.

The development of the Semantic Web involves not only the generation of semantic content (ontologies defining specific domains) but also the semantics of some functionality that allow software agents to discover, invoke, compose and monitor it with a high degree of automation.

The Web Engineering field [8] emerged to address some of the aspects that conventional software methods did not consider while generating Web applications. These aspects refer to Navigational and Presentation issues. These aspects became, within software methods, first-order citizens in the conceptual modelling step. In this sense, Web Engineering methods should now be prepared to provide solutions that tackle the modelling of this new dimension, which refers to the view over the system from the Semantic Web point of view. This new dimension will allow us to generate not only applications intended for humans, but also for automated software agents

* This work has been developed with the support of MEC under the project DESTINO TIN2004-03534 and cofinanced by FEDER.

which can understand application contents (data and functionality) because they are expressed in language that provides a vocabulary along with a formal semantics.

Most of Web Engineering methods (OOHDM [15], OOWS [4], OO-H [5] or WebML [2] among others) follow a Model Driven Development (MDD) approach in the sense that they are based on a set of models that cover those concerns that are necessary to specify Web Applications (structure, behaviour, navigation and presentation). The main benefit of adopting an MDD approach is that the system specification can be automatically transformed in terms of any implementation technology if the appropriate transformations have been defined.

Modelling ontologies and the semantics of services is a complex task that requires having a deep knowledge in the target domain. However, within the web engineering field, developers are dealing in every project with this kind of tasks. They are used to model system domains by means of class diagrams and activity diagrams. Therefore, we want to take profit of this knowledge in order to obtain a first draft of the semantic content that we want to generate.

We want to note that Web Engineering methods are approaches for modelling and producing web applications and not for the development of ontologies in itself. However, if we consider the definition provided by the ODM RFP [9], “An ontology can range in expressivity from a Taxonomy (knowledge with minimal hierarchy or a parent/child structure), to a Thesaurus (words and synonyms), to a Conceptual Model (with more complex knowledge), to a Logical Theory (with very rich, complex, consistent and meaningful knowledge)”. Therefore, considering that Web Engineering methods are based on conceptual modelling techniques, we can take advantage of their infrastructure to produce those ontologies that represent the system domain.

The main contribution of this work is to provide the Semantic Web community with a method for the generation of semantic content and functionality. Based on the use of Web Engineering methods, in particular the OOWS (Object Oriented Web Solution) approach [4], we propose to extend it to generate part of the system specification that is going to be accessible through Semantic Web technology. This generation can be performed since the OOWS approach includes a set of models that specify in precise and non ambiguous way the system structure and behaviour in the form of a conceptual schema. To achieve this goal, we have enriched the OOWS approach with a mechanism to define, at the modelling level, the system in terms of the Semantic Web point of view. This mechanism involves the introduction of a new dimension within the process modelling step. This new dimension means specifying the view over the system for external agents. It is defined by two models, one aimed at specifying the system domain (tourism, health, news, education, etc) and another for describing how external agents should use the functionality of the system exposed in business settings. These models refer to the OWL and the OWL-S metamodels respectively. Within the literature, there are different works aimed at providing a solution to the modeling process of ontologies. Most of these proposals [3, 6] are based on the use of UML profiles to overcome the task of defining ontologies. In this work we base on these profiles for the specification of these ontology models.

The remainder of the paper is organized as follows. Section 2 provides the background in the area of semantic web languages as well as in the context of the Web Engineering methods, in particular to the OOWS approach. Section 3 presents an overview of the proposal within the context of the OOWS approach. Section 4

presents, by means of an example, the use of the method for the generation of the semantic dimension. Finally, section 5 provides some conclusions and further work.

2 Background

In this section we provide an overview over the existing languages to define ontologies, as well as a brief overview over the OOWS approach in which we have contextualized the presented work.

2.1 Markup Ontology Languages

Ontology languages are formal languages used to construct ontologies. They allow the encoding of knowledge about specific domains and often include reasoning rules that support the processing of that knowledge.

In the last decade we have been witnesses of how different markup ontology languages (not the traditional ones used in the AI field, such as FLogic, KIF or OCML) have evolved to represent as much knowledge as possible. A good example of this evolution is the OWL [10] language, the successor of DAML+OIL and an extension to the RDFS language. However, the list of different ontology languages includes SHOE, XOL, DAML+OIL, RDF or RDF Schema.

Although there are already tools that allow manipulating ontologies (for instance the Protégé ontology editor - <http://protege.stanford.edu/>), for system designers it is still difficult to tackle directly with these rumbling low-level languages. This difficulty comes mainly to the fact that software engineering is moving towards the model driven design based on graphical notations and not on textual ones.

2.2 Service Ontology Languages

In the same way as ontology languages were developed to improve machine interoperability among applications that needed to process web data, within the Web Services community, there are different initiatives to define Web service properties and capabilities in unambiguous, computer-interpretable form. Among the major initiatives we found OWL-S [11], WSMO [14], WSDL-S [16], IRS-II [7], and METEOR-S [13].

All these initiatives share the same goal, which is to facilitate the automation of discovering, combining and invoking services over the Web. In the following paragraphs we make a review of the most interesting parts of each of those proposals. All these specifications overlap in some parts and are complementary in others. However, all of them are described using low-level lexical notations, what makes difficult their widespread use and adoption. Therefore, as the ODM [9] proposes, it is necessary to provide a graphical notation (usually by means of UML profiles) to specify ontologies in the different languages and then, by means of the application of mappings, to obtain the same service description (at least those overlapped parts of the specifications) in another specification. Although there is some work towards the convergence of OWL-S with the products of the WSMO/WSML research effort, until the time being there is not a unifying proposal in that sense.

2.3 OOWS

OOWS (Object Oriented Web Solutions) is the extension of the object-oriented software production method OO-Method [12] that introduces the required expressivity to capture the navigational and presentational requirements of web applications. This extended view of the OO-Method with OOWS provides full software development method for web applications that defines a set of activities to be fulfilled to properly specify the functional, navigational and presentational dimensions of web applications requirements. The whole method includes a set of models which are organized into three parts, which are:

- *Structure and Behavior Specification*: The set of models included in this part (Structural, Dynamic and Functional) are focused on the description of the domain being automated, characterizing the peculiarities inherent to a specific domain.
- *User Interaction Specification*: The models included in this part specify all the requirements related to the user interface. It is made up of two models, which are the Navigational and Presentation models.
- *External System Interaction Specification*: This part defines, at the modeling level, the functionality that is consumed by our system and that is provided by external partners.

However, application interoperability is becoming increasingly important due to the wide use of the Internet. Therefore, we should enrich our methods in order to generate web applications that satisfy current trends in the web.

3 Overview of the Extended Method

The OOWS approach, as most Web Engineering Methods, was designed to model Web Applications where the main logic resides in the own system. However, this idea of having monolithic applications is falling behind open and interoperable Web systems. To improve interaction between systems we have extended the OOWS approach with a new dimension. This new dimension refers to *semantic application specification* from the Semantic Web point of view. We include this dimension in order to manage, at the modeling level, the semantics of content and services that we are going to make available to external entities in a machine-processable format.

Usually, the content and functionality that we want to provide to external systems have already been specified in the system by means of the (1) structural and the (2) behavior specification. However, it must be clear that we can be interested in publishing only some parts of the system for external consumers. The naive assumption that all the application data and functionality is going to be revealed to a general audience cannot be accepted at all. Therefore, we needed, at the modeling level, a mechanism that allows us to specify what parts of the content and functionality is going to be accessible in a machine-understandable form. For this reason, we have included two new models within the approach which are gathered in what we call the *Ontology Models* (see Fig. 1).

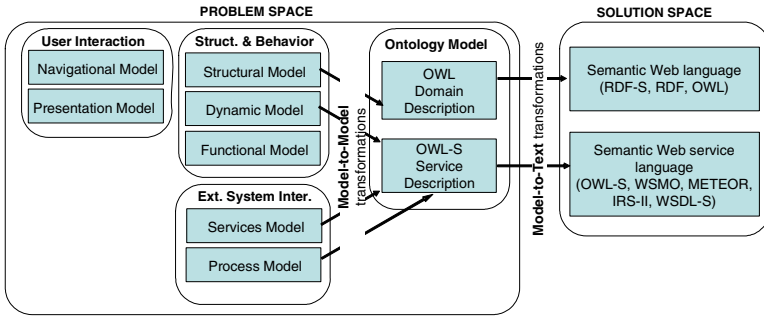


Fig. 1. Extended Method Overview

The Ontology Models are defined as views over the Structural and the Process Model. It is made up of two parts, one that refers to the domain, *OWL Domain Description* (concepts, attributes, relations between concepts), and another to functionality, *OWL-S Service Description*.

However, unlike software system models, shared worlds modeled with ontologies do not keep separation of metal-levels in a given model, resulting in specifications that contain both schemas and instances. If we would have to fit the Ontology Model within the four layer level architecture defined by the OMG, we would see that the ontology specification covers the M1 (refers to general concepts) and M0 (refers to concrete instances) layers.

The Ontology Model is represented graphically by means of a UML profile [3]. However, the profile used in this work does not include the UML stereotypes that refer to instances from the OWL Metamodel (Individual, datatypeSlot, objectSlot, differentFrom, SameAs and so on). This is because the part of the ontology that refers to instances is going to be automatically generated from the knowledge base that is kept in the persistence layer (usually in the form of a database). Therefore, the separation between layers make us to separate the domain ontology Metamodel in two parts, one that refers to general concepts and another that refers to concrete instances.

Domain Ontology Generation Process

Initially, from the class diagram specification we have to select those elements (concepts, attributes, relationships between classes and so on) that are going to be included in the ontology. This filtering process is performed manually by the developer using a tool. Once we have the structural part of the domain ontology represented graphically, we can manipulate it from its graphical representation to complete the ontology specification or to apply directly a model-to-text transformation process that allow us to obtain the document that contains the ontology description (owl or rdf-s files). However, as we have stated previously, ontologies frequently cross levels, and contain instances as well as schemas. For this reason we have included a third step in which we complete the domain ontology description with the instances that define the knowledge base. However, this step cannot be performed until the system had produced the corresponding information.

Service Ontology Generation Process

Regarding to the generation process of the Service Ontology, the process is quite similar to the previous one. For the Service ontology, the filtering process performed in the first step is made from the class diagram and the process model. The class diagram includes the definition of a set of class operations that will be defined in the service ontology as atomic operations. On the other hand, composite operations are taken from the definition of processes defined by means of activity diagrams in the process model.

Once the filtering process is completed, we apply a model-to-model transformation to extract, as much as possible, all the knowledge (conditions and effects) regarding the operations that are going to be provided as Semantic Web services. This knowledge is going to be extracted from the Dynamic Model. This model specifies the conditions under which an operation can be executed in order to change the state of a specific class. Finally, by the application of the corresponding model-to-text transformations we generate service descriptions based on different service ontology languages.

4 Semantic Dimension

This section explains how to obtain the models defined in the semantic dimension, and how these are related to the rest of models from the OOWS approach. Moreover, it outlines the transformations that are defined to obtain the content and services ontologies. In addition, the following subsections are accompanied with a Rental Car System (hereafter RCS) example.

The RCS is a Web Application that allows customers to rent vehicles electronically. In order to provide this service to customers, the website provides information regarding its fleet and prices, giving the customer the opportunity to request a reservation or receive a confirmation there. Therefore, the system searches to match the customer requirements, which are based mainly on price, vehicle category and collection and return dates. Once the customer has selected the vehicle that matches with his/her requirements, the RCS proceed with the reservation process by charging part of the total amount of the reservation. However, the RCS may wish to provide these features also to other applications (i.e. travel agencies) not just to users via the website.

4.1 Domain Representation: OWL

The domain ontology model only includes the ontology elements that represent concepts and types in general, not to any particular instance. As we have already stated, the ontology part that refers to instances is completed from the information kept in the persistence layer.

As we have stated previously, the semantic content is going to be reused/extracted from the specification defined in the structural model by means of the class diagram (see Fig. 2). This specification gathers those concepts and relationships that represent that part of the domain that our system is going to manage (vehicles, reservations and customers). However, in the ontology domain we only want to provide just part of the

system specification. For this reason, the first step of the process for ontology generation is to perform the filtering step. This step is performed by the modeler manually by selecting all those elements (concepts, attributes and relationships) that he wants to include in the domain ontology. As a result, we obtain a view over the class diagram in terms of an OWL model. This model represents, by means of a UML profile, the OWL metamodel. In the last step of the process, this model will be transformed into a document that will include just the elements that are going to be accessible in a machine processable format (see Fig. 3). The resulting ontology does not include for instance sensitive information such as customer personal data, which has to be regulated by the data protection law.

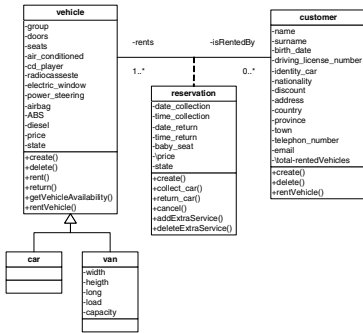


Fig. 2. An Excerpt of the RCS Class Diagram

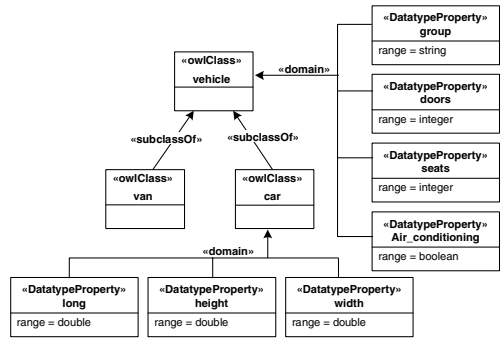


Fig. 3. An Excerpt of the Domain Ontology Model

The class concept within the ontology terminology can be defined in six different ways (as a class identifier, as an enumeration, as a property restriction, as a union, as a complement or as a pair wise distinct set of individuals). From these six ways, except the first one (class identifier), the rest class descriptors refer to individuals that form classes. Therefore, the ontology domain, at the modeling level, will only include classes defined as class identifiers, and not as individuals.

In order to have the maximum expressiveness without losing computational completeness and decidability we have decided to generate the ontology domain in the OWL DL sublanguage. This language has the characteristic that it includes all OWL language constructs with some restrictions such as strict separation of classes, properties, individuals and data values. However, this restriction is completely respected when defining the corresponding transformation rules.

4.2 Semantic Web Service Definition: OWL-S

Semantic Web services described in OWL-S are structured in three parts which consist of a *service profile* for advertising and discovering services, a *process model* which gives a detailed description of a service's operation and a *service grounding* which provides details on how to interoperate with a service via message exchange. Although it is unrealistic to think that we can obtain a complete semantic description

from the structural and dynamic models, at least, it is worth to obtain a first draft of the services that we want to expose as a Semantic web service.

The models that are going to be taken to obtain part of the semantic description of the functionality are the *Structural*, the *Process* and the *Dynamic* Models.

The *Structural Model*, which is defined by means of a Class Diagram, includes for each class, the definition of a set of operations whose execution results in changing the state of the class objects. This definition includes the operation signature, as well as the input and output parameters of the operation. Therefore, from this model we obtain input and output parameters from atomic services (services that are provided as black boxes to its potential consumers).

The *Process Model* defines, by means of the Activity Diagram, the internal logic of complex operations. It defines also the input and output parameters required by the complex operation.

The *Dynamic Model*, which is defined by means of a State Transition Diagram (STD), defines the different valid object-life sequence for each class of the system as well as the transitions that allow changes in the class state. These transitions are specified as follows: Operation [*if precondition*] [*when control-condition*]. From this specification, *operation* refers to an operation defined in any of the classes from the Class Diagram, *precondition* refers to the conditions that must be satisfied in order to execute the operation and *control-condition* are conditions used when for the same operation there are several alternatives (several transitions from the current state). Therefore, in these transitions we have defined the conditions under which an operation can be executed (defined by pre and control conditions). In addition, the valid states of the class objects state the effects that result from the execution of a particular operation.

However, to obtain a complete description, the process of modeling semantic web services should include both manual and automatic steps:

- First step: Select the system functionality that we want to provide as a Semantic web service. This selection is made from the operations defined in the Class Diagram (which will map to atomic processes) and in the Process Model (which will map to composite processes).
- Second step: Apply the transformations that build part of the Semantic web services description (input and output parameters, service preconditions and service effects). These transformations refer only to those operations that have been selected in the previous step. For all of them, we look for any occurrence within the STD. All these occurrences will give us information about the conditions under which an operation occurs as well as their effects within the system.
- Third step: Complete the service description with those elements (such as contact information or security capabilities, quality rating or service classification in a standard taxonomy which are specified in the Service Profile) that were not specified in the source models and that are necessary to obtain a complete description of the Semantic Web Service.

Fig. 4, on the left side, shows the STD defined for the Vehicle class modelled in the Class Diagram. This STD indicates the valid states (available, broken_down and rented) for the Vehicle class as well as the transitions that allow performing the state changes. On the other hand, the right side of the figure shows an excerpt of the owl file that represents the process for the rentVehicle operation.

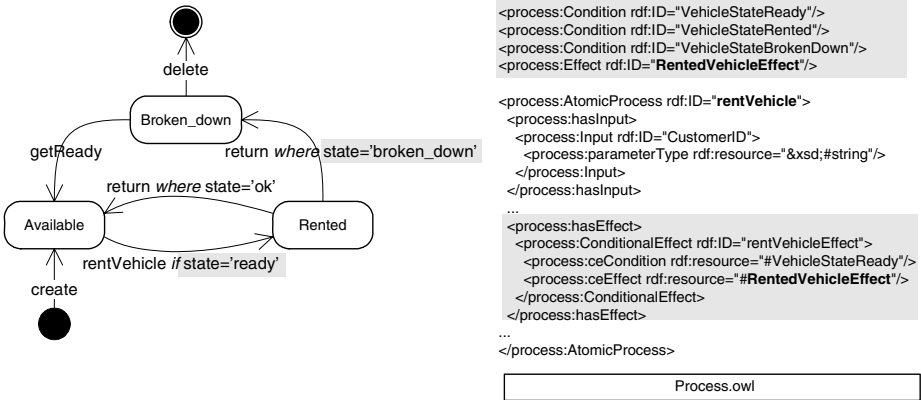


Fig. 4. Mapping between the STD and the OWL-S Service Model

5 Conclusions and Further Work

Since Tim Berners-Lee made public his vision about an extension of the current Web into the Semantic Web, we have not seen a wide use of the languages and technologies developed until the time being. Therefore, it is necessary that different research areas provide proposals that help developers to foster the use of semantic web technologies and contribute for the development of a real Semantic Web. Moreover, it must be clear that the first requirement to turn into reality the Semantic Web is to publish the content of the Web associated with clear semantics.

In this work we have presented an approach based on the OOWS method for the generation of a web application from the Semantic Web point of view. Our intention is to contribute to the use of the Semantic Web by introducing the languages and technologies developed in the context of the Semantic Web within the Web Engineering field. In order to make this introduction in the most natural way, we have applied the transformation mechanisms that we already used for the generation of the web system in a particular technology.

As further work we have planned to introduce this proposal to the OlivaNova Modeling Execution (ONME) Tool (<http://www.care-t.com>). This new feature will provide the ONME tool with the mechanisms to generate web applications not only for the use of humans, but also for external applications.

References

1. Berners-lee T., Hendler J. and Lassila, O. The Semantic Web. *Scientific American*, May 2001.
2. Ceri, S., Fraternali, P. and Bongio, A. "Web Modeling Language (WebML): a Modeling Language for Designing Web Sites", In *WWW9, Vol. 33 (1-6), Computer Networks*, 2000, pp 137-157.
3. Djuric, D. MDA-based Ontology Infrastructure. *Computer Science Information Systems (ComSIS)*, 1(1):91-116, February 2004.

4. Fons, J., Pelechano, V., Albert, M. and Pastor, O., "Development of Web Applications from Web Enhanced Conceptual Schemas", *Proc. Of the International Conference on Conceptual Modelling, 22nd Edition, ER'03*, Chicago, EEUU, 2003, pp. 232-245.
5. Gómez, J., Cachero, C. and Pastor, O. "Extending a Conceptual Modeling Approach to Web Application Design", *Proc. Conference on Advanced Information Systems Engineering, CAiSE'00*, Springer- Verlag, LNCS 1789, 2000, pp. 79-93.
6. Gronmo, R., Jaeger, M.C., and Hoff, H. Transformations between UML and OWL-S. *Eur. Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA 2005) Nuremberg, Germany*, (November 2005), 269-283.
7. Motta, E., Domingue, J., Cabral, L. and Gaspari, M. IRS-II: A framework and infrastructure for semantic web services. *2nd International Semantic Web Conference (ISWC2003)*. (October 2003) Springer Verlag.
8. Murugesan, S. and Deshpande, Y. (ed.): Web Engineering. Web Engineering, Software Engineering and Web Application Development. Lecture Notes in Computer Science, Vol. 2016. Springer.
9. Ontology Definition Metamodel. Revised submission to OMG/RFP ad/2003-03-40.
10. OWL Web Ontology Language Reference. W3C Recommendation 10 February 2004, Mike Dean, Guus Schreiber, eds. Latest version is available at <http://www.w3.org/TR/owl-ref/>.
11. OWL-S: Semantic Markup for Web Services. <http://www.daml.org/services/owl-s/1.1>
12. Pastor, O., Gomez, J., Insfran, E. and Pelechano, V. "The OO-Method Approach for Information Systems Modelling: From Object-Oriented Conceptual Modeling to Automated Programming". *Information Systems* 26, pp 507-534 (2001)
13. Patil, A., Oundhakar, S., Sheth, A., and Verma, K. METEOR-S Web service Annotation Framework, *Proceeding of the World Wide Web Conference, New York, NY, May 2004*, pp. 553-562
14. Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C. and Fensel, D. Web Service Modeling Ontology (WSMO). *Applied Ontology* 1 (2005) 77-106. IOS Press.
15. Schwabe, D. and Rossi, G. "An Object Oriented Approach to Web-Based Application Design", *Theory and Practice of Object System* 4(4), Wiley and Sons, New York, 1998, ISSN 1074-3224.
16. Web Service Semantics – WSDL-S. <http://www.w3.org/Submission/WSDL-S/>

Choreographies as Federations of Choreographies and Orchestrations*

Johann Eder, Marek Lehmann, and Amirreza Tahamtan

University of Vienna
Dept. of Knowledge and Business Engineering
`first_name.last_name@univie.ac.at`

Abstract. We propose a new conceptual model for choreographies of web-services. Choreographies are seen as virtual workflow models shared among participants. Subsets of these participants might have more refined models known only to them. So we see choreographies actually as federations of process models (choreographies as well as orchestrations). In this paper we discuss this layered concept, and present a metamodel with the following distinguishing features: It is fully distributed and does not require a central or global authority. It captures the control flow and the data flow aspects of the processes. Choreography models can be (re)used in several other choreographies. Additionally, we provide a procedure which checks whether choreographies fit together, i.e. the conformance of the federation relationship between models.

1 Introduction

Web Service composition has received recently a lot of interest, both in academia and industry. It enables software integration within and across boundaries of cooperating organizations. The real added value of Web Services lies in their composition and this is the place where several overlapping and competing standards have been proposed. Proposals like BPEL4WS [3], WS-CDL [8], WSCI [7] address some aspects of both choreography and orchestration, but the distinction between the two is not very clear. Moreover, there is still a big gap between choreography and orchestration models, i.e. it may be very difficult to answer whether a given orchestration really realizes a given choreography. A coherent and integrated view on both choreography and orchestration is still missing.

Web Services are loosely-coupled and autonomous. From this perspective it is essential that an organization is able to define its own business process described as a composed Web Service and control the execution of this process or at least the execution of parts of it. This is the task of the orchestration. On the other hand, the same organization must adjust its internal executable processes to protocols used in cooperative and interorganizational processes described by choreographies. Furthermore, this organization may take part in several choreographies and several choreographies may be reflected as a single orchestration.

* Work partly supported by the Commission of the European Union within the project WS-Diamond in FP6.STREP.

In this paper we propose a new metamodel which gives a consistent and integrated view on both choreographies and orchestrations and their mutual relationships. We use an extension of our workflow metamodel [6,9] which enables convenient and flexible representation and analysis of choreographies and orchestrations. Moreover, we propose a novel approach to combine several choreographies into more complex federated choreographies. Our federated choreographies are more flexible than typical compositional approaches.

2 Federated Choreographies

A Web Service choreography specifies a communication protocol for all involved partners. The outcome of the choreography is a virtual process definition viewed from the global perspective where all partners are treated equally [4]. On the other hand, Web Service orchestration refers to an executable process run by a single partner. It contains information about message exchanges, definition of business logic, execution order of activities and internal actions (e.g. data transformations) [11]. Each of the partners involved in a global choreography realizes the in the internal orchestration only its own parts of this choreography.

This typical scenario [4,11] assumes one choreography shared among all partners and a set of private orchestrations (one for each partner). Besides, two partners involved in one choreography may also take part in another choreography that is not visible to other partners in the first choreography, however essential for the realization of its goal. For example, when shopping online we take part in a choreography, where we know the following partners and steps: we order something at a seller, pay by a credit card and expect to receive the items from a shipper. At the same time the seller takes part in several other choreographies which are not visible to us, e.g. the seller and the shipper realize another protocol they agreed upon containing other actions like money transfer from seller's bank to the shipper for balancing shipment charges. These choreographies overlap in some parts but cannot be composed into a single choreography. Moreover, such choreographies must be realized by orchestrations of partners that take part in them. In the above example the seller implements an orchestration enacting the different interaction protocols with the buyer, shipper and the bank.

We propose a new approach where existing choreographies can be combined into a *federated choreography* and extended according to the need. We treat both choreographies and orchestrations as workflows. A choreography coordinates several orchestrations owned by different partners and a single orchestration may realize parts of several choreographies. Choreographies can be federated into more complex ones. Moreover, as all choreographies and orchestrations are workflows, they can be composed out of other choreographies by means of complex activities and control structures available in our workflow models.

The idea of federated choreographies is presented in Fig. 1. It consists of two layers. The first layer consist of federated choreographies shared between different partners, e.g. a *Purchase processing* choreography shared between Buyer, Seller, and Shipper. A choreography may support another choreography. This

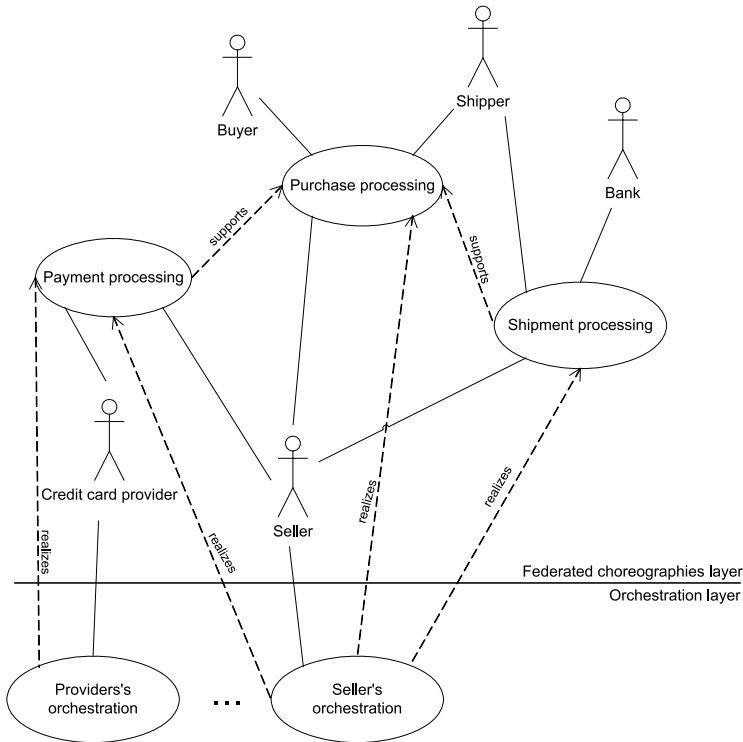


Fig. 1. Federated choreographies

means the former contributes to the latter and partially elaborates it. The second layer consist of orchestrations that realize the choreographies. Each partner provides its own realization of relevant parts of the according choreographies, e.g. the Seller has an orchestration which realizes its part in all three choreographies.

The presented approach is fully distributed. Each partner has local models of all choreographies in which it participates. All local models of the same choreography are identical. Additionally, each partner holds and runs its own model of the orchestration. There is no need for a centralized coordination.

In order to ensure the compatibility of the system, a conformance test between orchestrations and the choreographies that they realize as well as an intra-layer conformance test in the choreography layer is necessary.

The choreography in Fig. 2 describes the "Purchase Processing" choreography in Fig. 1. Three partners participate: a Buyer, a Seller and a Shipper. The Seller upon receiving a request quote from the Buyer answers if the requested item is deliverable or not. If the item is deliverable, the buyer places an order, the seller processes the order and forwards the shipment details to the shipper. The shipper ships the products to the buyer and informs the seller about the details. The seller upon receipt of information sends the bill to the buyer. When the buyer has received both the bill and the ordered goods, makes the payment to the seller. The process terminates upon receipt of the payment by the seller.

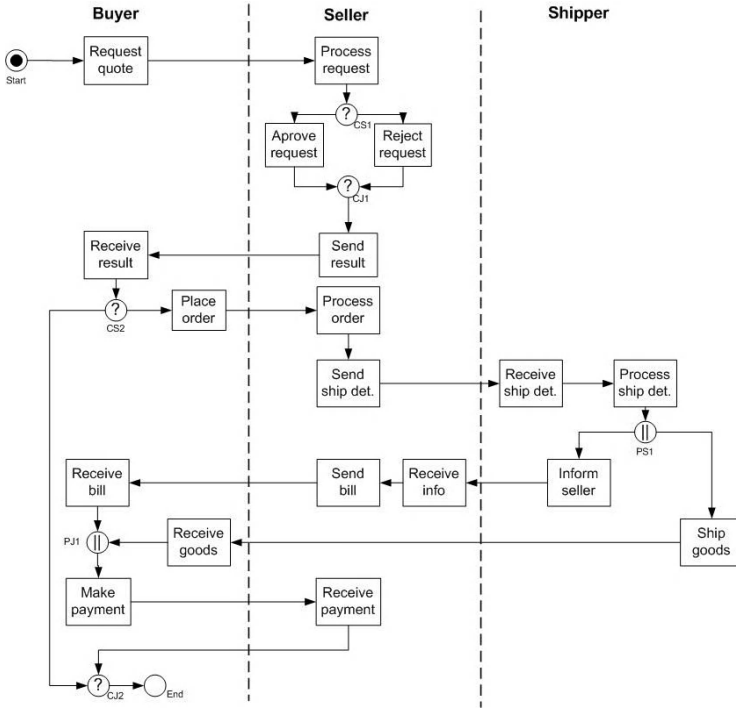


Fig. 2. *Purchase processing* choreography between Buyer, Seller and the Shipper

This choreography involves interactions between the Buyer and the Seller, the Seller and the Shipper and between the Buyer and the Shipper. Each of these interactions can be represented as a separate choreography. These choreographies may be combined by the means of composition as described in [8] where existing choreography definitions can be reused and recursively combined into more complex choreographies. But we claim that a relationship between choreographies can be more sophisticated than merely a composition. In our example the relationship between the Seller and the Shipper includes not only the passing of shipment details from the Seller to the Shipper but it also involves payment of shipment charges through the seller's bank. This is described by a separate choreography between the Seller, Shipper and Bank. This choreography shown in Fig. 3 has additional activities and partners which are not visible in the previous one. This choreography contributes to the "Purchase Processing" and elaborates the interaction between the Seller and Shipper.

Choreographies describe just abstract processes, whereas an orchestration describes an executable process run by one of the partners. Each partner involved in choreographies must provide their realization. The orchestration of the Seller is presented in Fig. 4. It implements the behavior of the Seller in both presented choreographies. Notice that the Shipper and other partners must provide their own private orchestrations which we omit here for the aim of brevity.

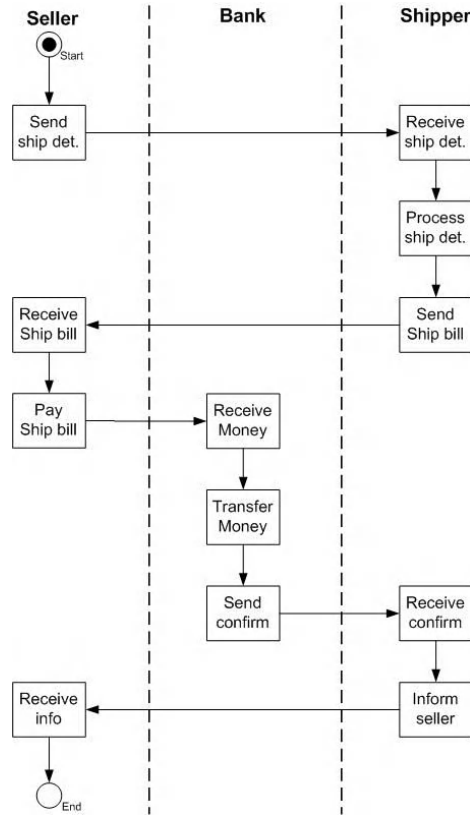


Fig. 3. The *Shipment processing* choreography between Seller, Bank and the Shipper

3 Metamodel

Our metamodel enables the representation of both choreographies and orchestrations which are described as workflow models. Therefore, choreographies and orchestrations can be modeled using typical workflow control flow structures. Moreover, it provides a coherent view on both choreographies and orchestrations and their mutual relationships, thus bridging the gap between abstract and executable processes. The metamodel allows to describe several choreographies on different levels of detail and an orchestration responsible for a private implementation of these choreographies. Choreographies and the orchestration can share the same activities. An activity visible in one choreography can be extended by its relationships with other activities in a federated choreography. On the other hand, an activity visible in a choreography can have a complex implementation described in an orchestration. Thus, choreographies and orchestrations together with their activities can be viewed on different levels of detail and in context of different relationships. Our metamodel presented in Fig. 5

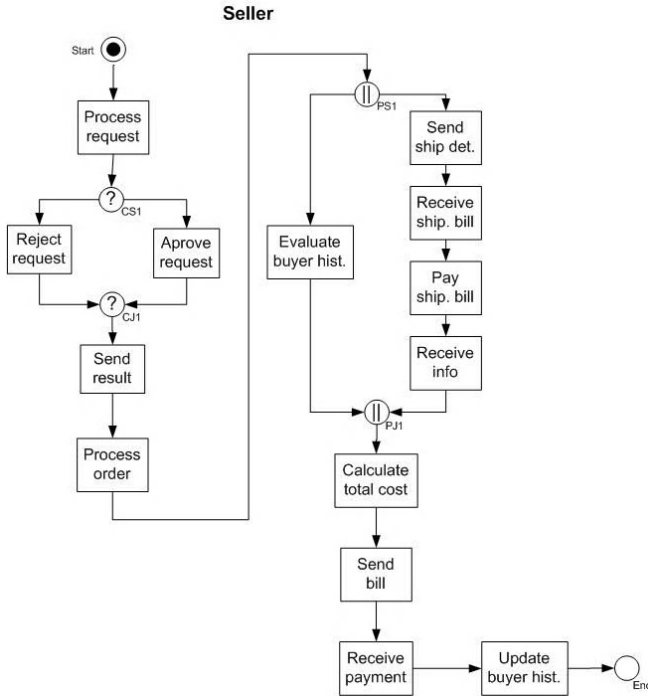


Fig. 4. Seller's Orchestration

focuses only on the control flow aspects of a workflow definition. Because of lack of space we had to omit the data aspects.

A workflow is either a *choreography* or an *orchestration*. A workflow uses *activities*. An activity is either a *task*, a *complex activity* or a *(sub-)workflow*. An activity can be used to compose complex activities and workflows. An activity occurrence in such a composition is represented by an *activity step*. One activity can be represented by several activity steps in one or several workflows or complex activities and each activity step belongs to exactly one activity. In other words, activity steps are placeholders for reusable activities. The control structure of a complex activity is described by its *type* (*seq* for sequence, *par* for parallel and *cond* for conditional).

An activity may be owned by a *partner*. Orchestrations and tasks must have an owner, whereas choreographies must not have an owner. A partner may have several *roles* and one role can be played by several partners. A role may take part in a workflow and call an activity step provided by another role. Thus a single parter can use different roles to participate in a workflow.

The notion of a *step* is very important for the presented metamodel. Both workflows and complex activities consist of steps. Between the subsequent steps there can be a *transition* from a predecessor to a successor which represents control flow dependencies between steps.

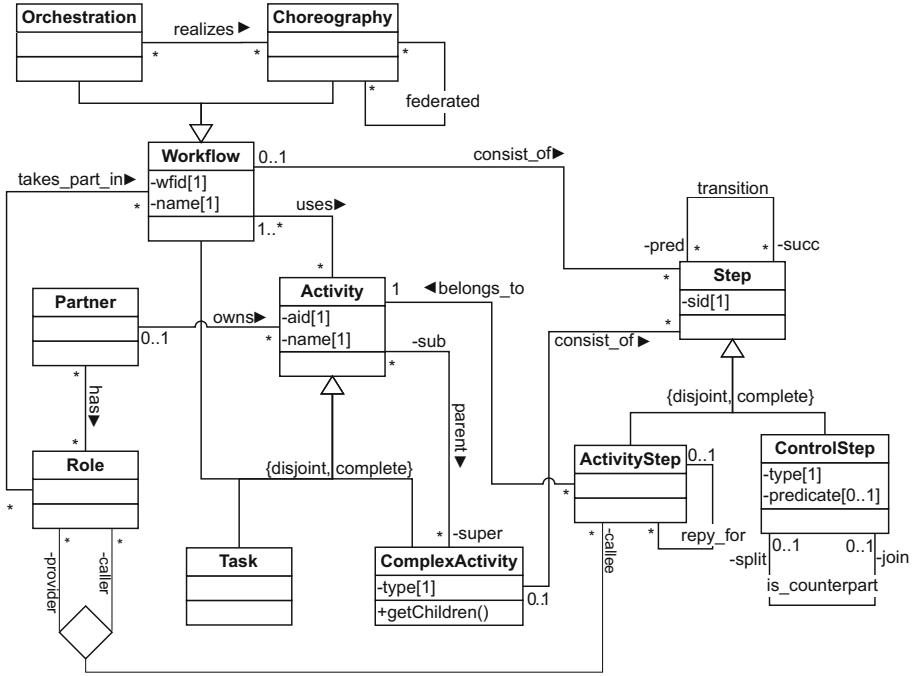


Fig. 5. Metamodel for Federated Choreographies

A complex activity may be decomposed in a given workflow into steps that constitute this complex activity only if all the activities corresponding to these steps are also used and visible in this workflow. Therefore, a workflow can be decomposed and analyzed on different levels of detail with complex activities disclosing their content, but without revealing protected information on the implementation of these complex activities. To allow a correct decomposition, a complex activity must have only one activity without any predecessors and only one activity without any successors. The same applies to workflows.

A step can be either an *activity step* or a *control step*. As mentioned above, activity steps are placeholders for reusable activities and each activity step belongs to exactly one activity. Activity steps can be called in a workflow definition. An activity step may be used as a reply for a previous activity step. A single activity step may have several alternative replies.

A control step represents a control flow element such as a split or a join. Currently we allow conditional and parallel structures, i.e. the type of a control step is one of the following: *par-split*, *par-join*, *cond-split* or *cond-join*. An attribute *predicate* is specified only for steps corresponding to a conditional split and represents a conditional predicate. Conditional splits have XOR semantics. A split control step may have a corresponding join control step what is represented by the recursive relation *is_counterpart*. This relation is used to represent well structured workflows [6] where each split node has a corresponding join node of the same type and vice versa.

A workflow model can be represented as a directed graph with two kinds of nodes corresponding to activity steps and control steps. The edges correspond to transitions between steps and determine the execution sequence. A complex activity can be decomposed into a subgraph with one input node and one output node. In the graphical notation we use additionally two control nodes: start node and end node places before the start activity and after the end activity of the workflow, respectively. A sample workflow graph is presented in Fig. 2.

The workflow graphs are a convenient and human readable representation of workflow models. For conformance tests we use workflow nets (WF-Nets) [2] which are an extension of classical Petri nets [10] and can be analyzed with all the techniques developed for Petri nets. A WF-Net contains exactly one place without any predecessors (source) and exactly one place without any successors (sink). Moreover, the net graph extended with an additional transition from the sink place to the source place is strongly connected, i.e. for each node n there exists a path from the source place to n and from n to the sink place. We use WF-nets with a set of special transitions added to express branching decisions in a more human readable form: AND-split, AND-join, XOR-split and XOR-join.

The mapping from the presented workflow graphs to WF-Nets is straightforward: start and end control nodes are mapped onto source and sink place respectively; each activity steps is mapped onto a labeled transition with a single input and output place; each split control node is mapped onto a corresponding split transition with a single input place and at least two output places and each join control node is mapped onto a corresponding join transition with at least two input places and exactly one output place. Edges in the original graph are mapped onto dummy transitions connecting subsequent transformed nodes. Finally, dummy transitions can be reduced by Fusion of Series Places (FSP) [10]. A sample mapping of a graph in Fig. 2 is presented in Fig. 6.

4 Conformance Test

Essentially the participating partners are autonomous organizations that may have existing workflows for their orchestrations as well as for their interactions with other organizations and they favor to use the existing workflows instead of designing new ones from scratch and integrate them in the organization. It is pivotal to ensure that utilization of these orchestrations and choreographies lead to no conflict with other choreographies and orchestrations. The basic requirement of the model is the inter and intra-layer conformance of orchestrations and choreographies. This means the orchestration and choreographies must maintain all the assumptions made by the choreography they support or realize. It is possible to check the structural conformance of the system based on the *projection inheritance* [1,5] and notion of *branching bisimulation* [12] as equivalence relation of workflows. The question if two workflows are branching bisimilar is a decidable problem. Because of space limitations we are not able to provide the full description of conformance tests in this paper.

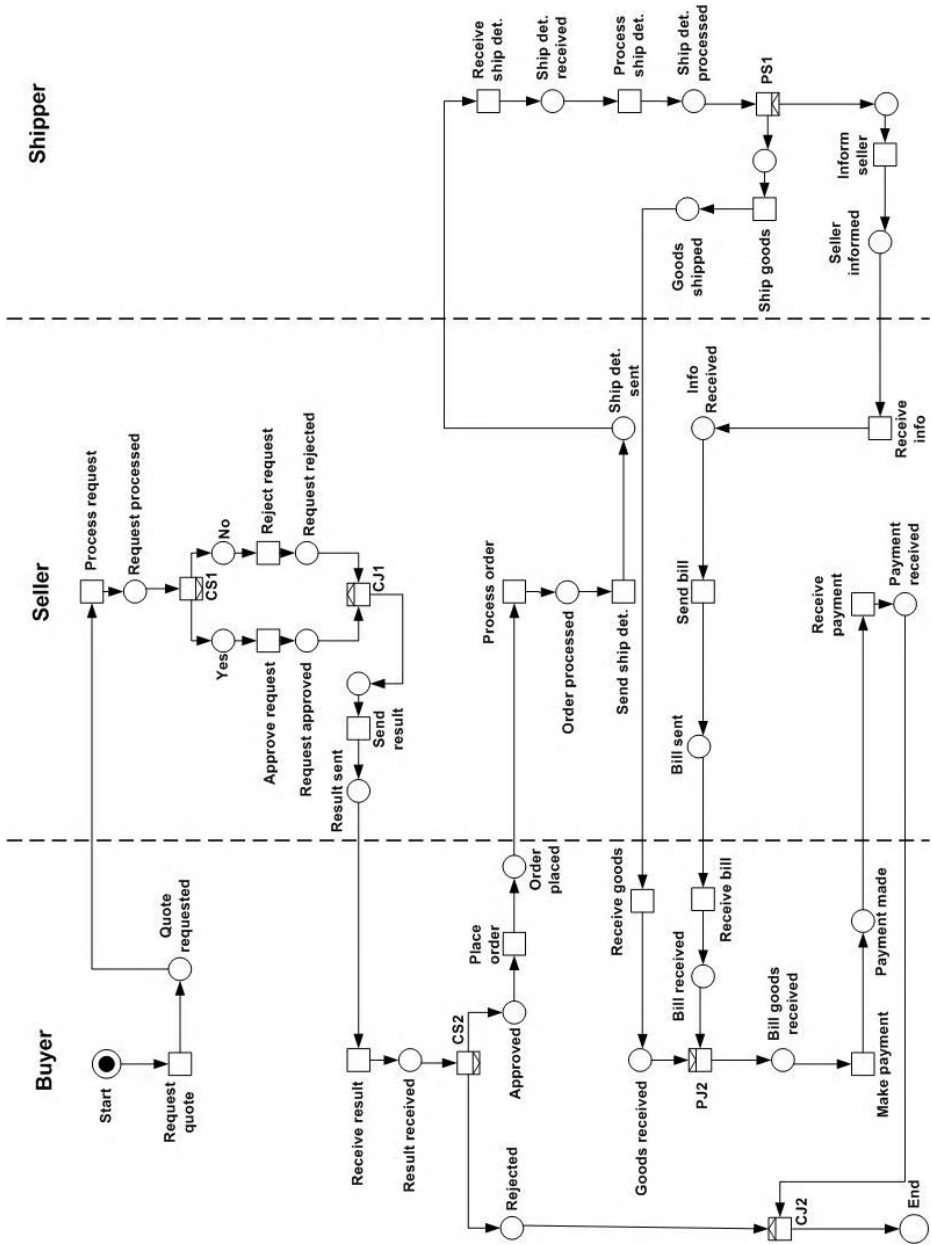


Fig. 6. The WF-Net of workflow graph in Fig. 2

5 Conclusions

Federated choreographies aim at modularizing web service choreographies. Federations can be built top-down - applying the famous "divide and conquer"

paradigm - or they can be built bottom-up, re-using existing choreographies for the construction or negotiation of larger choreographies. The meta-model we propose supports all these usages. The automated checking of conformance between choreographies and also orchestrations can be used for federations built top-down as well as for those built bottom-up. Design principles and processes for both ways of constructing choreographies which guarantee conformance are subject of ongoing research.

References

1. Wil M. P. van der Aalst and Twan Basten. Inheritance of workflows: an approach to tackling problems related to change. *Theor. Comput. Sci.*, 270(1-2), 2002.
2. Wil van der Aalst and Kees van Hee. *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2002.
3. Tony Andrews et al. Business process execution language for web services (bpel4ws). ver. 1.1, BEA, IBM, Microsoft, SAP, Siebel Systems, 2003.
4. Alistair P. Barros, Marlon Dumas, and Phillipa Oaks. Standards for web service choreography and orchestration: Status and perspectives. In *BPM 2005 Workshops, LNCS 3812*, Springer, 2005.
5. Twan Basten. *In Terms of Nets: System Design with Petri Nets and Process Algebra*. PhD thesis, TU Eindhoven 1998.
6. Johann Eder and Wolfgang Gruber. A meta model for structured workflows supporting workflow transformations. In *ADBIS 2002, LNCS 2435*, Springer, 2002.
7. Assaf Arkin et al. Web service choreography interface (wsci) 1.0. W3C, 2002.
8. N. Kavantzis et al. Web services choreography description language (ws-cdl) 1.0. W3C, 2004.
9. Marek Lehmann. *Data Access in Workflow Management Systems*. Number 94 in DISDBIS. Akademische Verlagsgesellschaft Aka GmbH, Berlin, 2006.
10. Tadao Murata. Petri nets: Properties, analysis and applications. *Proc. IEEE*, 77(4):541–580, April 1989.
11. Chris Peltz. Web services orchestration and choreography. *IEEE Computer*, 36(10):46–52, 2003.
12. Rob J. van Glabbeek and W. P. Weijland. Branching time and abstraction in bisimulation semantics. *J. ACM*, 43(3):555–600, 1996.

Designing Web Services for Supporting User Tasks: A Model Driven Approach*

Marta Ruiz, Vicente Pelechano, and Óscar Pastor

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camí de Vera s/n, Valencia-46022, España
{mruiz, pele, opastor}@dsic.upv.es

Abstract. Web Service (WS) developers must take several decisions to design Service Oriented Architectures. Some of these decisions are: which operations should provide the involved Web Service, how we can detect these operations, in which way this can be automated and which is the right granularity of the operations. To address some of these questions, we present in this paper a methodological approach that extends the OOWS web engineering method. Following a model-driven approach, we systematically identify and design the operations of a Web service, taking as the source: (1) the tasks that the user must be able to achieve and (2) the OOWS conceptual models.

Keywords: Web Services Design, Service Oriented Architecture, Model Driven Code Generation, Web Engineering.

1 Introduction

Nowadays, the design of Web Services is emerging as one of the most important tasks in the development of a Service Oriented Architecture. Several questions arise during the design of WS: which operations should they provide, how we can detect these operations, in which way this detection can be automated and which is the right granularity of their operations. To address some of these questions, we provide some solutions in the context of a Web Engineering method called OOWS (Object Oriented Web Solutions) [9]. Our method allows identifying and designing the operations of Web services following a model-driven approach.

Web development should be adapted to support the systematic design of Web Services. A Web service should provide public operations with an appropriate granularity level in order to provide flexibility and to facilitate its connection and integration into distributed business processes over the Internet. They have emerged as important components of modern Web applications.

OOWS [9] proposes a model driven approach to develop web applications. This method integrates navigational design with a classical OO conceptual modelling providing systematic code generation (following the strategy proposed in OO-Method [8]).

* This work has been developed with the support of MEC under the project DESTINO TIN2004-03534 and cofinanced by FEDER.

The structure of the paper is the following: section 2 makes an overview of the related work in the Web Engineering methods. Section 3 presents the methodological guide to obtain well designed Web services in a SOA through the Amazon case study. In our approach, a set of operations are identified taking the task diagrams and the OOWS conceptual models as a source. The operations are obtained in a model driven fashion. Section 4 presents an evaluation of our proposal. In this section we compare the operations that are obtained following our strategy with those published by Amazon. We conclude that our proposal can produce operations that are functionally equivalent to those provided by Amazon. Finally, we present some conclusions and further work in section 5.

2 Related Work

Current Web Engineering methods provide methodological guides for developing Web applications. Some of these methods have been adapted to support the development of Web services-based applications. We can distinguish approaches like OOHDM [12] and UMLGuide [2] that introduce some kind of syntactic mechanism to include Web services calls into navigational models but do not give support to their design and development. Other methods like OO-H [4] allow Web developers to generate Web services but they neither give support to their design. Finally, WebML [5] captures Web services invocations by means of visual representations. They describe the Web interactions using hypertext models but do not follow any strategy or guide to systematically design Web services.

This work introduces some contributions compared to other Web Engineering methods because we provide a methodological guide that helps Web designers in the design and development of Web services. We also want to emphasize that this methodological guide allows to systematically define Web services from the web conceptual models in a model driven fashion. The operations published by these Web services are obtained extracting all the relevant knowledge that the models include. Our approach is mainly based in detecting operations from task descriptions. Tasks descriptions are focused on specifying human interaction processes, in this way those operations give complete support to the implementation of web applications. We know that web services are not only conceived to be technology that allows the implementation of the server side of a web application. Web services also should interact with other software systems or services rather than with human users. In this work we are providing a partial view that is being complemented in the following work [14]. These works deal the design of complex interaction processes and the integration and interoperability with third parties.

3 A Methodological Guide to Design Web Services

This section presents the main contribution of the proposal: a methodological guide that allows us to obtain the operations that implement the requirements of a Web application in a SOA. To illustrate the application of our proposal we are going to use the Amazon case study.

3.1 Proposal Overview

OOWS [9] is a web engineering method that provides methodological support for the Web application development. In order to build a web application, OOWS proposes a development process that is divided into three main stages (see figure 1): (1) *User identification*: we identify which kinds of users are going to use the final web application. (2) *Task description*: we detect and describe which tasks must achieve each kind of user when interacting with the web application. This allows us to elicit and specify which requirements must satisfy the web application in order to give support to the user's needs. (3) *Conceptual modelling*: we describe at conceptual level the web application that gives support to each user requirement detected in the previous stage. To do this, the web application is described by using different abstract mechanisms that allow us to build web applications in an implementation independent way.

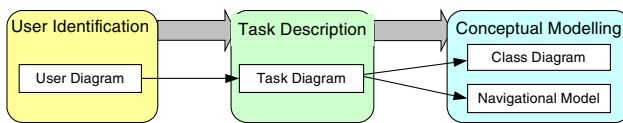


Fig. 1. OOWS Development Process

Analyzing these models and taking into account the kind of requirements that they capture, our proposal identifies a set of operations that define the public interface of the Web services. The OOWS approach is an extension of the OO-Method proposal [8] that provides a method with automatic code generation. A commercial CASE tool called ONME [6] generates the application logic in the form of EJB and .NET Components. The operations of the web services make use of the generated logic to provide the functionality to the web application.

Our proposal follows the next steps: (1) the first step analyzes the *task descriptions* (from the Task Diagram) and identifies the operations (and their arguments) that implement the functional requirements of a Web application. The arguments of the operations are extracted from the *information templates* and the *Class Diagram*. (2) In the second step we analyze the *OOWS Navigational Model*. From the navigational context specification we can identify operations to retrieve the information that must be shown in a web page.

3.2 Identifying Operations from the Task Descriptions

In the task description step, a Task Diagram is defined for each kind of user [15]. In this diagram, we describe in a hierarchical way which tasks the user can achieve by interacting with the system. To do this, we use the CTT (ConcurTaskTree) approach [10]. The operations detected from the task diagram provide operations to implement the functional requirements of a Web application. A task diagram identifies user goals and the activity a user performs to achieve these goals. Therefore, we think that this diagram can help us to identify the public operations that must offer our service in order to build Web applications that completely support user tasks. The potential “users” can be persons, Web pages or other external applications.

To provide a method that automatically obtains the operations that must publish the Web service, we have analyzed some works. These works come from two different areas: from the HCI community ([3], [7] and [13]) and from Web engineering methods ([1]). Analyzing these works we can state that the types of the tasks provide relevant information in order to choose which operations must publish our Web service. CTT categorizes tasks into four types that are determined by the way the tasks are accomplished. Next, we analyze these tasks taking into account if they can be candidates or not to generate the operations of the Web service:

-  **User tasks** are performed by the user without interacting with the system. Therefore, this type of task does not have any sense to be selected as a candidate Web service operation.
-  **Application tasks** are completely executed by the application. Hence we can conclude that this type of task does not need any interaction and it has no sense to be provided as a candidate operation.
-  **Interaction tasks** are performed by user interacting with the application. Even though they can be used as operations offered by a Web service, they will be generally fine grained operations.
-  **Abstract tasks** consist of complex activities whose subtasks are a mixing of user, application and interaction tasks. These tasks are the most recommendable to be candidate operations in the Web service because this mixture. In theory, an abstract task must have different subtasks' categories, but in most cases this rule is not applied.

Taking into the account these conclusions, we present a process that can be followed to automatically generate the operations. In the first place, we should traverse the task diagram in preorder from the root until the last leaf. It starts at the root node, traversing the left-hand subtask and then the right-hand subtask (e.g. in figure 2 we visit the task *Purchase Product*, then *Collect Items - Fill Shopping Cart - Select Product - View List - View Product Info* and so on). When we select a task as a candidate operation to be published, we move to its right-hand subtask to traverse this side of the task tree. For example if we select the task *Inspect Shopping Cart*, then we move to *Modify Shopping Cart* task to traverse this subtree.

While we traverse the task diagram, we should focus on those abstract tasks: (1) that have at least one interaction, application or user subtask or (2) that are a leaf task. To do this, we traverse the task tree looking for *interaction*, *application* or even *user* tasks and taking their parent as a candidate operation to be published in the Web service. If some of the tasks of the candidate's level are not abstract classes, then we backtrack to the tasks selected in this level and we take the parent task of them as a candidate operation.

The algorithm that implements the previous process is presented next. First, it is necessary to state that the input is a non-empty tree and the output is a list of operations (OP) detected from this tree. We call `node` to the current node of the tree; `node.name` is the name of the node; `node.child` is the collection of the subtrees' children of the current node and finally `node.parent` is the parent of the current node. Moreover we have two helper functions: `isAbstract` that returns true if the node is an abstract task and `isLeaf` that returns true if the task has no child.

Algorithm TaskDiagramOperations

```

if isAbstract(node) then
  if isLeaf(node) then
    OP.add(node.name)
  else
    for (i=0; i<count(node.child); i++)
      TaskDiagramOperations(node.child(i))
    endFor
  else
    OP.add(node.parent.name)
  endIf

```

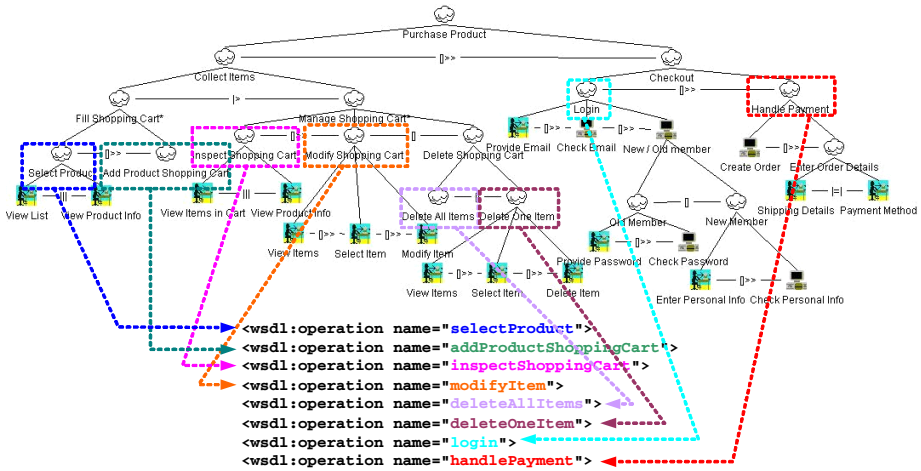


Fig. 2. Operations detected in the task diagram

In figure 2 we have the description of how a Client can buy a product. Following the algorithm previously presented, we traverse the task diagram showed in figure 2. We start at the *Purchase Product* task traversing the left-hand subtree; we then visit *Collect Items*, follow by *Fill Shopping Cart* and *Select Product* and finally, because we visit the interaction task *View List*, we select the abstract task *Select Product* as a candidate operation. We now start at the right-hand subtree of *Fill Shopping Cart*, that is *Add Product Shopping Cart* task, and we select it because it is a leaf task. At this moment we go after the task *Manage Shopping Cart*, afterward *Inspect Shopping Cart* and finally the interaction task *View Items in Cart*, so we select the abstract class *Inspect Shopping Cart* as a candidate operation. After that, we go for the task *Modify Shopping Cart* and successively the interaction task *View Items*. Thus the abstract task *Modify Shopping Cart* is selected as a candidate operation. Then we visit *Delete*

Shopping Cart task and we go to the leaf abstract task *Delete All Items* selecting it as candidate operation. To conclude this part of the tree we visit *Delete One Item* task and then the interaction task *View Items* selecting the abstract task *Delete One Item* as a candidate. Now we traverse the right-hand subtree. We start visiting the task *Checkout* followed by *Login* and the interaction task *Provide Email*, so we select the abstract task *Login* as a candidate. Finally we visit the task *Handle Payment* and the next task visited is an application task named *Create Order*, so we select its parent as a candidate operation.

To summarize, we define eight operations from the task tree in figure 2: `selectProduct`, `addProductShoppingCart`, `inspectShoppingCart`, `modifyShoppingCart`, `deleteAllItems`, `deleteOneItem`, `login` and `handlePayment`.

The operations detected do not have arguments. Arguments can be detected from *information templates* (IT). Valderas et al. [15] in addition to define a task diagram for each kind of user, complement task hierarchical representations with the description of the interaction that the user requires from the system in order to achieve each task. They propose the definition of an *IT* for each entity that has been identified in the description of an elementary task achievement. Using this *IT* together with the *class diagram*, we can obtain the arguments of each operation matching the entities that are specified in the IT with the class diagram. For each identified class we add its type to the arguments of the operation that has been detected from this branch of the task tree.

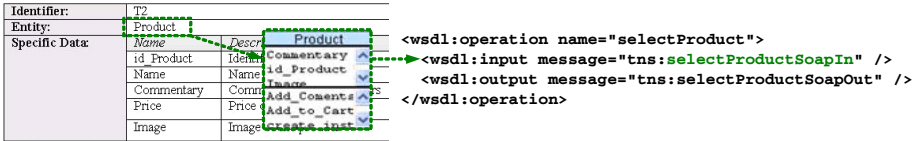


Fig. 3. The arguments of the `selectProduct`

Figure 3 shows the arguments for the operation `selectProduct`, which are taken from the IT of the tasks *View List* and *View Product Info* because they are the children of the *Select Product* task. These IT have the entity *Product* which maps to the *Product* class in the class diagram. So we add an argument of *Product* type to the arguments of the operation `selectProduct`. Table 1 shows the (input and output) arguments following the strategy commented previously.

Table 1. Arguments for the operations detected from the task diagram in figure 2

Operation	Input	Output
addProductShoppingCart	Product and ShoppingCart	
inspectShoppingCart	ShoppingCart	Product
modifyShoppingCart	ShoppingCart and Product	Product
deleteAllItems	ShoppingCart	
deleteOneItem	ShoppingCart and Product	Product
Login	Email, Passport and Client	
handlePayment	Order and Client	

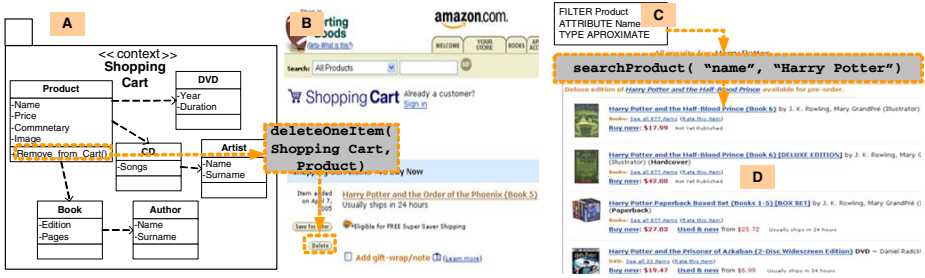


Fig. 4. The implementation of `deleteOneItem` and `searchProduct`

Figure 4-A shows the Navigational Context *Shopping Cart* defined in Amazon. In this context the user can see all the products that are in his/her cart. In addition, the system allows the user to activate the operation `deleteOneItem` in order to remove a product from his/her shopping cart. Figure 4-B shows the Web page that gives support to the execution of this operation.

3.3 Operations Identified from the OOWS Navigational Model

In the conceptual modeling step, we define a web conceptual schema that gives support to the tasks identified above. The OOWS Navigational Model is defined from a set of navigational maps (see figure 5-A) that describe the navigation allowed for each kind of user. The navigational contexts (see figure 5-B) show us the instances that we need to retrieve, the relationships between them and how the user wants to get all these information. Thus, we can identify operations to retrieve the information that must be shown in each navigational context (a web page in the running example).

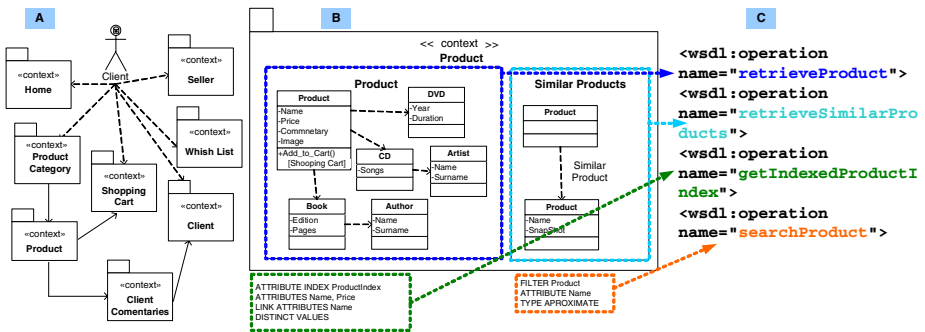


Fig. 5. Navigational Map and Navigational Context *Product*

For each navigational context, we can define:

1. The `retrieveViewName([attributeID])` allows obtaining the information specified in the navigational context views. It returns (1) a set of instances or (2) a specific instance of the classes defined in the context view. We define an

operation of this kind for each view defined in a navigational context. Figure 5-C shows the operations detected from the context *Product*: `retrieveProduct` and `retrieveSimilarProducts`.

2. The `getIndexedIndexName(attributes)` operation gives support for the *index mechanisms*. This operation returns the list of the summarized information allowing the user to choose one item (instance) from the list. It is defined for every index in a navigational context. Figure 5-C shows the operation `getIndexed-ProductIndex` identified from the index of the *Product* context.
3. The `retrievePopulationViewName (attribute*)` operation gives support to *population condition mechanisms*. This operation returns a set of instances that fulfil the condition specified by the user. We define an operation of this kind for each population filter defined in a navigational context.
4. The `searchFilterName(attribute, value)` operation gives support to *search filters mechanisms*. This operation returns the set of instances that fulfil the search conditions specified by the user. An operation is obtained for each filter of a navigational context. Figure 5-C shows the `searchProduct` operation identified from the filter defined in the *Product* context.

Figure 4-D shows an Amazon Web page in which the result of the operation `searchProduct` can be seen (defined from the filter of the context *Product* in figure 4-C). The filter condition is that the name of the product has the word “Harry Potter”. The web page shows all the products of Amazon related with “Harry Potter”.

4 Evaluation

In this section we compare the operations published by the AWS¹ with the operations obtained following our strategy for the Amazon case study. Amazon offers 17 operations and our approach offers 19. Next, we present those operations (following our approach) that give support to implement each Amazon operation.

- `BrowseNodeLookUp` is supported by `retrieveProductCategory` detected from a view defined in the *Product Category* context
- `CartAdd` is supported with `addProductShoppingCart`.
- `CartClear` is supported with the operation `deleteAllItems`.
- `CartCreate` is implicitly implemented in our `addProductShoppingCart`.
- We implement `CartGet` with `inspectShoppingCart`.
- `CustomerContentLookup` is implemented with `retrieveClientComments` detected from a view defined in the *Client Commentaries* context.
- `CustomerContentSearch` is supported by `retrieveClient` detected from a view defined in the *Clients* context.
- `Help` is not supported because we have not captured this requirement in our web conceptual model. This operation just provides a user manual.
- We support `ItemLookup` with `retrieveProduct`.
- `ItemSearch` is supported by `searchProduct`.

¹ www.amazon.com/gp/aws/landing.html

- We implement `SimilarityLookup` with `retrieveSimilarProducts`.
- `SellerLookup` is supported by `retrieveSeller` detected from a view defined in the *Sellers* context.
- `SellerListingLookup` and `SellerListingSearch` are related to the integration of Amazon with Third Party systems which is out of the scope of this work. Information about this can be found in [14].
- `TransactionLookup` is not supported because we have not considered information about financial operation in the web conceptual model specification.
- `ListLookup` is supported by `getIndexedProductIndex`.
- We implement `ListSearch` with `searchWishList` detected from a filter defined in the *Whish List* context.

The operations detected following our approach give support to implement 14 AWS operations. Furthermore, our approach detected several operations which do not exist in the AWS: `deleteOneItem`, `handlePayment` and `login` detected from the task diagram; and `retrieveHome` from the context *Home*. These operations are not provided by the ASW but they are accessible by means of their Web site. We want to note that ASW does not allow the confirmation of the order by means of the AWS. Client applications must login in the Web site and then confirm the order.

As a conclusion, we can see that our proposal offers a solution that is functionally equivalent to Amazon and it includes additional functionality. This additional functionality is necessary to achieve the final goal of the Amazon case study, that is, purchase a product. The weak point of our proposal is the integration of Amazon with Third Party systems, but this is a work in progress [14].

5 Conclusions and Future Work

This article has presented an approach to design Web services (WS) for supporting user tasks and it is based on OOWS navigational contexts. We have presented a methodological guide to obtain the operations that define the WS. This methodological guide can be generalized to other Web Engineering Methods, because the OOWS method shares with them the most common models and primitives taken as source to obtain the WS.

Three Web sites have been successfully developed using the methodology presented in this paper: the OO-Method Group Web Site (<http://oomethod.dsic.upv.es>), the intranet management system for the General Library of the Technical University of Valencia and the DSIC Department Web Site (<http://www.dsic.upv.es>). In all these cases, we have defined first the kinds of users that interact with the system in the user identification stage.

Several extensions to this work are under development. We are working on providing mechanisms that facilitate the integration of Web applications with Third parties systems at the conceptual level [14]. When third parties supply us their functionality as Web services, we proceed to the Web services composition to achieve integration. Another line of research concerns to the automatic implementation of the Web services logic in the OlivaNova CASE Tool [6]. We also want to enrich the code generation process to provide Web Services that include presentational aspects [11]. For

this purpose we have planned to use the presentation model defined within the OOWS approach. Our final objective is to generate reusable Web functionality that provides relevant information (content and presentation) to potential consumers.

References

- [1] De Troyer, O., Casteleyn, S., Modeling Complex Processes for Web Applications using WSDM. IWWOST (2003)
- [2] Dolog, P., Model-driven navigation design for semantic web applications with the UML-guide. In Maristella Matera and Sara Comai (eds.), Engineering Advanced Web Applications. (2004)
- [3] Er-Jongmanee T., XML-Driven Device Independent User Interface Build Rich Client Applications Using XML. Master Thesis (2005)
- [4] Gómez, J., & Cachero, C. , OO-H Method: extending UML to model web interfaces. In information Modeling For internet Applications. P. van Bommel, Ed. Idea Group Publishing, Hershey, PA, (2003). pp. 144-173.
- [5] Manolescu, I., Brambilla, M., Ceri, S., Comai, S., & Fraternali, P., Model-driven design and deployment of service-enabled web applications. ACM Trans. Inter. Tech. 5, 3, (2005) pp. 439-479.
- [6] OlivaNova Model Execution System. CARE Technologies. <http://www.care-t.com>.
- [7] Paquette, D., Schneider, K. A.: Interaction Templates For Constructing User Interfaces From Task Models. CADUI (2004) pp. 221-232
- [8] Pastor, O., Gomez, J., Insfran, E., & Pelechano, V.: The OO-Method approach for information systems modelling: from Object-Oriented conceptual modeling to automated programming. Information Systems 26, (2001) pp 507-534.
- [9] Pastor, O., Fons, J., Pelechano, V., & Abrahão, S. (2005). Conceptual modelling of Web applications: the OOWS approach, book chapter in Web Engineering - Theory and Practice of Metrics and Measurement for Web Development, Mendes E. (Eds.), Springer 2005, pp. 277-302.
- [10] Paternò, F., Mancini, C., & Meniconi, S. (1997). ConcurTaskTrees: a diagrammatic notation for specifying task models. In Proceedings of INTERACT'97, Chapman & Hall, 368-366.
- [11] Rojas, G., Pelechano, V., & Fons, J., A Model-Driven Approach to include Adaptive Navigational Techniques in Web Applications. V International Workshop on Web Oriented Software Technologies - IWWOST'05, Porto, Portugal. (2005).
- [12] Schwabe, D., Rossi, G. & Barbosa, D.J., Systematic hypermedia application design with OOHDM. Proc. ACM Conference on Hypertext. pp.166. (1996).
- [13] Sinnig, D., Forbrig, P., Seffah, A.: Patterns in Model-Based Development. INTERACT. (2003).
- [14] Torres, V., Pelechano, V., Ruiz, M., & Valderas, P., A model driven approach for the integration of external functionality in Web applications. The Travel Agency System. Workshop on Model-driven Web Engineering (MDWE). (2005).
- [15] Valderas, P., Fons, J., Pelechano, V., Developing E-Commerce Application from Task-Based Descriptions. 6th International Conference on Electronic Commerce and Web Technologies (EC-WEB). (2005)

Designing Service-Based Applications: Teaching the Old Dogs New Tricks ... or Is It the Other Way Around?

Vojislav B. Mišić

University of Manitoba
Winnipeg, Manitoba, Canada

Abstract. Service orientation is rapidly gaining acceptance as the dominant development paradigm for software applications. The current lack of methodologies suitable for the design and development of such applications is usually attributed to significant differences between service-oriented applications and more traditional ones: monolithic, component-based, and web-based ones. While some differences indeed exist, they are not substantial, and many among the old tricks—i.e., existing enterprise modeling and development techniques—can effectively be reused in the service-oriented context.

Service orientation: Businesses both small and large are more and more dependent on accurate, dependable, and timely functioning of various software systems, the complexity of which has been steadily increasing. A number of techniques were introduced over the years to manage this complexity, the most recent one being the approach known as service orientation or service oriented computing [3,5,8]. In this approach, software entities are identified with the service(s) they provide (with the desired functionality and performance characteristics) through standardized interfaces. While other, traditional paradigms rely on externally observable functionality as well, service orientation takes this reliance to the extreme. It uses services as the most basic units through which software functionality can be designed, developed, and used a dynamic, on-demand fashion. In this manner, it allows the development of software systems as loosely coupled associations of services that continuously adapt to the ever-changing business and quality goals.

The new hype: As has become customary in such cases, the service computing paradigm is generally touted as something completely different, a revolutionary change in the way software applications are designed, developed, and deployed. In that respect, service orientation is not that different from object orientation, component-based development, or electronic commerce, all three of which have enjoyed similar hype in their own times. (It is perhaps worth noting that service orientation does exhibit non-negligible overlap with all three of those approaches.) The hype thus created is simultaneously good and bad for further

advance of the paradigm and the community in general. On one hand, it attracts attention and helps create the momentum which is necessary to develop, assess, and exploit the benefits of the new paradigm. On the other hand, the (real or imaginary) perception that the paradigm is radically different from everything else is often taken to mean that all existing knowledge about software development is to be thrown away and the new body of knowledge is to be built from scratch. The risk incurred in this manner is twofold: first, existing knowledge which may be usable either ‘as is’ or, perhaps, with appropriate adjustment, is simply ignored. Second, the process of building a cohesive and comprehensive development approach takes time and effort which might be put to better use.

If standards are the answer, what is the question? These risks are further aggravated by the fact that the main supporting pillar of service oriented computing—and, indeed, its main selling point—is the interoperability between heterogeneous applications and systems. Interoperability requires standards, and it appears that the proliferation of standards to support it has been the favorite topic in both research and industry. Although quite a few service-related standards have been defined by various industry consortia and independent standards organizations, a number of important areas were left aside [9]. This includes supporting infrastructure [6] and architecture and architecture description languages [1], but also the crucial area of methods and techniques for modeling and design of service-oriented applications. Without such methods, the multitude of existing standards—which just state HOW to do something but not WHAT is to be done—will remain fairly ineffective tools to use in the development of service-oriented applications.

Reuse is the key . . . Instead of throwing away all that we know, we argue that most of the knowledge about systems development still holds, and that existing knowledge about modeling and systems development can easily be reused in the service-oriented context. (The venerated entity-relationship modeling is, after all, the basis for most, if not all, of the service-related standards, despite the clever disguise of XML and XML-compliant languages.) Reuse is also present in several attempts to apply well developed techniques and formalisms such as workflow modeling [4], hierarchical statecharts and other flavors of UML [2], architectural description languages [7], and Petri nets [10], to name just a few. But reuse can be attempted at other levels and with other techniques, including a few well known ones for layering and decomposition, analysis, and design at the conceptual level. Let us provide just one example: enterprise application are often modeled in a layered fashion, with service layer located between (and connecting) the business logic and application logic layers [5]. Then, analysis tasks include defining what services need to be built; defining what functionality (i.e., piece of business logic) should be encapsulated within each service; identifying the pieces of this logic that have the potential for reuse; and devising the way in which the services identified in the previous analysis can (or should) be interconnected to provide the desired functionality. Yet a number of questions both technology- and business-related are left unanswered: for example, *how* exactly the system should be decomposed into

layers, or *which* services are available from third parties. Many of these questions can be answered in the development process undertaken using existing methodologies.

... *but in creative way*: This is not meant to say that existing methodologies will solve all the problems in the process of service-oriented system design. There are quite a few aspects of this process that are specific to service-oriented systems, and there are quite a few legitimate topics for research and development, both in academia and in industry. However, the developers should carefully look into their method chest before trying to do everything from scratch. In this manner, we can leverage our existing knowledge and make the transition to service-oriented systems as fast and as painless as possible.

To summarize: we argue that (a) high-level conceptual design techniques are a must for successful development of service-oriented applications, and (b) existing techniques for systems development can be put to good use in the design of such applications, possibly with minor adjustments to cater to the peculiarities of the paradigm. Only then we can leverage the power of standards and fully develop the potential of the service oriented computing paradigm.

References

1. L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. The SEI Series in Software Engineering. Addison-Wesley, Reading, MA, 2nd edition, 2002.
2. B. Benatallah, M. Dumas, Q. Z. Sheng, and A. H. H. Ngu. Declarative composition and peer-to-peer provisioning of dynamic web services. In *Proc. 18th IEEE Int. Conf. Data Engineering ICDE'02*, pages 297–308, San Jose, CA, Feb. 2002.
3. N. Bieberstein, S. Bose, M. Fiammante, K. Jones, and R. Shah. *Service-Oriented Architecture (SOA) Compass : Business Value, Planning, and Enterprise Roadmap*. IBM Press, October 2005.
4. R. M. Dijkman and M. Dumas. Service-oriented design: a multi-viewpoint approach. *International Journal of Cooperative Information Systems*, 13(4):337–368, 2004.
5. T. Erl. *Service-Oriented Architecture : Concepts, Technology, and Design*. Prentice Hall PTR, August 2005.
6. E. Maximilien and M. Singh. A Framework and Ontology for Dynamic Web Services Selection. *IEEE Internet Computing*, 8(5):84–93, 2004.
7. V. B. Mišić and M. W. Rennie. Describing the architecture of service-oriented systems. In *Proc. First International Workshop on Design of Service-Oriented Applications WDSOA'05*, pages 9–16, Amsterdam, The Netherlands, Dec. 2005. IBM Research Report RC23819 (W0512-029).
8. M. P. Singh and M. N. Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. John Wiley & Sons, New York, NY, 2005.
9. W. van der Aalst. Don't go with the flow: Web services composition standards exposed. *IEEE Intelligent Systems*, 18(1):72–76, 2003.
10. J. Zhang, C. Chang, J. Chung, and S. Kim. WS-net: A Petri-net Based Specification Model for Web Services. In *Proceedings IEEE International Conference on Web Services*, pages 420–427, San Diego, CA, July 2004.

Preface for ECDM 2006

Fabio Grandi

University of Bologna, Italy

Change is a fundamental but sometimes neglected aspect of information and database systems. The management of evolution and change and the ability for database, information and knowledge-based systems to deal with change are an essential component in developing and maintaining truly useful systems. Many approaches to handling evolution and change have been proposed in various areas of data management and this forum seeks to bring together researchers and practitioners both from more established areas and those from emerging areas to look at this issue. The fourth ECDM workshop (the first three ECDM workshops were held with ER 1999 in Paris, France, with ER 2002 in Tampere, Finland, and with ER 2004 in Shanghai, China) dealt with the manner in which change can be handled, and the semantics of evolving data and data structure in computer-based systems, including databases, data warehouses and Web-based information systems.

With respect to the main ER conference, the ECDM workshop series aim at stressing the evolutionary aspects involved in conceptual modeling and in development and implementation of systems, ranging from the modeling of information dynamics to the dynamics of the modeling process itself. Another explicit aim of ECDM workshops is to bring together scientists and practitioners interested in evolution and change aspects in different research fields and, thus, often belonging to completely separate communities. As a result, such an interaction could become tighter and cross-fertilization more useful in the context of a collaborative workshop like ECDM.

Following the acceptance of the workshop proposal by the ER 2006 Organizing Committee, an international and highly Qualified Program Committee was assembled from research centers worldwide. As a result of the call for papers, the Program Committee received 11 submissions from 9 countries and after rigorous refereeing 5 high-quality papers were eventually chosen for presentation at the workshop and publication in these proceedings. Furthermore, the workshop program was enriched by a keynote speech given by John Roddick, who was also the founder of the ECDM series and (Co-)Chair of the first two editions.

I would like to express my thanks to the Program Committee members and the additional external referees for their timely expertise in reviewing, the authors for submitting their papers, and the ER 2006 organizers for their support.

Reduce, Reuse, Recycle : Practical Approaches to Schema Integration, Evolution and Versioning

John F. Roddick and Denise de Vries

School of Informatics and Engineering
Flinders University,
PO Box 2100, Adelaide, South Australia 5001
`roddick@infoeng.flinders.edu.au`

Abstract. Three themes are apparent in recent schema integration, evolution and versioning research. First, the need to reduce the number of schema changes that are necessary. The approach here has been to build into the conceptual and data models the scope to accommodate modest changes to definition. Second, research that aims to reuse the current schema definition through procedures that mask the changes through sophisticated wrappers or techniques for multiple extensional data. Finally, techniques that enable schema change to be accommodated as seamlessly and as painlessly as possible. All these approaches have their limitations and strengths. This paper investigates each of these approaches and outlines the current research directions in schema integration, evolution and versioning.

1 Introduction

Changes in the functional requirements for software systems commonly result in changes to the underlying database. Similarly, changes to the storage schema for a database often necessitates amendments to the software systems to accommodate the changes. Aside from any data loss that may result, these changes cause both application and database to age resulting in higher maintenance and redevelopment costs [1]. Schema evolution and versioning research thus aims to reduce the effect of changes to schema on the system. Schema versioning is related to schema evolution in that it aims to provide not only evolution (change) capability but also the provision of multiple schema instances for data update and retrieval.

Schema versioning requires data integration which takes data held under multiple local schemata (perhaps different versions) and provides a unified view of those data, while schema (or view) integration takes multiple user views and provides a common conceptual schema. Thus in many respects, schema and data integration and schema evolution and versioning can be viewed as cognate problems in that there are multiple sources of data which are held, and which need to be viewed, on demand, through multiple schemata [2].

Research in these fields has been relatively steady now for two decades, as illustrated in Figure 1 and can broadly be classified under a *3R-hierarchy* which is broadly speaking, in order of desirability:

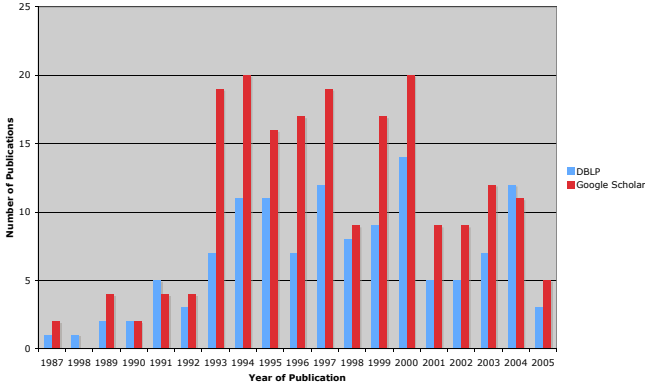


Fig. 1. Approximately publications by year in schema evolution and schema versioning (source DBLP, Google Scholar)

- **Reduce.** Those strategies that attempt to reduce the number of database schema changes that need to be made by building into the conceptual and data models the capability to accommodate changes to definition.
- **Reuse.** Those strategies that reuse the schema definition (and the data held under that definition) by providing translation mechanisms, such as wrappers.
- **Recycle.** Those strategies that ease the pain of integration and evolution, including techniques for data coercion.

This paper explores those three themes. Given the surveys that are available [3,4,5,6,7], the paper focusses specifically on techniques that have been developed in the past few years. Despite useful work in a variety of other areas (see for example [8,9,10]) we limit ourselves here to a discussion on those approaches to database schema management.

2 The Practical and Theoretical Limits of Schema Evolution

As discussed by Qian [11], Miller *et al.* showed that in order to update data stored under two different schemata using the opposite schemata, theoretically they must be equivalent – all valid instances of some schema S_1 must be able to be stored under S_2 and vice-versa [12,13]. Specifically, $S_1 \equiv S_2$ iff $I(S_1) \rightarrow I(S_2)$ is bijective where $I(S)$ is the set of all valid instances of S – essentially the *information capacity* of S . This means that, in theory, *full* schema versioning is unattainable and much research in the area adopts the weaker concept of *partial schema versioning* in which data stored under any historical schema may be viewed through any other schema but may only be updated through one specified schema version - normally the current or active schema. This is essentially the old *view update* problem.

However, in practice, many schema changes that expand or reduce the information capacity of a schema can be done without loss of information. This is the case for example, for domains defined too large for the data, for the creation of subclass relations from a single relation where the subclass type attribute already exists, and so on. It is common practice where there is some ambiguity in the requirements definition of a system to allow for data that once implemented never materialises and, as a result, changes to schemata to adhere to the data collected are not uncommon [7].

Thus the limits for *practical schema versioning* in a database $\mathfrak{D}$ are that $S_1 \stackrel{p}{=} S_2$ iff $I'(\mathfrak{D}|S_1) \rightarrow I'(\mathfrak{D}|S_2)$ is bijective where $I'(\mathfrak{D}|S_n)$ is the set of all instances of S_n inferable from $\mathfrak{D}$ given the constraints of S_n . This means that whether the integration of two schema is possible is dependent on the data held as well as the schema definition and while this makes the ability to undertake wholesale change less predictable, it may provide an acceptable level of support in many practical situations. A number of schema matching techniques have been suggested [14] and the ones detailing instance matching [15,16] may be able to be modified to determine whether instance level equivalence exists. The use of Armstrong relations [17] or induced dependencies [18] may also assist. This is an area for further research.

3 Reducing the Requirement for Schema Change

Since it is changes that result in system and schema modification and thus in the ageing of a system, one approach is to avoid the need for change by developing schemata that can handle a modest level of change through data modification. One approach which has been proposed as a solution to this is the concept of mesodata [19,20]. Mesodata aims to add power to the relational data model by providing greater semantics to the *domain* (note, not the type) of an attribute by allowing attributes to be defined over domains with complex data structures. The mesodata concept does not add complex data types to the relational model – the *type* of an attribute remains a simple scalar type while the *domain* of the attribute allows the values taken by the attribute to be placed within some complex structure. For example, while the code for a disease might be defined as CHAR(5), disease codes exist within an agreed international classification (such as ICD10) a tree-structure that relates diseases and other observations by group.

When a schema changes two events typically occur - the application is modified and recompiled to deal with the changes and the data may be converted to a new format [21,22]. By adding a mesodata layer to the structure, the domains over which the data are defined may be modified leaving untouched both the data and the attribute definition. Even simple country, organisation or region name changes may be one-to-one, one-to-many, many-to-one or many-to-many and more complex changes might involve the reorganisation of the entity.

The typical solution is to convert all old values and replace them with new values. This could be an ongoing task and results can result in an irreversible loss of information. The mesodata solution would be to use the mesodata type TREE

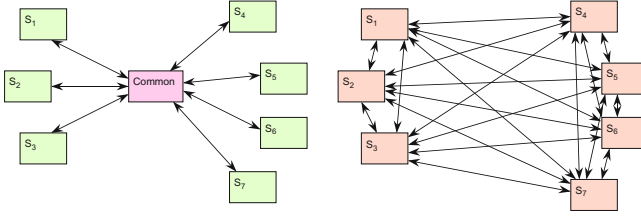


Fig. 2. Basic Architectures for Schema Reuse

to map old values to the new values. Extended SQL operators could be used to retrieve data values with the advantage that the original values are not lost.

4 Reusing Schema Definition and Data

Coercing data to a common schema is often not a practical solution and thus the largest group of techniques focusses on retaining data and schema definition and providing interfaces using either indirect *Source-Common-Target* (SCT) [23,24] or direct *Source-Target* (ST) architectures. The SCT architecture benefits from requiring fewer interfaces than the SC architecture ($2n$ as opposed to $n^2/2$, where n is the number of schemata) but can suffer from a greater level of information loss as the common schema is often non-maximal. Wrappers [25,26] and XML-based markups are effectively a version of the SCT approach in which the interface provided adheres, at least in principle, to a more or less agreed commonality of schema.

Implicit in some work in schema versioning is the concept of a completed schema C (similar to the completed relation of Clifford and Warren [27]) which contains the minimal union of all attributes which have been defined during the life of the relation. The domain of each attribute in C is considered syntactically general enough to hold all data stored under every version of the relation scheme R and the implicit primary key of C is defined as the maximal set of key attributes for the scheme over time. Versions of the schema (denoted S_{time}) can be seen to be views of C . Thus the relation scheme active during the interval t_1 , $R_{t_1} = (S_{t_1}, T)$. The current schema through which updates may be performed is denoted S_{now} . A view function V_{t_1} maps C to a subset of the attributes in a schema S_{t_1} active during t_1 with a converse function W_{t_2} mapping from S_{t_2} to C .

Grandi [28,29] provides a solution that enhances this view by providing a *multi-pool* solution. In this proposal, each schema version can have different extensional data and thus the same object might have an independent representation.

Rosenthal et al. [30] recommend that we must go beyond after-the-fact semantic integration to actively guiding semantic choices with semantic management by:

- producing new areas of semantic agreement not just correspondences between existing systems,
- considering the needs of people in multiple roles (e.g. owners, architects, users, developers) to have a greater share of what the data mean,

- broadening the definition of *semantics* to describe what data instances are collected and desired, not just concept definitions and relationships.

For temporal databases, which lend themselves well to schema versioning through the provision of a temporally-aware schema, data coercion is inapplicable due to their write-once, append-only nature. Work in this area has included Jensen and Böhlen [31,32] who discuss two versions of a lossless method of schema versioning – first, a full but exponentially complex, multi-dimensional conditionally evolving schema (MD-CES) and, secondly, a restriction on this that is tractable for one and two-dimensional temporal datasets.

5 Strategies for Accommodating Schema Modification

As would be expected, the desire to retain information means that data coercion is avoided in many proposals. However, it is a popular practical solution and, although not accommodated in some models, we would argue that there are some circumstances in which data coercion would be the most desirable mechanism of evolution, particularly when a consistent interpretation of the source data is required and/or there is confidence that historical information will not be required.

Early approaches to data coercion investigated either a strict conversion [33] or a lazy conversion [22] of data to the new format. Recent work has improved on this by providing a more detailed view of how this may occur. The TVSE Model [34], for example, uses temporal and versioning concepts to manage schema evolution in object-oriented databases. The model assists users to manage the evolution history of both intensional and extensional databases and proposes a language to derive and modify schema versions and to update the data associated with them.

It is also useful if the decisions made during database design can be consulted and the users can interact with schema changes at a high level. Hick and Hainaut [35] investigate how requirements changes can be propagated to database schemas, to data and to programs through reference to the design process. Other work, particularly that of Bernstein et al. [36,37] and Madhavan and Halevy [38], provides support for meta data management that provides a higher level view in which models can be mapped to each other.

6 Conclusions

Despite the research to date, schema integration, evolution and versioning are far from being solved as evidenced by the almost total lack of functionality in commercial DBMS. This lack of support may be for a number of reasons. First, the lack of strategies for dealing with schema versioning in practice which may be because the motivation for schema change is seldom examined and, given many possible solutions, the best outcome is often related to the motivation for change. Second, the need for manual intervention and the incompleteness of most solutions. Many solutions deal with specific situations only and most proposals, if implemented, would require at least a check that the process has provided an

adequate solution. Finally, many solutions target the object-oriented (cf. [39,40]) rather than the (commercially more popular) relational model cf. [28]).

We would argue that one of the next steps in schema handling is to transfer the body of research from theory to practice including extensions to commercial DBMS and SQL. To do this the user's motivation needs to be involved at a more fundamental level with, for example, links back to the original conceptual design.

References

1. Lientz, B.: Issues in software maintenance. *ACM Computing Surveys* **15** (1983) 271–278
2. McBrien, P., Poullovassilis, A.: Schema evolution in heterogeneous database architectures, a schema transformation approach. In: CAiSE'02, Birkbeck College and Imperial College (2002)
3. Lautemann, S.E.: An introduction to schema versioning in OODBMS. *Database and Expert Systems Applications*, 1996. Proceedings., Seventh International Workshop on (1996) 132–139
4. Lemke, T.: Schema evolution in OODBMS: A selective overview of problems and solutions. Technical Report IDEA.WP.22.O.002, University of Bonn (1994)
5. Li, X.: A survey of schema evolution in object-oriented databases. In: 31st International Conference on Technology of Object-Oriented Language and Systems, Nanjing, China, IEEE (1999) 362–371
6. Roddick, J.F.: A survey of schema versioning issues for database systems. *Information and Software Technology* **37** (1995) 383–393
7. Shankaranarayanan, G., Ram, S.: Research issues in database schema evolution - the road not taken. Technical Report Technical Report 2003-15, University of Arizona (2003)
8. Fan, H., Poullovassilis, A.: Schema evolution in data warehousing environments a schema transformation-based approach. In Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.W., eds.: *International Conference on Conceptual Modeling*. Volume 3288., Shanghai, Springer (2004) 639–653
9. Noy, N.F.E., Klein, M.E.: Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems* **6** (2004) 428–440
10. Haase, P., Stojanovic, L.: Consistent evolution of OWL ontologies. In Gómez-Pérez, A., Euzenat, J., eds.: *Second European Semantic Web Conference*. Volume 3532 of LNCS., Heraklion, Greece, Springer (2005) 182–197
11. Qian, X.: Correct schema transformations. In Apers, P.M.G., Bouzeghoub, M., Gardarin, G., eds.: *Advances in Database Technology - EDBT'96*, 5th International Conference on Extending Database Technology. Volume 1057 of LNCS., Avignon, France, Springer (1996) 114–128
12. Miller, R., Ioannidis, Y., Ramakrishnan, R.: The use of information capacity in schema integration and translation. In Agrawal, R., Baker, S., Bell, D., eds.: *19th International Conference on Very Large Data Bases, VLDB'93*, Dublin, Ireland, Morgan Kaufmann (1993) 120–133
13. Miller, R.J., Ioannidis, Y.E., Ramakrishnan, R.: Schema equivalence in heterogeneous systems: Bridging theory and practice. *Information Systems* **19** (1994) 3–31
14. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. *VLDB Journal* **10** (2001) 334:350

15. Li, W.S., Clifton, C.: SemInt: a tool for identifying attribute correspondences in heterogeneous databases using neural network. *Data and Knowledge Engineering* **33** (2000) 49–84
16. Doan, A.H., Domingos, P., Halevy, A.Y.: Reconciling schemas of disparate data sources: a machine-learning approach. *ACM SIGMOD International Conference on Management of Data* (2001) 509–520
17. Armstrong, W.W.: Dependency structures of data base relationships. In: 6th International Federation for Information Processing Congress (IFIP). Volume 74., North-Holland, Amsterdam (1974) 580–583
18. Roddick, J.F., Craske, N.G., Richards, T.J.: Handling discovered structure in database systems. *IEEE Transactions on Knowledge and Data Engineering* **8** (1996) 227–240
19. de Vries, D., Roddick, J.F.: Facilitating database attribute domain evolution using mesodata. In Grandi, F., ed.: 3rd International Workshop on Evolution and Change in Data Management (ECDM2004). Volume 3289 of LNCS., Shanghai, China, Springer (2004) 429–440
20. de Vries, D.: Mesodata : Engineering Domains for Attribute Evolution and Data Integration. PhD thesis, Flinders University (2006)
21. Ferrandina, F., Meyer, T., Zicari, R.: Implementing lazy database updates for an object database system. In Bocca, J.B., Jarke, M., Zaniolo, C., eds.: 20th International Conference on Very Large Data Bases, VLDB'94, Santiago, Chile, Morgan Kaufmann (1994) 261–272
22. Tan, L., Katayama, T.: Meta operations for type management in object-oriented databases - a lazy mechanism for schema evolution. In Kim, W., Nicolas, J.M., Nishio, S., eds.: 1st International Conference on Deductive and Object-Oriented Databases, DOOD'89, Kyoto, Japan, North-Holland (1989) 241–258
23. Bergamaschi, S., Castano, S., Vincini, M.: Semantic integration of semistructured and structured data sources. *SIGMOD Record* **28** (1999) 54–59
24. Cavallone, D., De Giacomo, G., Lenzerini, M., Nardi, D., Rosati, R.: Information integration: Conceptual modeling and reasoning support. In: 3rd IFCIS International Conference on Cooperative Information Systems (CoopIS), New York City, NY (1998) 280–291
25. Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J.D., Widom, J.: The TSIMMIS project: Integration of heterogeneous information sources. In: 16th Meeting of the Information Processing Society of Japan, Tokyo, Japan (1994) 7–18
26. Hammer, J., Garcia-Molina, H., Nestorov, S., Yerneni, R., Breunig, M., Vassalos, V.: Template-based wrappers in the TSIMMIS system. *SIGMOD Record* **26** (1997) 532–535
27. Clifford, J., Warren, D.: Formal semantics for time in databases. *ACM Transactions on Database Systems* **8** (1983) 214–254
28. Grandi, F.: A relational multi-schema data model and query language for full support of schema versioning. In: National Conference on Advanced Database Systems, Isola d'Elba, Italy (2002) 323–336
29. Grandi, F.: SVMgr: A tool for the management of schema versioning. In Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.W., eds.: 23rd International Conference on Conceptual Modeling (ER2004). Volume 3288., Shanghai, China, Springer-Verlag (2004) 860–861
30. Rosenthal, A., Seligman, L.J., Renner, S.: From semantic integration to semantics management: case studies and a way forward. *SIGMOD Record* **33** (2004) 44–50

31. Jensen, O.G., Böhlen, M.H.: Evolving relations. In: Database Schema Evolution and Meta-Modeling, Proc. International Workshop on Foundations of Models and Languages for Data and Objects. Volume 2065 of LNCS. Springer (2001) 115
32. Jensen, O.G., Böhlen, M.H.: Multitemporal conditional schema evolution. In Grandi, F., ed.: 3rd International Workshop on Evolution and Change in Data Management (ECDM2004). Volume 3289 of LNCS., Shanghai, China, Springer (2004) 441–456
33. Penney, D., Stein, J.: Class modification in the gemstone object-oriented dbms. OOPSLA '87 (SIGPLAN Notices) **22** (1987) 111–117
34. Edelweiss, N., Moreira, A.: Temporal and versioning model for schema evolution in object-oriented databases. Data and Knowledge Engineering **53** (2005) 99–128
35. Hick, J.M., Hainaut, J.L.: Database application evolution: A transformational approach. Data and Knowledge Engineering **Article in Press** (Preprint)
36. Bernstein, P.A.: Applying model management to classical meta data problems. Conference on Innovative Data Systems Research (CIDR) (2003) 209–220
37. Melnik, S., Rahm, E., Bernstein, P.A.: Rondo: a programming platform for generic model management. In: 2003 ACM SIGMOD International Conference on Management of data, San Diego, California, ACM Press (2003) 193–204
38. Madhavan, J., Halevy, A.Y.: Composing mappings among data sources. In Freytag, J.C., Lockemann, P.C., Abiteboul, S., Carey, M.J., Selinger, P.G., Heuer, A., eds.: 29th International Conference on Very Large Data Bases (VLDB), Berlin, Germany, Morgan Kaufmann (2003) 572–583
39. Franconi, E., Grandi, F., Mandreoli, F.: A semantic approach for schema evolution and versioning in object-oriented databases. In Lloyd, J.W., Dahl, V., Furbach, U., Kerber, M., Lau, K.K., Palamidessi, C., Pereira, L.M., Sagiv, Y., Stuckey, P.J., eds.: 1st International Conference on Computational Logic, CL'00. Volume 1861., London, UK, Springer (2000) 1048–1062
40. Grandi, F., Mandreoli, F.: A formal model for temporal schema versioning in object-oriented databases. Data and Knowledge Engineering **46** (2003) 123–167

A DAG Comparison Algorithm and Its Application to Temporal Data Warehousing

Johann Eder¹ and Karl Wiggisser²

¹ University of Vienna, Dep. of Knowledge and Business Engineering
`johann.eder@univie.ac.at`

² Alps Adria University Klagenfurt, Dep. of Informatics-Systems
`wiggisser@isys.uni-klu.ac.at`

Abstract. We present a new technique for discovering and representing structural changes between two versions of a directed acyclic graph (DAG). Motivated by the necessity of change detection in temporal data warehouses and inspired by a well known tree comparison algorithm, we developed a heuristic method to calculate an edit script transforming an old version of a graph to the new one. This edit script is composed of operations for inserting and deleting nodes and changing labels and values of nodes as well as for inserting and deleting edges to cover rearrangements of nodes (moves). We present the prerequisites of our approach, the different phases of the algorithm and discuss some evaluation results gained from a prototypic implementation. Our approach is applicable to arbitrary labeled DAGs in any context, but optimized for rooted, ordered and labeled acyclic digraphs with a small rate of changes between the DAGs to be compared.

1 Introduction

Graphs are powerful data structures to model complex concepts in various contexts. Such modeling capabilities often come along with the need for efficient comparison of models, to either be able to determine the similarity of two models or to find differences between them. For instance, in many pattern matching approaches, the patterns are represented by graphs and one wants to find the best fitting predefined model out of a model database to identify the pattern. Another use case for such an algorithm is to find differences between two versions of a model. The integral part when finding model differences is to separate elements that have *changed* from elements that have been *deleted* or *inserted*. This is necessary for the traceability of elements through different versions of the model. We present a graph based approach for comparing two models and finding the differences between them in terms of changed, deleted and inserted nodes and edges. Furthermore, we present an application of this approach in the context of temporal data warehousing.

Data warehouses are specialized databases supporting complex analytical queries for decision support. The most popular architecture for a data warehouse are multidimensional data cubes, where transaction data (*facts* or *measures*) is described in terms of master data (*dimension members* or *members*).

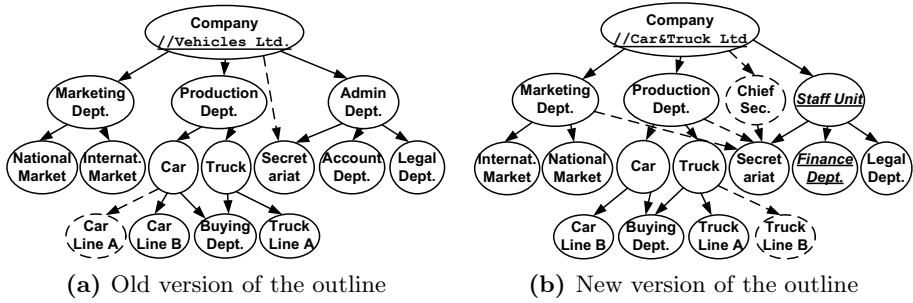


Fig. 1. Two versions of a motor company's data warehouse outline

Usually, members are hierarchically organized in dimensions, e.g., *company* $\leftarrow$ *division* $\leftarrow$ *department*, where $B \leftarrow A$ means that A rolls-up to B . Most data warehouses contain a time dimension, so available data warehouse systems can handle changing measures, e.g., changing turnover. Surprisingly, today's systems are not able to deal with structural modifications in the dimensions, e.g., the creation of a new division, or the join of two departments.

In [1] we presented the COMET temporal data warehouse metamodel, which allows to represent not only changes of transaction data, but also of schema and structure data. The COMET model can then be used as basis of OLAP tools which are aware of structural changes and therefore permit correct query results spanning multiple periods and thus different versions of dimension data. Temporal data warehouses, however, need a representation of the changes that occurred between contiguous structure versions of the dimension data. Frequently, there is no complete change log available, but the data warehouse administrator has to manually find a mapping between snapshots of the dimension structure as they are extracted from the data sources - a cumbersome and tedious job in real world size data warehouses. Our motivation for this work was to support data warehouse administrators during this task. The dimension structure of data warehouses can be represented by directed acyclic graphs. So in this paper, we present a comparison algorithm for multidimensional data by means of graph edit scripts. It is able to detect changes like insertion, deletion, or rearrangement of dimension members. One of the main advantages of our approach is that it also supports detection of key modifications (e.g., renaming of departments). Besides such structural changes, there are also semantical changes, like splitting or merging of members, which cannot be found by analyzing structural data only. For such changes we have developed a data mining based approach which analyzes outliers in cell data [2]. As this method's computational costs are rather high, it is desirable to find structural changes first, by mere structure comparisons with more efficient algorithms. Such a comparison algorithm can then be used in a system for the maintenance of temporal data warehouse models [3]. It is obvious that such an approach can only be *heuristic* and never complete: First, because graph isomorphism is considered to be an $\mathcal{NP}$ -complete problem and second, it touches on the identity problem of entities - i.e., to decide whether something is new or merely (key) attributes of an existing entity changed. Therefore, this

method is intended to support the administrator, but not to fully automate the task. Since dimension members can be numerous (e.g., product catalog), the productivity gain will be significant.

Figure 1 shows examples for an old and a new version of a motor company's data warehouse outline with a number of changes. Nodes and edges which are not present in both graphs are illustrated by dashed lines. Changes in node values or labels are underlined. If no value change happened, the value is omitted for sake of simplicity. We will explain the changes in detail when describing the various parts of our algorithm.

2 Related Work

In [4] we presented an approach for detecting changes in tree structured multi-dimensional outlines. We showed how the hierarchies may be presented as trees and how a particular tree comparison algorithm can be applied to find differences. We furthermore presented a highly customizable method for comparing single dimension members in order to find a good matching between the trees. But as not all outlines, e.g., parallel hierarchies, can be represented as trees we need another representation for them, namely directed acyclic graphs.

Graph matching and graph comparison is a long known problem. Because the graph isomorphism problem is considered to be $\mathcal{NP}$ -complete [5], there are several approaches comparing two graphs and/or determine their edit distance using some heuristics or restricting the data structure. For instance, the approaches of Wang et al. [6], Zhang et al. [7], Bunke et al. [8], Shoubridge et al. [9], Cordella et al. [10], Hlaoui and Wang [11] or Gori et al. [12] are only some of them. We evaluated these algorithms but they all have some shortcomings which make them either completely unusable for our purpose or at least very hard to adapt to our problem. Some of the approaches are defined on cyclic graphs and others are missing operations essential for our purpose. If an adaption was possible, main advantages of the algorithms would have vanished.

So, to the best of our knowledge there is no algorithm which can easily be adapted to calculate the edit operations between two DAGs as we need them for our temporal data warehouse system. So we developed a new algorithm inspired by the tree comparison algorithm of Chawathe et al. [13,14]. It is built upon the same principles, but with major enhancements to support the comparison of directed acyclic graphs. The renaming detection component is adapted from our previous work in this area [4].

3 Comparison of DAGs

3.1 Graph Definition and Transformations

We define our graph to be a *rooted, ordered, directed, acyclic graph*, i.e., a DAG with exactly one node having no parents and the ordering of the children of a node to be important. A graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V} = \{m_1, \dots, m_m\}$ is a set of

nodes and $\mathbb{E} = \{e_1, \dots, e_n\}$ is an *ordered set* of edges. A node m_i represents a dimension member and is defined as $m_i = \langle id, label, value \rangle$. The *id* is a unique identifier within the graph, the *label* is the member's name, and the *value* is an object holding all other member characteristics, like formulae or comments. An edge e_i is representing a hierarchical connection between two members and defined as $e_i = \langle m_i, m_j, inOrder \rangle$, where m_i is the parent of m_j and *inOrder* is a boolean attribute which denotes if m_j is on the correct position among its siblings under m_i . The ordering of $\mathbb{E}$ defines the order of a node's children. We define the following six operations on DAGs:

1. $INS((m_i, l, v), \mathbb{A})$ with $\mathbb{A} \subseteq (\mathbb{V} \times (\mathbb{V} \cup \{\mathbf{null}\}))$ inserts the node m_i with label l and value v into the graph. For each pair $(m_x, pos_x) \in \mathbb{A}$ m_i is added as child to m_x after sibling pos_x . If pos_x is $\mathbf{null}$, m_i gets the first child of m_x .
2. $DEL(m_i)$ deletes the node m_i and all its incoming edges. A node can only be deleted if it does not have any children.
3. $IED(m_i, m_j, pos)$ creates an edge from parent m_i to child m_j , and m_j is inserted to m_i after the sibling pos . If pos is $\mathbf{null}$, m_j becomes the first child of m_i . With the new edge no cycle must be created.
4. $RED(m_i, m_j)$ removes the edge from parent m_i to child m_j .
5. $UPD(m_i, value)$ changes the value of node m_i to *value*.
6. $REN(m_i, label)$, renames the node m_i to *label*.

We use the update and renaming operations to distinguish whether an element was changed or an element residing on the same position is something completely new. This knowledge enables us to define what we call *transformation functions* that transform cell data between different structure versions [1].

To present the algorithm, we now have to introduce a few terms used later on: We call the old and new version of the DAG d_1 and d_2 . The nodes of d_1 and d_2 are elements of the sets $\mathbb{V}_1$ and $\mathbb{V}_2$ respectively. The *partner* of a node x is the node y in the other graph to which x is matched. $x.label$ and $x.value$ denote the label and value of a node, $compare(v_1, v_2)$ compares the two node values v_1 and v_2 and returns a float in $[0; 1]$ as degree of similarity, where 0 means completely different and 1 means equal. $|x|$ gives the number of leaf descendants nodes of x and $common(x, y)$ gives the set of common leaves – i.e., leaf descendants of x whose partners are leaf descendants of y .

3.2 Matching of DAG Nodes

First of all, we have to find a good matching between the graphs, i.e., finding nodes that represent the same member in both graphs. As the member's name often acts as key, we use it as primary matching constraint, i.e., two nodes cannot match if their labels differ. Furthermore, we expect the node's hierarchical position not to change much, i.e., a leaf will rarely become an inner node and vice versa. Thus, leaves and inner nodes cannot match. Finally we assume that the value and the descendants of nodes will not change too much. Thus, we define the method *similar*(x, y) with $x \in d_1$ and $y \in d_2$ to return **true** iff one of

the following conditions is true: (i) x and y are leaves and $x.label = y.label$ and $compare(x.value, y.value) \geq f$, where $f \in]0; 1]$ is the user defined *Minimum Similarity*. (ii) x and y are inner nodes and $x.label = y.label$ and $\frac{|common(x,y)|}{\max(|x|, |y|)} \geq t$ where $t \in]\frac{1}{2}; 1]$ is the user defined *Minimum Common Leaves* value. As the member's name is the main matching criterion, the number of equally named members directly influences the calculation time and the quality of the matching. In the data warehousing domain it is a good practice not to use duplicate member names, so we assume the number of equally labeled nodes to be very small.

Referring to our example in Fig. 1, we see that e.g., the roots of the graphs can be matched because of their identical name and having 7 out of 9 leaves in common. The difference in the description (*Version 2004* vs. *Version 2005*) does not matter here. On the other hand, the nodes *Admin Dept.* $\leftrightarrow$ *Staff Unit* cannot be matched, because of their different name.

We also have to define the order to visit and compare the nodes. We use a bottom up traversal – so we have matched leaves when comparing inner nodes – and traverse children from left to right in each graph version. From these two node sequences we build the Longest Common Subsequence (LCS) [15] using $similar(x, y)$ as equality check. All pairs contained in the LCS get matched. The remaining nodes are checked with each other for similarity and if seen as similar also considered as a matching. This gives the *Matching Set* $\mathbb{M}$, which contains pairs of nodes (x, y) with $x \in d_1$ and $y \in d_2$ being sufficiently similar to be seen as representing the same member. This technique does not guarantee the best matching, but a “good one” [13].

3.3 Renaming of DAG Nodes

Because the node's label is the main matching criterion, all nodes that cannot be matched could have been renamed. So we want to find pairs of nodes with different labels but which are so similar with respect to their value and structure that they may represent the same element nonetheless.

First, we have to find the set of possible renamed nodes $\mathbb{P}$. $\mathbb{P}$ contains all pairs (a, b) of unmatched nodes $a \in d_1$ and $b \in d_2$ which have enough matched common parents or common parents which are possible renamed. For the formal definition of $\mathbb{P}$ we need the set of common parents $cPar(x, y) = \{(a, b) | a \in x.parents \wedge b \in y.parents \wedge (a, b) \in \mathbb{M}\}$, the set of possible renamed parents $prPar(x, y) = \{(a, b) | a \in x.parents \wedge b \in y.parents \wedge (a, b) \in \mathbb{P}\}$, the set $oldNames$ $\mathbb{O} = \{a \in d_1 | \nexists (a, \_) \in \mathbb{M}\}$ and the set $newNames$ $\mathbb{N} = \{b \in d_2 | \nexists (\_, b) \in \mathbb{M}\}$. With these sets we define $\mathbb{P} = \{(x, y) | x \in \mathbb{O} \wedge y \in \mathbb{N} \wedge \frac{|cParents(x,y)| + |prPar(x,y)|}{\max(|x.parents|, |y.parents|)} \geq c_R\}$, where c_R is a customizable factor in the interval $]0; 1]$. We split $\mathbb{P}$ into two disjoint subsets $\mathbb{P}_L$ containing leaf node pairs and $\mathbb{P}_I$ containing inner node pairs. Pairs containing an inner node and a leaf are neglected. Then we calculate similarity for each pair in $\mathbb{P}_L$ and $\mathbb{P}_I$ like in the matching phase, of course without looking at the labels. All pairs $(x, y) \in \mathbb{P}_L$ with $compare(x.value, y.value) \geq f_R$ (f_R is the Minimum Similarity for renamed nodes) are moved to the set *likelyRenamings*, $\mathbb{L}$, all others are moved to the set *unlikelyRenamings*, $\mathbb{U}$. In

case that for one node more than one possible renaming occurs in $\mathbb{P}$ – but of course only one renaming can have occurred between d_1 and d_2 – we only move the pair with the highest similarity to $\mathbb{L}$, all other pairs containing one node of a likely renaming go to $\mathbb{U}$. For inner nodes we again check their common descendants, but take into account also likely renamed leaves, i. e., pairs of leaf nodes from $\mathbb{L}$, and the nodes' values.

As this approach is a heuristic one, in production environments the user has to acknowledge detected renamings. For each pair (a, b) , the user accepts as renaming, an operation $\text{REN}(a, b.\text{label})$ is generated, applied to d_1 and appended to the edit script and (a, b) is added to $\mathbb{M}$. Surely, the user may also accept pairs completely discarded by the algorithm or pairs from $\mathbb{U}$ as renaming.

Again referring to the example in Fig. 1, the renaming of *Account Dept.* to *Finance Dept.* can be detected because of two reasons: *Account Dept.* and *Finance Dept.* have a possible renamed common parent (*Admin Dept.* / *Staff Unit*) and their value (not shown in the example) is the same, so they are seen as similar. No renaming is detected for *Car Line A* and *Truck Line B* because they do not have any parents in common. For *Admin Dept.* and *Staff Unit* a renaming is suggested because they have all leaves in common – one is part of a likely renaming – and their value (not shown) is the same.

3.4 Comparing Two DAGs

After the matching and renaming is done, we have the basis for detecting differences between the two graphs. The change detection is split into four phases: detecting (i) inserted nodes, (ii) updated nodes, (iii) inserted and deleted edges, and (iv) deleted nodes. In each of the phases, whenever an operation is generated it is immediately applied to d_1 and appended to the edit script.

In the *Insert Phase* let x be the current node when traversing d_2 in topological order. If x is not contained in any matching – i. e., x does not have a partner in d_1 – it has been inserted. Let $\mathbb{Y}$ be the set of parents of x and $\mathbb{Z}$ the set of partners of the nodes in $\mathbb{Y}$. As we traverse the graph in topological order, we can be sure that every parent of x has already been visited and thus has a partner in d_1 . We now can use $\text{FindPos}(x, y)$ defined in Fig. 2a, to find the correct position for x within each parent $y \in \mathbb{Y}$. Then we create the an $\text{INS}(w, x.\text{label}, x.\text{value}, \mathbb{A})$, where $\mathbb{A}$ contains the parents and the respective position. Looking at our example graphs, we see that the nodes *Truck Line B* and *Chief Sec.* are inserted.

In the *Update Phase* let x be the current node in a topological traversal of d_2 and $x.\text{partner} = w$. If $x.\text{value} \neq w.\text{value}$ generate an operation $\text{UPD}(w, x.\text{value})$. In the example, only the description of the root has changed, which results in a single update operation.

The *Edge Changing Phase* is split into two parts: First, ensure that every node is located under the correct parents, and second, (re)create the order of the children. Let x again be the current node in a topological traversal of d_2 and $\mathbb{Y}$ its set of parents. Then let w be the partner of x and $\mathbb{U}$ the set of w 's parents. We now have to compare these two sets in the method $\text{CorrectParents}(w, x)$ given

1. If x is leftmost child of $parent$ whose edge is marked *in order* Return **null**
2. Let v be the rightmost sibling to the left of x whose edge is *in order*
3. Return $v.partner$

(a) The Method FindPos($x, parent$)

1. Let $\mathbb{Y} = x.parents$ and $\mathbb{U} = w.parents$
2. $\forall y \in \mathbb{Y} : y.partner \notin \mathbb{U}$ generate the operation **IED**($y.partner, w, pos$)
where $pos = \mathbf{FindPos}(x, y)$
3. $\forall u \in \mathbb{U} : (u.partner = \mathbf{null} \vee u.partner \notin \mathbb{Y})$ generate the operation **RED**(u, w)

(b) The Method CorrectParents(w, x)

1. Mark the edges from w and x to all of their children *out of order*
2. Let $S_1(S_2)$ be the sequence of children of $w(x)$ whose partners are children of $x(w)$
3. Define the function $equal(a, b)$ to be true iff $(a, b) \in \mathbb{M}$.
4. Let $S = LCS(S_1, S_2, equal)$.
5. For each $(a, b) \in S$, mark the edges $\langle w, a, \mathbf{true} \rangle$ and $\langle x, b, \mathbf{true} \rangle$.
6. For each $a \in S_1, b \in S_2$ such that $(a, b) \in M$ but $(a, b) \notin S$
 - (a) Let $pos = \mathbf{FindPos}(b, x)$.
 - (b) Generate the operations **RED**(w, a) and **IED**(w, a, pos)

(c) The Method AlignChildren(w, x)

Fig. 2. PseudoCode for Methods used during DAG Comparison

in Fig. 2b. The method **AlignChildren**(w, x) (re)creates the correct ordering for the children of the matched nodes w and x and is defined as in Fig. 2c.

In our running example, we see some edge changing operations: The *Secretariat* is now attached also to the *Production Dept.*, the *Marketing Dept.* and the *Chief Sec.*, but no longer directly to the *Company*. This results in three edge creates and one edge delete. Furthermore we see that the *National Market* and *Internat. Market* switched their place. This results in one edge delete and one edge create for one of these sub departments.

In the *Delete Phase* let w be the current node in a post-order traversal of d_1 . If w is not matched, generate an operation **DEL**(w). In our example, we see that the node *Car Line A* is still unmatched and thus is deleted in this phase.

3.5 The Complete Algorithm

Now we can present the whole procedure. The first part is finding the matching set $\mathbb{M}$. Then we do the renaming detection and update $\mathbb{M}$ with the detected renamings. The graph comparison itself consists of four phases – insert, update, change edges, and delete. The first three of them can be combined in one single topological graph traversal. The delete phase needs one extra run through the graph. When all phases are finished, d_1 is isomorph to d_2 and the edit script holds all found differences. The pseudocode for the comparison is shown in Fig. 3. Figure 4 shows one possible edit script for our running example.

As said above this is a heuristic approach. The matching of nodes and the renaming detection rely on the assumption that nodes do not change their value and their position too much. Of course such a heuristic may fail in some cases,

1. Traverse d_2 in topological order // *Insert, Update and Edge phase*
 - (a) Let x be the current node in the traversal
 - (b) If x has no partner // *x was inserted*
 - i. $\mathbb{A} = \emptyset$
 - ii. For all nodes $y \in x.parents$
 - A. $pos = \text{FindPos}(x, y)$
 - B. $\mathbb{A} = \mathbb{A} \cup \{(y.partner, pos)\}$
 - iii. Add $\text{INS}(w, x.label, x.value, \mathbb{A})$
 - iv. $\mathbb{M} = \mathbb{M} \cup \{(w, x)\}$
 - (c) else // *x has a partner*
 - i. Let w be $x.partner$
 - ii. If $w.value \neq x.value$ generate $\text{UPD}(w, x.value)$
 - iii. If x is not the root of d_2 $\text{CorrectParents}(w, x)$
 - iv. $\text{AlignChildren}(w, x)$
2. Traverse d_1 in post order // *delete phase*
 - (a) Let w be the current node in the traversal
 - (b) If w has no partner generate $\text{DEL}(w)$

Fig. 3. The complete change detection algorithm

```

REN(Admin Dept, Staff Unit) REN(AccoutDept, Finance Dept)
INS((Chief Sec., Chief Sec, attributes), {(Company, Production Dept)})
INS((Truck Line B, Truck Line B, attributes), {(Truck, Truck Line A)})
IED(Marketing Dept, Secretariat, National Market)
IED(Production Dept, Secretariat, Truck) IED(Chief Sec, Secretariat, null)
REDCompany, Secretariat UPD(Company, newattributes) DEL(Car Line A)

```

Fig. 4. Possible edit script for running example

so the algorithm does not guarantee to provide a perfect graph matching. Thus, in a production environment, the user will have to check the matchings and renamings for correctness to prevent consecutive faults.

4 Evaluation

For the evaluation we need two versions of a graph with a known set of differences. We randomly generate a rooted DAG (old graph version) and a number of edit operations, representing the differences, which we apply to the old version. This gives us the new graph version. Within the generated edit script the operations are distributed as follows: 55% are updates, 11%-12% for each of inserts, deletes and renames and 5%-6% for insert and remove edges.

As our graphs represent dimension structures, and data warehouses do not tend to change very fast, we assume the difference rate not to be greater than 10%. The difference rate is given as percentage of the number of nodes. If the difference rate is greater than that, this can be seen as major reorganization. Such situations should be accomplished by methods, providing a complete documentation of the process and therefore substituting the graph comparison.

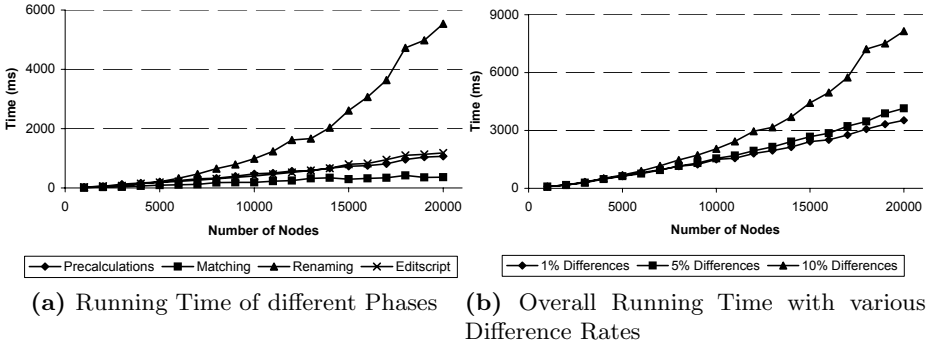


Fig. 5. Evaluation Charts

We evaluated our approach on graph sizes from 1000 to 20000 nodes and difference rates of 1%, 5% and 10%. For each graph size we did 10 runs with new graphs and differences.

4.1 Evaluation Results

Figure 5a shows the average calculation time of the algorithm’s phases at a difference rate of 10%. The *Precalculations* are a set of calculations over the graph, done before the matching starts, e. g., calculating all labels contained in the graph. The *Matching* is the matching set calculation, *Renaming* is detection of renamings and *Edit script* is the calculation of the remaining edit script. Figure 5b shows the average overall running time for the algorithm for each tested difference rate. It can be seen that the running time depends on the number of differences between the two graphs as well as on the number of nodes.

We also calculated the error rate of our algorithm which is represented as the percentage of errors in the detected edit scripts with respect to the generated differences. The error rate does not only cover the absolute number of found edit operations but each operation is checked for its correctness. The error rate grows with the percentage of difference but does not strongly depend on the number of nodes in the graph. The average error rates are about 0.11% for 1% differences, 0.20% for 5% differences and 0.55% for 10% differences.

All of the wrong inserts, deletes and edge operations are a direct result of errors in the matching or renaming detection. If the algorithm calculates wrong matchings, additional operations for reestablishing the “correct” structure will be generated. Thus, we have to ask the user to acknowledge all detected matchings. This will foreclose many of the resulting errors.

5 Conclusion

Graph change detection and graph isomorphism are two of the “known-to-be-hard problems” in computer science, so no efficient solution is possible unless

$\mathcal{P} = \mathcal{NP}$. Motivated by the necessity of having a way to find and describe differences between two versions of a data warehouse structure, we presented an heuristic approach for comparing directed acyclic graphs and determining the differences between them. Our algorithm is optimized to work on labeled and attributed DAGs with change rates below 10%. We described the various phases of the algorithm, i. e., node matching, renaming detection, detection of inserts and deletes of nodes and edges and change detection within nodes. An extensive evaluation of the algorithm in our prototypical implementation showed very positive results. Both accuracy of the computed edit script with the factual changes and the performance and scalability of the algorithm were measured surprisingly well. We are confident that variations of this algorithm are also suitable for other application areas with similar requirements and characteristics, in particular for describing evolution in systems represented by directed acyclic graphs, e. g., changing ontologies.

References

1. Eder, J., Koncilia, C.: Changes of Dimension Data in Temporal Data Warehouses. In: Proc. of the 3rd DaWaK. (2001) 284–293
2. Eder, J., Koncilia, C., Mitsche, D.: Automatic Detection of Structural Changes in Data Warehouses. In: Proc. of the 5th DaWaK. (2003)
3. Eder, J., Koncilia, C., Wiggisser, K.: Maintaining temporal warehouse models. In: Proc. of the IFIP Int'l Conf. on Research and Practical Issues of Enterprise Information Systems. (2006)
4. Eder, J., Koncilia, C., Wiggisser, K.: A Tree Comparison Approach to Detect Changes in Data Warehouse Structures. In: Proc. of the 7th DaWaK. (2005) 1–10
5. Garey, M., Johnson, D.: Computers and Intractability – A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York (1979)
6. Wang, J.T.L., Zhang, K., Chirn, G.W.: Algorithms for approximate graph matching. *Information Sciences* **82**(1-2) (1995) 45–74
7. Zhang, K., Wang, J., Sasha, D.: On the editing distance between undirected acyclic graphs. *Int'l Journal of Foundations of Computer Science* **7**(13) (1996)
8. Messmer, B., Bunke, H.: A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Trans. on PAMI* **20**(1998) 493–505
9. Shoubbridge, P., Kraetzl, M., Ray, D.: Detection of abnormal change in dynamic networks. In: Proc. of Information Decision and Control, IEEE Inc. (1999) 557–562
10. Cordella, L., et al.: Performance evaluation of the vf graph matching algorithm. In: Proc. of the 10th Int'l Conf. on Image Analysis and Processing. (1999) 1172–1177
11. Hlaoui, A., Wang, S.: A new algorithm for inexact graph matching. In: Proc. of the 16th Int'l Conf. on Pattern Recognition (ICPR'02) - Volume 4. (2002)
12. Gori, M., Maggini, M., Sarti, L.: Exact and approximate graph matching using random walks. *IEEE Trans. on PAMI* **27**(7) (2005) 1100–1111
13. Chawathe, S., Rajaraman, A., Garcia-Molina, H., Widom, J.: Change detection in Hierarchically Structured Information. In: Proc. of the ACM SIGMOD. (1996)
14. Chawathe, S., Garcia-Molina, H.: Meaningful change detection in structured data. In: Proceedings of the 1997 ACM SIGMOD. (1997) 26–37
15. Myers, E.: An $O(N D)$ Difference Algorithm and Its Variations. *Algorithmica* **1**(2) (1986) 251–266

Handling Changes of Database Schemas and Corresponding Ontologies

Andreas Kupfer*, Silke Eckstein, Karl Neumann, and Brigitte Mathiak

Technical University of Braunschweig, Germany
Institute of Information Systems

{a.kupfer, s.eckstein, k.neumann, b.mathiak}@tu-bs.de

Abstract. Currently, knowledge from biological research is stored in hundreds of databases, counting only public accessible ones. Finding specific data in these is a challenging task which can be supported by ontologies describing them. The maintenance of a corresponding ontology is time consuming manual work, because research database schemas change rapidly. Our project will reduce the work by automating tasks, like a generation process and applying schema changes to the corresponding ontology. We call the proposed method coevolution, because database schema and ontology are allowed to evolve independently without ever losing their connection to each other. Our method consists of initial ontology generation, manual annotation and change propagation steps.

1 Introduction

Among other things, the database schema of a relational database contains the structure of the stored data. This allows to select specific information by using descriptions like table or column names. However, those names are just characters and do not have a defined semantical meaning, nor does the database schema contain additional annotations. There are no common naming conventions across multiple databases. So when we work with several databases e.g. for a data warehouse or when we simply want to know which one could contain information about a topic that we search for, semantic annotation would help [1]. Ontologies describe the concepts and relations of a specific domain. They are used for structured data like XML documents [2] and can be used to describe the structure and the conceptual model of a database as well. So, our aim is to annotate biological research databases with ontologies, in order to add semantical information to them.

Even though ontologies offer great benefits, they are rarely used for database schemas for two reasons. First, most database management systems do not support ontologies as a semantic annotation for the database. Second, if the ontology is stored in an external file, the problem must be solved, how it can be updated

* The first author was funded by the German Federal Ministry for Education and Research, BMBF, Grant No. 031U210C as a member of the Interonomics Center (<http://www.intergenomics.de>).

on schema changes. Occurring database schema evolution and desirable ontology evolution do not fit together well. To avoid manual work we present a synchronisation mechanism between both. It is impossible to generate a new ontology on each change, because it would neglect the annotations already made.

To handle changes of database schemas and corresponding ontologies, we start by mapping each database schema to an ontology, to provide a link between the database and annotated semantic concepts. This mapping is done in an automatic process and allows a quick start of the annotation process. We use the current relational schema of a database as input, allowing us to use our method on already running databases without the dependence on a conceptual model. Those models are most likely not available or outdated. For our ontologies we use the Web Ontology Language (OWL) [3], so the stored annotations are not application specific, but can be used in several OWL applications. We do not create a global schema, but each of our database ontologies contains instances describing the corresponding database schema and they can be combined simply. Various methods and methodologies for creating ontologies exist, a good overview can be found in [4]. Although it is desirable for an ontology to grow by manual maintenance, ontologies are often created for already existing data. Our approach is similar to the relational ontologies discussed in [5], except the language we use is OWL Lite, instead of OWL Full. This allows us easier subsequent processing of the generated ontologies.

There are more than 850 public biological research databases [6]. All contain specific research data, which can be combined usefully [7]. Describing the topics and their location in the database is more precise than just using database catalogue information. Since the context can be expressed in a machine-readable form, a program integrating such databases could decide if the database is about molecules in a specific organism, e.g. human, mouse or yeast, independent from table names. Currently, ontologies are mainly used to build a common vocabulary for describing, sharing and linking biological annotations [8].

Biological research databases are not static, but change constantly. The main focus of our work are strategies to cope with upcoming changes. For data integration purposes there already exist solutions. But they use specialized data structures, like the hypergraph data model in [9], instead of ontologies. We assume that relational database systems are capable of applying any schema changing SQL statement. These reflect all database evolution constructs regarding relational database systems [10], which have to be considered for an attached ontology. A structural change in a database, a schema evolution, causes a problem for database ontologies, because they are only useful if they match to the database schema.

For our task, the ontology and the schema have to remain synchronised. The ontology describing a database must be updated after each database schema change. It is worthwhile to reduce the required work by automating tasks. This is not trivial, because the ontology cannot be regenerated after each change in the schema, as we would lose the annotations. Instead, we have to analyze the changes in the schema and apply them to the ontology as well.

The enriched ontology may serve as groundwork for other applications. Connecting diverse data sources is a challenging task which can be supported by ontologies describing them [11]. By describing the domain semantically and additionally the structure of those databases syntactically in an ontology, applications like data integration or semantic query processing across heterogeneous databases can be made possible [12].

In the following section we explain how we produce the initial ontology from a database schema. Section 3 goes into detail on how annotations can be made on the ontology in order to store semantics. In Section 4 we explore, how the schema changes can be reproduced in an annotated ontology. We conclude with some remarks about our future work.

2 Mapping a Database Schema to an Ontology Instance

This section describes an abstract database ontology and the automatic ontology generation which represents a database schema. The generated ontologies are a prerequisite to our coevolution framework. These created ontologies serve as a basis for the annotation and allow added semantic concepts to point to specific parts in the database schema for information retrieval.

An ontology provides an abstract view to the information space [13]. So we begin to design one ontology to describe database schemas in general. It is implemented with the smallest subset of OWL, named OWL Lite. This allows it to be used in every dialect of OWL. In [5] a similar ontology for relational databases is proposed which uses specialised relations between all concepts. In our approach we use a generic relation between our concepts allowing us to restrict our language requirement to OWL Lite.

Each database schema with all relations and data type information is mapped to instances in an individual ontology document in OWL Lite format, using concepts of the abstract database ontology. The common ontology defines concepts for the terms **Database**, **Relation** and **Attribute** and an object property **consistsOf** to create a hierarchy between them. The language OWL lite does not provide sufficient restrictions, which enforce the structure of our hierarchy in the instances, which will be created next. But since these instances are generated automatically, there is no need to check for consistency.

In our abstract database ontology a database consists of several relations (tables) which consist of several attributes (columns). Each table is identified by its unique name and each column additionally by an XML schema data type. Special data types, views and security settings are not needed for search operations and are not transformed at the moment. We will extend the database ontology later on. Table and column names are stored as an ID of the respective relation and attribute instance. Identifiers of instances in OWL should be document wide unique. This is not true in databases where columns in different tables can have the same name. To cope with this, we add the relation name to each attribute as its suffix. The original name is stored as a label of the instance, so editors can show the usual attribute name.

The automatic generation process extracts the database schema from a given database and stores the extracted schema information as instance data in an ontology using the concepts described above. We use the SQL statements from a database dump as input for our ontology generation program. The schema dump can be obtained from various database management systems, e.g. in PostgreSQL with `pg_dump`. To outline some examples in this paper, we use an example project from a database textbook [14], because a biological database is too large. It depicts a database for an art gallery named View Ridge (as shown in Fig. 1).

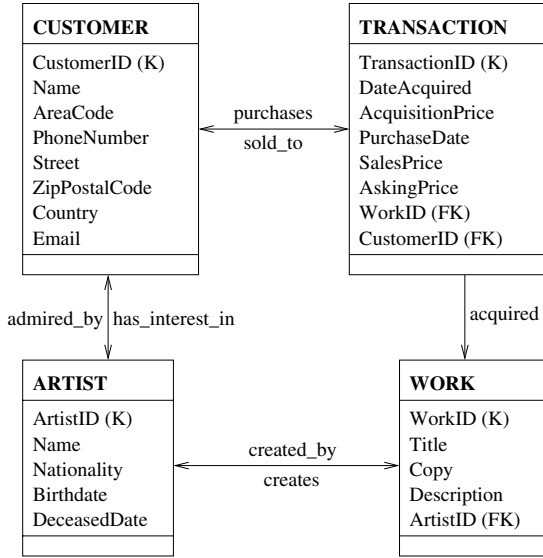


Fig. 1. Simplified class diagram of the View Ridge example database [14]

The conversion process creates an instance of the concept relation for each occurring table and an instance of the concept attribute for each column. It is implemented as a Java application using the XOM library to generate the XML document of the ontology. First the program parses each SQL statement of the given database schema dump and creates an instance of **Database**. The conversion process creates an instance of the concept **Relation** for each occurring table and an instance of the concept **Attribute** for each column. Each attribute references an XML Schema data type. The data types from the schema are not mapped exactly as predefined data types in XML schema are more generic, e.g. any char, varchar or text data type is mapped to an XML Schema string. The hierarchy structure between the instances is created by connecting them with the object property `consistsOf` according to the abstract database model.

Furthermore, the conversion process handles foreign keys. They are very helpful for the annotation process, because they depict connections of each table to other tables. The generation process can not decide what kind of semantical connection there is, but we store a placeholder which can be annotated later on. The example below shows an excerpt of a schema dump which is used as input

and the the corresponding lines from the output. For each foreign key constraint (input line 3) in the schema dump input the program generates an instance of the relation named `isAssociatedWith` (output line 5) in the generated output. The relation is added to the `Attribute` instance and refers to the instance corresponding to the stated column. We do not use the constraint identifier as relation identifier, because it does not contain semantics. So it is discarded and the same placeholder relation is used for all foreign keys, allowing us to find them easily when the ontology is edited later on.

Input:

```
1 CREATE TABLE WORK (  
2 ArtistID int NOT NULL,  
3 CONSTRAINT ArtistFK FOREIGN KEY(ArtistID) REFERENCES ARTIST (ArtistID),  
...  
4 );
```

Output:

```
...  
1 <Relation rdf:ID="WORK"> <consistsOf>  
2 <Attribute rdf:ID="ArtistID__WORK">  
3 <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">  
ArtistID</rdfs:label>  
4 <consistsOf rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>  
5 <isAssociatedWith rdf:resource="#ArtistID__ARTIST"/>  
6 </Attribute>  
...
```

3 Annotation Process to Add the Conceptual Model

Once the database ontology for a specific database is created, semantic information about the context and concepts can be annotated. For this process we have to create the conceptual model by reverse engineering and connect it to the relational model in the database ontology. These annotations are introducing new concepts and are establishing connections between concepts. Even connections between our concepts and the concepts of other ontologies can be expressed. With these annotations the ontology grows and describes the given domain a bit more specific. The semantic concepts can be connected via relations directly to the instances of our database ontology and thus creating links from a concept to a part of the database schema or vice versa.

To improve the ontology, we may add semantic information to our generated instances. For this purpose it is a good idea to use a conceptual model. Usually, this requires some reverse engineering, because UML models about the concepts of the research database are not available or are outdated. In this paper we skip this step, because we have a conceptual model available, which can be used as a starting point for the manual annotation process.

In this example we begin with the relations that were lost during the transformation from a conceptual model to a relational model. The UML class diagram of the View Ridge Gallery (see Fig. 1) also depicts relations between

the tables which could not be extracted from the database schema directly, but only as generic foreign key associations. A **Customer** purchases with a specific **Transaction**. In a **Transaction** a piece of **Work** is acquired which was **created_by** an **Artist** and so on. These relations contain semantic information about the database which are now added as an example to the according database ontology.

First, we add new definitions for each new relation. We add some lines to the ontology (e.g. by using an ontology editor) defining a relation for the association **created_by** between **Artist** and **Work** and we do the same for the association **creates** in the inverse direction. Then an instance of the new defined relations is created for each pair of classes connected. A relation r connects a concept c_1 with a concept c_2 . In OWL c_1 is the instance in which the relation is instantiated and c_2 is referenced. So in this example `<creates rdf:resource="#Work"/>` is added into the relation **Artist** and in the instance for **Work** we add `<created_by rdf:resource="#Artist"/>`. Relations can target at any instance in our database ontology and are not limited to instances of our concept relation for tables. They can also describe a specific column by targeting at the corresponding attribute instance.

So far, the ontology only consists of concepts describing a relational database. It can be further enriched by adding new concepts describing the domain and relating them to the generated instances from the database. For example, a customer as well as an artist is a person. In order to represent this information in the ontology, we create a new concept **Person** and a new relation **is_about**. The instance of the relation can be added to the instance **Artist** and the same line can be added to **Customer**.

Other than creating our own concept, things get more interesting when we link to external concepts in other ontologies. For data integration it is the key to relate to concepts which are widely adopted. In OWL we can easily use Uniform Resource Identifiers (URI) to refer to remote resources like other ontologies. Domains like biology or knowledge management already have ontologies to relate to, like GeneOntology, Protein Ontology, BioPAX etc., so we may simply link to them using URIs. With the statement `owl:equivalentClass` we can define that classes from different ontologies describe the same concept. To link our class of a person (the example in section 3) to an external one we can easily add it in the declaration:

```
1 <owl:Class rdf:ID="Person">
2   <rdfs:subClassOf rdf:resource="#TopClass"/>
3   <owl:equivalentClass rdf:resource=
      "http://www.mindswap.org/2003/owl/swint/person#Person"/>
4 </owl:Class>
```

By adding line 3 in the example above we have linked our concept to a similar concept in an external ontology. A reasoning tool may now use additional semantic information which is defined in the other ontology. The use of common concepts in different ontologies is a key feature in data integration.

4 Synchronisation on Database Schema Changes

Now that the ontology is automatically produced and manually enriched, we have to consider the database evolution. A research database schema is regularly prone to changes and the ontology is a stand-alone file which would become outdated, if the database schema changes. In this section we present our approach to update the already changed ontology without losing information.

So instead of generating a new ontology without semantic annotations, we apply the database schema changes to the annotated ontology. This is done by classifying all possible schema changes which affect the database ontology into change primitives and describing how each primitive will be applied to the ontology. A similar approach has been used for hypergraph data models instead of ontologies [9].

Any operation m which changes a database from a schema s to a schema s' can be expressed by a statement of the data definition language of the database system. Instead of using a formal logic to express the changes like it is done in [15], we have to use SQL. The SQL statement for a specific operation m can be deducted from s and s' , but we assume for simplicity that the evolution of the database schema is stored in a log file that contains all schema changing SQL commands given to the database.

There is a mapping p which maps all change operations m to one or more change primitives m' which we use for classification. Each change primitive m' can be implemented to apply the modification to the corresponding database ontology. The mapping p can be completely defined on m , because m is always finite on any database language.

In the following we discuss a subset of the most common change primitives to handle all creations, deletions and alterations of tables or columns. Also the implementation of each change primitive to an ontology change is discussed. In some cases an algorithm can not decide, if a change operation m of the user should change the associated semantical concept or not as noted below. In general, maintenance of ontologies can not be done completely without manual expert support [16].

The most obvious change operation is to create a new table by a **CREATE TABLE** command. The mapping creates a change primitive for the table and one for each attribute. The implementation of the change primitives for the creation can simply use the already implemented methods to create new instances of the concept **Relation** for each new table as discussed in section 2. **ALTER TABLE ADD COLUMN** works similarly with respect to attributes. A primitive for each new attribute is created. The implementation creates a new instance for the new attribute and inserts it in the corresponding relation.

SQL allows the deletion of columns and tables. When an element of the database schema is removed, the corresponding instance in our database ontology has to be removed. In order to delete something, we first have to find it. The identifier of the original instance is computed and then it is searched for in the ontology. In case of a table, its identifier is equal to its name. If it is an attribute, the identifier is equal to its name and the name of the table. Also,

we have to find all references (like `rdf:resource`, cf. section 3) which link to the deleted one, as not to leave broken links. We delete all relations which refer to the deleted instance. After that, the instance will be removed as well as all annotations it contains.

With the SQL statement `ALTER TABLE` various changes are possible on the database schema. They can be classified in three categories: First, we filter out some changes which do not affect the current database ontology, e.g. changes of constraints, default values or storage related changes. These are discarded at the moment, we will add change primitives for them later on. The three categories which are transferred to the ontology are: renaming, foreign key structures and data type changes. The problem, whether the alteration changes only syntax or semantics as well, will be discussed in these cases.

Tables and columns can be renamed. The content in the database will not be changed by this operation. All rename operations are mapped to a change primitive for renaming. Since attributes are concepts by their own right, all alterations are treated as implicit deletions and creations. The decision that remains, is whether the relation is containing the same concept as before. Although it may seem tempting to assume that a name change just means that the name changed and the actual concept stayed the same, this may not necessarily be true. An example is shown below:

```
ALTER TABLE ARTIST RENAME COLUMN Nationality TO CountryOfBirth;
```

We rename in the table **Artist** the column **Nationality** to **CountryOfBirth**. With this information alone we can not decide if it is the same for all related concepts. If this attribute only contains a relation to a spacial concept like political area it would not matter. But there is a semantic difference between nationality and the country of birth, which might have been annotated.

The implementation offers the option of either deleting all involved annotations or not to the user in any case. In the case of changing an existing instance, we first look for the instance based on its identifier and then make the needed changes. If a column is renamed we generate a new identifier based on its new name and replace it in the instance of attribute. If we change the name of a table we also have to change all identifier of its attributes, because the table name is used in the identifiers of its attributes as well. When we change the identifier of an instance, it would retain all relations which relate from the instance but lose all relation which relate to this instance. Therefore if we want to delete or rename the annotations we delete or rename all additional elements from the instance and delete or rename all relations in the ontology which target the old identifier.

So as an example we would rename the table **Work** to **Artwork** and use the relations from section 3. Then the relation **created_by** is still valid and will remain or will be deleted by the users decision. Also our program must search for links to the old identifier and will find `<creates rdf:resource="#Work"/>` in the instance of the relation **Artist**. According to the decision this relation will either be updated to the new identifier or deleted. We can only check references in the current ontology. To cope with references from external ontologies we

create an empty dummy instance and an equality link with `owl:sameAs` to point external references to the new identifier.

On a change of the foreign key structure, there is most likely a change of the conceptual model. Our `isAssociatedWith` properties can be updated easily, but the user is warned if there are any annotated relations found for this table.

The data type of a column can be changed. Transferring this kind of change to the database ontology is quite easy. For example, we want to allow more than 3 digits in the area code for international customers like this:

```
ALTER TABLE CUSTOMER ALTER COLUMN AreaCode TYPE char(8);
```

In fact, we just compute the new corresponding XML schema data type, which would be a string, and replace the old type declaration in the specified instance of attribute. For a user it is a quite obvious conclusion that this change would most probably not change any added semantic context. But we can not be sure about it so our program lets the user decide whether the semantic annotation can remain.

5 Conclusions

In this paper we showed how ontologies and relational database schemas can be evolved parallel in a coevolution. We motivated our approach, presented an automatic mapping from database schemas to database ontologies. Then we discussed the evolution of these database ontologies by adding semantic context and finally illustrated the coevolution process in which changes of the database schema can be applied to the related database ontology.

Currently, we are working on several active biological research databases, like PRODORIC, where we have access to the database schema and its evolution. On this data we are testing our implementation of the ontology generation process. For the evolution process we use 242 logged schema changes over the past 16 months. Yet we can not evaluate our approach, because the database ontology is currently hardly annotated and thus all schema changes can be applied successfully. Other tasks like logging of schema changes or support in schema annotation, will be implemented in near future.

Now, the next step will be to annotate a database ontology with expert support. Editing tools are available, like Protégé [17], to work with the generated ontologies, but ontology evolution is more complex than schema evolution [18]. An annotated ontology will help us to find out, which kind of change operations to the ontology we can allow, while maintaining the automatic update on database schema changes. Currently the versioning of ontologies is not needed, but this may be subject to change.

Later on we plan to extend our approach towards processing of OWL-DL ontologies allowing us to map further integrity constraints from the database schema to restrictions in the ontology. There is also more information in the database schema available like views, which could be added to the abstract database ontology and exported to the database ontologies as well. So far, we

concentrated on relational databases, because they are most common, but we are confident that similar approaches can be made to other database models.

Thinking beyond the scope of this project we would like to add semantic query processing for the ontologies [19]. This would allow users to easily find data across all databases which use these ontologies.

References

1. Spyns, P., Meersman, R., Jarrar, M.: Data modelling versus ontology engineering. *SIGMOD Rec.* **31** (2002) 12–17
2. Erdmann, M., Studer, R.: How to structure and access XML documents with ontologies. *Data Knowl. Eng.* **36** (2001) 317–335
3. Bechhofer, S., v. Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L.: *OWL web ontology language reference* (2004) W3C.
4. Gómez-Pérez, A., Fernández-López, M., Corcho, O.: *Ontological Engineering*. Springer (2004)
5. de Laborda, C., Conrad, S.: Relational.OWL - A Data and Schema Representation Format Based on OWL. In: *Conceptual Modelling 2005 (APCCM05)*, Australian Computer Society (2005) 89–96
6. Galperin, M.Y.: The Molecular Biology Database Collection: 2006 update. *Nucl. Acids Res.* **34** (2006) D3–5
7. Lacroix, Z., Critchlow, T.: *Bioinformatics – Managing Scientific Data*. Morgan Kaufmann (2003)
8. Stevens, R., Wroe, C., Lord, P., Goble, C.: *Ontologies in Bioinformatics*. Int. Handbooks on Information Systems. Springer Verlag, Heidelberg (2004) 635–657
9. Fan, H., Poulouvasilis, A.: Schema evolution in data warehousing environments - a schema transformation-based approach. In: *ER 2004, 23rd Int. Conf. on Conceptual Modeling, Proc. Volume 3288 of LNCS.*, Springer (2004) 639–653
10. Tresch, M.: *Evolution in Objekt-Datenbanken*. Teubner, Stuttgart (1995)
11. Lu, J.J., Hsu, C.N.: Query answering using ontologies in agent-based resource sharing environment for biological web information integrating. In: *Proc. of IJCAI-03 Workshop on Information Integration on the Web.* (2003) 197–202
12. Stuckenschmidt, H., v. Harmelen, F.: *Information Sharing on the Semantic Web. Advanced Information and Knowledge Processing*. Springer Verlag (2005)
13. Castano, S., Antonellis, V.D.: A Discovery-Based Approach to Database Ontology Design. *Distrib. Parallel Databases* **7** (1999) 67–98
14. Kroenke, D.: *Database Processing. Fundamentals, Design, and Implementation*. Pearson Prentice Hall, New Jersey, 10th edition (2005)
15. Franconi, E., Grandi, F., Mandreoli, F.: Schema evolution and versioning: A logical and computational characterisation. In: *Database Schema Evolution and Meta-Modeling, 9th Int. Workshop. Volume 2065 of LNCS.*, Springer (2001) 85–99
16. Ding, Y., Foo, S.: Ontology research and development, Part 2 - A review of ontology mapping and evolving. *Journal of Information Science* **28** (2002) 375–388
17. Knublauch, H., Fergerson, R., Noy, N., Musen, M.: The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In: *Int. Semantic Web Conf. Volume 3298 of LNCS.*, Springer (2004) 229–243
18. Noy, N.F., Klein, M.: Ontology evolution: Not the same as schema evolution. *Knowl. Inf. Syst.* **6** (2004) 428–440
19. Necib, C.B., Freytag, J.: Query Processing Using Ontologies. In: *17th Int. Conf. CAiSE 2005. Volume 3520 of LNCS.*, Springer (2005) 167–186

Evolving the Implementation of ISA Relationships in EER Schemas^{*}

Eladio Domínguez¹, Jorge Lloret¹, Ángel L. Rubio², and María A. Zapata¹

¹ Dpto. de Informática e Ingeniería de Sistemas.
Facultad de Ciencias. Edificio de Matemáticas.
Universidad de Zaragoza. 50009 Zaragoza. Spain
{noesis, jlloret, mazapata}@unizar.es

² Dpto. de Matemáticas y Computación. Edificio Vives.
Universidad de La Rioja. 26004 Logroño. Spain
arubio@dmc.unirioja.es

Abstract. One of the most severe problems related to database evolution is how to reflect in the data level the changes that have occurred in the conceptual schema of a database. This is specially relevant when evolution operations affect ISA relationships. In this paper we present our view of the evolution of ISA relationships, focusing on the artifacts that generate the sentences for changing the data in a consistent way.

1 Introduction

Database evolution activities are still a source of challenges for both industry and academy [5]. One of the most severe problems related to database evolution is how to reflect in the extensional level (i.e. data level) the changes that have occurred in the conceptual schema of a database, in such a way that consistency and validity of data are guaranteed. As is recognized in Hainaut et al. [7], data conversion is usually left to the programmers.

A special case of this situation is that of ISA relationships. These are a general construct proposed in many conceptual models (either object-oriented or not). However, the more usual logical models (like the relational model) and database languages (like SQL) do not provide an explicit representation of ISA relationships. Because of this, translation strategies must be established in order to transform these kinds of structures into suitable ones in the target context. Whenever evolution operations have to be introduced at the conceptual level, it is essential to be provided with mechanisms that, as far as possible, keep automatically the consistency of data regarding the corresponding ISA relationships.

In the present paper we describe our approach to the evolution of ISA relationships, focusing on the artifacts that are necessary for changing the data in a

^{*} This work has been partially supported by DGI, project TIN2005-05534, by the Government of La Rioja, project ANGI2005/19, by the Government of Aragon and by the European Social Fund.

consistent way. To our knowledge, this is the first work that addresses this problem specifically. Our solution is based on an architecture presented partially in Domínguez et al. [3], which is based on a metamodeling approach. In this paper we detail how the different components of the architecture, and in particular how the extensional component, deal with the manipulation of data and structures that have to be carried out to evolve ISA relationships. It is worth noting that although in this paper we use an Extended Entity-Relationship -EER- model as conceptual model, our architecture would allow the use of other models -OO, semantic models- as well.

The structure of the paper is as follows. Section 2 first reviews the basic ideas of our architecture for database evolution, and secondly presents the initial running example. Section 3 deals with the particular metamodel we use in the example and with the evolution transformations, specially with regard to ISA relationships, that can be applied to models of that metamodel. We devote section 4 to detailing the technical aspects of the propagation subalgorithm of the extensional component. Finally, related work, some conclusions and future work are outlined in Sections 5 and 6.

2 Context

As has been suggested in the introduction, the goal of this paper is to describe how a physical database schema changes after having modified the EER schema, focusing on modifications that affect ISA relationships.

In order to reach this goal, we have used as scaffolding an architecture described in Domínguez et al. [3]. The architecture uses a metamodeling approach for its four components: *conceptual component*, *translation component*, *logical component* and *extensional component*. We will describe briefly the meaning and purpose of each component (see [3] for details).

The *conceptual component* captures machine-independent knowledge. In this work, it deals with EER schemas. The *logical component* captures tool-independent knowledge describing the data structures in an abstract way. Here, it deals with schemas from the relational model by means of standard SQL. The *extensional component* captures tool dependent knowledge using the implementation language. Here, this component deals with the specific database in question, populated with data, and expressed in the SQL of Oracle. One of the main contributions of our approach is the *translation component*, that not only captures the existence of a transformation from elements of the conceptual component to other elements of the logical one, but also stores explicit information about the way in which concrete conceptual elements are translated into logical ones.

The way of working is as follows: given an EER schema in the conceptual component, the translation algorithm is applied to it and creates 1) a set of elementary translations in the translation component, 2) the logical database schema and 3) the physical database schema. During the execution of the translation algorithm, the user may choose the translation rules which are applied to each building block of the EER schema. For example, for ISA relationships

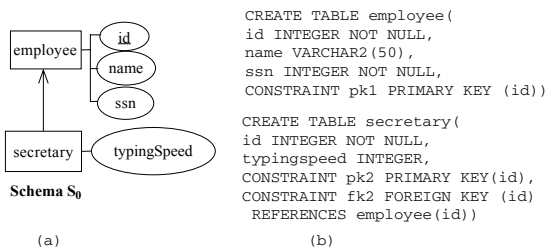


Fig. 1. Example of EER schema and tables of the physical db schema

the user can choose, among others, between the **tablePerEntity** translation rule (‘one table for each entity type’) or the **tableForRoot** translation rule (‘only one table for the root of the ISA hierarchy’). Additional details about the translation algorithm can be found on Domínguez et al. [3].

Then, the physical database is populated with data. For various reasons, the data structure may need to be changed. In this case, the data designer must issue the appropriate evolution transformations to the EER schema (see Section 3 for a catalog of transformations). The propagation algorithm (see Section 4) propagates automatically the changes to the other components. In this way the physical database schema is changed as well as the extension to conform to the new EER schema.

Running example. We will consider an EER schema S_0 with employees and secretaries (taken from Elmasri et al. [6]). Each employee has an **id**, **name** and **ssn**. The secretary has a **typingSpeed** attribute. There is an ISA relationship between the entity type **secretary** and the entity type **employee** (see Figure 1(a)).

When the translation algorithm is applied to the schema S_0 of our running example, the tables shown in Figure 1(b) are generated (we do not show the generated elementary translations nor the logical database). We suppose that, for the translation of the ISA hierarchy, the translation rule **tablePerEntity** (‘one table for each entity type’) has been applied.

3 EER Metamodel and Catalog of Transformations

We have mentioned in the previous section that the application of the evolution architecture requires the use of metamodels for each of the conceptual, translation and logical components. So we need a metamodel of the EER model.

This metamodel has been created following some of the guidelines of the Noesis Metamodeling Technique [4]. The metamodel includes a concept called *IsaSet*, which is used to refer to a set of ISA relationships that follows the same criterion of specialization and that shares the same ‘parent’. A similar notion to *IsaSet* has been included in the recent UML 2.0 specification [11]. Furthermore, the metamodel includes the concept of *IsaHierarchy*, which refers to a set of ISA relationships, all of them descending directly or indirectly from the same ‘parent’ (root). You can ask the authors for the complete version of the paper with additional details about the EER metamodel.

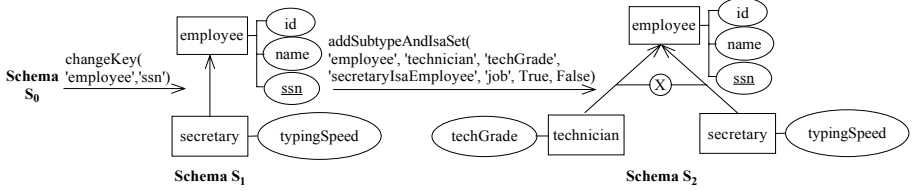


Fig. 2. Evolved schemas

In order to carry out EER schema evolution we have enhanced this metamodel by endowing it with a family of transformations (based on Al-Jadir et al. [1]). For example, the primitive *changeKey*(E : entityType, LA : listOfAttributes) changes the primary key attributes of an entity type, the primitive *addSubtypeAndIsaSet*(P : entityType, C : entityType, LA : listOfAttributes, LI : listOffIsas, [Dis : discriminant], [$Disj?$: boolean], [$Cov?$: boolean]) adds a new subtype to an entity type. Additionally, we have defined the transformation *changeTranslationRule*(E : conceptualElement, T_1 : translationRule) which changes the translation rule applied to an element during the translation process.

An example of evolution appears in Figure 2, in which, starting from the schema S_0 , the attribute **ssn** is set as the new key of the entityType **employee**, a new subtype of **employee** is defined. Besides, the translation rule *tableForRoot* (‘only one table for the root of the ISA hierarchy’) is established.

4 Evolving the Implementation of ISA Relationships

The changes in the EER schema are propagated to the physical database by means of the propagation algorithm. This is split into the propagation subalgorithms for the translation, for the logical and for the extensional components. In this paper, we will explain in more detail the last subalgorithm.

The first two subalgorithms change, respectively, the translation and the logical components. Due to the fact that we have adopted a metamodeling perspective, the logical component is based on a metamodel of the relational model (not shown here) and a set of logical transformations (see examples in Table 1). Then, every conceptual transformation produces in the translation component a set of added, deleted and affected elementary translations. This information is notified to the logical component giving rise to a set of logical changes expressed by means of the logical transformations. Additional details on the subalgorithms and about the elementary translations can be seen in Domínguez et al. [3].

For instance, for the transformation *changeKey* of our running example, the logical changes applied by the propagation subalgorithm can be seen in Table 1(a). Similarly, the conceptual transformations *addSubtypeAndIsaSet* and *changeTranslationRule* trigger the changes indicated in Tables 1(b)(c). We are aware that other changes (e.g., related with relationship types) could be necessary but we consider that they are not relevant to the issues dealt with in this paper.

Table 1. Logical changes after applying the conceptual transformations

(1) <code>changeLogicConstrTarget('pk','employee','id','ssn')</code>	(12) <code>addColumn('job','varchar(10)','employee')</code>
(2) <code>addColumn('ssn','integer','secretary')</code>	(13) <code>addColumn('techgrade','varchar2(10)','employee')</code>
(3) <code>changeLogicConstrTarget('pk','secretary','id','ssn')</code>	(14) <code>addColumn('typingSpeed','integer','employee')</code>
(4) <code>changeLogicConstrSourceAndTarget('fk','secretary.id','secretary.ssn','employee.id','employee.ssn')</code>	(15) <code>deleteConstraint('pk','secretary','ssn')</code>
(5) <code>deleteColumns('id','secretary')</code>	(16) <code>deleteConstraint('pk','technician','ssn')</code>
(a) Logical changes for <code>changeKey('employee','ssn')</code>	(17) <code>deleteConstraint('fk','secretary.ssn','employee.ssn')</code>
(6) <code>addColumn('techGrade','varchar(10)','technician')</code>	(18) <code>deleteConstraint('fk','technician.ssn','employee.ssn')</code>
(7) <code>addColumn('ssn','integer','technician')</code>	(19) <code>deleteColumns('ssn','typingSpeed','secretary')</code>
(8) <code>addTable('technician')</code>	(20) <code>deleteColumns('ssn','techGrade','technician')</code>
(9) <code>addConstraint('pk','technician','ssn')</code>	(21) <code>deleteTable('secretary')</code>
(10) <code>addConstraint('fk','technician.ssn','employee.ssn')</code>	(22) <code>deleteTable('technician')</code>
(11) <code>addConstraint('check-excl','employee','secretary','technician')</code>	(c) Logical changes for <code>changeTranslationRule('employee','tableForRoot')</code>
(b) Logical changes for <code>addSubtypeAndIsaSet('employee','technician','techGrade','secretaryIsaEmployee','job',True,False)</code>	

Table 2. Sketch of the propagation subalgorithm for the extensional level

INPUT: Set of changes in the logical database and information about the added, deleted or affected elementary translations

OUTPUT: Set of SQL sentences to be executed against the physical schema

For each logical change *c* of the INPUT

Find the correspondence rule *r* (if exists) such that:

- the event of *r* matches the logical change *c* AND
- the condition of *r* is met

Execute the action of *r*

End For

The logical changes produced by the conceptual transformations are the input for the propagation subalgorithm for the extensional component. This subalgorithm (see Section 4.1) triggers, for each logical change, the correspondence rule that has to be executed. The execution of a correspondence rule changes the structure of the physical database as well as the extension. This last task is done by the data load procedures for the extensional component (see Section. 4.2).

4.1 Propagation Subalgorithm for the Extensional Component

In Table 2 we show a sketch of the propagation subalgorithm for the extensional component. This is based on a set of correspondence rules. Basically, a correspondence rule (see an example in Table 3) is an ECA rule, Elmasri et al. [6], with the following meaning for its components:

1. The event that triggers the rule. This is a transformation on the logical component.

Table 3. An example of correspondence rule

1. **Name:** populateNewIsaKeyColumns
2. **Event:** Add the columns newCols with datatypes **datatypes** to a table t
3. **Condition:** (the event AffectedElementaryTranslation(e) has been notified by the translation component) AND (in the elementary translation e the conceptual element is an ISA hierarchy h and the translation rule applied is tablePerEntity) AND (the transformation changeLogicConstrTarget changes the primary key of the table of the root of h) AND (t is the translation of an entity type et which belongs to the ISA hierarchy h but it is not the root)
4. **Action:**

```

tr←tableOf(root)
SQLNewColumns(newCols,datatypes,t)
columnDataLoad(targetColumns→newCols,
  sourceColumns→newPkColsOf(tr), targetTable→t,
  sourceTable→tr, targetJoinColumns→oldPkColsOf(t),
  sourceJoinColumns→oldPkColsOf(tr))

```
2. The condition that determines whether the rule action should be executed. This is expressed in terms of the conceptual and logical elements involved in the affected elementary translations and in terms of the logical changes.
3. The actions. These are the procedures which generate SQL sentences that change the data in a consistent way.

With respect to the action part of the example rule of Table 3, the procedure `SQLNewColumns` adds new columns to the physical database and the procedure `columnDataLoad` copies data between tables (see its description in the next subsection).

The determination of the complete set of correspondence rules has been conducted following a pattern of study. According to this pattern, the effect on the physical database has to be analyzed starting from the initial schema, the translation rule applied and the conceptual transformation. As regards this pattern, we want to highlight that, thanks to the translation component of our architecture, the extensional changes are analyzed in terms of conceptual changes. As a consequence, the application of this pattern to the particular case of ISA relationships (see Table 4) allows us to analyze ISA evolution issues within the relational model (which lacks the ISA concept). From our point of view, this is one of the most noteworthy contributions of our approach.

Our algorithm at work. We have built it in such a way that, for each logical transformation, at most one correspondence rule is executed. For each logical change, the subalgorithm searches the correspondence rule (if any) with the same event as the event of the logical change and for which the condition of the rule evaluates to true.

For instance, for the logical transformation (2) of Table 1 the rule with the same event and whose condition evaluates to true is `populateNewIsaKeyColumns`. In this case the action is formed by the SQL sentences (c) and (d) of Table 6. For the logical transformation (8) of Table 1 the action corresponds to the case (2) of the pattern example of Table 4 in which the new `isaSet`

Table 4. Pattern for studying the ISA evolution at the physical level

1. **Description of the initial schema:** A parent entity type *p*, a set of subtypes and ISA relationships between them.
2. **Translation rule applied:** `tablePerEntity`
3. **Conceptual transformation:** `addSubtypeAndIsaSet(p, d, la, li, disc, -, -)`
4. **Effect on the physical database:** The effect depends on the constraints for the newly created isa set. We distinguish four cases:
Case 1. New isa set constraints: `isDisjoint=true, isCovering=true`.
 Action: (a) Every row of the table *p* which is not in the rest of tables is copied to table *d*. This process can be done automatically.
Case 2. New isa set constraints: `isDisjoint=true, isCovering=false`.
 Action: (b) The user informs about which rows of the table *p* which are not in the rest of tables are copied to table *d*.
Case 3. New isa set constraints: `isDisjoint=false, isCovering=true`.
 Actions: (a) of case 1 and (c) the user informs about which rows of the rest of tables must be copied into table *d*.
Case 4. New isa set constraints: `isDisjoint=false, isCovering=false`.
 Actions: (b) and (c)

constraints are `isDisjoint=true` and `isCovering=false`. For this case, the user informs about which rows of table `employee` which are not in table `secretary` must be in table `technician`. In the example, the action executes the SQL sentences (i) for creating the table `technician` and (j) for adding the only employee that the user has informed is also a technician.

For the complete running example, the SQL sentences generated by this sub-algorithm can be seen in Table 6. Sentences (a) to (h) correspond to the logical changes due to the conceptual transformation `changeKey`, sentences (i) to (l) to the transformation `addSubtypeAndIsaSet` and sentences (m) to (t) to the transformation `changeTranslationRule`.

4.2 Data Load Procedures for the Extensional Component

In the action part of the ECA rules, the changes in the extension of the database are done by the data load procedures (see an example in Table 5). Each procedure has a precondition associated to it which states the circumstances in which it can be applied. In this respect, the main issue is that whenever a data load is used in an ECA rule, the condition of the ECA rule must ensure that the precondition of the data load holds. The procedure example of Table 5 adds values to new columns of a target table taking these values from columns of other source table. In this case the precondition establishes that a unique row exists in the source table each time values have to be obtained from the source columns.

A particular context where our data load procedure example appears is when we change the key of the root of an ISA hierarchy. Then, at the logical level, new columns (which will be the new columns of the primary key of the table) must be added to each table of each subtype of the ISA hierarchy. The values for these columns must be drawn from the values of the corresponding columns in the

Table 5. An example of data load procedure

Procedure	columnDataLoad
Parameters	targetColumns, sourceColumns, targetTable, sourceTable, targetJoinColumns, sourceJoinColumns
Precondition	In sourceTable exists an unique row that satisfies the join condition
Description	This procedure loads data in the targetColumns. For each row <i>r</i> of targetTable, the value of targetColumns is s.sourceColumns where <i>s</i> is the unique row of sourceTable such that s.sourceJoinColumns= <i>r</i> .targetJoinColumns
Code	UPDATE targetTable SET target_columns=(SELECT sourceColumns FROM sourceTable WHERE sourceJoinColumns=targetJoinColumns)

table of the root of the ISA hierarchy. In order to do this the `columnDataLoad` procedure can be used since the particular context of this change ensures the precondition of the SQL procedure. In our example, this procedure is applied when the key of `employee` is changed from `id` to `ssn`, since a new column `ssn` is added to the table `secretary` of Figure 1(b). Then, for each row *r* of the table `secretary` we have to complete the column *r.ssn* with the value of the unique row *e* of table `employee` such that *e.id=r.id*. This task is performed, inside the `populateNewIsaKeyColumns` correspondence rule, by the following procedure:

```
columnDataLoad(
  targetColumns→'secretary.ssn', sourceColumns→'employee.ssn',
  targetTable→'secretary', sourceTable→'employee'
  targetJoinColumns→'secretary.id', sourceJoinColumns→'employee.id')
```

5 Related Work

In our current work, we aim to contribute to the evolution context by studying the evolution of ISA relationships. Several recent papers have addressed similar matters (for example Rodríguez et al. [8]). Following the ideas proposed in Ram et al. [10], we are going to compare our approach with these other proposals along two different dimensions: (1) the data models used and (2) the schema evolution activities covered.

Three different types of data model are considered in Ram et al. [10]: object oriented, relational and semantic models. In the case of ISA evolution, the majority of research is proposed for object oriented data models [1,8]. Fewer works analyze the case of ISA evolution within semantic models (Claypool et al. [2]) and, since the relational model lacks the ISA concept, no work can be found in this case.

The most significant difference of our approach as opposed to other proposals is precisely that, thanks to the fact that our evolution architecture fits together a semantic model (the EER in the case we are explaining) and the relational model, we can analyze the propagation of the conceptual evolution of ISA

Table 6. Generated SQL sentences which change the physical component

- a. ALTER TABLE employee DROP CONSTRAINT pk1 CASCADE
- b. ALTER TABLE employee ADD CONSTRAINT pk1 PRIMARY KEY (ssn))
- c. ALTER TABLE secretary ADD ssn INTEGER
- d. UPDATE secretary SET secretary.ssn= (SELECT employee.ssn FROM employee WHERE employee.id=secretary.id)
- e. ALTER TABLE secretary DROP CONSTRAINT pk2
- f. ALTER TABLE secretary ADD CONSTRAINT pk2 PRIMARY KEY(ssn)
- g. ALTER TABLE secretary ADD CONSTRAINT fk2 FOREIGN KEY (ssn) REFERENCES employee(ssn)
- h. ALTER TABLE secretary DROP COLUMN id
- i. CREATE TABLE technician(ssn INTEGER,techgrade VARCHAR2(10))
- j. INSERT INTO technician VALUES (10,'Superior')
- k. ALTER TABLE technician ADD CONSTRAINT pk3 PRIMARY KEY(ssn)
- l. ALTER TABLE technician ADD CONSTRAINT fk3 FOREIGN KEY (ssn) REFERENCES employee(ssn)
- m. ALTER TABLE employee ADD job VARCHAR2(10)
- n. ALTER TABLE employee ADD techgrade VARCHAR2(10)
- o. UPDATE employee e SET e.techgrade=(SELECT t.techgrade FROM technician t WHERE t.ssn=e.ssn), e.job='tech' WHERE EXISTS (SELECT * FROM technician t WHERE t.ssn=e.ssn)
- p. ALTER TABLE employee ADD typingspeed INTEGER
- q. UPDATE employee e SET e.typingspeed=(SELECT s.typingspeed FROM secretary s WHERE s.ssn=e.ssn), e.job='secr' WHERE EXISTS (SELECT * FROM secretary s WHERE s.ssn=e.ssn)
- r. ALTER TABLE employee ADD CONSTRAINT ckeck2 CHECK(((job='tech') AND (typingspeed IS NULL)) OR ((job='secr') AND (techgrade IS NULL)) OR ((job IS NULL) AND (techgrade IS NULL) AND (typingspeed IS NULL)))
- s. DROP TABLE secretary
- t. DROP TABLE technician

relationships to the relational model and to the extension. The present paper is, to our knowledge, the first approach to dealing with this research issue.

As for the research activities related with schema evolution, three types are distinguished in [10]: core schema evolution, version management and management of applications. Since core schema evolution is concerned with the propagation of schema changes to the data, the ISA evolution issues analyzed in this paper lie within this research activity. Activities related with the manipulation of applications and schema versions represent a goal for further development.

The various papers dealing with keeping data in a consistent state after schema changes occur differ in the way they preserve this consistency. The data conversion mechanism we have adopted is the strict conversion method, in which changes are immediately propagated to the data. As is pointed out in Roddick et al. [9], this mechanism, in contrast to the other two options (lazy and logical), has longer schema modification time but reduces subsequent data access time.

In this respect, we want to stress the fact that the consistency is kept taking into account changes propagated from the conceptual level to the physical level, through the logical level. Besides, this propagation is carried out in a traceable

way, so that information stored about the translation process allows the propagation to be done more easily. Finally, we want to highlight the fact that we follow a metamodeling approach with the aim of being generalizable and adaptable when other activities and data models are used.

6 Conclusions and Future Work

This paper has focused on the definition, description and implementation of evolution operations that could be applied to the evolution of ISA relationships. We have shown how starting from the issue of an evolution primitive at the conceptual level, our architecture propagates the changes. In this work we have focused especially on physical, data aspects. We could enrich our setting by investigating how to include in it issues such as version management and management of applications. Also, we will have to do further research into the setting of quality measurements (for instance by the use of some metrics) in order to decide which are the most suitable evolution possibilities and implementations. Formal discussions about the families of transformations considered in our framework remains another open issue.

References

1. L. Al-Jadir, M. Léonard, Multiobjects to Ease Schema Evolution in an OODBMS, in T. W. Ling, S. Ram, M. L. Lee (eds.), *Conceptual modeling, ER-98*, LNCS 1507, Springer, 1998, 316–333.
2. K. T. Claypool, E. A. Rundensteiner, G. T. Heineman, ROVER: flexible yet consistent evolution of relationships, *Data Knowl. Eng.* 39(1), 2001, 27–50.
3. E. Domínguez, J. Lloret, A. L. Rubio, M. A. Zapata, Elementary translations: the seesaws for achieving traceability between database schemata, in [5], 377–389.
4. E. Domínguez, M. A. Zapata, Towards a Situational Method Engineering Technique, accepted for publication in *Information Systems*.
5. Proceedings of the ECDM 2004 Workshop, in Shan Wang et al. (eds.), *Conceptual Modeling for Advanced Application Domains: ER 2004 Workshops*, LNCS 3289, 2004, 349–456.
6. R. A. Elmasri, S. B. Navathe, *Fundamentals of Database Systems (4th ed.)*, Addison-Wesley, 2003.
7. J. L. Hainaut, J. M. Hick, V. Englebert, J. Henrard, D. Roland, Understanding Implementations of IS-A Relations, in B. Thalheim (ed.), *Conceptual Modeling-ER'96*, LNCS 1157, 1996, 42–57.
8. E. Rodríguez, A. Abelló, M. Oliva, F. Saltor, C. Delgado, E. Garvi, J. Samos, On operations along the generalization/specialization dimension, in *Int. Workshop on Engineering Federated Information Systems, EFIS*, IOS Press, 2001, 70–83.
9. John Roddick, A survey of schema versioning issues for database systems, in *Information Software Technology*, 37 (7), 1995, 383–393.
10. Sudha Ram, G. Shankaranarayanan, *Research issues in database schema evolution: the road not taken*, working paper # 2003-15, 2003.
11. OMG, UML 2.0 Superstructure Spec., formal/05-07-04, 2005, www.uml.org.

Schema Change Operations for Versioning Complex Objects Hierarchy in OODBs

Sang-Won Lee*

School of Information and Communication Engineering, Sungkyunkwan University
300 Cheoncheon-dong, Jangan-gu, Suwon, Gyeonggi-do 440-746, Korea
swlee@skku.edu

Abstract. In this paper, we propose a set of schema change operations which allows to restructure complex object hierarchy in object-oriented databases. This model extends a schema version model, called *RiBS*, which is based on the concept of *Rich Base Schema*. In the *RiBS* model, each schema version is in the form of updatable class hierarchy view over one base schema, called the *RiBS* layer, which has richer schema information than any existing schema version in the database.

In this paper, we introduce new operations for restructuring composite object hierarchy in schema versions, and explain their semantics. The originality of this paper lies in that 1) we introduce several new operations to restructure composite object hierarchy, and 2) this extended *RiBS* model operations raise the concept of data independence in OODBs upto the schema level.

1 Introduction

Recently, many new database applications such as CAD, CASE and WWW have emerged. One common requirement from these applications is to model complex objects and the relational databases are not adequate for them. In order to support these applications, many object-oriented data models have been developed since the mid 1980s, and the advantages of object-oriented database systems (OODBMSs) have been widely recognized in terms of rich modeling power and performance. Unlikely from the traditional relational applications, OODB applications require powerful schema management such as dynamic schema changes and flexible schema management. Therefore, there has been much work on schema evolutions in OODBMSs [10], and some commercial OODBMSs (e.g. O2 [10] and ObjectStore [7]) support basic schema evolution operations.

With these systems, however, only a single schema can exist at any point in time: if a schema evolution operation completes, the previous schema state is no longer valid. This single schema modification mechanism has several drawbacks. First, schema updates may invalidate programs written against old schema. Second, because all the users share a single schema, a schema update by one user

* This research was supported by MIC, Korea under ITRC IITA-2005-(C1090-0501-0019).

may change the views of all other users. These shortcomings can be summarized as the lack of logical data independence in OODBMSs. In order to overcome these drawbacks, schema version mechanism has been introduced [3].

Recently, schema version has been re-motivated in several new OODB applications such as Repositories, Portable Common Tool Environment (PCTE), and WWW. Thus, in order for OODBMSs to be adopted in the new applications, the functionality of schema versions is essential. In fact, a commercial OODBMS, POET, supports basic schema version mechanism [6]. There have been several approaches to schema version mechanisms in OODBs [3,5,8,1], but they are not satisfactory yet. In particular, these works have mainly focused on schema versions over class inheritance hierarchy. To our best knowledge, there has been no work on schema versions for complex object hierarchy. In fact, many OODBMS applications requires schema versions over complex object hierarchy, rather than over class hierarchy.

In [4], we developed a simple-yet-powerful schema version model, *RiBS*, which is based on two concepts of 'rich base schema' and 'updatable class hierarchy view'. We call this model as the basic *RiBS* model, because it supports schema versions only for class hierarchy. This paper proposes an extended schema version model, called the extended *RiBS* model, to restructure the complex object hierarchy. Our contribution is to propose a set of schema evolution operations for complex object hierarchy and to give the semantics of each operation. There has been no work on this issue. However, in relational databases, Roddick et al. [9] proposed a schema change taxonomy which is very similar to our model. And, the work by Grandi et al. [1] provide some schema change operations for versioning complex object hierarchy, but they do not provide a comprehensive taxonomy.

This paper is organized as follows. Section 2 briefly explains the object model and the basic *RiBS* model. Section 3 proposes new schema evolution operations for restructuring complex object hierarchy, and explain their semantics. Section 4 concludes this paper.

2 Object Model and Basic *RiBS* Model

2.1 Object Model

A user-defined class, as well as the built-in data type, can be used as the domain of an attribute of a class, and an instance object of this class may have the object identifier of an object of that class as its value of the attribute. The object-oriented data model, in contrast to the relational data model in which the relationships between entities are implicitly represented using data values, explicitly represents the relationships using object identifiers. These reference relationships between classes make a complex object hierarchy. Even though the term 'complex object' is generally used to represent the *part-of* relationship as in [2], it is also used, in this paper, to cover simple references between objects.

Figure 1 shows an example of complex object hierarchy throughout this paper. The complex object hierarchy has the VEHICLE class as its root class and each of the other classes as its part class. In Figure 1, for the simplicity, we use the

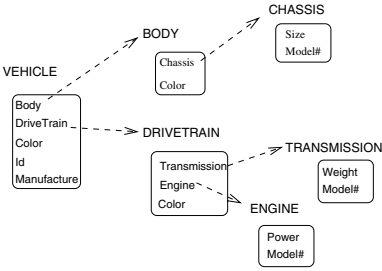


Fig. 1. VEHICLE Complex Object Hierarchy

same name for a class and an attribute which refers the class. A dotted arrow in Figure 1 represents a reference relationship from a starting class to an end class. Each attribute in a class can be classified into either a leaf or a non-leaf attribute depending on the type of its domain: that is, an attribute whose domain is a built-in class, such as integer and string, is called a leaf attribute, while an attribute whose domain is a user defined class is a non-leaf attribute.

2.2 Basic RiBS Model

Figure 2 shows the structural component of the basic *RiBS* model. It has a three-layered architecture: (1) (extensional) object base layer, (2) *RiBS* layer, and (3) schema version layer [4]. Each user is interested only in each schema version in the schema version layer. In turn, each schema version is in the form of a class hierarchy view over the next *RiBS* layer. The *RiBS* layer accumulates all the necessary schema information ever defined in any schema version, including classes, inheritance, attributes, and relationships. Finally, the object base layer physically stores all the instance objects of the *RiBS* layer.

With the *RiBS* model, user can execute arbitrary schema evolution operations against current schema version. The *RiBS* model supports all the schema change operations in the taxonomy of Orion OODBMS [2]. As shown in Figure2, direct schema updates on schema versions are allowed, and their effects are, if necessary, automatically propagated down to the *RiBS* layer and/or to the object base layer. At a certain point of time, either a user or an application can access and manipulate database through a specific schema version, which we call *current schema version (CSV)*. A schema version is actually a logical view over the *RiBS* layer in the sense that all the objects visible through a schema version are derived from the objects stored in the extensional object base. Thus, an application program or a query over a schema version, before its execution, need to be automatically rewritten into one over the *RiBS* layer.

Meanwhile, it should be noted in Figure 2 that any schema change within a schema version has no effect on other schema versions in the database. That is, every schema change operation in the basic *RiBS* model has effects only on the schema version itself and the *RiBS* layer. One important characteristic of the *RiBS* model is to allow direct schema updates over a schema version which is a

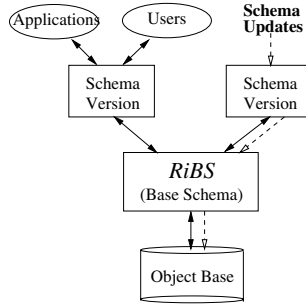


Fig. 2. *RiBS* Model: Architecture

class hierarchy view over the *RiBS* layer. This allows us to avoid some disadvantages of the existing models citekim88,monk93 1) storage overhead due to many replicated copies for an object, 2) the complexity in schema change process in view approaches, and (3) schema management overhead in class version model.

3 Restructuring Complex Object Hierarchy

In this section, we give a set of operations for restructuring complex object hierarchy; its taxonomy and a detailed semantics of each operation. The following operations are introduced for restructuring complex object hierarchy. The first two operations are originally included in the basic *RiBS* model, but we include them in this taxonomy because they can be regarded as to restructure complex object hierarchy. The remaining four operations are newly introduced in the extended *RiBS* model.

1. Add a part class into complex object hierarchy
2. Remove a part class from complex object hierarchy
3. Pull an attribute of a part class to root class: **pull**
4. Unnest a part class on root class: **unnest**
5. Nest part attributes into a new virtual part class: **nest**
6. Move a part attribute to a part class: **move**

Please note that only the first operation will affect the *RiBS* layer, while others have no effect on the *RiBS* layer. The last five operations are intended to customize complex object hierarchy, and thus we need to simply change the mapping information between *RiBS* layer and current schema version.

Add a Part Class into Complex Object Hierarchy. As we already mentioned, the operation to add a new attribute into a class version, which comes from the basic *RiBS* model, can be regarded as an operation to add a part class into complex object hierarchy. That is, when the domain of the new attribute pv of class version CV_1 is another class version CV_2 on the current schema version CSV , the complex object hierarchy which contains CV_1 has class version CV_2 as its new part class. In this case, it should be noted that a new base class bp corresponding to pv is added to the *RiBS* layer.

Remove a Part Class from Complex Object Hierarchy. A part class can be removed from a complex object hierarchy using the operation *drop an existing attribute version v from a class version*, which also comes from the basic *RiBS* model. Using this operation, we can drop an attribute version *pv* whose domain is the class version to be removed from a complex object hierarchy. In this case, in contrast to adding a part class into a complex object hierarchy, only the information about *pv* is removed from the current schema version, and the corresponding base attribute still remains in the *RiBS* layer.

Pull a Property of a Part Class to Root Class. In the object-oriented data model, an attribute on a superclass is inherited into a subclass downwards along the class hierarchy via the inheritance mechanism. Similarly, an attribute in a part class on a complex object hierarchy can be conceptually viewed as an attribute of its root class; that is, ‘an attribute in a part class *is also an attribute of its root* (or any whole) class upwards along the complex object hierarchy’.

For easy access to an attribute which are deeply nested from its root class, users would like to view the attribute as if it is defined on root class. This is very similar to a relational view which is defined using several complex join conditions and thus can be used as a shorthand for long and complex queries. The following operation allows users to do this kind of restructuring.

```
pull path_exp [as new_attribute_name] in class class_name;
```

The semantics of **pull** operation is that on a complex object hierarchy having the root class *class_name*, the leaf attribute over path expression *path_exp* is pulled to the root class. When the leaf attribute causes a name conflict with an attribute which is locally defined in the root class, this operation is rejected. In order to avoid this kind of name conflict or to assign a more meaningful name to the attribute, the user can use the optional **as** clause. For example, the following operation pulls the attribute of *Model#* in class version *CHASSIS* in Figure 1 to root class *VEHICLE*, and renames the attribute *ChassisModel* in the root class

```
pull Body.Chassis.Model# as ChassisModel in class VEHICLE;
```

Figure 3 shows the result of this operation (for the simplicity of presentation, we assume here that before running this operation, the structure of the complex object hierarchy rooting from class version *VEHICLE* is exactly the same as that of the *RiBS* layer). As the result of the **pull** operation, a new attribute **ChassisModel** is introduced in class version *VEHICLE*, while the attribute version *Model#* is removed from class version *CHASSIS*, the reason for which is given later in this section. Please note that the domain of new attribute *ChassisModel* is same to that of *Body.Chassis.Model#*.

The above example shows how to upwardly pull (that is, move) a leaf attribute in a complex object hierarchy. In addition, the **pull** operation can be used to the non-leaf attribute, as shown below.

```
pull DriveTrain.Engine in class VEHICLE;
```

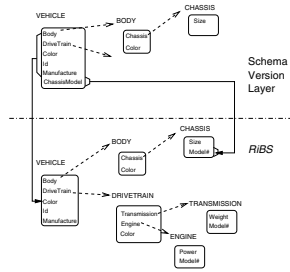


Fig. 3. Restructured Complex Object Hierarchy using the **pull** Operation

Moreover, the **pull** operation can be applied to the cases where the path expression is cyclic. Figure 4 exemplifies how the **pull** operation is used to introduce a new attribute *GrandFather* in class *PERSON* which is a root class of a complex object hierarchy with a cyclic path.

pull Father.Father as GrandFather in class PERSON;

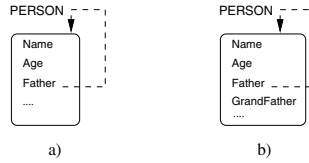


Fig. 4. Restructuring a Cyclic Complex Object Hierarchy using the **pull** Operation

Unnest: Unnest a Part Class on Root Class. The **unnest** operation generalizes the **pull** operation described previously. It allows information which was modeled as a part class to be represented by a set of attributes of the root class.

unnest path_expression in class class_name;

Using the above syntax, all of the attributes of the class corresponding to the leaf node of the path expression *path_exp* are pulled upwards into the root class *class_name*. For this operation to be valid, the path expression should not end with a leaf attribute. The following example shows how this **unnest** operation is used to restructure the information modeled in a part class *ENGINE* in Figure 1 as attributes of root class *VEHICLE*.

unnest DriveTrain.Transmission in class VEHICLE;

This operation changes the complex object hierarchy of *VEHICLE* into the status shown in Figure 5; while all of the attributes of class *ENGINE* move upwards to class *VEHICLE* preserving their names and domains, the attribute *Engine* is deleted from class *DRIVETRAIN* and class *ENGINE* itself disappears from the complex object hierarchy. The two attributes circled in class *VEHICLE* has come from the unnested class *ENGINE*.

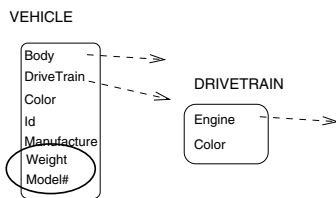


Fig. 5. Restructured Class *VEHICLE* using **unnest** operation

Note that the upwardly pulled attribute *Weight* in Figure 5 models the weight of the engine of a vehicle, not that of the vehicle itself, and thus the name of the attribute needs to be changed. To do this, we can use the operation **change the name of an attribute version** from the basic *RiBS* model.

The **unnest** operation is similar to the **pull** operation in that both operations move attributes of part classes to the root class, but there exists a semantic difference between them. In the **unnest** operation, a part class is removed from the complex object hierarchy, while, in the **pull** operation, only an attribute is moved to the root class with its origin class surviving. In the latter case, the class version is still to be modelled, even after the operation **pull**.

Nest: Nest Part Properties into a New Virtual Part Class. The **nest** operation is converse to the **unnest** operation. The **nest** operation allows us to create a new virtual part class by combining several attributes from root class and part classes, and then models the new class as the domain of an attribute of the root class.

nest *new_class_name*{*attribute_list*} **as** *new_attribute_name* **in** class *class_name*;

In the above syntax, *new_class_name*, *new_attribute_name* and *attribute_list* represents the name of the new virtual class, the name of the new attribute, and the list of the attributes to be included in the virtual class, respectively. *attribute_list* is a list of attributes, whose syntax is as follows:

{*path_exp* [**as** *new_attribute_name*] }*

We call the new class version resulting from the **nest** operation as a virtual class version because the new class version does not have its corresponding base class in the *RiBS* layer. In fact, its attributes may originate from several different class versions and thus the attributes have their corresponding base attributes spreaded over different base classes. In contrast, each class version in the basic *RiBS* model has its corresponding base class in the *RiBS* layer, and in this respect, all the class version in the basic *RiBS* model are non-virtual.

The following example shows how a virtual class *PARTMODEL* is created, using the **nest** operation: the **nest** operation collects all the model numbers of the parts of class *VEHICLE* into the class *PARTMODEL*. Figure 6 shows the result of this operation, where a new attribute *PartModel* is added to the class *VEHICLE* and it has the new virtual class *PARTMODEL* as its domain.

```

nest PARTMODEL { Body.Chassis.Model# as ChassisModel#,
                  DriveTrain.Transmission.Model# as TransModel#,
                  DriveTrain.Engine.Model# as EngineModel# }
as PartModel in class VEHICLE;

```

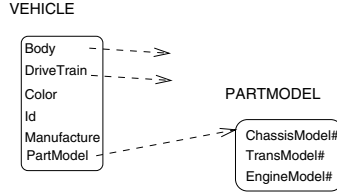


Fig. 6. An Example of the **nest** Operation

As we have noted above, the **nest** operation creates a virtual class version in the current schema version. However, the virtual class version raises a subtle issue with regard to schema evolutions. Since a virtual class version is also a class version, all valid schema evolution operations for class versions should also be applicable to this virtual class version. However, unlike the basic *RiBS* model where each class version has its corresponding base class in the *RiBS* layer, each virtual class version in the extended *RiBS* model does not have its base class in the *RiBS* layer, and thus, for example, when the schema evolution operation **add a new attribute version** is imposed on it, it does not have base class to which a corresponding base attribute is added. We take a simple approach to this problem, that is, when the operation *add a new attribute version* is imposed on a virtual class version, we create a corresponding base class in the *RiBS* layer with a base attribute corresponding to the new attribute version, and the class version becomes non-virtual.

Move: Move a Part Property to a Part Class. In some cases, a user may want to customize his/her own view for a complex object hierarchy in a different way so that an attribute version of a part class is modeled as an attribute version of another part class which is usually in a path from root class different from the original part. In order to support these requirements, we introduce the following **move** operation.

```

move src_path_exp to dest_path_exp [as new_attribute_name] in class
      class_name;

```

The above operation moves the (leaf or non-leaf) attribute in path expression *src_path_exp* to the class in path expression *dest_path_exp*. *dest_path_exp* must be a non-leaf node. In this case, the attribute being moved disappears from the original class, according to the principle of the *RiBS* model described above. The **as** clause is optionally used to change the name of the attribute.

The following exemplifies the usage of the **move** operation: the *Chassis* attribute of class *BODY* in Figure 1 moves to class *DRIVETRAIN*. Figure 7 shows

the result of this operation, where the new attribute *BodyChassis* is added into the class *DRIVETRAIN*.

move Body.Chassis to DriveTrain as BodyChassis in class VEHICLE;

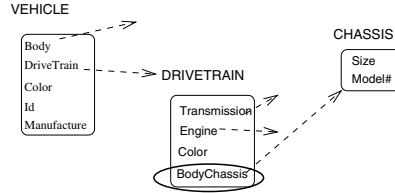


Fig. 7. The **move** Operation: An Example

However, with regard to the **move** operation, there is a constraint on the two path expressions *src_path_exp* and *dest_path_exp*, that is, *src_path_exp* should not be a part of *dest_path_exp*. The following example violates this constraint.

move DriveTrain to DriveTrain.Transmission in class VEHICLE;

The above operation, as shown in Figure 8, causes two problems; 1) class *DRIVETRAIN* is no longer a part class of the complex object hierarchy, and 2) class *DRIVETRAIN* becomes a part class of class *TRANSMISSION*. As a result

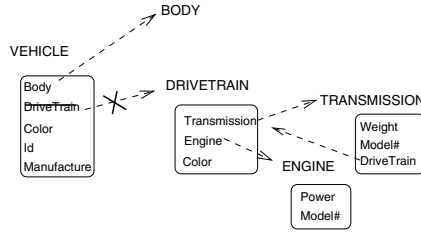


Fig. 8. An Illegal Usage of the **move** Operation

of the first problem, the information available in the complex object hierarchy with *DRIVETRAIN* as its root is unreachable from the root class *VEHICLE*. The second problem introduces the relationship $IsPartOf(B, A)$, which is contradictory to the relationship $IsPartOf(A, B)$. Due to these problems, we do not allow the **move** operation which violates the constraint.

4 Conclusions

In this paper, we proposed a schema version model for complex objects. The model includes a set of new operations for restructuring complex object hierarchy, with which a user can customize database schema according to his/her own

view. Because our model has taken into account the concept of complex object, it is more comprehensive than any other schema version model ever developed. Also, these schema evolution operations for complex objects can be used as an alternative mechanism for view definitions in OODBMSs.

References

1. F. Grandi and F. Mandreoli. A formal model for temporal schema versioning in object-oriented databases. *Data Knowledge Engineering*, 46(2), 2003.
2. W. Kim. *Introduction to Object-Oriented Databases*. MIT Press, 1990.
3. W. Kim and H. Chou. "Versions of Schema for Object-Oriented Databases". *Proceedings of VLDB*, 1988.
4. S. Lee and H. Kim. "Rich Base Schema(RiBS) :A Unified Framework for OODB Schema Version". *Journal of Database Management*, pages 33–41, Jan. 2000.
5. S. Monk and I. Sommerville. "Schema Evolution in OODB Using Class Versioning". *SIGMOD Record*, 22(3), 1993.
6. POET Software. "POET Object Server Suite". *TR*(www.it-analysis.com/researcharchivepdf.php?id=309), 2004.
7. Progress Software. "ObjectStore Data Sheet, Version 6.1". *TR*(www.objectstore.net/products/docs/objectstore_datasheet.pdf), 2004.
8. Y.-G. Ra and E. A. Rundensteiner. A transparent schema-evolution system based on object-oriented view technology. *IEEE Trans. Knowl. Data Eng.*, 9(4), 1997.
9. J. Roddick, N. Craske, and T. Richards. A taxonomy for schema versioning based on the relational and entity relationship models. *12th ER Conference*, 1993.
10. R. Zicari and F. Ferrandina. "Schema and Database Evolution in Object Database Systems". In *Part6, Advanced Database Systems*, 1997.

Representing Versions in XML Documents Using Versionstamp

Luis Jesús Arévalo Rosado, Antonio Polo Márquez, and Juan M^a Fernández González

Department of Computer Science, University of Extremadura, Spain¹
{ljarevalo, polo, jmfergon}@unex.es

Abstract. The problem of managing versions in XML documents can be approached through traditional adapted procedures, based on managing XML operations (deltas) or using timestamped markups to represent the validity of each versioned tag within the document. The first solution entails a high reconstruction cost for any version different from the current one. Whereas the second solution, due to the linear nature of time, implies that these techniques do not support branched versioning. In this work, the XML data model is extended for the representation of different versions of XML documents that consists of marking the tags with a versionstamp instead of using a timestamp. This technique is based on two ideas: on the one hand storing the ancestral relations of the versions (version tree) produced a new version is generated and on the other hand the version validity of each versioned tag is defined based on this tree (versionstamp). The easy management of multiple versioning, the wide number of queries in XML standard query languages and its implementation only using XML technology, are some of the advantages of the proposed technique.

1 Introduction

The standard XML [1] has been consolidated as a tool for information representation and interchange, due to both its flexibility and the management and conversion facilities that the documents offer in this standard. Since its specification a great variety of languages with different functionalities have arisen in research and commercial areas (SOAP, WSDL, XSLT, XML-Schema, etc) [1].

In essence, any XML document can be considered as semi-structured data to store information, so that as it occurs in any other information system, they are susceptible to change. The versions of an XML document can be managed through traditional adapted procedures, based on XML operations change (delta XML) [2, 3] or integrating the different versions of the XML document into a single file using temporal aspects [6, 7]. The first solution prevents the validation and query of the different versions by means of the XML language standard and the second one, due to the linear nature of time causes these techniques not to support branched versioning and for this reason this technique cannot be used in collaborative scenarios [4] where new versions can be generated from any past version.

¹ This work has been financed by the Spanish CICYT project "TIN2005-09098-C05-05".

In this work, we extend the XML data model for the representation and management of different versions of XML documents that consists of marking the tags with a versionstamp instead of using a timestamp. In order to do this effectively it is necessary to store the versions included in the document and their relations (ancestral relation), thereby the version validity for each mark in the document is defined based on it. The following advantages should be highlighted: the easy management of non linear versioning, the possibility to query the different versions using XML query languages standards and it runs in whatever XQuery/XSLT engine with id() support[1].

The remainder of this paper is organized as follows: we begin by summarizing the current solutions for the management of XML versions. Then, we show a data model for XML which is extended to a version model and later we show how to map it to XML documents. Next, we describe different queries that can be made on versioned documents. Finally we offer our conclusions and a look at our future works.

2 State of the Art

The problem of XML document version management combines the issues of document version management [2, 3, 4, 5] and temporal databases [10].

Delta XML management: Based on traditional change operation procedures adapted to XML [2, 3]. It consists of obtaining and storing the XML differences between two versions (delta XML). The high reconstruction cost for any version different from the current one and the impossibility to validate and query the document using standard XML languages are some of their disadvantages.

Multiversion XML: [4, 5] define an indexing technique for branched versioning which they called *BT-Tree* and *BT-ElementList* respectively, however they cannot be used in XML Standard Query languages (XQuery or XPath[1]).

Temporal XML Representation: Some works has recently focused on how to represent and manage historical information in XML. In [11] a technique for managing temporal web documents is shown using an XML/XSLT infrastructure. A data model is proposed for temporal XML documents [9] where leaf data nodes can have alternative values; however how to support different structures for non-leaf nodes is not discussed. Extensions of XPath data model are exposed in [12,13] to represent and query transactional and valid time respectively by the addition of several temporal axis.

In [6] a technique for the management of changes in XML documents using a notion similar to time-interval is described which is called *context specifiers*; however how to support queries is not discussed. A temporally-grouped data model is shown in [7] that provides a way to represent the evolution of the content of a database using XML timestamps; nevertheless non-linear versioning is not supported.

However these timestamped techniques cannot be used in collaborative scenario [4] due to the fact that in this scenario users can update any version of the document generating a new version either from the current one or discard it and reuse an old version. Our technique represents the validity of each tag by means of a versionstamp, which provides us: 1) an easy support for branched versioning 2) a wide set of queries based on the relation between the included versions and 3) It runs in whatever XQuery/XSLT engine with id() support.

3 Representing Versions in XML Documents with Versionstamp

In this section, we begin by summarizing an XML data model, which is extended to a version model, and finally a versioned graph is converted to a V-XML document.

3.1 XML Data Model

An XML document [1] is a collection of elements delimited by a start-tag and an end-tag where between the tags of each element might be either its content or other elements, that is, these elements are nested and cannot overlap. An XML document is defined in [16] as:

Definition 1. An XML document D is a rooted directed graph $D=(V,E,root)$ where V is a finite set of vertices in the graph, E is a finite set of directed edges between vertices in V and $root \in V$ is the root of the directed graph.

Each vertex $v \in V$ has two properties: $type(v) \in T_V$ and $value(v) \in D(type(v))$, where T_V is a finite set of types $T_V = \{element, ID, IDREF, IDREFS, string, int, \dots\}$ and every type t in T_V has a domain of values, denoted as $D(t)$. Vertices are of type *element* or *data value*.

Edges, which have the form $e=(v_1, v_2)$ being $parent(e)=v_1$ and $child(e)=v_2$, have two properties *type* and *name* with $type(e) \in \{E, A, R\}$ that classifies E in three disjoint sets (elements, attributes and references), and $name(e)$ denotes the set of valid names for every type of edge. There is an order relation that represents the total order between edges of a particular class E, A or R and connect parent element to its children.

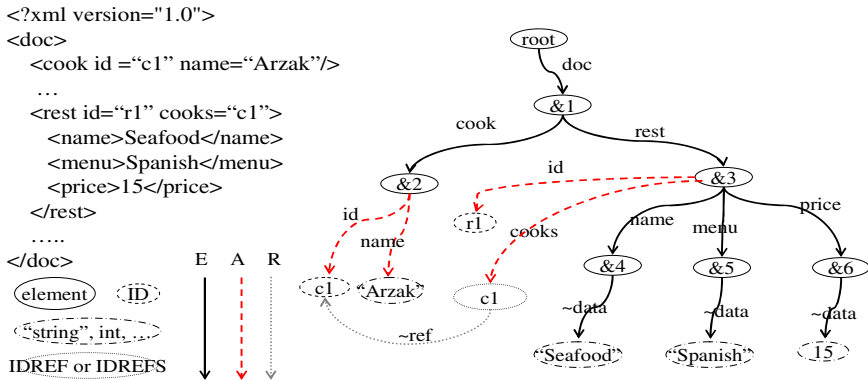
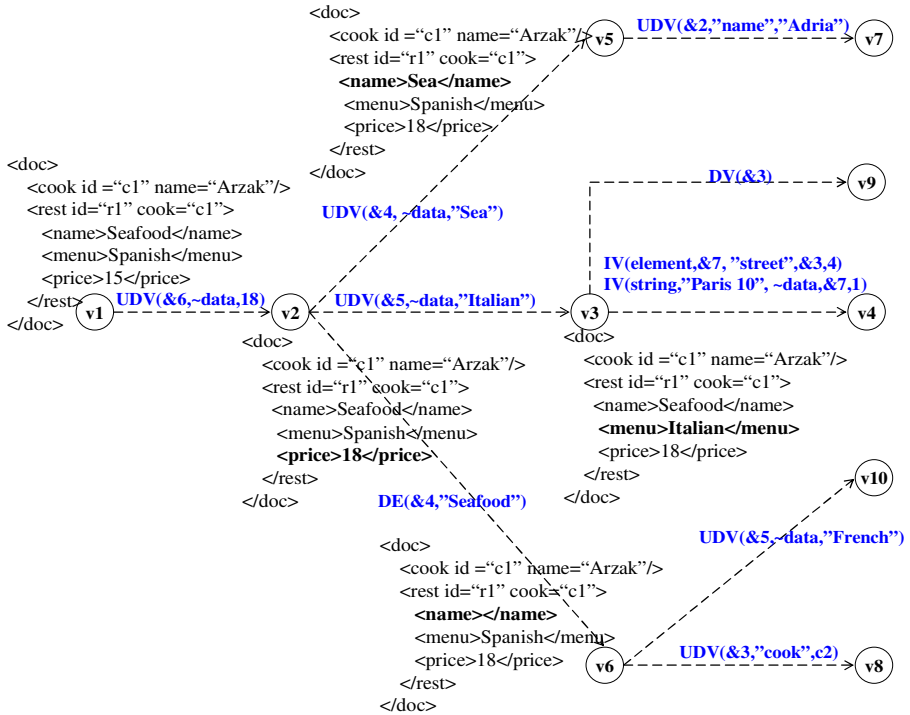


Fig. 1. An XML document and its graph

Figure 1 shows an XML document with information about cooks and restaurants of a hotel and its graphical representation in this data model. XML documents are not static: inserts, deletes or updates can modify them [14]. Beginning from the initial state of the document (version 1), the new versions are then established by applying a number of changes to the current version that we will call *version change transaction* and it must guarantee the XML specification for the new versions of the document. In table 1 we briefly describe the updates operations for this data model where move operation is represented by a delete and an insert operation.

Table 1. Primitives of changes

Update Operations	Description
(IV) InsertVertex(t, v, ne, p, o)	It adds a new vertex with the specific type t and value v and also adds a new edge with the name ne from the parent vertex p to the new vertex in the position o .
(DV) DeleteVertex(v)	It removes the vertex identified by v . All descendant vertices and edges are deleted.
(UDV) UpdateDataVertex(p, ne, nv)	It changes the value of a data vertex with parent p and name of the inedge ne to a new value nv .
(DE) DeleteEdge(p, c)	It removes the edge from p to c . All descendant vertices and edges are deleted.

**Fig. 2.** Snapshots of an XML document

When a new version is produced, the new vertices and edges are integrated with the previous versions, being necessary thereby to establish for each versioned item its validity. In the Figure 2 several change operations have been made to the XML document shown in the Figure 1, generating different versions identified by V_i , as shown inside the circle, representing different snapshots of the document and establishing thereby what is known as a *version tree* in traditional document versioning systems. Edges represent the changes made from a specific version to a specific descendant from table 1 and the order represents the ancestral relation between the defined versions. For example in Figure 2, the data of the vertex &5 has changed in versions V_3 and V_{10} and it has been deleted in V_9 .

3.2 V-XML Document Model

In this paper our objective is how to represent different versions of an XML document in a single document, which we called *V-XML Document*, using versionstamp.

Definition 2. (*V-XML Document*) A V-XML Document $D_v=(V,E,v\_document)$ is an XML document where $v\_document$ is the root of the D_v document that has only two outedges: ‘version_tree and ‘versioned_doc’.

- A *version tree* $V_t=(V',E',\&V_0)$ is an XML document where $\&V_0$ is the root of the version tree and it has as many element vertices as versions stored. A version tree is characterized: 1) We denote each vertex as $\&V_i$ 2) Each vertex has an outedge $e \in E'_A$ such that $name(e)=\text{'id'}$ and $value(child(e))=V_i \in D$ (ID) (a version identifier) 3) Each vertex has a set of outedges labeled (‘version’) that represent the ancestral relation between them. 4) There is a special vertex $\&V_{now}$ that is child of $\&V_0$ and has no children. Notice that we could consider adding more information to each edge of the version tree in order to enrich it.
- A *versioned doc* $D=(V, E_v, \&0)$ is an XML document where $\&0$ is the root and each edge of the graph has a new property VV , which we called as *V-Edge* and represents its *version validity*, that defines in which versions of the version tree the edge is valid. VV will be defined as a *version region* (definition 3).

To represent this validity we can choose either a positive representation defined by means of those versions of the version tree where the edge is valid or by a negative representation where the edge is not valid. In Figure 3 the positive and negative representation for the edge between vertices $\&5$ and ‘Spanish’ is shown $VV=\{V_1,V_2,V_5,V_6,V_7,V_8\}$ where, in both solutions, the solid circles indicate which versions are in VV . Both solutions are acceptable. The first case (positive) allows us to represent the validity by means of a tree (a connected graph) instead of having to handle a forest (several connected graphs) as it happens in the negative representation.

To define the version validity we could choose between the following solutions: 1) A *list of version identifiers* that represent which versions the edge is valid for. Thus, version validity for the edge between vertices ‘ $\&5$ ’ and ‘Spanish’ value is $VV=\{V_1,V_2,V_5,V_6,V_7,V_8\}$ 2) A *list of lineal version intervals* that consists of a set of pairs $[v1-v2]$ and represents a path in the version tree where its origin is the node with the $v1$ identifier and its end is the node with the $v2$ identifier. The version validity for the edge between vertices ‘ $\&5$ ’ and ‘Spanish’ could be $VV=\{[V_1-V_7],[V_6-V_8]\}$.

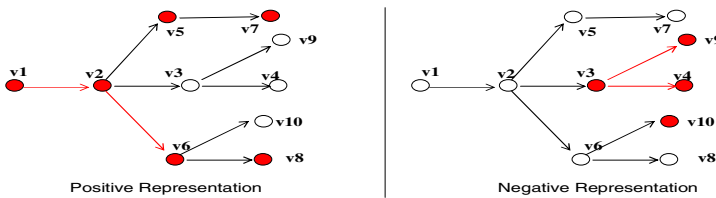


Fig. 3. Positive and negative representation for the edge between vertices $\&5$ and ‘Spanish’

Both representations present the same problem derived from the necessity to *update* the version validity of all affected edges in the graph when a new version is produced (i.e: If a new version is generated, with version identifier V_{11} , from the version

identified by V_8 , this forces us to change the validity version $[V_I-V_8]$ to $[V_I-V_{II}]$. Obviously this situation entails a high reconstruction cost for all affected edges, because, although we use a tree representation to describe the included versions, the version validity is defined by a lineal representation (with only one start and one end). For this reason it is necessary to use a non-linear representation to define the versions associated to an edge. To do this effectively it has used a mixed representation, positive and negative, that allows us to define several non-linear areas in the version tree which we call *version region* as in [5].

Before defining it, it is necessary to set the following notations. Given a version tree $V_t=(V',E',\&V_0)$ and given a ' $v \in V'_{\text{element}}$ ':

- Given a $x \in D(ID)$, if $\exists v \in V'_{\text{element}} / \exists e=(v,y) \in E'_A, (name(e)='id') \wedge (value(y)=x)$, then we denote this vertex v as $\gamma(x)$. If this vertex exists for x , it must be unique.
- $\succ(v) = \text{descendant}(v, V_t) \subset V'_{\text{element}}$ as all descendant element vertices of ' v '.
- $\prec(v) = \text{ancestor}(v, V_t) \subset V'_{\text{element}}$ as all ancestor element vertices of ' v '.
- $\succeq(v) = (\text{descendant}(v, V_t) \cup \{v\}) \subset V'_{\text{element}}$ as all descendant element vertices of v including itself.
- $\preceq(v) = (\text{ancestor}(v, V_t) \cup \{v\}) \subset V'_{\text{element}}$ as all ancestor element vertices of v including itself.
- $\succ, \prec, \succeq, \preceq$ can also work on a set of version identifiers, i.e. $\succ\{v_i \in I\} = \cup_{i \in I} \succ(v_i)$.

Definition 3. (*Version region*) Given a version tree $V_t=(V',E',\&V_0)$, a *version region* is a $[start-End]$ pair where the *start* value is a version identifier that represents the origin node of the valid area in the version tree (positive) and *End* is a set of version identifiers that indicate when each area has stopped being valid (negative) representing the different non-linear areas. They must carry out the properties:

1. $\exists \gamma(start) \in V'_{\text{element}}$ and $\forall v_i \in End, \exists \gamma(v_i) \in V'_{\text{element}}$.
2. $\forall v_i \in End, \gamma(v_i) \in \succ \gamma(start)$ and $\forall v_i \in End, \gamma(v_j) \notin \succeq \gamma(v_i), \forall i \neq j$.

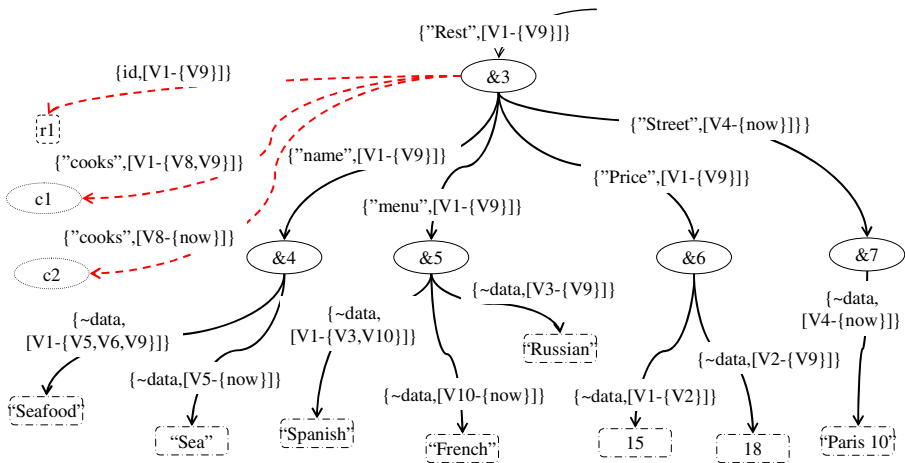
Thus a version region stands for a subset of non-linear areas in the version tree and each edge in the document is formed by a version region indicating its validity. If the set of values of *End* only contains the special value 'now', this indicates that this region is valid for all descendants of versions in the version tree whose area begins in the *start* identifier. When a new version is produced the version region of the affected edges is modified as it is shown in table 2 for every update operations shown in table 1. As in [13] the version validity of any edge e is a superset of the union of the version validity of all the edges in $\text{outedges}(\text{child}(e))$.

Notice that a version region defines a connected subset of a version tree since only an origin version identifier is allowed. If a versioned edge occurs intermittently in the version tree it is represented as a different edge with different version region. The next step is to represent the version region as a connected subset of a graph (formed by a list of non-linear version region intervals) similar to temporal element idea [8].

Table 2. Meaning of changes in the version validity property

Update Operations	Description	Edge.start	Edge.end
InsertVertex	It creates a vertex and an edge	Id. New version	Now
DeleteVertex	All affected edges and vertices	No affected	Id. New version
UpdateDataVertex	Current edge	No affected	Id. New version
DeleteEdge	All affected edges and vertices	No affected	Id. New version

The previous ‘menu’ example of Figure 2 has three V-Edges: The first one $VR_1 = [V_1 - \{V_3, V_{10}\}]$ with the ‘Spanish’ value, the second one $VR_2 = [V_{10} - \{\text{now}\}]$ with ‘French’ and the third $VR_3 = [V_3 - \{V_9\}]$ with ‘Italian’. The first region is formed by those versions defined by all descendants of the version identified by V_1 except V_3 and V_9 and all their descendants. The resulted versioned graph is shown in figure 4.

**Fig. 4.** A versioned graph document with some versions

3.3 Representing a V-XML Graph in a V-XML Document

To represent a graph document in XML the root of the graph maps to the root element of the document and for each element vertex, there is an XML element, labeled with the name of the containment incoming edge. If the element vertex has an edge to a value vertex, its value is included in the element. For each attribute or reference edge there will be an attribute where its name will be the edge name and its value will be the data value vertex or the vertex being referenced respectively.

We have represented a *version tree* by means of an XML document where each $\&V_i$ is represented as a tag and the nesting of these version tags stands for the relationship that may exist between them. The XML representation of the version tree showed in Figure 2 is shown in Figure 5.a

To represent the versioned edges in XML we have used two techniques: versionstamped attributes as in [7, 12] and versionstamped child nodes as in [13]. A version region is turn into two attributes, $v:start$ and $v:end$. The first one is defined as an IDREF datatype attribute which refers to a version identifier and the second one defined as IDREFS

datatype allows us to represent a set of version identifiers from the version tree. Depending on the type of the versioned edge the transformation will be:

1. Containment edges incoming to an element vertex, an XML element with the name of the edge is created with its corresponding v:start and v:end attributes.
2. Containment edges incoming to a data value vertex, a new child XML element 'v:data' is created with its corresponding v:start and v:end attributes, where its value is included in the element.
3. For reference edges, a new XML element 'v:isref' is created with its v:start and v:end attributes where name and value of the reference are as attribute.
4. For attribute edges, a new child XML element 'v:attrib' is created with its corresponding v:start and v:end attribute, where the name and value of the attribute are represented as an attribute value.

<pre> <?xml version="1.0"?> <v_document> <version_tree> <version xml:id="V1"> <version xml:id="V2"> <version xml:id="V3"> <version xml:id="V4"/> <version xml:id="V9"/> </version> <version xml:id="V5"> <version xml:id="V7"/> </version> <version xml:id="V6"> <version xml:id="V8"/> <version xml:id="V10"/> </version> </version> </version> </version_tree> <version xml:id="now"/> </v_document> </pre>	<pre> <versioned_doc> <rest v:start="V1" v:end="V9"> <v:attrib v:start="V1" v:end="V9" name="id" value="r1"/> <v:isref v:start="V1" v:end="V8 V9" name="cook" value="c1"/> <v:isref v:start="V8" v:end="now" name="cook" value="c2"/> <name v:start="V1" v:end="V9"> <v:data v:start="V1" v:end="V5 V6 V9">Seafood</v:data> <v:data v:start="V5" v:end="now">Sea</v:data> </name> <menu v:start="V1" v:end="V9"> <v:data v:start="V1" v:end="V3 V10">Spanish</v:data> <v:data v:start="V3" v:end="V9">Italian</v:data> <v:data v:start="V10" v:end="now">French</v:data> </menu> </rest> </doc> </versioned_doc> </v_document> </pre>

Fig. 5.a. A version tree. 5.b. A V-XML Document with some versions.

4 Version Retrieval

In this section we begin by showing how to retrieve the valid edges that belongs to a version in a versioned graph and later we make some queries in XQuery and XPath [1].

4.1 Version Retrieval in a V-XML Document Model

In this section, we describe how we can verify if a VV belongs to a given version and how to retrieve the valid tags for a given version. To retrieve the valid labels for a given version it will be necessary to analyze which versions are included in a version region and check if the requested version belongs to them. We have considered that this occurs only if 1) the given version is among the descendants in the 'start' version identifier in the version tree or even is itself and 2) the given version is not among the descendants or is itself in all version identifiers for 'end' attribute. The following proposition can be used to determine if a version region contains a given version.

Proposition 1. A version region $VR=[start-End]$ contains an identifier version $x \in \mathcal{D}(ID)$ if it satisfies that: $(\gamma(x) \in \succeq (start)) \wedge (\gamma(x) \notin \succeq (End))$

i.e: we want to find out which edge between the vertices &5 and its data value vertices is valid in V_4 . From the version tree shown in Figure 2 the second version region $VR_2=[V_3, V_9]$ fulfils the previous proposition since $\gamma(V_4) \in \succeq (V_3)$ and $\gamma(V_4) \notin \succeq (V_9)$.

- Spanish. $VR_3=[V_1-\{V_3, V_{10}\}]$, $\succeq (V_1)=\{V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8, V_9, V_{10}\}$, $\succeq \{V_3, V_{10}\} = (\succeq (V_3) \cup \succeq (V_{10})) = \{V_3, V_4, V_9, V_{10}\}$
- French. $VR_I = [V_{10}-now]$, $\succeq (V_{10})=\{V_{10}\}$, $\succeq (now)=\emptyset$
- Italian. $VR_2 = [V_3-V_9]$, $\succeq (V_3)=\{V_3, V_4, V_9\}$, $\succeq (V_9)=\{V_9\}$.

4.2 Version Retrieval in XQuery and XPath in a V-XML Document

Firstly, the ‘menu’ history will be shown and then we will make a snapshot query.

Query1: Version Projection query: Retrieve menu's history.

```
1. for $s in doc("rest.xml")//rest/menu
2.   return $s
```

Query 2: Snapshot version query: retrieve the menu tag on V_8 version

To do this effectively, we have to obtain which versions are in a version region and check if they belong to the requested version. We use the `id()` function provided by XPath [1] to obtain the versions by means of dereference the version/s in the version tree which $v:start$ and $v:End$ refer to (they are defined as IDRef and IDRefs datatypes respectively) and thereby we can easily obtain their descendants and check the constraints said before.

```
1. <data>{
2.   for $s in doc("rest_xml_id.xml")//rest/menu//v:data
3.     [ (not(f:descen_self_end(., "V8")) and (f:descen_self_start(., "V8"))) ]
4.   return $s
5. }</data>
```

We have defined two version operators as user-defined functions (figure 6) (line 1,4) that return the descendants or itself of the $v:start$ and $v:End$ attribute respectively. Using them, we can check in this query (line 2) if the given version belongs only to the $v:start$ attribute and not to the $v:End$ attribute. From the V-XML Document shown in Figure 5, it retrieves the first ‘v:data’ label (Spanish) since the V_8 version descends from V_3 and V_8 does not descend from V_9 .

```
1. declare function f:descen_self_end($p,$v) as xs:boolean{
2.   let $desc:=$p/id($p/@v:end)/descendant-or-self::version/@xml:id
3.   return ($desc=$v)
4. };
5. declare function f:descen_self_start($p,$v) as xs:boolean{
6.   let $desc:=$p/id($p/@v:start)/descendant-or-self::version/@xml:id
7.   return ($desc=$v)
8. };
```

Fig. 6. ‘descen_self_start’ and ‘descen_self_end’ user-defined functions

The same query (Query 2) in XPath is:

```
//menu/v:data[not(id(./@v:end)/descendant-or-self::version/@xml:id='V8') and(id(./@v:start)/descendant-or-self::version/@xml:id='V8')]
```

The special value ‘*now*’ in the attribute *End* indicates ‘*no changes until now*’, it means; the version region is formed by all version descendants from the *start* attribute. In order not to change the previous queries, a new node ‘*V_{now}*’ has been added in the version tree and is characterized by having neither descendants nor ancestors.

All previous queries can be improved if instead of recovering in each query the ancestors and descendants of a referenced version, they will be stored as attribute in the version tree, reducing the time of the query since it does not work out. These queries are executable in any Xquery/XPath/XSLT engine that supports the id() function (tested in Exist, Saxon, XsltProc).

5 Conclusions and Future Work

The current solutions for the management of versions for XML documents use timestamp to represent the validity of each versioned tag in the document. Due to the linear nature of time, this solution cannot support branched versioning. In this work a technique for the management of versions of XML documents has been put forward using versionstamp. When a new version is generated the ancestral relations of the versions produced are stored by using a version tree and the version validity of each versioned tag is defined based on it.

The following advantages should be highlighted: the easy management of branched versioning, its implementation is carried out with XML technology and the possibility to query the versions using XML standard query languages. The following steps are:

- To represent the version region as a connected subset of a graph
- To measure this technique in large documents and compare the results against traditional document versioning systems and XML solutions.
- To analyze, study and develop new version queries.
- To add the valid/ transaction time for each version included in the version tree.
- To implement a versioning system based on XUpdate [14].

References

1. W3C. <http://www.w3c.org>.
2. Amelie Marian, Serge Abiteboul, Gregory Cobena, and Laurent Mignet. Change-centric management of versions in an xml warehouse. The VLDB Journal. 2001.
3. Chien S., Tsotras V., Zaniolo C. XML Document Versioning. ACM Sigmod. 2001.
4. Zografoula Vagena and Mirella M. Moro and Vassilis J. Tsotras. Supporting Branched Versions on XML Documents. in RIDE 2004.
5. Betty Salzberg, Linan Jiang, David B. Lomet, Manuel Barrena, Jing Shan and Evangelos Kanoulas. A Framework for Access Methods for Versioned Data. In EDBT 2004.
6. Manolis Gergatsoulis and Yannis Stavarakas. Representing changes in XML documents using dimensions. In XSYM 2003.
7. F. Wang and C. Zaniolo. XBiT: An XML-based Bitemporal Data Model", in ER 2004.

8. S. K. Gadia. A homogeneous relational model and query languages for temporal databases. *ACM Transactions on Database Systems*. 1988.
9. Toshiyuki Amagasa, Masatoshi Yoshikawa, and Shunsuke Uemura. A data model for temporal XML documents .In *DEXA 2000*.
10. Richard T. Snodgrass. *The TSQL2 Temporal Query Language*. Kluwer, 1995.
11. Fabio Grandi and Federica Mandreoli. The valid web: An XML/XSL infrastructure for temporal management of web documents. In *ADVIS*. 2000.
12. Curtis. E. Dyreson. Observing transaction-time semantics with TTXPath. In *WISE*. 2001.
13. Shuohao Zhang and Curtis. E. Dyreson. Adding valid time to XPath. In *DNIS*. 2002.
14. Igor Tatarinov, Zachary. G. Ives, Alon. Y. Halevy, and Daniel. S. Weld. Updating XML. In *ACM Sigmod*. 2001.
15. Alberto O. Mendelzon, Flavio Rizzolo, Alejandro A. Vaisman: Indexing Temporal XML Documents. In *VLDB 2004*:
16. D. Beech, A. Malhotra, and M. Rys. A formal data model and algebra for XML. *W3C XML Query Working Group Note*, September 1999.

Preface for OIS 2006

Martin Hepp¹, Miltiadis Lytras², and V. Richard Benjamins³

¹ University of Innsbruck, Austria

² Athens University of Economics and Business, Greece

³ iSOCO, Spain

A major bottleneck in business applications of Semantic Web technology is the lack of industry-strength ontologies that go beyond academic prototypes. The design of such ontologies from scratch in a textbook-style ontology engineering process is in many cases unattractive, for it would require significant effort and because the resulting ontologies could not build on top of existing community commitment. Also, real-world problems of data and systems interoperability can only be overcome using Semantic Web technology if ontologies exist that represent the very standards currently in use in systems and databases. Of such standards, there exist at least four major categories:

1. XML schema definitions for message exchange (e.g., BMEcat, ebXML, RosettaNet, ...)
2. Non-XML message format standards (e.g., UN/EDIFACT, X12, SWIFT, VDA, SEDAS, EANCOM, CIF, OASIS UBL, ...)
3. Taxonomies and thesauri (e.g., eCl@ss, UNSPSC, RosettaNet Technical Dictionary, ...)
4. Numbering schemes and other shallow vocabularies (e.g., EAN, UPC, DUNS, ILN, ISO 639, ...)

These specifications, though mostly informal in nature, are likely the most valuable asset on the way to real business ontologies that can help solve real business interoperability problems, since they reflect some degree of community consensus and contain, readily available, a wealth of concept definitions. However, the transformation of such standards into useful ontologies is not as straightforward as it appears, because of the following reasons:

1. The specifications come in a variety of formats, e.g., XML, CSV, Microsoft Access, PDF, or plain text, and often lack a formal meta-model.
2. The specification and the documentation can be very voluminous, rendering manual translation unfeasible.
3. The same standard can be used in different ways with incompatible semantics in different contexts.

With this workshop, organized in conjunction with ER 2006, we aimed at advancing the state of the art in mechanized and semi-automated transformation of existing industrial standards into useful ontologies.

Every paper was thoroughly reviewed by three members of the Program Committee, who dedicated a lot of time and effort for their excellent reviews. Out of

ten full paper submissions, we selected five for inclusion in the official workshop proceedings, which results in an acceptance rate of 50%. Promising additional work was presented in the form of short papers, posters, and demos during the workshop.

The accepted papers address an interesting range of highly relevant research issues in this new field. Cantador, Castells, and Lara analyze the transformation of XBRL financial taxonomies into OWL ontologies, which is a very important issue for increased interoperability of financial and accounting systems.

Hakkarainen et al. discuss how the ISO standard 15926 can be represented using OWL DL. ISO 15926 aims at supporting the integration of life-cycle data for process plants including oil and gas production facilities.

Lee et al. present their work on modeling product data using ontology languages, which is at the core of future e-business applications.

Foxvog and Bussler report on their work of ontologizing the semantics of the X12 EDI specification. This work, if completed successfully, will prepare the ground for the use of semantic technology for real-world business integration challenges in many industries.

Finally, the paper by Kopecky summarizes lessons learned from mapping WSDL to RDF and derives best practices from this task for developing ontologies from existing standards.

We would like to thank everybody who contributed to this workshop. It is quite obvious that the success of this workshop is thanks to the great support of all members of the Program Committee, who dedicated a lot of time and effort in preparing their excellent reviews. The members of the PC were Chris Bizer (Free University of Berlin), Chris Bussler (Cisco), Jorge Cardoso (University of Madeira), Oscar Corcho (University of Manchester), Jos de Bruijn (DERI Innsbruck), Doug Foxvog (DERI Galway), Fausto Giunchiglia (University of Trento), Karthik Gomadam (University of Georgia), Michel Klein (Free University of Amsterdam), Paavo Kotinurmi (Helsinki University of Technology), and York Sure (University of Karlsruhe).

We feel confident that this first workshop helped establish a respective research community; at the same time, we are fully aware of the fact that most of the research work is still to be accomplished. We are still widely lacking conceptual approaches and algorithms for deriving ontologies from thesauri and inconsistent taxonomies and XML schema definitions and best practices for the axiomatization of such ontologies using human-language technologies on textual descriptions and other HL elements. The same holds for workarounds and modeling patterns in RDF-S and OWL and other popular ontology languages or the semi-automated and fully mechanized extraction of standards from human-readable specifications (e.g., PDF).

We are looking forward to working on these challenges with all colleagues in this emerging research community.

XBRL Taxonomies and OWL Ontologies for Investment Funds

Rubén Lara¹, Iván Cantador², and Pablo Castells²

¹ Tecnología, Información y Finanzas, Madrid, Spain
rlara@afi.es

² Universidad Autónoma de Madrid, Spain
{ivan.cantador, pablo.castells}@uam.es

Abstract. The analysis of investment funds information requires the availability of homogeneous information of the funds considered, which is usually generated and provisioned by different parties and in different formats. In this context, the gathering and integration of information from disparate, heterogeneous sources becomes a key task that can be considerably eased by the availability of explicit and shared information models. Furthermore, the analysis process leads to the generation of added-value information, whose consumption by other parties can also benefit from the existence of agreed information models. In this paper, we present our work on building explicit information models for investment funds in the Spanish market. An XBRL taxonomy of funds, and a translation process of XBRL taxonomies into OWL ontologies that has been applied to this taxonomy in order to obtain an OWL ontology of funds, are presented. The relative benefits of using OWL ontologies or XBRL taxonomies for the exchange and analysis of investment funds information are also discussed.

1 Introduction

Companies devote considerable efforts to the management of their information [2], requiring the integration of information from disparate and heterogeneous sources. In the financial field, a conceptually rich domain where information is complex, huge in volume and a highly valuable business product by itself [4], the exchange and integration of information for its posterior analysis is a key task for financial analysts. In particular, the analysis of investment funds requires the availability of homogeneous and consistent information, both up-to-date and historical, on the descriptive aspects (Net Asset Value -NAV-, commissions, etc.) of the funds subject of analysis.

Tecnología, Información y Finanzas (TIF), in cooperation with AFINet Global¹, acts in the Spanish market as a provider of analytical information. For providing this service, TIF continuously receives and aggregates information from the national stock markets, from firms managing investment funds, and from the national market supervisor (the CNMV)², covering all the investment funds currently commercialized in Spain with a

¹ <http://www.grupoanalistas.com>

² <http://www.cnmv.es>

10-years historical base (over 6000 investment funds at the time of writing). The information received includes descriptive aspects of a fund when it starts to be commercialized (entity commercializing the fund, investment policy, etc.), changes on any of these data, and the NAV of the fund at different points in time.

The different parties from which TIF receives information use heterogeneous information models and formats. This makes the reception, validation, and aggregation of information a difficult task, and requires ad-hoc validation procedures and a costly maintenance. As depicted in Figure 1, descriptive information about funds commercialized in the Spanish market is provided by the CNMV, and periodic information such as the NAV of a fund is provided by national stock markets (Madrid, Bilbao, Valencia and Barcelona) and by the firms managing the funds. This information is validated and transformed, leading to the creation of an aggregated and consistent information base that is ready for analysis. The analysis process leads to added-value information consumed by agents such as management firms, sellers, or investors.

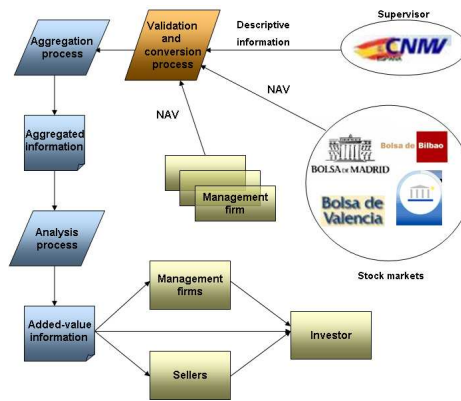


Fig. 1. Information life cycle

A gain in efficiency in the life cycle of Figure 1 can be achieved if the validation and conversion process, instead of dealing with heterogeneous information, would receive information according to a shared model so that ad-hoc processing can be avoided and maintenance needs are reduced. Furthermore, if the analytical, added-value information produced also follows an agreed model, the consumption of such information by different agents can be considerably eased.

The CNMV is considering the definition of XBRL [1] taxonomies for modelling some descriptive aspects of investment funds. However, these taxonomies would not include analytical information. Furthermore, OWL [3], the W3C recommendation for the definition of ontologies [9], has not been considered so far as an alternative for defining shared information models for investment funds. In this setting, we have worked on: a) an XBRL taxonomy that includes descriptive *and* analytical information of funds, and that can serve as a basis for possible future developments led by the CNMV or for their extension, and b) on the evaluation of OWL as an alternative to XBRL.

The paper is structured as follows: XBRL is introduced in Section 2, as well as the XBRL taxonomy created. Section 3 presents the process designed for translating XBRL taxonomies into OWL ontologies. Finally, Section 4 discusses the advantages and disadvantages of using XBRL or OWL for modelling investment funds information and provides some concluding remarks.

2 An XBRL Taxonomy of Investment Funds

In this section, we introduce XBRL and present the XBRL taxonomy developed for the modelling of descriptive and analytic aspects of investment funds.

2.1 XBRL in a Nutshell

XBRL builds on top of XML, XML Schema and XLink. **Taxonomies**, which provide the elements that will be used to describe information, and **instances**, which provide the real content of the elements defined, are the main ingredients of XBRL.

An **XBRL taxonomy** is constituted by an XML Schema and the XLink linkbases contained in or referenced by that schema, and it can be part of a set of related taxonomies called a Discoverable Taxonomy Set (DTS) [1]. The XML Schema in a taxonomy defines reporting concepts, which are given a name and a type, as XML Schema element definitions. For example, the *NAV* concept of a fund would typically have a *monetary* type. Additional constraints on how concepts can be used (e.g. instant/duration period, debit/credit balance) are documented by other XBRL attributes on the XML Schema element definitions.

Linkbases are collections of XLink extended links, which provide further information about the meaning of the concepts by expressing relationships between concepts (inter-concept relationships) and by associating concepts to their documentation. Taxonomies make use of five types of XLink linkbases, namely: definition, calculation, presentation, label, and reference linkbases.

Definition links describe relations among concepts in a taxonomy, such as general-special relations, that provide information on what an element actually. Calculation linkbases provide information on how some elements are calculated in terms of some other elements. Presentation linkbases contain relations such as parent-child that are exclusively used for presentation purposes.

Label links define labels in natural language for concepts in the taxonomy; the same concept can have labels in different languages. Reference links point to legal or other type of documentation that explains the meaning of a given taxonomy element. As it can be seen, both label and reference links do not define relations among elements but document elements in a taxonomy.

Actual facts are reported as **XBRL instances**, which organize reporting information using two main elements: XBRL items and XBRL tuples. *Items* are defined as extensions of primitive data types (String, Integer, Boolean, etc.), and they represent atomic information elements of an XBRL instance. However, some facts are dependent on each other and they must be grouped for a proper and complete understanding. For instance, in reporting information of a fund, each deposit entity name has to be properly

associated to a correct deposit entity identifier. Such sets of facts (entity name, entity identifier) are called *tuples*. Tuples may contain both items and other tuples.

In addition to the actual values of a fact, XBRL instances provide contextual information necessary for interpreting such values e.g. "NAV is 50 *today*" through the use of XBRL context elements. Furthermore, for numeric facts, XBRL instances can document measurement units e.g. "NAV is \$50" by using XBRL unit elements.

2.2 An XBRL Taxonomy of Investment Funds

The lack of explicit and shared models for exchanging information in the investment funds market and the promotion and increasing adoption of XBRL by Spanish regulators and supervisors e.g. Bank of Spain and CNMV led us to consider XBRL as a candidate language for creating an explicit information model. We started by evaluating and reviewing the information model used by TIF in order to define a revised model that could meet the needs of different agents in the market. We counted with the cooperation of Analistas Financieros Internacionales³, a leading company in the analysis of the Spanish financial market, and Gestifonsa⁴, a funds management firm.

The revised model has been described using XBRL. The reuse of existing taxonomies (IPP⁵, DGI⁶, and ES-BE-FS⁷) has been evaluated, and it has been concluded that the DGI taxonomy can be reused for the description of the entities that commercialize or manage a given fund. Figure 2 shows the DTS of the taxonomy built, where *dgi-lc-es-2005-03-10.xsd* contains the information elements of the imported DGI taxonomy in Spanish and its respective linkbases, and *dgi-lc-int-2005-03-10.xsd* contains the international elements of the DGI taxonomy. The information elements of the taxonomy created have been divided into the following groups:

- Descriptive information: models all the descriptive aspects of a fund, such as the name of the fund, the entity managing the fund, etc.
- Relevant facts information: models relevant facts about a given fund, such as changes in its investment policy.
- Periodic descriptive values: models descriptive information periodically updated, such as the NAV of the fund or the number of unit holders.
- Analytic information: models the analytic values associated to a fund, such as performance measures, the rating of the fund in its category, etc.

The reason for identifying these four distinct groups of information (being the root of each group an XBRL tuple) is that the information they contain has a different nature, the sources providing the information are different, and the periodicity with which each group of information is produced is diverse. Besides the information elements created, presentation, label, calculation and reference linkbases have been defined.

³ <http://www.afi.es>

⁴ <http://www.cajacaminos.es/>

⁵ <http://www.xbrl.org.es/informacion/ipp.html>

⁶ <http://www.xbrl.org.es/informacion/dgi.html>

⁷ http://www.xbrl.org.es/informacion/es_be_fs.html

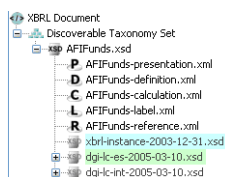


Fig. 2. DTS of investment funds

Definition links have not been used as: a) the use of links of type *requires-child* is not recommended in [10], b) there are no equivalent elements in the taxonomy, so links of type *essence-alias* have not been used, c) no use was found for links of type *general-special*, and d) there are no similar tuples for which a link of type *similar-tuples* makes sense. The latest version of the taxonomy can be found at <http://www.tifbrewery.com/tifBrewery/resources/XBRLTaxonomies.zip>.

3 Translating XBRL Taxonomies into OWL Ontologies

We have identified OWL as a potential alternative to the use of XBRL which might present some features that are of practical interest in the investment funds market. In order to evaluate the use of OWL ontologies we have developed a generic translation process of XBRL taxonomies into OWL ontologies so that existing and future taxonomies can be easily converted into OWL ontologies.

3.1 Description of the Translation Process

XBRL taxonomies provide explicit and shared information models and, thus, they are very similar to ontologies except that they do not have a formal semantics for all the aspects of the model. Similarly, XBRL instances can be seen as ontology instances and expressed as such. Therefore, we have designed a translation process of XBRL taxonomies into OWL ontologies, and of XBRL instances into OWL instances. In the following, we will restrict ourselves to the translation of taxonomies into ontologies.

An automatic translator has been implemented based on the process that will be presented. It has been tested by translating not only our funds taxonomy but also other XBRL taxonomies, including the DGI, IFRS-GP⁸, ES-BE-FS and IPP taxonomies. The latest version of the ontologies obtained can be found at <http://www.tifbrewery.com/resources/OWLOntologiesv2.zip>.

As XBRL is an XML based technology, the first step in the translation process is to parse the XML elements. Using JDOM⁹, an XML parsing module obtains the XML elements in the XBRL taxonomies, instances, and links to be translated. The translation steps described below are then applied to the obtained elements, resulting in a Jena¹⁰

⁸ <http://xbrl.iasb.org/int/fr/ifrs/gp/2005-05-15>

⁹ <http://www.jdom.org>

¹⁰ <http://jena.sourceforge.net>

model that corresponds to the OWL ontologies and instances derived from the XBRL taxonomy and instances given, which is saved to text files.

The correspondence between the upper level XBRL elements and the OWL classes generated in summarized in Table 3.1. In the following, we describe the steps for the automatic translation of XBRL elements, following this correspondence. For the sake of simplicity, we will refer to the DGI taxonomy in the explanations. The translation process for other taxonomies is analogous.

Table 1. Summary of parsed taxonomy element translations

Parsed taxonomy element	Root OWL class	Direct OWL subclasses
XML complex types	DGI_ComplexType	A subclass for each complex type
XBRL Tuples	DGI_Element	DGI_Tuple
XBRL items		DGI_Item
XLink links	DGI_Link	DGI_LabelLink DGI_PresentationLink DGI_CalculationLink
XBRL Contexts	Context (range of properties is subclass of ContextElement)	Subclasses of ContextElement: ContextEntity ContextEntityElement (Identifier) ContextPeriod ContextScenario
XBRL units	Unit (range of properties is subclass of UnitElement)	Subclass of UnitElement: UnitMeasure

1. *Declaration of a root OWL class Element* from which complex (tuples) and simple (items) information parts of the taxonomy will inherit, named DGI_Element for the DGI taxonomy. This class has associated a property *xbml_id*, corresponding to the XBRL attribute *id* common to all XBRL elements.

2. *Declaration of DGI_Tuple and DGI_Item subclasses of DGI_Element.* XBRL tuples and items correspond to OWL subclasses of DGI_Tuple and DGI_Item, respectively. The attributes of XBRL Item are translated into the OWL properties: *xbml_balance*, with possible values "credit" and "debit"; *xbml_periodType*, with possible values "instant" and "duration"; *xbml_contextRef*, whose range is the OWL class Context (step 11); and *xbml_unitRef*, whose range is the OWL class Unit (step 12).

3. *Declaration of a root OWL class DGI_ComplexType.* XML complex types are translated into classes that inherit from DGI_ComplexType, having OWL properties: *xml_name* to store the name of the complex type, *xbml_periodType*, with possible values "instant" and "duration", and *xbml_contextRef*, whose range is the Context class.

4. *Syntactic translation of XML complex types into OWL subclasses of DGI_ComplexType.* The names of the obtained subclasses are those stored in the XML attribute *name* of the complex type elements. Each subclass of DGI_ComplexType has a property whose name is the concatenation of the complex type name and the word "value", and whose type is the primitive data type associated to the complex type (xsd:string,

xsd:integer, xsd:boolean, etc.). Additionally, they contain those properties defined in the primitive XBRL data types (xbri:stringItemType, xbrli:integerItemType, xbrli:booleanItemType, etc.). For example, in the DGI taxonomy, the class *AddressFormatCodeItemType* has the property *length* with a fixed value of 2, indicating that the possible values of the data type can only have 2 characters.

5. *Syntactic translation of XBRL Items into OWL subclasses of DGI_Item*. The names of the obtained subclasses are those stored in the XML attribute “name” of the item elements. Each subclass of *DGI_Item* has a property for storing the value of the item, and whose range is the type of the XBRL item.

6. *Record XBRL Tuples as OWL subclasses of DGI_Tuple*. Initially, they are created empty, and their properties are added in step 7. The reason is that tuple properties will reference other tuples, which might be not yet created and which will have to exist in the OWL model that is being built.

7. *Syntactic translation of the XBRL tuple attributes into OWL object properties*. The attributes of the tuples are added to the subclasses of *DGI_Tuple* as OWL object properties. These properties will have as range a class associated to a complex type of step 4, a class created in step 5 or a class recorded in step 6.

8. *Declaration of a root OWL class DGI_Link*. Its instances, which correspond to the XLink links of the XBRL taxonomies, contain the properties: *xlink_from*, created for the translation of the XLink attribute *from*, stores the origin element of the link; *xlink_to*, created for the translation of the XLink attribute *to*, indicates the destination element of the link; *xlink_role*, created for the translation of the XLink attribute *role*, indicates the role assigned to the link: “label”, “calculation”, etc.

9. *Declaration of OWL subclasses of DGI_Link*. Subclasses of *DGI_Link* are built for each type of link: *DGI_LabelLink*, *DGI_PresentationLink*, *DGI_CalculationLink*, *DGI_ReferenceLink*, and *DGI_DefinitionLinks*.

10. *Syntactic translation of XBRL linkbases into instances of the corresponding subclasses of DGI_Link*. Links in XBRL linkbases are translated into OWL instances of the different subclasses of *DGI_Link* (for reasons of space, only the translation of label, presentation and calculation linkbases is presented):

- Label links are translated into OWL instances of *DGI_LabelLink*. In addition to the common link properties (*from*, *to*, *role*), label links have properties: *xbml_label*, obtained from the translation of the XBRL attribute *label* and used to store the text of the label, and *xml_lang*, obtained from the translation of the XML attribute *lang* and used to indicate the language of the label.
- Presentation links are translated into instances of *DGI_PresentationLink*. Besides common link properties, presentation links have properties: *xbml_order*, from the translation of the attribute *order* and used to store the relative position of the destination element within the presentation of the origin element, and *xbml_preferredLabel*, obtained from the translation of *preferredLabel*.
- Calculation links are translated into OWL instances of *DGI_CalculationLink*. Additionally to common link properties, calculation links have properties: *xbml_order*, obtained from the translation of the XBRL attribute *order* and used to store the relative position of the destination element value within the calculation of the origin element value, and *xbml_weight*, obtained from the translation of the XBRL attribute

weight and used to store the weight of the destination value within the calculation of the origin element value.

11. Syntactic translation of XBRL contextRef elements. In order to translate XBRL contexts, a new ontology has been created, which will be imported by the ontologies resulting from the translation of XBRL taxonomies. This ontology contains a main class Context. The Context class has the following properties: a) *xbml_id*, of type *xsd:ID*, for the translation of the XBRL attribute *id* to identify each context, b) *xbml_entity*, of type ContextEntity, defined for the translation of *entity*, c) *xbml_period*, of type ContextPeriod, defined for the translation of *period*, and d) *xbml_scenario*, of type OWL Thing, and defined for the translation of *scenario*. Other classes such as ContextPeriod (with subclasses ContextForeverPeriod, ContextInstantPeriod, and ContextStartEndPeriod), ContextEntityElement and ContextScenario are defined corresponding to the types of values that define an XBRL context.

12. Syntactic translation of XBRL unitRef elements. For the translation of units defined in an XBRL taxonomy, an independent OWL ontology has been created. This ontology will be imported by ontologies resulting from the translation process. Its main class is Unit, which has a property *xbml_unitMeasure* of type UnitMeasure and whose content is the definition of the associated unit. The UnitMeasure class, used to define the units added in a given context, does not have properties. Its subclasses distinguish the different types of units: Divide for units defined by means of a ratio (with properties *xbml_unitNumerator* and *xbml_unitDenominator*), and Measure for simple units (with property *xbml_measure*).

Besides the order of steps presented above, the hierarchy and relationships between elements within a taxonomy, and the relationships among different taxonomies, will define their translation order.

4 Discussion

The translation process presented in the previous sections helps to identify similarities and differences between XBRL and OWL, described below.

XBRL items and tuples. There is a correspondence between XBRL items and tuples and OWL classes. Items correspond to classes that only have one value (besides information such as the period, context, etc.), and tuples correspond to classes with object properties that store the constituent parts of the tuple. In this sense, items and tuples can be naturally represented by OWL classes.

XBRL contexts and units. An important feature of XBRL is the possibility of associating contexts and units to elements. This can be done in OWL by creating ontologies for contexts and units, as presented in the previous subsection, and by including appropriate object properties in OWL classes representing XBRL items and tuples.

Reference and label links. They can be represented in OWL by creating appropriate classes and instances, as done in our translation. As these links are intended for documentation, no formal semantics is associated to them. Furthermore, no application of a possible formal semantics for this type of links is envisioned.

Definition links. Definition links can be represented by creating instances of the classes introduced in the previous subsection. Special attention deserves the representation of *general-special* definition links which, even though they are currently translated into instances of definition link classes, naturally correspond to subclass relations in ontologies. However, existing taxonomies e.g. IPP, DGI, or IFRS-GP hardly make use of general-special definition links. A reason for this is that this type of links is not exploited by current XBRL tools to infer additional information, as this kind of relation does not currently have a formal semantics. We believe that the formalization of subclass relations can be of interest in practical applications, and that general-special definition links could be given formal semantics by using OWL.

Calculation links. Calculation links can be represented in the way outlined in the previous section. However, these links have a formal, mathematical semantics in XBRL, while in OWL this semantics is not supported. Therefore, we believe that for OWL ontologies to be adopted in the financial domain, where mathematical relations are highly relevant for data validation, building of mathematical support on top of OWL would be required.

Presentation links. Presentation links can be represented as described by our translation process. However, unless OWL visualization tools are adapted to take into account presentation information, they will be meaningless.

Open-World Assumption (OWA) vs Closed-World Assumption (CWA). The semantics of OWL is based on classical First-Order Logic (FOL) [8], and the OWA is made i.e. information is not assumed to be false if it cannot be proven to be true. However, in an industrial setting the CWA is widely made e.g. in relational databases. In fact, XBRL users are expected to intuitively make the CWA when, for example, querying for particular information of an investment fund. Due to his background, an average user would most likely see natural a "no" answer to the question "Is the investment fund *myFund* classified in category *myCategory*?" if, from available information, the investment fund is not classified under this category. Locally closing the world using an epistemic operator for OWL could be a solution to this problem [7]. In addition, OWL does not define constraints but restrictions, as explained in [6]. However, for validation purposes we believe that the use of constraints is required.

Summarizing, the major advantage we see from the use of OWL is its formal semantics, which can be exploited for the automatic classification of funds if general-special relations are used and represented as OWL subclass (or subsumption) relations. As implicit subsumption relations can be automatically inferred using Description Logics reasoners [11], customers or analysts can e.g. formally define the characteristics of funds they are interested in and appropriate funds will automatically and precisely be found. In particular, we are investigating the application of formal semantics to personalization in the reception of information in the investment funds market and to the automated classification of funds. For this purpose, we plan to analyze subsumption relations present in current taxonomies but not explicitly declared. However, the Open-World semantics of OWL and the use of restrictions instead of constraints can hamper the use of OWL for querying investment funds information and for validating information reported. This problem will be further investigated in the future. Additionally, we

believe extensions of OWL to incorporate and validate mathematical relations in the style of XBRL are necessary for the use of OWL ontologies in the financial domain.

Our conclusion is, thus, that extensions to OWL are required in order to fulfill all the requirements of financial information reporting, and that while its semantics can be appropriate e.g. for investment funds classification, it might be problematic for e.g. validation purposes. We believe that the XBRL community has accomplished a remarkable success in the definition of agreed, shared models; the existence of these models is actually good news for the semantic Web community, which seeks similar goals. However, XBRL can be improved in the direction of adding formal semantics to it and, thus, benefit from the work done by the semantic Web community. Similarly, the semantic Web community can identify paths for improvement and development of OWL by studying the increasing adoption of XBRL in the financial domain, proposing extensions and modifications targeting at this domain.

Future work will concentrate on evaluating alternative languages for the formal description of investment funds, especially the use of the WSML family of languages [5], which provides a basic interoperability layer and extensions in the direction of Descriptions Logics and in the direction of Logic Programming.

Acknowledgements. This work has been partially funded by the PROFIT programme of the Spanish Ministry of Industry, Tourism and Commerce (FIT-340000-2005-256).

References

1. eXtensible Business Reporting Language. Technical report, XBRL International.
2. V. Alexiev, M. Breu, J. de Bruijn, D. Fensel, R. Lara, and H. Lausen. *Information Integration with Ontologies : Experiences from an Industrial Showcase*. Wiley, 2005.
3. S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL Web Ontology Language Reference. Technical report, 2004.
4. P. Castells, B. Foncillas, and R. Lara. Semantic web technologies for economic and financial information management. In *ESWS 2004*, Heraklion, Greece, May 2004.
5. J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel. The Web Service Modeling Language WSML. Technical report, WSML, 2005.
6. J. de Bruijn, A. Polleres, R. Lara, and D. Fensel. OWL DL vs. OWL Flight: Conceptual Modeling and Reasoning for the semantic Web. In *WWW conference*, 2005.
7. F. M. Donini, M. Lenzerini, D. Nardi, W. Nutt, and A. Schaerf. An epistemic operator for Description Logics. *Artificial Intelligence*, 100(1-2):225–274, 1998.
8. M. Fitting. *First order logic and automated theorem proving*. Springer, 2nd edition, 1996.
9. T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.
10. W. Hamscher, M. Goodhand, C. Hoffman, B. Homer, J. MacDonald, G. Shuetrim, and H. Wallis. Financial reporting taxonomies architecture 1.0. Technical report, XBRL, 2006.
11. D. Nardi, F. Baader, D. Calvanese, D. L. McGuinness, and P. F. Patel-Schneider (eds.), editors. *The Description Logic Handbook*. Cambridge, January 2003.

A Semantic Transformation Approach for ISO 15926

Sari Hakkarainen¹, Lillian Hella¹, Darijus Strasunskas¹, and Stine Tuxen²

¹ Norwegian University of Science and Technology (NTNU),
NO-7491, Trondheim, Norway
{sari, hella, dstrasun}@idi.ntnu.no

² Bekk Consulting AS,
NO-0150 Oslo, Norway
stine.tuxen@bekk.no

Abstract. Traditional industries anticipate supporting cross-organizational co-operation as applied in the semantic Web and Web services environments. However, the international standards used in organisations are often developed prior to or independent from the novel technologies. We demonstrate a solution, namely to perform controlled semantic transformations enabling both technologies. A study in the semantic relationship between an international standard, ISO 15926 Part 2, and a semantic Web technology, OWL DL is reported. Three alternative semantic transformation approaches are identified where two are further elaborated. Accordingly, two Transformation Protocols are developed in order to transform ISO 15926 components to OWL. Further, two Inverse Transformation Protocols are developed. The transformations are analysed and the alternative transformation protocols are evaluated.

1 Introduction

There is a need for efficient sharing and usage of information. Large traditional industries, such as oil industry are dependent on internationally agreed standards to maintain a semantically equal understanding of shared domains in and across organizations. However, the international standards currently used are often developed prior to or independent from the novel technologies. One possible solution is to develop a language that is powerful enough to express and differentiate the technological elements as well as the underlying semantics. A further requirement is that the language must be computer and human readable. An initiative to translate the international standard ISO 15926 to the semantic Web technology OWL has been taken by the Norwegian Oil Industry¹. Here we study and demonstrate an alternative solution, namely to facilitate controlled semantic transformations for models enabling co-existence of both technologies. The semantic relationship between the ISO 15926 Part 2 and OWL DL is studied in particular. The objective is to study the semantic relationship between ISO 15926 and OWL in order to facilitate in- and cross-organizational co-operation as applied in the semantic Web and Web Service environments.

¹ Integrated Operations at OLF – Norwegian Oil Industry Association, <http://www.olf.no>

Two alternative transformation protocols are defined. Further, in order to examine possible loss of semantics during the transformation two inverse transformation protocols are defined. The transformations and the inverse transformations are analysed and the alternative transformation protocols are evaluated comparatively by their semantic performance.

The outline is as follows. In section 2, the studied representations and related work are described and the prerequisites presented. In section 3, transformation and inverse transformation protocols are defined. In section 4, the alternative protocols are analysed and the ability to preserve semantics is compared between the approaches. Finally, in section 5, conclusions and directions for future work are discussed.

2 Describing the Semantics

The diversity and multitude of resources and applications in organizations and the Internet places elaborated requirements on methods and tools for efficient generation, manipulation and compositional usage of information and services. OWL and ISO 15926 that are in the foci of this study are representing the novel models for reasoning about and for specifying information systems, respectively.

2.1 ISO 15926

The purpose of the international standard for Industrial automation and integration – Integration of life-cycle data for process plants including oil and gas production facilities (ISO 15926) is to facilitate integration of data to support the life-cycle activities and processes of process plants. It has defined syntax and graphical representation yet not formal semantics. It is built on EXPRESS (ISO 10303-11) to specify its data model yet facilitates more accurate distributed specification.

ISO 15926 Part 2 [9] specifies a data model that semi-formally defines the meaning of the life-cycle information in a single context supporting the views that process engineers, equipment engineers, operators, maintenance engineers and other specialists may have on the plant [8]. Table 1 summarizes the main components.

Table 1. Components of ISO 15926 Part 2

ISO 15926	Comment
Main components	
Entity data type	The standard consists of 201 Entity data types, the top level Entity data type is thing, with its subtypes possible_individual and abstract_object. All other entities are subtypes of these.
Attribute	Each Entity data type can have Attributes that are specified in the Attribute definition part of the description of an Entity data type.
Entity data type restriction	
Subtype	Subtype is specified before “is a” in the textual description of an Entity data type. All the Entity data types are part of a Subtype-Supertype hierarchy.
Supertype	Supertype is specified after the “is a” in the textual description of an Entity data type. All the Entity data types are part of a Subtype-Supertype hierarchy.
One-of	Entity data types can be mutually exclusive, and this is specified in the textual description of an Entity data type.

Table 1. (continued)

Transitivity	An Entity data type can be defined as transitive, referred to as Transitivity.
Attribute restrictions	
Attribute-belongs-to	An Attribute can not exist independently from an Entity data type. The Entity data type that defines an Attribute within an Attribute definition is specified through Attribute-belongs-to.
Attribute-relates-to	An Attribute can relate an Entity data type to either another Entity data type or Simple Type. The related Entity data type or Simple Type is specified through Attribute-relates-to.
Redeclared Attribute-relates-to	An Attribute-relates-to is re-declared when it relates to a different Entity data type than its Supertype Entity data type does, i.e. Redeclared Attribute-relates-to.
Simple types	
STRING	STRING is a Simple Type, defined in ISO 10303-11:1994
INTEGER	INTEGER is a Simple Type, defined in ISO 10303-11:1994
REAL	REAL is a Simple Type, defined in ISO 10303-11:1994
LOGICAL	LOGICAL is a Simple Type, defined in ISO 10303-11:1994
BOOLEAN	BOOLEAN is a Simple Type, defined in ISO 10303-11:1994
BINARY	BINARY is a Simple Type, defined in ISO 10303-11:1994

2.2 Prerequisite for the Analysis

Semantic Web technologies allow the definition of logical relationships against which reasoning can occur, something EXPRESS does not fully support. EXPRESS does not formally distinguish between attributes and relationships [12], a feature that is often required. Hence, a transformation from ISO 15926 to an ontology language is anticipated to facilitate reasoning. OWL was chosen as an ontology language for transformation being a recommendation by W3C showing its relative maturity level and suitability for Web applications. Furthermore, OWL version of ISO 15926 is recommended by [12].

2.3 Related Work

The long-term objective for the work is to establish a semantic relationship between ISO 15926 and OWL in order to facilitate in- and cross-organizational co-operation in traditional domains as applied in the novel semantic Web and Web Service environments. The semantic gap between ISO 15926 and OWL is seemingly narrow when considering the object model part of the languages.

The models in the traditional modelling standards such as ISO 15926 have been used for human interpretation with graphic representation and semi-structured documentation. The emerging semantic Web recommendations such as OWL are primarily intended for machine interpretation, enabled utilizing the high expressiveness of the syntax when modelling. Thus, the target models should contain more semantics than can be captured in a pure syntactic transformation process and further, a simple 1:1 mapping between language constructs introduce non-intended semantics to the target model. This problem has not been recognized in the current body of related work.

The Integrated Information Platform (IIP) project [10] extend and formalize ISO 15926 using OWL. OWL Full is chosen for maximum expressiveness in the resulting

representation [3]. The POSC Caesar² Intelligent Data Sheet (IDS) project defines product models for data sheets based on ISO 15926 Part 4. A resulting Reference Data Library (RDL)³ is planned to consist of approx. 100,000 EXPRESS standard classes, be defined in ISO 15926 Part 4, and mapped to Part 7 OWL format.

Further, there are efforts to formalize and ontologize other international standards. [4] for instance, have taken an approach to ontologize Electronic Data Interchange (EDI), i.e. to model meaning of the messages and create semantically enabled messages yet maintaining backward compatibility to traditional systems. [6] specifies an OWL ontology to cover the Geographic Information – Metadata (ISO 19115) standard, simply expressing each ISO 19115 as a single OWL construct. These concurrent approaches seek to define a new language in OWL and assume a 1:1 relationship between the languages not considering the problem of general modelling purpose and style.

A first step in the direction of achieving that long-term objective, namely a study of possible approaches for transformation of ISO 15926 into OWL with high preservation of human interpreted semantics is presented in the sequel.

3 Semantic Transformations

A method for transforming the basic components of ISO 15926 to their proper representation as language constructs in OWL is required. [5] identifies three distinct ways to perform the transformation, all of which have different interpretations and consequences for the preservation of semantics in the resulting data models. Only, two of them are described below, since the third requires access to domain models. In the transformation (and inverse) protocol tables below, the first columns specify the source and the second specify the target. The last columns specify transformation conditions being empty when a construct can be transformed unconditionally. Further, a source construct is NOT MAPPED if no target construct to represent the source exists. A transformation is IMPLICIT if a source construct is indirectly transformed to a target construct through transformation of other constructs. Assumptions are described in [5].

3.1 Transformation and Inverse Using Method TM1

Transformation method one (TM1) considers the *semantic interpretation* of the source language construct when defining a transformation rule to a target construct. Basically, one ISO 15926 component corresponds to exactly one OWL primitive. The assumption is that there exists a one-to-one relationship between the models. The inverse transformation (ITP) using the TM1 method is based on the OWL constructs that the transformation protocol TM1-TP resulted in, i.e. only the language constructs occurring as target in TM1-TP are assigned transformation rules in TM1-ITP. The set of possible target constructs in TM1-ITP is not restricted by the results of using TM1-TP.

² <http://www.posccaesar.com>

³ <http://www.infowebml.ws>

Table 2. Transformation protocol TM1-TP

ISO 15926	OWL	Transformation condition
Main components		
Entity data type	<i>owl:Class</i>	
Attribute	<i>owl:DatatypeProperty</i>	IF Attribute relates Entity data type to a Simple Type
	<i>owl:ObjectProperty</i>	IF an Attribute relates an Entity data type to another Entity data type AND NOT already translated
	IMPLICIT	IF already translated
Entity data type restriction		
Subtype	<i>rdfs:subClassOf</i>	
Supertype	NOT MAPPED	
One-of	<i>owl:disjointWith</i>	
Transitivity	<i>owl:TransitiveProperty</i> ^c	
Functional	<i>owl:FunctionalProperty</i> ^d	
Attribute restrictions		
Attribute-belongs-to	<i>rdfs:domain</i>	
Attribute-relates-to	<i>rdfs:range</i>	IF NOT already translated
	IMPLICIT	IF already translated
Redeclared Attribute-relates-to	<i>owl:allValuesFrom</i>	
Simple types		
STRING	<i>xsd:string</i>	
INTEGER	<i>xsd:integer</i>	
REAL	<i>xsd:float</i>	
LOGICAL	<i>xsd:boolean</i>	
BOOLEAN	<i>xsd:boolean</i>	
BINARY	<i>xsd:integer</i>	

Table 3. Inverse Transformation protocol TM1-ITP

OWL	ISO 15926	Transformation condition
<i>owl:Class</i>	Entity data type ^a	
<i>rdfs:subClassOf</i>	Subtype	
<i>owl:disjointWith</i>	One-of	
<i>owl:ObjectProperty</i>	Attribute	
<i>owl:DatatypeProperty</i>	Attribute	
<i>owl:TransitiveProperty</i>	Transitivity ^b	
<i>owl:FunctionalProperty</i>	Functional ^b	
<i>owl:allValuesFrom</i>	Redeclared Attribute	
<i>rdfs:domain</i>	Attribute-belongs-to	
<i>rdfs:range</i>	Attribute-relates-to	
<i>xsd:string</i>	STRING	IF <i>xsd:string</i> is used in conjunction with <i>owl:DatatypeProperty</i>
<i>xsd:integer</i>	INTEGER	
<i>xsd:float</i>	REAL	
<i>xsd:boolean</i>	BOOLEAN	

3.2 Transformation and Inverse Using Method TM2

Transformation method two (TM2) considers *extended semantic interpretation* of the source language construct when defining a transformation rule to a target construct. Basically, one ISO 15926 component potentially corresponds to several OWL primitives and vice versa. The assumption is that there exists a many-to-many relationship

between the models. The inverse transformation using the TM2 method is based on the OWL constructs that the transformation protocol TM2-TP resulted in. The set of possible target constructs in TM2-ITP is not restricted by the results of using TM2-TP.

Table 4. Transformation protocol TM2-TP

ISO 15926	OWL	Transformation condition
Main components		
Entity data type	<i>owl:Class^a</i>	IF NOT relationship AND NOT Subtype of relationship AND NOT EXPRESS_string AND NOT EXPRESS_integer AND NOT EXPRESS_real AND NOT EXPRESS_logical AND NOT EXPRESS_boolean AND NOT EXPRESS_binary
	<i>owl:ObjectProperty</i>	IF relationship OR Subtype of relationship
	<i>owl:DatatypeProperty</i>	IF EXPRESS_string OR EXPRESS_integer OR EXPRESS_real OR EXPRESS_logical OR EXPRESS_boolean OR EXPRESS_binary
Attribute	<i>owl:DatatypeProperty</i>	IF Attribute-relates-to a <i>Simple type</i> AND NOT Attribute-belongs-to relationship AND NOT Attribute-belongs-to a Subtype of relationship AND NOT Attribute content
	<i>owl:ObjectProperty</i>	IF Attribute-relates-to an Entity data type AND NOT Attribute-belongs-to relationship AND NOT Attribute-belongs-to a Subtype of relationship AND NOT Attribute content
	IMPLICIT	IF Attribute-relates-to relationship OR Attribute-belongs-to a Subtype of relationship or already translated OR Attribute content
Entity data type restriction		
Subtype	<i>rdfs:subClassOf</i>	IF NOT Entity data type relationship AND NOT Subtype of Entity data type relationship
	<i>owl:subPropertyOf</i>	IF Entity data type relationship OR Subtype of Entity data type relationship
Supertype	NOT MAPPED	
One-of	<i>owl:disjointWith</i>	IF NOT relationship AND NOT relationship Subtype
	NOT MAPPED	IF relationship OR Subtype of relationship
Transitivity	<i>owl:TransitiveProperty</i>	
Functional	<i>owl:FunctionalProperty</i>	
Attribute restrictions		
Attribute-belongs-to	<i>rdfs:domain</i>	IF NOT relationship AND NOT Subtype of relationship AND NOT Attribute content
	IMPLICIT	IF NOT relationship OR Subtype of relationship OR Attribute content
Attribute-relates-to	<i>rdfs:range^b</i>	IF NOT relationship AND NOT Subtype of relationship AND NOT already translated
	<i>rdfs:range^b</i>	IF (relationship OR Subtype of relationship) AND it is the second Attribute AND NOT already translated
	<i>rdfs:domain^b</i>	IF (relationship OR Subtype of relationship) AND it is the first Attribute AND NOT Attribute-relates-to a Subtype of relationship AND NOT already translated

Table 4. (continued)

ISO 15926	OWL	Transformation condition
	<i>rdfs:domain</i> ^b	IF (relationship OR <i>Subtype</i> of relationship) AND the first <i>Attribute</i> AND <i>Attribute-relates-to</i> relationship AND <i>Attribute-relates-to</i> a <i>Subtype</i> of relationship AND NOT already translated
Redeclared Attribute-relates-to	<i>owl:allValuesFrom</i>	IF NOT <i>Attribute-belongs-to</i> relationship AND NOT <i>Attribute-belongs-to</i> a <i>Subtype</i> of relationship
	<i>rdfs:range</i>	IF (<i>Attribute-belongs-to</i> relationship OR <i>Attribute-belongs-to</i> a <i>Subtype</i> of relationship) AND it is the second <i>Attribute</i>
	<i>rdfs:domain</i>	IF (<i>Attribute-belongs-to</i> relationship OR <i>Attribute-belongs-to</i> a <i>Subtype</i> of relationship) AND it is the first <i>Attribute</i>
Simple types		
STRING	<i>xsd:string</i>	
INTEGER	<i>xsd:integer</i>	IF NOT Entity data type representation_of_Gregorian_date_and_UTC_time
	<i>xsd:gYear</i>	IF Attribute year belonging to Entity data type representation_of_Gregorian_date_and_UTC_time
	<i>xsd:gMonth</i>	IF Attribute month belonging to Entity data type representation_of_Gregorian_date_and_UTC_time
	<i>xsd:gDay</i>	IF Attribute day belonging to Entity data type representation_of_Gregorian_date_and_UTC_time
	<i>xsd:time</i>	IF Attribute hour belonging to Entity data type representation_of_Gregorian_date_and_UTC_time AND Attribute second NOT already translated AND Attribute minute NOT already translated THEN Attributes second, minute and hour are translated
	<i>xsd:time</i>	IF Attribute minute belonging to Entity data type representation_of_Gregorian_date_and_UTC_time AND Attribute second NOT already translated AND Attribute hour NOT already translated THEN Attributes second, minute and hour are translated
REAL	<i>xsd:float</i>	IF NOT Entity data type representation_of_Gregorian_date_and_UTC_time
	<i>xsd:time</i>	IF Attribute second belonging to Entity data type representation_of_Gregorian_date_and_UTC_time AND Attribute minute has not been translated AND Attribute hour has not been translated THEN Attribute second, minute and hour are translated
LOGICAL	<i>xsd:boolean</i>	
BOOLEAN	<i>xsd:boolean</i>	
BINARY	<i>xsd:integer</i>	

Table 5. Inverse Transformation protocol TM2-ITP

OWL	ISO 15926	Transformation condition
<i>owl:Class</i>	Entity data type ^a	
<i>owl:ObjectProperty</i>	Entity data type	IF relationship OR <i>subPropertyOf</i> relationship
	Attribute	IF NOT relationship AND NOT <i>subPropertyOf</i> relationship

Table 5. (continued)

OWL	ISO 15926	Transformation condition
<i>owl:DatatypeProperty</i>	Entity data type	IF EXPRESS_string OR EXPRESS_integer OR EXPRESS_real OR EXPRESS_logical OR EXPRESS_boolean OR EXPRESS_binary
	Attribute	IF NOT EXPRESS_string AND NOT EXPRESS_integer AND NOT EXPRESS_real AND NOT EXPRESS_logical AND NOT EXPRESS_boolean AND NOT EXPRESS_binary
<i>rdfs:subClassOf</i>	Subtype	
<i>owl:subPropertyOf</i>	Subtype	
<i>owl:disjointWith</i>	One-of	
<i>owl:TransitiveProperty</i>	Transitivity ^b	
<i>owl:FunctionalProperty</i>	Functional ^b	
<i>owl:allValuesFrom</i>	Redeclared Attribute	
<i>rdfs:domain</i>	Attribute-belongs-to	IF NOT used in conjunction with <i>owl:ObjectProperty</i> relationship AND NOT used in conjunction with <i>subPropertyOf owl:ObjectProperty</i> relationship
	Attribute-relates-to	IF used with <i>owl:ObjectProperty</i> relationship OR used in conjunction with a <i>subPropertyOf owl:ObjectProperty</i>
<i>rdfs:range</i>	Attribute-relates-to	
<i>xsd:string</i>	STRING	IF <i>xsd:string</i> is used in conjunction with <i>owl:DatatypeProperty</i>
<i>xsd:integer</i>	INTEGER	
<i>xsd:float</i>	REAL	
<i>xsd:boolean</i>	BOOLEAN	
<i>xsd:gYear/gMonth/gDay</i>	INTEGER	
<i>xsd:time</i>	INTEGER	IF hour specified by <i>hh</i> OR minute specified by <i>mm</i>
	REAL	IF second specified by <i>s.sss</i>

4 Semantic Analysis of Transformation Methods

4.1 Preserved and Altered Semantics

The detected deviations in the transformations are described below.

IMPLICITLY translated components. Using TM1, both Attribute and Attribute-relates-to may be implicitly defined. Some Entity data types have Attributes with identical names. If the Attribute is already translated we claim that the other Attributes with identical names are implicitly translated through the first Attribute. Further, Attribute-relates-to and Attribute-belongs-to are used in conjunction with the Attribute. If the Attribute is implicitly defined then Attribute-relates-to is also implicitly defined.

In TM2, Both Attribute and Attribute-relates-to may be implicitly defined. If it is Attribute content defined inside Entity data type EXPRESS_string, EXPRESS_integer, EXPRESS_real, EXPRESS_logical, EXPRESS_boolean or EXPRESS_binary it is implicitly defined since these Entity data types are translated to *owl:DatatypeProperties* that relates to the same Simple Type as Attribute content. Even though the semantics of Attribute content are lost, it is of no relevance since it is only used as an intermediary between the Entity data type and the associated Simple Type, while TM2-TP uses the Entity

data type itself as the intermediary. This rule does only apply when the Attribute belongs to an Entity data type relationship or a Subtype of relationship.

When an Entity data type in TM2 is translated to *owl:Class*, and its Attributes are translated to *owl:ObjectProperties*, the *owl:Properties* are defined independent of the *owl:Class*. Thus *rdfs:domain* is used to specify the class that the ObjectProperty belongs to. However, when an Entity data type is translated to *owl:ObjectProperty*, its Attributes are not translated, but the value they relate to is specified by *rdfs:domain* and *rdfs:range*. Since *rdfs:domain* and *rdfs:range* are used inside *owl:ObjectProperty* it is not necessary to explicitly define the property they belong to.

NOT MAPPED components. ISO 15926 component Supertype is not mapped in the transformations. Subtype is translated to *rdfs:subClassOf* thus automated reasoning is still possible and the semantics is preserved even though Supertype is not mapped.

4.2 Comparative Analysis of TM1 and TM2

The major difference between TM1-TP and TM2-TP is that TM1-TP only considers the semantics interpretation of the ISO 15926 components, while TM2-TP also considers the semantics of each instance of the components.

Relationship. Entity data type relationship and Subtypes of relationship are translated to *owl:Class* according to TM1-TP and *owl:ObjectProperty* according to TM2-TP. When the inverse translations are performed the names of the Attribute are preserved by using TM1-ITP. By using TM2-ITP the Attributes are created, but a new name has to be defined as the original names are not used.

Attribute. ISO 15926 Entity data type EXPRESS_string, EXPRESS_integer, EXPRESS_real, EXPRESS_logical, EXPRESS_boolean and EXPRESS_binary are translated to *owl:Class* according to TM1-TP while they are translated to *owl:DatatypeProperty* according to TM2-TP. The Attribute content is defined within each Entity data type. However, Attribute-relates_to associated with each Attribute relates to different Simple Type components. Thus when the Attributes are translated using TM1-TP it is not possible to take advantage of the OWL feature that allows multiple domains to be defined for an Attribute.

TM2-TP translates the Entity data types to *owl:DatatypeProperty*. Thus the Attribute content is lost in the translation. However, the loss of semantics is of no relevance as the Attribute is an intermediary between the Entity data type and the associated Simple Type, while TM2-TP use the Entity data type itself as the intermediary.

The Simple Types. Both TM1-TP and TM2-TP define six Simple Types. While TM1-TP translates these components into four OWL language constructs, they are translated into eight language constructs using TM2-TP. The difference is the translation of the Simple Types used in conjunction with Attribute-relates-to of the Attributes defined within Entity data type representation_of_Gregorian_date_and_UTC_time. Compared to TM1-TP, TM2-TP in a larger extent takes advantage of the language constructs in OWL. However, TM1-TP is preferred if it is necessary to keep hour, minute and second as separate values.

5 Concluding Remarks

Two views for transforming ISO 15926 into OWL are analysed. The first alternative considers the semantic interpretation of the ISO 15926 components. The second

alternative considers extended semantic interpretation, i.e., it also considers the semantics of each instance of the components. The alternative views were tested through definition of alternative transformation protocols with corresponding inverse transformation protocols and analysed according to their ability to preserve semantics.

The analysis shows that Transformation Method one (TM1) results in a seemingly direct representation of ISO 15926 in OWL, and enables full specifications, whereas Transformation Method two (TM2) takes more advantage of the language constructs in OWL. TM1 is most appropriate if the goal is to achieve a representation of ISO 15926 in OWL where the quantity of *Entity data types* is equal to the number of the corresponding OWL language construct, i.e. where a one-to-one mapping is desired. TM2 is most appropriate if the transformation is performed in order to take advantage of the reasoning facilities provided by OWL, thus adding functionality not natively present in ISO 15926.

Transformation Method three (TM3) should be tried out with both semantic and extended semantic transformation method in the future. TM3 is based on the *contextual semantic interpretation* on existing domain models. A transformation to target language is based on mapping the components of the domain model in source language to semantically equal language constructs in target language. In this case not solely analytic, but empirical observations could be made in conjunction with appropriate statistical analysis.

References

1. Antoniou, G., van Harmelen, F. Web Ontology Language: OWL. Staab, S. and Studer, R. (Eds.), Handbook on Ontologies in Information Systems, Springer Verlag (2003)
2. Bechofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A. OWL Web Ontology Language Reference. W3C Rec.
3. Christiansen, T., Jensen, M., Valen-Sendstad, M. Defining ISO 15926-4 Reference Data Library classes in OWL. The Norwegian Oil Industry Association
4. Foxvog, D., Bussler, C. Ontologizing EDI: First Steps and Initial Experience. In proceedings of International Workshop on Data Engineering Issues in E-Commerce, 2005, pp. 49-58
5. Hella, L., Tuxen, S.M. Semantic Web Technology and ISO 15926 – A Semantic Transformation Approach. MSc thesis, IDI, NTNU, Trondheim, Norway, 2004
6. Islam, A.S. et al. Ontology for Geographic Information - Metadata (ISO 19115:2003). URL:
7. McGuinness, D.L., van Harmelen, F. OWL Web Ontology Language Overview. W3C Rec.
8. Price, D., Sandsmark, N. Industrial automation systems and integration - Integration of life-cycle data for process plants including oil and gas production facilities - Part 1: Overview and fundamental principles. ISO/Final Draft International Standard FDIS 15926-1:2003(E)
9. Price, D., Sullivan, J. Industrial automation systems and integration - Integration of life-cycle data for process plants including oil and gas production facilities - Part 2: Data model. ISO/Final Draft International Standard FDIS 15926-2:2003(E)
10. Sandsmark, N., Mehta, S. Integrated Information Platform for Reservoir and Subsea Production Systems. In Proceedings of the 13th Product Data Technology Europe Symposium (PDT 2004), October, Stockholm 2004
11. Smith, M.K., Welty, C., McGuinness, D.L. OWL Web Ontology Language Guide. W3C Recommendation.
12. Wix, J., Liebich, T. Final Draft CWA3 Proposal European eConstruction Meta-Schema (EeM), Task T2 of the eEurope Pilot Project "SPICE" towards the CEN/ISSS Workshop on eConstruction.

Modeling Considerations for Product Ontology*

Hyunja Lee¹, Junho Shim¹, Suekyung Lee², and Sang-goo Lee²

¹ Dept of Computer Science
Sookmyung Women's University
Seoul 140-742, Korea

{hyunjalee, jshim}@sookmyung.ac.kr

² School of Computer Science & Engineering
Seoul National University
Seoul 151-742, Korea

{sklee, sglee}@europa.snu.ac.kr

Abstract. An issue in utilizing ontology in product information domain is how to make it operational. This requires considering the features to be included in a pragmatic product information database which should contain the key concepts and the relationships in the domain. Ontological modeling of product information is a starting as well as a most important point for practical deployment. It often involves problems such as the choice and representation of the proper semantic relationships, handling of product properties, and product descriptions associated with their units of measure. In this paper, we introduce an approach to those problems. We recently engaged in a project to build an operational product ontology system. The directions to problems, which are presented in this paper, were obtained from this project, and may serve a role as a reference model for similar projects in future.

1 Introduction

Product information is an essential component in e-commerce. It contains information such as pricing, features, and terms about products traded between partners. A base of clearly defined product information is a necessary foundation for collaborative business processes. Furthermore, semantically enriched product information may enhance the quality and effectiveness of business transactions. The utility and potential benefits of an ontology to product information have been suggested [10,4,12].

However, there are still gaps in applying the ontology in a real-life commercial environment. The problems are somewhat difficult especially in such domains as product information management, which requires mission critical applications operational. We recently participated in a project to build an operational product ontology system for a government procurement service [11,8]. The major challenges for building a practical ontology may necessitate moderation between the precision and richness of domain knowledge and its representation and processing complicatedness, and also achieving a balance between built-in functionality and domain-specific customization [14,7]. Then, a most important step for deploying a product ontology should start

* This work was supported by the Ministry of Information & Communications, Korea, under the Information Technology Research Center (ITRC) Support Program.

from the construction of an operational ontological model for product information. It requires domain expertise and experience, as for any other domain, and involves modeling choice problems such as the choice and representation of the proper semantic relationships, handling of product properties, and product descriptions associated with their units of measure. In this paper, we present our modeling approach for building operational product ontology and also provide informative directions with which to handle those modeling problems.

The rest of this paper is structured as follows. Section 2 describes the related work. In Section 3 we present each of the modeling problems and also provide our approaches to handle them. And finally, conclusions are drawn in Section 4.

2 Related Work

Product information is one of the most adequate domains within e-commerce where ontology can play a vital role. In [3] they list the difficult aspects of building, maintaining, and integrating product information, and propose that an ontological approach may be the answer. In [12] they propose to use cross-industry standard classifications as the upper ontology and industry specific classifications as the lower ontology, thus achieving the generality and specificity.

The importance of attributes in product information management is well-introduced in [4,9]. In [4], the quality of product classification standards is evaluated by a number of factors including the quality of their attribute lists. In [9], it is pointed out that a classification hierarchy is a representation of just one of many views over the set of products, and that a product's identity and property are not decided only by how the product is classified.

For any semantic modeling to be suitable for its application domain, it is crucial to investigate what semantic concepts and relationships are desirable for the domain and to capture them in the model. Concepts presented in this paper were developed through a real project that we participated in [11], and their formal representation is presented using description logics [13]. [10] is a recent work to point out that building an OWL knowledgebase is not pragmatically adequate for a large-scale ontology. Instead they propose a relational database approach in which reasoning is supported by storing facts in tables and representing rules in triggers. Their work is different from ours in that they focus on providing a programmatic framework including APIs and query languages rather than on modeling related issues.

E-procurement may benefit from well defined product information. The entire process of registering a product, searching for a product, registering a purchase request, and ordering requires accurate product information. We have built an ontology system helping data integration for business processes for the Public Procurement Services of Korea, which is responsible for procurement for government and public agencies [11,8]. The system consists of a product ontology database and two subsystems; the ontology construction & maintenance system and the ontology search system. The ontology database is modeled using a meta-modeling approach, and includes the key semantic concepts including products, classification schemes, attribute requirement for each product, units of measure and their relationships among one other.

3 Modeling Choices: Practical Encounters

While modeling this product ontology, we frequently encountered problems such as 1) the choice of proper semantic relationship 2) handling of product attributes, and 3) taxonomy of Attributes and UOMs. In the following sections, we introduce each problem and present modeling guidelines for each of them.

3.1 Choice of Semantic Relationship

This problem means to assign a proper semantic relationship between concepts. Examples include whether A should be modeled as a class inclusion of B or member of B, or whether A should be modeled as a part of B, and if so which part-whole relationship should be chosen to convey the proper semantics. For example, HDD and CPU are parts of a computer, and beef-stew has beef, garlic, and onion. The different semantic interpretations of this part-whole relationship have been recognized. In the previous example, the former part-whole means that computer has a HDD and a CPU as its *components*, while the later means that the beef-stew has beef, garlic, and onion as its *substances*.

We found *component*, *substance*, *member* are most frequently used in the field [11]. Table 1 summarizes the types of part-whole relationships, their examples, and discretion criteria based on [15], where each criterion means the following:

- Functional: if the part plays a functional role with respect to the whole.
- Separable: if the part can be physically disconnected from the whole.
- Homogeneous: if the parts are similar to each other and to the whole.

Table 1. Types of Part-Whole relationships

Types	Examples in the domain	Classification criteria		
		functional	separable	homogeneous
component	HDD vs. computer	Yes	Yes	No
substance	beef (garlic and onion) vs. beef-stew	No	No	No
member	umbrella (hat or belt) vs. clothing accessory	No	Yes	No

Along with different semantics associated with each part-whole relationship, the properties of part-whole relationships may also be different from each other. Properties to be investigated may include the following, where *pw* stands for the part-whole:

- Transitive: $x \text{ pw } y \ \& \ y \text{ pw } z \Rightarrow x \text{ pw } z$
- Inverse: $x \text{ pw}_1 y \Rightarrow y \text{ pw}_2 x$, where $((\text{pw}_1^{-1})^{-1}) = \text{pw}_2$
- Asymmetric: $x \text{ pw } y \Rightarrow \neg (y \text{ pw } x)$

For example, consider the former case (*component*). Let us say that a computer has a HDD as its component, and a HDD has a motor as its component. A query is to find all parts of a computer. Whether the motor should be treated as a part of the computer may be controversial, in that user's intention of querying the "part" relationship is ambiguous, i.e., Does the part in the query represent a direct part only, or any part

contained in another part? In addition, should it turn out to process the indirect and transitive component relationships, and then we should employ transitive property to the relationships.

The *member* relationship represents the concept that a set of members is considered an object in its own right. It can be easily confused with the class inclusion relationship because both involve membership of individuals in a larger set. The member relationships are determined on the basis of extrinsic characteristics of the individual members. Such criteria may include spatial proximity or social connection, or naïve intention to create such a collection [13]. In contrast, the class inclusion is determined by similarity to other members based on an intrinsic characteristic.

Table 2. Class inclusions vs. member inclusions

UNSPSC	Title	UNSPSC	Title
43211500	Computers	53102500	Clothing accessories
43211501	Computer servers	53102501	Belts or suspenders
43211502	High end computer servers	53102502	Ties or scarves or mufflers
43211503	Notebook computers	53102503	Hats
43211504	Personal digital assistant	53102504	Gloves or mittens
43211505	Point of sale POS terminal	53102505	Umbrellas
43211506	Thin client computers	53102506	Sweat bands
43211507	Workstations	53102507	Clothing hangers
43211508	Personal computers	53102508	Armbands

Consider the UNSPSC 432115 computer class-commodity inclusion hierarchy (Table 2, left). In this case, the relationship is the class inclusion in that member objects, such as servers, desktops, or notebooks, contain intrinsic similarity to one another. In contrast, under the UNSPSC 531025 clothing accessory class-commodity (Table 2, right), it is more natural to regard the relationships as members, since there are little shared intrinsic characteristics between commodities, and it just seems that a variety of commodities are collected in a named class.

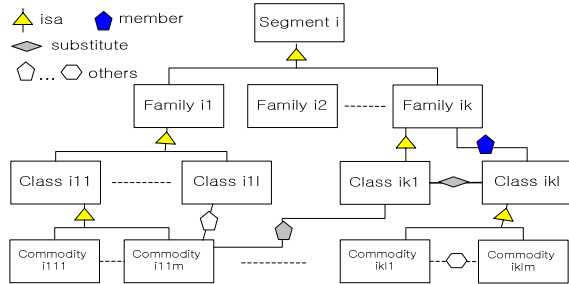


Fig. 1. Extended classification model with various semantic relationships

The major goal of a product classification model is to produce a standard product code system, for example numbered *segment*, *family*, *class*, and *commodity* codes in UNSPSC. A major drawback of such a code-based model is that the model is not able to explicitly adopt various semantic relationships but only to include the inclusion relationship. We claim that in order to provide richer and more precise semantic product information, the model should be extended to explicitly adopt various semantic relationships like ours. Figure 1 illustrates this idea.

3.2. Modeling Product Attributes

While modeling a property of product class, it is not clear at times whether a property should be modeled as an attribute or as another product and a relationship associated with the product.

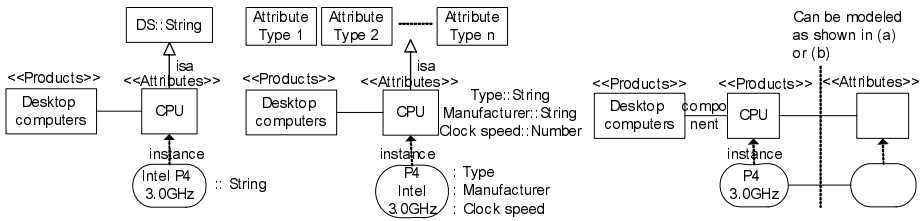


Fig. 2. (a) Atomic attribute (b) General attribute (c) Product and component relationship

For example, consider the CPU property for desktop computers. One way is to model the CPU as an attribute class. The simplest way to achieve this is to have or confine CPU information in a basic data type such as a string. For example, a CPU can be *owl:DatatypeProperty* with a restriction of *rdfs:range::STRING* in OWL (Figure 2-a). Then just having a string value may implicate less semantics than a general attribute would. In Figure 2-b CPU is an attribute class which contains manufacturer and clock speed information as separate properties. By representing CPU as attribute class, it may be much easier to support such a query as “find desktop computers with CPUs which are manufactured by Intel and faster than 3.0GHz”.

An alternative is to create a product class called CPU and set up component relationship between Desktop computers and CPU (Figure 2-c). Or instead of having a component relationship, we may also have a relationship, let us call it *CPU-component*, between Desktop computers and CPU and declare it as a role inclusion of a component, i.e., *CPU-component* $\sqsubseteq$ *component* in logic representation. This may be more complicated than the others but be beneficial in following situations:

- To consider a CPU as a product by itself rather than a descriptive attribute of desktop computers. Consider the case in which a CPU may not only be an inseparable part of a desktop computer but a sellable part as a separate product.
- To capture the component semantic relationship between a desktop computer and a CPU. For example, we might further break down the part information of a CPU, i.e., a CPU is composed of chipset, cache, and cooler. This way is beneficial for exploiting the component relationship. For example, consider the query “find any product which contains Intel 955X chipset”. Then we may find not only CPU products with the

chipset and all desktop products which contain those CPUs, but also find other products such as high-speed network switches that contain those CPUs as a component.

Common attributes are ones which appear in all the products belonging to the same product class or category. For example, under a UNSPSC classification system we may declare common attributes for each level of *segment*, *family*, *class*, and *commodity* categories. Here an industry standard such as GDAS (Global Data Alignment System) [1] from EAN/UCC or “attribute list” in eCI@ss can be used as a source standard for all the different product attributes, in order to achieve interoperability between catalogues. The attribute taxonomy of GDAS is as follows in Table 3.

Table 3. A taxonomy of common attributes

Attributes Category	Examples	Attributes Category	Examples
Product Identification	Global Trade Item Number, Replaced Product Global Trade Item Number	Product Hierarchies	GTIN of Next Lower Level Configuration
Product Classification	International or Local Product Classification/ Code/ Agency	Handling information	Product Handling information
Parties	GLN of Supplier/ Manufacturer, Supplier or Manufacturer's Name/Homepage	Commercial Data	Sale/ Catalog Price, Currency of Price, Tax Rate, Type of Tax, Warranty
Dates	Start/ End Availability Date	Packaging	Product Packaging Type
Product Descriptions	Product Name, Name Language, Brand Name, Certified Specification	Packing Configuration	Number of Layers per Pallet/ Number of Trade Items per Pallet
Product Measures	Product Depth/ Height/ Width/ Diameter/ Gross Weight/ Volume	Order & Delivery	Order Quantity Minimum, Lead Time

It may be a simple task to get different catalogs interoperable if products are modeled only upon common attributes (but not practicably achievable whatsoever). A product may have unique features to make it distinguishable from other products, albeit those products belong to the same product class, and those unique features may not be enumerated in the description of a common attribute. Attributes, which particularly reside in the ‘Product Descriptions’ and ‘Product Measures’ categories of GDAS, such as ‘type of material’ or ‘length of sleeve’, can be good individual attributes representing a peculiar property of an individual product. Then we need to express those features of individual products using additional attributes, which are called *individual attributes*.

Some researchers categorized the scope of attributes instead of distinguishing common and individual attributes, because this determines where and how respective statements shall be stored and how frequently the statements need to be updated. Such attributes can be (1) valid for every make and model that falls into a specific product category, (2) vendor- or model-specific, or (3) characteristics of single product instances. For example, model-specific attributes will be assigned in a vendor-specific product model ontology, while attributes of product categories will rather be part of the TBox part of a domain ontology. Additionally, some product attributes may be dynamic, others may be static.

For attributes where an arbitrary string is not sufficient to capture the value in a semantically unambiguous way, most classification systems maintain a list of supported values in a separate collection. For example, GPC(Global Product Classification) published standards contain the attribute "Type of Athletic Footwear" (ID 20001935) for "Athletic Footwear – Specialist" category. The set of supported values contains four entries that are recommended for the use with this attribute, "BOOT" (ID 30007405), "SHOE" (ID 30007406), "UNCLASSIFIED" (ID 30002515) and "UNIDENTIFIED" (ID 30002518). The mapping between recommended values and such attributes is usually kept in a separate relation.

3.3 Units of Measure

As the attributes are categorized, so can UOM. For example, in KOCIS we have over six thousand different units for particular quantities, which contain both physical and countable quantities. A *physical quantity* is a measure of some quantifiable aspect of the modeled world characterized by a physical dimension such as length, mass, or time. Quantities can be expressed in various units of measure e.g. meter, inch, kilogram, pound, etc. We define the relations between quantities and units of measure so that different units for same physical dimension may be transformed into one another. A *countable quantity* is a measure for the cardinality of a set of entities defined by the common characteristic of its members [2]. Units using countable quantities include bundle, dozen, each, or can.

To categorize UOMs for physical quantities, we follow the International System of Units(SI), which has four different types: (1) *base quantities* (2) *simple derived quantities* (3) *derived quantities with special unit symbols* and (4) *derived quantities whose symbols include derived units with special names and symbols* [16]. Units for eight *base quantities* are the foundation of units for derived quantities. Each base quantity is mutually independent of each other and has a corresponding base unit (Table 4). For example, mass base quantity has *kg*(kilogram) for a *base unit* and also contains other types of mass units such as *g*(gram), *karat* (mass for gems, 1 karat= 20mg), *pound*, and *ton*. *Simple derived quantities* have units which can be derived from the eight base units and a system-defined equation expression rule. Examples include *area* for m^2 (square meter), *velocity* for m/s(meter per second), and *current density* for A/m^2 (ampere per square meter). For the convenience, quantities which have units derived from the base units may be given *special names and symbols*, such as *frequency*($Hz=s^{-1}$), *force*($N=kgm/s^2$) and *energy*($J=Nm=kgm^2/s^2$). Then those *units whose symbols contain derived units with special symbols* includes *moment of force*(Nm, Newton meter) and *angular velocity*(rad/s, radian per second).

UOMs may be *compatible* each other if they belong to the same category of quantity and there exists a conversion relationship which is bi-directional and can be expressed in a conversion equation. For example, in length base quantity, we have *cm*, *inch*, and *cha* (a traditional Korean length unit) which are compatible; i.e., $inch=2.54cm$ and $cha=30.3cm$. Each equation is represented as a convertedTo relationship in KOCIS, and we maintain about nine-hundred conversion equations in a relational table. In addition, we maintain synonym relationships between UOMs that are in fact equal but frequently appear in different formats. For example, *KM*, *kilo meter*, *kilometer*, and *k meters* are synonyms for each other. These relationships are currently maintained in a relation in our case.

UOMs in the same quantity may also use different magnifying prefixes. We currently use 10 selected decimal prefixes for general UOMs and K, M, G and T binary prefixes only for computer related UOMs (Table 5).

Table 4. Eight base quantities and their base units

Quantity	Name	Base Unit
length	meter	m
mass	kilogram	Kg
time	second	s
electric current	ampere	A
thermodynamic temperature	kelvin	K
amount of substance	Mole	mol
luminous intensity	candela	cd
computer data	Byte	byte

Table 5. Magnifying prefixes

Factor	Name	Symbol	Factor	Name	Symbol
10^{12} (2^{40})	tera	T	10^{-2}	centi	c
10^9 (2^{30})	giga	G	10^{-3}	milli	m
10^6 (2^{20})	mega	M	10^{-6}	micro	μ
10^3 (2^{10})	kilo	K	10^{-9}	nano	n
10^0			10^{-12}	pico	p

To provide compatibility and conversion relationships between UOMs, we model UOM to contain *StdUnit*, *BaseType*, *MultipleBound* and *SubmultipleBound* properties (Figure 3).

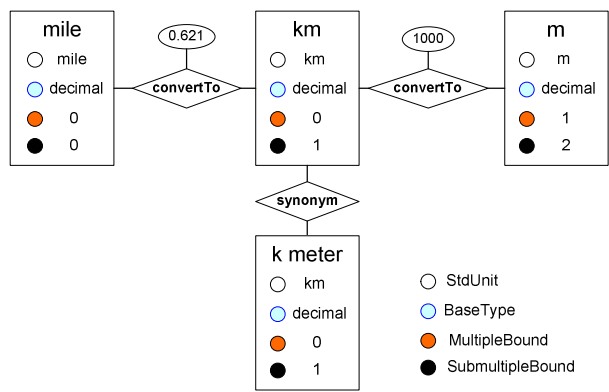


Fig. 3. Conversion relationships between compatible UOM

StdUnit denotes a standard symbol, and the search system uses the corresponding standard unit instead of original one. For example, the system utilizes *Km* when it meets the unit *k meter* since *k meter* has value *Km* for its *StdUnit* property.

BaseType, *MultipleBound* and *SubmultipleBound* properties are related to conversion among units which use different magnifying prefixes. *BaseType* denotes the multiple type of SI prefixes, decimal or binary. For example, *byte* uses a binary prefix but *meter* uses a decimal prefix. *MultipleBound* and *SubmultipleBound* denote the upper bounds of multiplying and submultiplying the prefix measure, respectively. For example, *Km* has 0 and 1 for its *MultipleBound* and *SubmultipleBound*. Then *Km* may not be magnified to a larger prefix such as *Mm* or *Gm*, since *MultipleBound* is 0; while *Km* may be submultiplied up to *m*, since *m* is the 1st submultiple of *Km*.

In that some combinations of prefixes with quantities are rarely used, we have multiple or submultiple bounds to achieve a limited-degree of conversions for a given quantity and unit. For example in Figure 3, let us say that a given quantity and prefix is 2 *Km*. Then it may be converted into 2,000m or 1.242 miles, while not to 200,000cm. This might be arguable in principle to be legal, but we employ this conversion model from an observation that such a combination of quantity and prefix (200,000cm) may be rarely found in a scientific or engineering context [16,6]. Nevertheless we are investigating this issue in a real product information domain at KOCIS.

Though it is yet implemented but worth note that a proper conversion equation for two derived units can also be derived from the conversion equations for their base units. For example, let us say that we have kcal/m^2 as a valid derived unit, and have conversion equations that $\text{kcal} = 1,000 \text{ cal}$ and $\text{m} = 100 \text{ cm}$. Then later we add a new derived unit cal/cm^2 into the database but no explicit conversion equations between kcal/m^2 and cal/cm^2 . Then we may calculate the proper equation $\text{kcal/m}^2 = 0.1 \text{ cal/cm}^2$ from the conversion equations between those base units.

Industries have delivered some standardization efforts to listing and grouping UOMs. "ISO 31: Quantities and units" and "ISO 1000, SI units and recommendations for the use of their multiple and of certain other units" are the most widely respected style guide for the use of UOMs [5,6]. NIST published SP 811 and SP 330 to guide the use of the International System of Units and many industries rely on NIST for these standards [16]. As yet, neither has taken ontological approaches like ours and provides few guidelines to handle UOM related semantic processing.

4 Conclusion

Our ontology model includes the definitions, properties, and relationships of the concepts that are fundamental to describe a product ontology. The ontology database is modeled and implemented to be operational, and it may function as a knowledge base for searching for products.

In this paper, we present our modeling approach for building an operational product ontology and also provide informative directions to handle three modeling problems. The modeling choice problems presented in this paper are the choice and representation of proper semantic relationships, the handling of product properties, and product descriptions associated with their units of measures. The presented direction to each problem is attained through a project that we participated in. Especially for the semantic relationships, we learned that standard product classification schemes

contain in their taxonomy structures different types of semantic relationships, which should be interpreted and distinguished accordingly to their own semantics. In addition to including product attributes within a product model, we also found it more 'ontological' to have UOMs associated with the attributes. We are currently developing an analytic report on organizing and employing UOMs in our product ontology and plan to publish it as a separate volume.

References:

1. EAN/UCC, "Global Data Alignment System", https://eankorea.koreannet.or.kr/kor/infocenter/info_gds_1.jsp, 2000.
2. W H Emerson, "One as a 'unit' in expressing the magnitudes of quantities", *Metrologia*, Vol. 41, IOP, 2004.
3. D. Fensel, B .Omelayenko, Y. Ding, E. Schulten, G. Botquin, M. Brown, A. Flett, "A Product Data Integration in B2B Electronic Commerce", *IEEE Intelligence System*, Vol.16, 2001.
4. M. Hepp, "Measuring the Quality of Descriptive Languages for Products and Services", in *E-Business – Standardisierung und Integration*, F.-D. Dorloff, et al(Editors), Tagungsband zur Multikonferenz Wirtschaftsinformatik, 2004.
5. ISO, Quantities and units, ISO 31:1992 International Organization for Standardization, 1992.
6. ISO, SI units and recommendations for the use of their multiples and of certain other units, ISO 1000:1992, International Organization for Standardization, 1992.
7. D. Kim, S. Lee, J. Shim, J. Chun, Z. Lee, and H. Park, "Practical Ontology Systems for Enterprise Application", 10th Asian Computing Science Conference Data management on the Web (ASIAN2005)/LNCS Vol. 3818, Springer-Verlag, 2005.
8. KOCIS, Korea Ontology-based e-Catalog Information System, Accessible online at <http://www.g2b.go.kr:8100/index.jsp>.
9. S. Lee, "Design & Implementation of an e-Catalog Management System", Tutorial at 9th International Conference on Database Systems for Advances Applications (DASFAA 2004), 2004.
10. J. Lee and R. Goodwin, *Ontology Management for Large-Scale E-Commerce Applications*, Electronic Commerce Research and Applications, Elsevier, 2006
11. Lee, S. Lee, T. Lee, S. Lee, D. Kim, J. Chun, H. Lee, and J. Shim, "Practical Issues for Building a Product Ontology System", *International Workshop on Data Engineering Issues in E-Commerce (DEEC2005)*, IEEE Society, 2005.
12. L. Obrst, R.E. Wray, and H. Liu, "Ontological Engineering for B2B E-Commerce", *International Conference on Formal Ontology in Information Systems (FOIS'01)*, ACM Press, 2001.
13. J. Shim, H. Lee, and S. Lee, *Conceptual and Formal Modeling of Product Ontology*, Technical Report, Center for e-Business Technology, 2005.
14. S. Staab and R. Studer (Eds.), *Handbook on Ontologies*. International Handbooks on Information Systems, Springer-Verlag, 2004.
15. V. C. Storey, "Understanding Semantic Relationships", *VLDB Journal*, Vol. 2, VLDB Endowment, 1993.
16. B. N. Taylor, "Guide for the Use of the International System of Units (SI)", NIST Special Publication 811, U.S. Government Printing Office, 1995.

Ontologizing EDI Semantics

Doug Foxvog¹ and Christoph Bussler²

¹ Digital Enterprise Research Institute, Galway, Ireland

² Cisco Systems, Inc., San Jose, CA, USA

¹ doug.foxvog@deri.org, ² cbussler@cisco.com

Abstract. Electronic Data Interchange systems are used to transmit business messages in a compact form, with the meaning of message components positionally determined. We propose translating EDI messages into “semantic” forms in which the statements encoded in the compact forms are explicitly expressed, allowing reasoning based on such messages by any program using the semantic language, allowing their generation and use by Semantic Web services through the use of mediators. We present a system for semantically encoding EDI messages through the use of semantic templates.

Keywords: Ontology, EDI, semantic encoding, semantic mediator, semantic template, WSML, Semantic Web Services.

1 Introduction

From the point where communication between two or more computer systems was required, there was a need to establish precise definitions of the communication. This included not only precisely defining message exchange patterns, but message content, so that the sender and receiver both understand the information content of the message in the same way.

The traditional approach was for the sender and receiver to agree on message syntax so that each could parse the message successfully. Soon it became clear that agreement on message content definitions is as important as agreement on syntax.

A sender of business messages populates them with business content which the receiver must interpret. The burden of interpretation lies on the receiver; and if it has many partners, there can be quite an effort to interpret the many incoming messages. Misinterpretation with resultant negative economic impact is a likely consequence.

A more accurate approach is to describe message structure and content independent from application context. Once an Electronic Data Interchange (EDI) system is semantically defined, senders and receivers can construct and interpret messages uniformly so that misinterpretation does not occur [1]. Communication should then work flawlessly for their mutual benefit.

This approach is followed in this work. The ANSI X12 EDI standard is used since it has been used world-wide for over 30 years. There is significant experience with the use of X12 and the meaning of its messages. The recent surge in works on semantics makes EDI a perfect area for combining these worlds to make communication precise.

1.1 Ontologizing EDI

Ontologizing EDI semantics involves the creation of a system that, given an EDI message, enables the meanings of each encoded statement to be generated in a semantic language. This can be done using templates for expressing the meanings of each of the subcomponents of a message and the relationships among them.

The meanings of the simplest data elements may be simple data values, individuals, classes, or relations that may hold among such values. This meaning may either be looked up in a code set or be a known or new instance of the type specified in the template. If the meaning does not come from a code set, the relationship between the message text and the represented thing needs to be defined, e.g. the name of a ship, or a number of grams. The main referent of a more complex message component is taken to be its basic “meaning”.

Templates for complex message components would normally specify the meanings for their slots and the relations that interrelate the meanings of various components. Either the relations which define the templates must take relations and classes (as well as individuals and simple data values) as arguments or such templates must be expressed as rules. The use of standard forms instead of rules to define templates limits the choice of ontology language since many ontology languages do not permit classes or relations to be used as arguments to relations. The desire to use ternary relation further restricts the choice of language for expressing the templates.

Approach. EDI systems have message and subcomponent types. ANSI X12 [2] has over 300 message types (called “transaction sets”), while EDIFACT [3] has around 200. Message subcomponent types number in the thousands. Because of the large size of these EDI systems, ontologizing their semantics is a massive task, but can show benefits long before being completed.

Since few companies need to transmit student records, US Customs documents, and voter registration documents – all X12 message types – the semantics for these would reasonably be defined in different ontologies. On this basis, templates for the most common messages can be created first, with sets of semantically-related messages added in turn. The same data elements and data segments recur in unrelated message types. The semantics of such components should be defined in separate ontologies referenced by the message type ontologies. Knowledge bases which define named terms (e.g. airports) mentioned in messages should be imported along with the ontologies for the associated message type whenever the template is used.

1.2 Sample EDI Message

The Terminal Information transaction set (#319) is one of the simplest defined in the X12 standard. It contains only four data segments (including mandatory transaction set header and trailer), 32 data elements, and no inner loops.

A sample message instantiating this transaction set is shown in Figure 1. The first segment is a header segment. The second is a Beginning Segment for Cargo Terminal Information. The third is a Cargo Detail segment. The fourth segment (fifth line) is a trailer segment. This message will be explained and used as an example below.

```

ST*319*000123>
BA2*LYKL*7*7A*Dubai Ports LTD*FedEx*01*Pier 1*T*New Orleans>
CD1*BXGL*012345**BOLNUM123456*LT****TerminalLoc4321*1*BBL94*6
9***T*BR*1.02*****A*LYKL*SO*Shipment02345*YD*doug foxvog>
SE*4*000123>

```

Fig. 1. Sample Instantiation of X12 Terminal Information Transaction Set

1.3 Document Layout

The rest of this document lays out what is necessary so that a semantic system could take in an EDI message and produce a set of statements expressing the encoded information in a logical language. Section 2 describes a design and ontology for semantic templates for information structures. Section 3 presents illustrative template encodings of the message type illustrated in Figure 1. Section 4 describes the tasks of the mediator in producing the semantic translation given an input EDI message and knowledge bases of semantic templates and presents logical statements generated for the above message using these templates. Section 5 presents related work. Section 6 summarizes the work that has already been done and outlines the work to follow.

2 Templates

A template for an information structure provides all the information needed to generate a sentence in the logical language which is implicit in an instantiation of that structure. A complex structure will have several templates for different implicit sentences. Each template specifies the predicate and arguments for a sentence. Each value may be derived from the instantiated information structure or be explicitly fixed in the template.

We have created a set of relations and classes for expressing such templates as described below and encoded them in WSML-Flight and CycL. Their definitions can be found at <http://www.wsmo.org/TR/d27/v0.2/ontologies>.

CycL allows for consistency checking and other reasoning during the creation of the ontologies. WSML was selected because it is a Semantic Web language which permits ternary relations and allows for classes and relations as arguments. An OWL encoding is not provided – that would require decomposing ternary relations into multiple binary properties, making template encoding more complex.

2.1 Template Classes

The classes to be defined for encoding the meaning of EDI messages are templates, code sets, formats, and positions in formatted structures. Terminology used for specifying EDI syntax described in [4] is used. Classes defined for templates include:

CodeSet: A mapping between short text strings and a set of things.

ComponentTemplate: A structure for determining a statement encoded in an instantiation of an Information Structure.

ComponentTemplate-Matching: A `ComponentTemplate` in which the generated statement must already exist in the data base. This can be used to bind a term being generated to an existing element. If all terms are already bound, it is a request to verify that the statement is known.

FormulaArgPosition: A position in a formula or formatted structure.

2.2 Template Relations and Function

The basic template relations are:

componentOf[Sub]Type indicates that an instantiation of the specified info structure means an instance/subclass of the specified class.

subcomponentOf[Sub]Type indicates that the filler for the Nth position of an instantiation of the specified info structure means an instance/subclass of the specified class. This may be a sub-concept of the class to which the filler structure is restricted.

hasSameMeaningAs indicates that the filler for the first specified position in the structure has the same “meaning” as the filler for the second. The same term will be used for each in the derived sentences. This is a ternary relation.

functionalPredicateEncodes indicates that the “meaning” of a filler for an instantiation of the specified structure is the unique thing related to the filler by the specified predicate, e.g., `passportNumber`. This is a ternary functional predicate.

directlyEncoded indicates that the “meaning” of an instantiation of the Nth slot in an info structure is the same as that of whatever fills it. This is the normal case.

[subcomponent]usesCodeSet indicates that the specified code set maps the filler for a specified [slot of an] info structure to its “meaning”. This is not a functional predicate – sometimes codes may come from any one of several code sets.

templateForComponent indicates one of possibly several templates for a logical sentence encoded by instantiations of a given information structure.

templateRelation[-Encoded] indicates the relation for a given template. The relation may either be specified or encoded at a specified position of the info structure. These are functional predicates.

templateArg<N>[-Encoded/Value] (for N 1, 2, or 3) indicates that the Nth argument for a given template is the specified value (or encoded or present but not encoded) at the specified position of the referenced info. Functional predicates.

One function is necessary to denote positions in a formatted structure, whether in the referenced structure, or nested in subcomponents or supercomponents of the structure.

FormulaArgPositionFn. The arguments are numbers which indicate the subcomponent number within the indicated structure. A zero indicates the next higher level. Thus, `FormulaArgPositionFn(3,2)` is the second component of the third component, and `FormulaArg-PositionFn(0)` is the structure itself, not a subcomponent. The enclosing structure is `FormulaArg-PositionFn(0,0)` and its third component is `FormulaArgPositionFn(0,0,3)`.

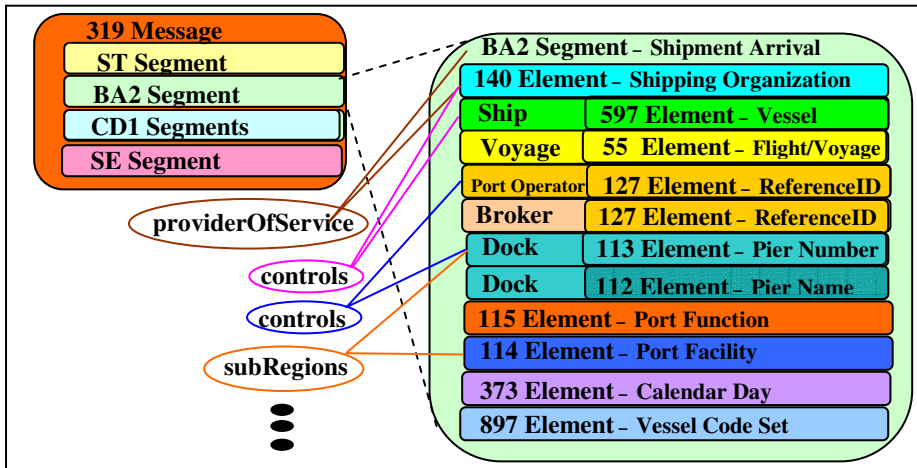


Fig. 2. Format of Transaction Set 319

3 Example Encoding of EDI Message Template

Example encodings of templates for an X12 transaction set are presented below, including representative templates for the transaction set and some of its subcomponents. The use of the templates is explained in Section 4. Statements generated by the templates from a sample message of Figure 1 are presented in Chapter 5. The complete templates (in WSMML) of all subcomponents of this message type can be found at <http://www.wsmo.org/TR/d27/v0.2/ontologies>.

```
instance X12-TS-319 componentOfType ShipmentArrival
instance X12-BA2-DS componentOfType ShipmentArrival
instance X12-CD1-DS componentOfType Shipment
relationInstance hasSameMeaningAs(X12-TS-319,
    FormulaArgPositionFn(0),FormulaArgPositionFn(2))
instance X12-TST-319-2 memberOf ComponentTemplate
templateForComponent hasValue X12-TS-319
templateRelation hasValue transportees
templateArg1-Encoded hasValue FormulaArgPositionFn(0)
templateArg2-Encoded hasValue FormulaArgPositionFn(3)
```

Fig. 3. Sample Templates for Transaction Set 319

Figure 2 shows the format of the Terminal Information transaction set (#319). ST is the Transaction Set Header data segment; BA2 is the Beginning Segment for Cargo Terminal Information; CD1 is the Cargo Detail data segment; and SE is the Transaction Set Trailer data segment. All data segments are mandatory, but the CD1 segment may occur multiple times, while each other segment appears only once. The ST and SE segments are members of every transaction set, providing only meta-information about the message. A few relations implicit in the message are shown.

A 319 message is about a shipment arrival. The BA2 segment is about the same shipment arrival. CD1 segments are about different shipments which arrive in the same ship. Figure 3 shows how this is encoded using the classes and relations described above. For clarity, namespace references are removed. The mediator generates a transportees statement using this template. This is the most common template pattern – relating one attribute with its two arguments. Additional templates relate items specified in the BA2 segment to items specified in the CD1 segments.

```
relationInstance subcomponentOfType (X12-BA2-DS, 2, Ship)
" subcomponentOfType (X12-BA2-DS, 1, ShippingOrganization)
" subcomponentOfType (X12-BA2-DS, 6, Dock)
" subcomponentOfType (X12-BA2-DS, 7, Dock) . . .
" hasSameMeaningAs (X12-BA2-DS, FormulaArgPositionFn(6),
                    FormulaArgPositionFn(7))
```

Fig. 4. Types of Subcomponents of BA2 Data Segment

A BA2 segment starts with seven mandatory data elements followed by four optional ones. It starts with codes for the shipping organization and for identifying the ship itself. Each component enters into at least one template for a sentence explaining its meaning within a 319 transaction set.

Although the data elements may store codes or names, their “meaning” is taken to be the thing that the code or the name represents. For example, data element 140 is called “Standard Carrier Alpha Code”, but its “meaning” is the cargo carrier, not the four-character SCAC assigned to that carrier. The BA2 segment gives two identifiers for the pier at which the ship docks: a number and a name. The `hasSameMeaningAs` relation ensures the same term will be generated from the templates by the mediator for the two slots. Types associated with message components are encoded as in Figure 4; component formats as described in [4].

```
instance X12-DE-140
    usesCodeSet hasValue SCACCodeSet
    componentOfType hasValue ShippingOrganization
instance X12-DE-597
    componentOfType hasValue TransportationDevice
relationInstance subcomponentOfType (X12-BA2-DS, 2, Ship)
instance X12-DET-597 memberOf ComponentTemplate
    templateForComponent hasValue X12-DE-597
    templateRelation hasValue identificationStrings
```

Fig. 5. Example Template for Data Element

The meaning of the instantiations of data elements may either be determined from code sets or obtained through templates. For example, the first data element (140) of the BA2 data segment (597) is encoded using the Standard Carrier Alpha Code, while the second data element may be identified by name or other ID. Figure 5 shows the representations for these two cases. Note that Data Element 597 represents a transport

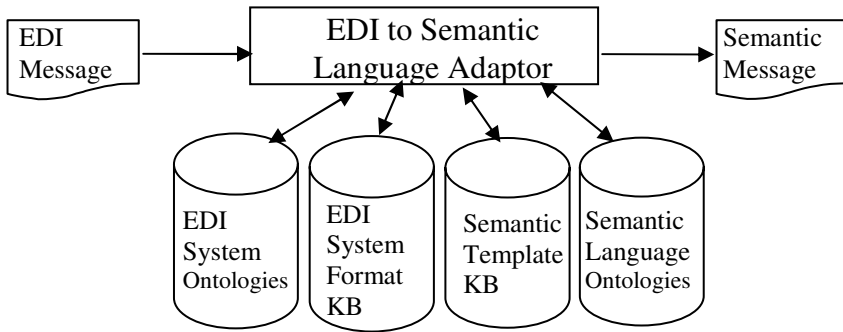


Fig. 6. Adaptor Data Sources

device, but is restricted to a ship in the context of BA2. The templates for the rest of Transaction Set 319 are presented at <http://www.wsmo.org/TR/d27/v0.2/ontologies>.

4 Generation of Logical Statements from EDI Messages

The task of a Semantic Mediator is to convert a message between two formats or ontologies, at least one of the forms being semantic [5]. Such a mediator uses a pre-established mapping between terms and structures in the two systems and may convert between languages if necessary. To generate logical statements from EDI messages, a mediator is needed that uses the template rules and the formats to convert individual messages into a set of logical statements (in WSML in our case).

The task of the X12 to WSML mediator is to separate the components of the X12 message programmatically; using the syntax defined in WSML for EDI components to determine which semantic templates to use. The mediator matches the templates against the extracted message components to generate the WSML statements which express the meaning of the X12 message. See Figure 6.

The mediator in the other direction, given a WSML message, uses the same templates to generate an X12 message, using the syntax described for the message type and its subcomponents. This mediator may have to add information required by mandatory message elements from data existing in the knowledge base (e.g., a party's contact information) or which it calculates (e.g. the number of data segments being transmitted) if that information had not been included in the input WSML message.

Variations of these mediators could handle other EDI formats such as EDIFACT or different semantic languages such as OWL. Features needed by the mediator for expressing the templates (ternary relations, functions, and relations and classes as arguments) are not needed in either the input or output languages.

5 Generation of Logical Statements from X12 Messages

A portion of the intended output of the conversion of the sample X12 message shown in Figure 1 using the WSML templates is presented below. Although semantic output

is far less compact than the X12 format, it allows for reasoning about the message content. The mediator creates new terms from the names of the instantiated concepts.

The BA2 segment of Figure 1 encodes the following: the shipper for this shipment arrival has SCAC code LYKL (which turns out to be Lykes Lines Ltd). Ship no. 7 is the carrier on voyage no. 7A, which ends with this arrival. Dubai Ports Ltd. is the port operator and FedEx is the broker for the shipment. Ship no. 7's arrival occurs at Import Pier 1 (numbered "01") at the port of New Orleans. The arrival of this ship is part of transshipment – the port is not the final destination of its shipment lots.

```
instance ShipmentArrival_1 memberOf ShipmentArrival
      providerOfService hasValue LykesLinesLimited
      transshipmentPort hasValue PortOfNewOrleans
instance Dock_1 memberOf Dock
      identificationStrings hasValue "01"
      placeName-ShortForm hasValue "Import Pier 1"
```

Fig. 7. Encoded Meaning of BA2 Segment

Figure 7 presents the encoding of part of this information, which the mediator should generate from the templates. The `memberOf` statements are generated based on the `subcomponentOfType` assertions of Figure 4. A template relating the zeroth and first argument of the BA2 segment generates the `providerOfService` statement. Similar templates are responsible for the other statements. Statements are similarly generated from the CD1 segment.

```
instance X12TS_1 memberOf X12TransactionSetInstantiation
      instantiationOfAIS hasValue X12_TS_319
      identificationStrings hasValue "000123"
relationInstance verify(segmentCount, X12TS_1, 4)
      " verify(identificationStrings, X12TS_1, "000123")
```

Fig. 8. Encoded Meaning of ST Segment

The Transaction Set Header of Figure 1 indicates that the message is an instantiation of a 319 transaction set with ID string "000123". The trailer templates generate two statements that should be verified by the system: the number of segments included in the message and its control number. No new information is provided by the trailer. Figure 8 shows the WSMML code produced by the templates for these data segments.

```
instance ShipmentArrival_1 memberOf ShipmentArrival
      transportees hasValue Shipment_1
instance FedEx controls hasValue Shipment_1
```

Fig. 9. Encoded Meanings at the Message Level

Figure 9 lists the meaning of several statements generated at the message level. The shipment defined in the CD1 segment is transported in the arrival defined in the BA2 segment and the broker specified in the BA2 segment controls the shipment defined in the CD1 segment. With multiple CD1 segments, similar statements would be generated using the same BA2 data. If OWL-like blank nodes were used to generate the terms referred to in each segment, the generation of such statements linking terms defined in different segments would be more difficult.

6 Related Work

Beginning in the 1990s, problems with EDI led to the development of systems intended as state-of-the-art replacements for the apparently antiquated X12 and EDIFACT systems. The new systems were not designed to be backward compatible with X12 or EDIFACT to allow translation of such messages into the new forms.

Few of the newer systems are semantically described. They mainly introduce a more open syntax (XML) and transmission medium (the Internet) and thus maintain those disadvantages of traditional EDI which result from semantic opacity. For an overview of some of the major systems, see [4].

Some private companies market proprietary software that produce and accept business messages in both traditional EDI and more recent XML-based formats [6, 7, 8]. However, these systems appear not to be based on ontologies.

The ASC X12 Committee saw a problem with a proliferation of redundant XML-defined message systems and wanted to ground the new systems as much as possible on a standard industry-neutral set of XML concepts. They created a Context Inspired Component Architecture (CICA) [9, 10] as a syntax-neutral architecture for XML-based business messaging. They have started converting traditional X12 message types into the new architecture, releasing their first five X12 XML message schemas in December 2005. This is an important development as the major US EDI standards organization starts providing EDI semantics and moves away from syntactically rigid message requirements. However, the semantics is only at the broadest level of major message types and most common message components (such as person and address). A semantic description of most of the content of business messages is not provided.

The Universal Business Language (UBL) is an effort started in 2003 to unify the plethora of XML standards for business documents [11]. UBL 1.0 was officially declared an OASIS Standard in November of 2004 with eight message types. UBL has XML definitions for many information components (such as addresses and product codes) that are present in a variety of common business messages (such as invoices and shipping manifests). Use of UBL for messages between industries was envisioned in conjunction with industry-specific messages being used within an industry. However, mappings of message types and their components to definitions of equivalent terms used in XML-based EDI (RosettaNet, HL7, CDIX, etc.) are not provided. No translator is provided to map X12 or any other standard into UBL. Nor is a method provided for specifying the mapping between different languages.

In the 1990s, Cycorp took two large medical term taxonomies, SNOMED¹ and MeSH², and mapped them to an anatomical sub-ontology which it was developing [12]. Use of the taxonomies aided in the production of the ontology; however, it was highly enriched from the bare taxonomic forms of the sources and the majority of terms in the taxonomies was omitted as the scope of the target ontology was narrower. This project was not intended to create a semantic encoding of an existing system, but to extract a specific subset of information encoded in two systems to aid in developing an ontology for a field and to establish a mapping between the semantically defined terms and the pre-existing terms in the taxonomies.

7 Summary and Future Work

We have demonstrated that semantic templates can be manually created for traditional EDI systems that would enable a data mediator to produce a semantic description of a message encoded in that system. Such a mediator would use these templates and a semantic description of the EDI message format in the translation.

We plan to use these techniques to ontologize standard subsets of EDIFACT for the Sixth Framework Programme specific targeted research project TripCom³ as a basis for business to business process integration. TripCom will allow business data to be represented as RDF triples for inter-business interactions.

Acknowledgments. This work is funded by the EU projects the DIP (FP6 - 507483) and TripCom (FP6 - 027324) and by Science Foundation Ireland under the DERI L  n project.

References

1. Bussler, C: B2B Integration, Springer-Verlag, Heidelberg (2003)
2. X12/DISA Information Manual, ASC X12S/92-707, Data Interchange Standards Association (DISA), Alexandria, VA (1992)
3. Berge, J.: The EDIFACT Standards. NCC Blackwell: Manchester, England (1991)
4. Foxvog, D., Bussler, C.: "Ontologizing EDI: First Steps and Experiences"; Proceedings of the International Workshop on Data Engineering Issues in E-Commerce (DEEC 2005), Tokyo, Japan, 9 April 2005, 49-58
5. Mocan, A. and Cimpian, E.: Web Service Modeling Execution Environment - Conceptual Model (WSMX_O), DERI Technical Report D13.3 v0.3
6. Meehan, M.: Aerospace Group Backs New EDI-to-XML Bridge, Computer World, 12 October 2001
7. Extreme Translator, <http://www.xtranslator.com>
8. Mercator Is First Integration Solution Provider to Offer ACORD, XML, EDI, HIPAA, and SWIFT, Business Wire, 21 May 2002

¹ <http://www.snomed.org/snomedct/>

² <http://www.nlm.nih.gov/mesh/meshhome.html>

³ <http://tripcom.org/>

9. Accredited Standards Committee X12 (2004). The Future of Standards Development: Context Inspired Component Architecture – Using CICA Architecture for Building XML Messages. Available from <http://www.disa.org/x12org/MEETINGS/x12trint/cica.cfm> .
10. ASC X12 Releases First Set of W3C-Compliant XML Schemas Based on CICA, Insurance News Net, 15 December 2005. Available at http://www.insurancenewsnet.com/print.asp?a=featured_pr&id=55171
11. Crawford, M.: UBL, 2001. Available from <http://www.oasis-open.org/committees/download.php/4610/xml2003.pdf>
12. Lehmann, F. & Foxvog, D.: Putting Flesh on the Bones: Issues that Arise in Creating Anatomical Knowledge Bases with Rich Relational Structures, Knowledge Sharing across Biological and Medical Knowledge Based Systems, Papers from the 1998 Workshop: Technical Report WS-98-04, Fifteenth National Conference on Artificial Intelligence, Madison, WI, 26 July 1998, 41-50

WSDL RDF Mapping: Developing Ontologies from Standardized XML Languages

Jacek Kopecný

DERI Innsbruck

Technikerstrasse 21a, 6020 Innsbruck, Austria

`jacek.kopecny@deri.org`

Abstract. The W3C is working on a Recommendation for Web Services Description Language, based on XML. The Working Group is chartered to produce a normative mapping of WSDL into RDF, effectively a WSDL ontology. While this work is still ongoing, in this paper we propose guidelines for creating such ontologies coming from XML standards.

1 Introduction

In the World Wide Web Consortium (W3C), the Web Service Description Working Group¹ has been working on version 2.0 of the popular Web service description language WSDL [8]. This XML language allows Web service providers to describe the functional interface of the service, that is, the types and sequences of input and output messages.

In order to add WSDL data to the Semantic Web, the working group was chartered to produce an RDF mapping for the XML-based WSDL documents. This mapping specification has two main parts: a WSDL ontology and a formal mapping from WSDL documents to RDF data. The author of this paper is the main editor of the WSDL RDF mapping specification [2].

While working on this specification, we have learned several lessons. For example, our initial draft of the WSDL ontology closely followed the formalization of WSDL, the so-called *component model*, and was largely rejected by Semantic Web practitioners² for its complexity and opacity. The current version tries to convey the intention of WSDL in what should be a much better RDF form.

In this paper, we describe the WSDL RDF mapping effort and propose a number of guidelines for creating ontologies for standard XML-based vocabularies. In our experience, XML-based standards share an interesting characteristic: while the XML language itself is usually fairly user-friendly, it is often backed by a strong and detailed formalization. The intricacies of the language are only grasped by experts, yet many non-experts will use the language interoperably only thanks to an intuitive understanding of the syntax.

¹ <http://www.w3.org/2002/ws/desc/>

² The initial draft was presented to the Semantic Web Interest Group of W3C at its meeting in Boston, March 2005.

We can demonstrate this characteristic on XML Schema [1]. In limited forms it is used and even implemented in most systems working with XML data, yet complete and correct support is lacking, and cooperating systems often have conflicting limitations. The W3C has started a working group called XML Schema Patterns for Databinding³ in an attempt to specify an interoperable subset of XML Schema that would be supported reliably in heterogeneous systems.

We argue that when mapping an XML language to RDF, attempting to capture all the details of the formalization is counterproductive. Ontology engineering methodologies (cf. [5]) use the notion of *minimal ontological commitment* to limit the level of constraints that are expressed in an ontology, and we argue that ontologies based on XML languages need to aim for even lower explicit ontological commitment.

In the following sections, we first give an overview of WSDL in Section 2, and we describe the draft WSDL ontology in Section 3. In Section 4 we propose our guidelines for working on standards-based ontologies, followed by showing how the proposed guidelines apply to our draft WSDL ontology in Section 5. Finally we discuss related work in Section 6 and we conclude the paper in Section 7.

2 WSDL Overview

WSDL 2.0 is a language for describing Web services. In particular, it can describe the structure of the messages the service accepts and produces, simple message exchanges (called operations) and all necessary networking details. On top of this, extensions in WSDL documents can specify that additional features are supported or even required by the service. In effect, WSDL specifies a limited contract that the service adheres to.

The WSDL specification is written in terms of a *component model*, presenting any WSDL document as a set of components with local properties. The core WSDL is a simple set of components: the top-level Description component represents a WSDL document together with all imported and included documents. Every Description may contain Interface, Binding and Service components⁴.

An Interface component describes the abstract interface of a Web service, i.e. the operations, messages and faults. An operation is represented with an Interface Operation component, which has a property {message exchange pattern} to point to the MEP that this operation follows. An MEP in WSDL prescribes the number and directionality of messages, and the operation populates them with concrete XML elements in its Interface Message Reference components. Each Interface Message Reference represents a single message, refers to the appropriate MEP message and contains an {element declaration} property which points to a schema. Most MEPs also allow fault messages for expected application-level

³ <http://www.w3.org/2002/ws/databinding/>

⁴ Formally, the Description component contains properties {interfaces}, {bindings} and {services}, each of which is a set of the respective Interface, Binding and Service components. We use the notation {property name} to indicate component properties, as the WSDL specification does the same.

errors. Faults are modeled on the same level as operations, that is, an interface defines a number of faults (Interface Fault components), which are then used within operations (with Interface Fault Reference components).

An interface is specified on the level of XML messages; the networking details about how the messages are represented on the wire are specified in the next top-level WSDL component, Binding. The Binding component follows the structure of Interface and uses extensions to specify any networking parameters.

The last top-level WSDL component, Service, provides a number of endpoints where a service is available. And endpoint specifies an actual address, together with a Binding that indicates how to communicate with that address.

Figure 1 shows a WSDL document (adopted from the WSDL Primer [4]) that describes a simple hotel service. This interface provides a single operation for checking the availability of rooms for a given date. Abstractly (i.e., on the Interface level), the operation accepts the dates of check-in and check-out and the required type of room, and it returns the daily rate of available

```
<description targetNamespace="http://hotel.example.com/wsdl" ...>
  <types>
    <xs:schema ...>
      <xs:element name="checkAvailability" type="tCheckAvailability"/>
      <xs:complexType name="tCheckAvailability">
        <xs:sequence>
          <xs:element name="checkInDate" type="xs:date"/>
          <xs:element name="checkOutDate" type="xs:date"/>
          <xs:element name="roomType" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
      <xs:element name="checkAvailabilityResponse" type="xs:double"/>
      <xs:element name="invalidDataError" type="xs:string"/>
    </xs:schema>
  </types>

  <interface name="hotelIface">
    <fault name="invalidDataFault" element="ghns:invalidDataError"/>
    <operation name="checkAvailability" pattern="http://www.w3.org/2006/01/wsdl/in-out">
      <input element="ghns:checkAvailability"/>
      <output element="ghns:checkAvailabilityResponse"/>
      <outfault ref="tns:invalidDataFault"/>
    </operation>
  </interface>

  <binding name="hotelSOAPBinding"
    interface="tns:hotelIface"
    type="http://www.w3.org/2006/01/wsdl/soap"
    s:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP">
    <fault ref="tns:invalidDataFault" s:code="soap:Sender"/>
    <operation ref="tns:checkAvailability"
      s:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
  </binding>

  <service name="hotelService" interface="tns:hotelIface">
    <endpoint name="hotelEndpoint" binding="tns:hotelSOAPBinding"
      address="http://hotel.example.com/service"/>
  </service>
</description>
```

Fig. 1. Example hotel service description

rooms, or zero if nothing is available. In case the room type or input dates are erroneous, a fault can be generated. The Binding level specifies that the data will be transmitted with SOAP over HTTP, plus several other details. Finally the Service level gives one endpoint address, at which the hotel service is available.

WSDL is, by design, a very extensible language, and in fact some parts of the standard are built as predefined extensions (see [3]). There are three distinct kinds of extensibility in WSDL: extension points, Features and Properties, and generic XML-based extensions.

Extension points are those places in WSDL descriptions where a number of options is defined by WSDL, but the list is open; for example, every WSDL operation follows some message exchange pattern, and while WSDL provides a list of eight predefined patterns, WSDL users are free to define new ones, if necessary. Extension points have specific and non-overlapping scopes, for example the mentioned MEPs guide the ordering of messages, *operation styles* generally restrict message schemas, and *binding types* specify the networking details with different underlying protocols.

Features and Properties form a specific extensibility mechanism in WSDL. A Feature is an abstract piece of functionality (e.g. making communication confidential) and a Property is a concrete parameter of that functionality (e.g. the required strength of confidentiality). Every component in WSDL may *offer* or *require* the use of specific Features.

Apart from the scoped extension points and the abstract Features and Properties, WSDL is open to XML-based extensibility, i.e. any WSDL element can contain any number of attributes or elements from a foreign (non-WSDL) namespace. Such extensions are not constrained in what they may mean. In general, extensions add properties to the existing WSDL components, so that the processor can use the extended information. As we cannot expect all WSDL processors to know all the extensions they might encounter, extension elements in WSDL are by default *optional*, but they may be marked *mandatory*. Optional extensions can be ignored by a processor that does not recognize them, whereas mandatory extensions must be understood by a processor that needs to process the parent WSDL element. More significantly, mandatory extensions are even allowed to change the meaning of the parent WSDL element.

WSDL is specified as an extensible component model as opposed to an XML grammar in order to abstract away, among others, many ambiguity issues of XML serialization and document includes and imports. Apart from the actual specification prose, the component model of WSDL is additionally defined using the Z notation [7]. The Z notation was used to help ensure that the test suite covers all important WSDL component model constraints, and it can be used for validation of WSDL documents.

The Z notation is also useful to us when we work on the WSDL ontology; it serves as a concise listing of all the components, their properties and all the component model constraints from WSDL.

3 WSDL Ontology

This section describes the WSDL ontology defined in the WSDL RDF mapping specification [2], highlighting the main differences between the component model and the ontology structure. The ontology is defined in OWL [6], but due to the layering of the Semantic Web languages, we expect that WSDL documents mapped into this ontology will be useful even to generic RDF processors that do not support OWL inference.

The core components of WSDL are mapped to corresponding classes with matching names, that is, the ontology defines classes **Description**, **Interface**, **Binding** and **Service** etc. Instances of these classes are connected using properties whose names are the same as those of their range classes, only beginning with lower case, so for instance the property **interface** links a concrete instance of **Description** with the concrete instance of **Interface** that it contains.

In the component model of WSDL, components are usually identified by their type, their name, potentially the names of the components in the owner hierarchy, and with the namespace of the defining WSDL file. The example WSDL

```
@prefix rwsdl: <http://www.w3.org/2005/10/wsd1-rdf#> .
@prefix rsoap: <http://www.w3.org/2005/08/wsd1/soap#> .

<http://hotel.example.com/wsd1#wsdl.description()>
  a rwsdl:Description ;
  rwsdl:interface <http://hotel.example.com/wsd1#wsdl.interface(hotelIface)> ;
  rwsdl:binding <http://hotel.example.com/wsd1#wsdl.binding(hotelSOAPBinding)> ;
  rwsdl:service <http://hotel.example.com/wsd1#wsdl.service(hotelService)> .

<http://hotel.example.com/wsd1#wsdl.interface(hotelIface)>
  a rwsdl:Interface ;
  rwsdl:interfaceFault
    <http://hotel.example.com/wsd1#wsdl.interfaceFault(hotelIface/invalidDataFault)> ;
  rwsdl:interfaceOperation
    <http://hotel.example.com/wsd1#wsdl.interfaceOperation(hotelIface/checkAvailability)> .
<http://hotel.example.com/wsd1#wsdl.interfaceFault(hotelIface/invalidDataFault)>
  a rwsdl:InterfaceFault .
<http://hotel.example.com/wsd1#wsdl.interfaceOperation(hotelIface/checkAvailability)>
  a rwsdl:InterfaceOperation ;
  rwsdl:messageExchangePattern <http://www.w3.org/2006/01/wsd1/in-out> ;
  rwsdl:interfaceMessageReference
    <http://hotel.example.com/wsd1#wsdl.interfaceMessageReference
      (hotelIface/checkAvailability/In)> .

<http://hotel.example.com/wsd1#wsdl.binding(hotelSOAPBinding)>
  a rwsdl:Binding ;
  a <http://www.w3.org/2006/01/wsd1/soap>;
  rwsdl:interface <http://hotel.example.com/wsd1#wsdl.interface(hotelIface)> ;
  rsoap:protocol <http://www.w3.org/2003/05/soap/bindings/HTTP> .

<http://hotel.example.com/wsd1#wsdl.service(hotelService)>
  a rwsdl:Service ;
  rwsdl:interface <http://hotel.example.com/wsd1#wsdl.interface(hotelIface)> ;
  rwsdl:endpoint <http://hotel.example.com/wsd1#wsdl.endpoint(hotelService/hotelEndpoint)> .
<http://hotel.example.com/wsd1#wsdl.endpoint(hotelService/hotelEndpoint)>
  a rwsdl:Endpoint ;
  rwsdl:binding <http://hotel.example.com/wsd1#wsdl.binding(hotelSOAPBinding)>;
  rwsdl:address <http://hotel.example.com/service> .
```

Fig. 2. An excerpt of the RDF form of the example WSDL from Figure 1

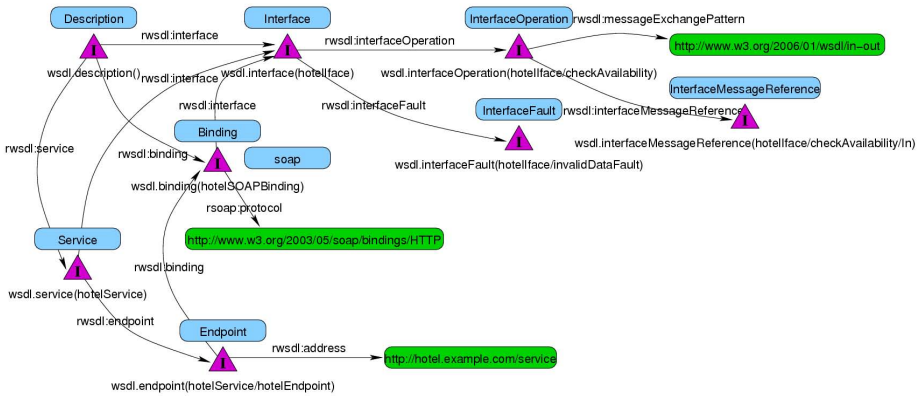


Fig. 3. Graph of the RDF triples from Figure 2

document in Figure 1 contains the Interface component named `hotelIface` in the namespace `http://hotel.example.com/wsdl`. For identifying the components with URIs, as is required by RDF, the WSDL specification provides a set of URI references formed from the namespace, component type and all the relevant names. The example interface is identified as `http://hotel.example.com/wsdl#wsdl.interface(hotelIface)`. For more information on WSDL component URIs see appendices A.2 and C in [8]. It is important to note here that the ontology only uses these composite component identifiers—the names and namespaces are not modeled separately, but they can be reconstructed from the URI references, if necessary.

Figure 2 shows the N3 form of an excerpt of the RDF form of the core components from Figure 1. A graphical representation of the same triples is in Figure 3. For readability we only show a limited part of the RDF data.

Because major pieces of WSDL are specified as standard extensions, the RDF mapping also defines how these predefined extensions are represented in RDF. For example, the RDF ontology contains a class of *operation styles* along with the property `operationStyle` for pointing to particular instances of the class, like the three styles defined by the standard: RPC, IRI and Multipart.

Both standard WSDL bindings (SOAP and HTTP) are also extensions that define additional properties for WSDL components, and these properties are translated into RDF in a straightforward manner.⁵ The two RDF figures show how the extensibility attribute `s:protocol` (specifying the underlying protocol that carries the SOAP messages) is mapped into a simple RDF property `protocol`.

As we already mentioned, WSDL allows arbitrary XML element and attribute extensions. All such extensions should also define how they are represented in RDF, as for instance the HTTP and SOAP bindings do. As WSDL

⁵ Unfortunately the word *property* is overloaded here: the component-model properties from WSDL are represented as RDF properties; and soon the Property components come into play as well.

```

<wsdl:binding name="hotelSOAPBinding" ...>
  <ns:ext xmlns:ns='http://example.com/'/>
    ...
</wsdl:binding>

@prefix rwsdl: <http://www.w3.org/2005/10/wsdl-rdf#> .
<http://hotel.example.com/wsdl#wsdl.binding(hotelSOAPBinding)>
  a rwsdl:Binding ;
  rwsdl:extensionElement
    "<ns:ext xmlns:ns='http://example.com/'/>" .

```

Fig. 4. An example WSDL binding with an unknown optional extension, its RDF form in N3

explicitly allows unknown extensions, we can expect that WSDL documents with unknown extensions may also be mapped into RDF. For this case, the mapping has a fallback mechanism to deal with unknown extensions. Optional unknown extensions are represented in RDF using an XML literal (for extension elements) or an instance with a QName and a literal value (for extension attributes), and the parent components point to these extensions using the properties `extensionElement` and `extensionAttribute`. An example with an unknown extension element is shown in Figure 4. Mandatory extensions, however, may change the meaning of their parent components, therefore the RDF mapping explicitly skips any components that contain mandatory unknown extensions, as the actual meaning of the component must be treated as unknown as well.

To summarize, the RDF mapping is mostly straightforward, mapping components to class instances and component properties to RDF properties, but there are several deviations; notably in component identification, where the RDF uses URIs that conflate the usual QNames of components. Another interesting difference between the component model and the ontology is that a few components (e.g. HTTP Header components) are not present in the RDF form at all, replaced with direct links that should improve the usability of the ontology.

We are currently working on an XSLT stylesheet that will transform WSDL files into the appropriate RDF data. Our stylesheet cannot handle extensions not defined in the WSDL 2.0 specifications, and it has only limited support for imports and includes, therefore we do not plan to make this stylesheet a normative reference implementation.

4 Guidelines for Standards-Based Ontologies

Based on our experience with the RDF mapping of WSDL, we propose the following two guidelines for creators of ontologies for standardized XML data languages:

1. the ontology should follow the *intent* of the language, not necessarily any particular formalization of it,

2. the ontology need not model all data constraints present in the underlying standardized language,

The subsections below explain the guidelines in more detail.

4.1 Intent, Not Formalization

Because we assume that standardized XML languages will have some kind of underlying formalization, it is tempting to translate such a formalization directly into an ontology, given that ontological data is intended for automated computer consumption. We suggest, however, that the ontology should eschew artifacts of the formalization, like component sets and the strict distinction between components and their properties in WSDL.

The rationale for this point lies in the differences in expressivity between the underlying language formalization model and the ontology model. The formalization model is often simple (like WSDL components with properties), whereas ontology models (for example RDF/OWL) can natively capture relationships like class membership and sub-class relations, and this power should be used.

For instance, we modeled properties like binding type or message direction as class membership (`rdf:type`), i.e., binding extensions (HTTP, SOAP or any new ones) are classes, and a given binding component (`<wsdl:binding>`) is an instance of one of these classes. Further, some enumerated properties were combined with container relationships, for example in the WSDL component model, a component can contain a **feature** which is required or optional, and in the ontology the component *offers* or *requires* the feature.

4.2 Not All Constraints

Most languages, not only standardized ones, put many constraints on the data. The constraints can be either explicitly spelled out or implied by the structure of the language. In WSDL, for example, every Service component points to an Interface (explicit constraint), and an Interface Operation can only be defined as part of an Interface (implicit, structural constraint).

Not all of these constraints need to be modeled in the ontology. Language constraints are formalized for validation, so that documents can be automatically checked for correctness. But the mapping to ontological data is expected to work with valid documents, so any RDF graph resulting from the mapping of a valid WSDL file will, by definition, be valid, rendering constraints like OWL cardinalities or even property ranges and domains redundant.

One could assume that all the language constraints should be present in the ontology in order to avoid unintended interpretations of the data; this is part of the principle of minimal ontological commitment. This principle is very useful if the ontological data stands alone; however when creating an RDF mapping for an XML-based language, the RDF data is effectively a read-only RDF view on the original XML data. XML is proven to be an interoperable basis for machine-processible languages, therefore if an application needs to check or access all the intricacies of the language, it should use the XML form.

The RDF form is intended for combining with other ontological data in the Semantic Web. Here, modeling all the available XML constraints could in fact be harmful. For instance, in the WSDL component model, a single Description component contains all the other components, however when we combine information from multiple (unrelated) WSDL documents, we can find one interface reused in multiple places. This is a real scenario that the WSDL specification considers out of its scope (in order to simplify the language).

In other words, when the RDF data is going to be backed by the original XML form, the ontology may beneficially drop some of the XML language constraints, so that the RDF data can better play its role in the Semantic Web. While ontologies are often viewed as complete domain models, we argue that ontologies for XML standards need not be complete in this regard.

5 WSDL RDF Mapping Case Study

Having formulated the guidelines in the previous section, we can now re-apply them to the WSDL RDF mapping. Even though the guidelines stem from our RDF mapping experience, it is useful to revisit them in an iterative process and show where they have helped, and whether they might help even more.

As a good example of overformalization (Guideline 1), our initial draft modeled a Description component with a single property **interfaces** which pointed to a set containing Interface components (description hasInterfaces X; X contains interface1 etc.), which is how the component model is specified. The current draft takes advantage of the RDF set-of-triples model and simply uses multiple **interface** properties on Description, removing the indirection through an explicit set, making the data easier to use.

The current version of the WSDL ontology still contains some cardinality constraints. As discussed in Guideline 2, this may be redundant, making the ontology more complex than is really necessary. We have already identified some of them to be removed in the next version, as direct benefit of having the guideline written down in this paper, yet some of the cardinality constraints seem to remain useful even in the RDF data. In particular, some properties have local cardinality restricted to 1, indicating that the property will always be present, simplifying the use of the ontology.

6 Related Work

Even though ontologies have been created for pre-existing standards like RosettaNet and eCl@ss, we are not aware of any directly related work suggesting a methodology for creating ontologies for reuse of formalized XML languages. However, in the general scope of ontology engineering methodologies, there is some overlap and even conflict with our first two suggestions.

The first guideline could be rephrased as “the existing formalization most probably wasn’t intended for reuse and inference, so it may be improper as the basis for an ontology.” While some could consider it unnecessary effort, we

suggest that redoing the conceptualization and formalization of the underlying language, with focus on ontological modeling, will yield positive results.

The second guideline seems to be contrary to conventional ontological engineering where more information in the ontology is beneficial, but we favor ontology simplicity (increasing the chances of reuse) to complete inference power for the constraints, especially since we assume the constraints are enforced in validation before a piece of data is transformed into the ontological form.

7 Conclusions

In this paper, we described our work on the RDF mapping of the Web Service Description Language (WSDL), and we share the guidelines that we learned from this work, for the benefit of others who map standardized languages into an ontological form, which need not necessarily be limited to RDF/OWL.

Standardized languages share one specific property: they are commonly based on some formalization intended to limit or eliminate ambiguities in interpretation. While very beneficial for ensuring interoperability, such formalizations are not necessarily very convenient for working with actual data, therefore our first guideline suggests that the ontology should focus on the intent of the language, reusing the appropriate constructs available in the ontology language (e.g. multiple class membership and subclassing) even if the underlying formalization does not make use of such constructs.

The second guideline proposes that the ontology need not repeat all the data constraints of the standardized language, because such constraints are in fact enforced before a document is mapped into the ontological form, and may in fact be harmful when combining data from multiple documents.

Even though our work on WSDL RDF mapping is not finished yet, we believe our proposed guidelines can already be useful. If our guidelines are followed, it should be easier to seed the Semantic Web with useful data coming from existing standard languages.

Acknowledgements

This work is partially funded by the European Commission under the project DIP (FP6 – 507483).

References

1. XML Schema Part 1: Structures. Recommendation, W3C, October 2004. Available at <http://www.w3.org/TR/xmlschema-1/>.
2. Web Services Description Language (WSDL) Version 2.0: RDF Mapping. Working Draft, W3C, November 2005. Available at <http://www.w3.org/TR/2005/WD-wsdl20-rdf-20051104/>.
3. Web Services Description Language (WSDL) Version 2.0: Adjuncts. Candidate Recommendation, W3C, January 2006. Available at <http://www.w3.org/TR/2006/CR-wsdl20-adjuncts-20060106/>.

4. Web Services Description Language (WSDL) Version 2.0: Primer. Candidate Recommendation, W3C, January 2006. Available at <http://www.w3.org/TR/2006/CR-wsdl20-primer-20060106/>.
5. M. Fernandez-Lopez, A. Gomez-Perez, and N. Juristo. METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In *Workshop on Ontological Engineering, Spring Symposium Series, AAAI97, Stanford, USA*.
6. D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview. Recommendation 10 February 2004, W3C, 2004. Available at <http://www.w3.org/TR/owl-features/>.
7. J. M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall, 1992.
8. Web Services Description Language (WSDL) Version 2.0. Candidate Recommendation, W3C, January 2006. Available at <http://www.w3.org/TR/2006/CR-wsdl20-20060106/>.

Preface for QoIS 2006

Samira Si-Saïd Cherfi¹ and Geert Poels²

¹ CEDRIC-CNAM, France

² University of Ghent, Belgium

These proceedings contain the papers of the second edition of the QoIS workshop held in conjunction with the 25th International Conference on Conceptual Modeling (ER2006).

Information system quality is expected to have a strong impact on information system effectiveness, which can be defined as the degree to which the information system meets its intended purpose. With proper business/IT alignment, an effective information system is one that successfully plays a central role in the fulfillment of strategic business goals such as cost leadership, increased profitability or sustained growth.

The papers chosen for presentation were selected very carefully by the Program Committee in order to ensure a high-quality workshop. They address the following topics:

- Information system quality: covering conceptual model quality as well as Web engineering approaches, quality measurement or requirements, and model, quality evaluation.
- Data quality: including data quality, data quality requests, data management problems, meta-data quality evaluation and information quality monitoring and measurement.

The workshop attracted 15 papers from over 12 countries and the Program Committee finally selected 6 high-quality papers. We would like to thank all the authors who submitted papers to our workshop. We would also like to thank Steering Committee and Program Committee members and all the referees who gave up their valuable time to review the papers and help in proposing the workshop program. Finally, our thanks go out to Organizing Committee of ER2006 for recognizing the relevance of this workshop and ER2006 Workshop Chairs for their helpful work.

We hope this workshop stimulated exchange between researchers and practitioners about information system quality.

Information Quality, System Quality and Information System Effectiveness: Introduction to QoIS'06

Geert Poels¹ and Samira Si-Saïd Cherfi²

¹ Faculty of Economics and Business Administration
Ghent University - UGent
Hoveniersberg 24, B-9000 Gent, Belgium
Geert.Poels@UGent.be

² CEDRIC-CNAM, 292 Rue Saint Martin, F-75141 Paris Cedex 03
sisaid@cnam.fr

Abstract. This paper is an introduction to the 2nd International Workshop on Quality of Information Systems. It introduces the 6 papers accepted to the workshop via a well-known model of information system success.

1 From IS Quality to IS Success

Information system quality is expected to have a strong impact on information system effectiveness, which can be defined as the degree to which the information system meets its intended purpose. With proper business/IT alignment, an effective information system is an information system that successfully plays a central role in the fulfillment of strategic business goals like cost leadership, increased profitability or sustained growth.

A framework that explains the relationship between information system quality and information system effectiveness is the DeLone and McLean (D&M) Model of IS Success [1] (Figure 1). Since its conception, numerous empirical tests have been conducted that support the constructs and relationships of the D&M Model and that contribute to a better understanding of information system effectiveness and its dimensions [2]. According to the D&M Model, information system quality affects both the use of the information system (i.e. whether the information system is adopted by its intended users) and how satisfied the intended users are with this use. The amount of use can be indicative of the level of user satisfaction, and vice-versa the current level of user satisfaction might predict future or continued information system use. Following the causal chain, use and user satisfaction are direct antecedents of the information system's impact on individual users (e.g. in terms of improved task performance). Finally, it is through these impacts on the individuals of the organization that the information system eventually has an organizational impact (e.g. reduction of operational costs, shorter time to market).

An important feature of the D&M Model is that it distinguishes two dimensions of *information system quality*: *information quality* and *system quality*. The former construct is concerned with the quality of the information that the information system delivers to its users and is measured in terms of accuracy, completeness, currency and

format [3], whereas the latter construct relates to the quality of the system that produces the information output, which can be expressed in terms of accessibility, reliability, flexibility, integration and response time [3] and has also often been measured as the ease of use or user friendliness of the system [4]. According to [1] “system quality and information quality singularly and jointly affect both use and user satisfaction” (p. 83). Given the importance of information system quality in the ultimate success of the information system, the QoIS workshops at the ER conferences intend to provide a forum for high-quality and innovative research in all areas related to information system quality.

The next section introduces the papers accepted to the QoIS 2006 workshop and positions them relative to each other along the lines of the D&M Model of IS Success.

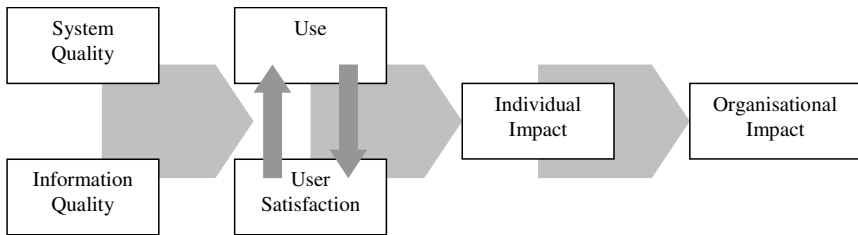


Fig. 1. D&M Model of IS Success [1]

2 The Dual Aspects of IS Quality

Consistent with the D&M Model, the 6 papers accepted to this year’s workshop (out of a total of 15 submitted papers, i.e., an acceptance rate of 40 %) can broadly be classified into a first subset of papers dealing primarily with information quality and a second group focusing on system quality.

2.1 Information Quality

Four workshop papers concern information quality, or even more specifically, data quality aspects. Dong *et al.* propose in *Expressing and Processing Timeliness Quality Aware Queries: The DQ²L Approach* an extension of SQL which enables users to include data quality requests in their queries. The authors also propose a query processing framework that can serve as an implementation of their ideas of quality aware queries. The new query language and processing framework are illustrated using queries that incorporate data timeliness requirements.

In *Quality of Material Master Data and its Effect on the Usefulness of Distributed ERP Systems*, Knolmayer and R  thlin present a case-study on master data management in a large-scale organization characterized by a distributed and heterogeneous ERP infrastructure, running 10 different instances of SAP R/3. Based on the data management problems observed and their effect on master data quality and ERP system usefulness, the authors present their vision on how to improve the quality of the material master data. Their proposed solution takes the form of a master data process that is in accordance with corporate goals, that respects structural data

model constraints, and that assists users in the fulfilment of their data management tasks without interrupting business processes.

In the *Towards Automatic Evaluation of Learning Object Metadata Quality* by Ochoa and Duval, the focus shifts to meta-data. The authors developed and experimentally validated a set of metrics for the automatic evaluation of the quality of meta-data records that describe learning objects. The quality of a meta-data record that describes a learning object affects the probability of the object being found. Being able to automatically evaluate quality properties of meta-data (e.g. completeness, information content, readability, ...) is crucial given that learning object repositories may contain up to millions of meta-data records (in which case manual inspections are infeasible). The authors show that one of the proposed metrics, the entropy-based Textual Information Content metrics, correlates particularly well with meta-data quality assessments performed by human experts. They further describe a number of possible applications and conclude that users searching for a learning object should be presented mainly the free text meta-data fields describing the object.

The final paper related to information quality is *HIQM: a Methodology for Information Quality Monitoring, Measurement, and Improvement* by Cappiello *et al.* The authors present an elaborate information quality management and improvement methodology, called Hybrid Information Quality Management (HIQM), that is highly original in the sense that it supports data error detection and correction at run-time. The paper describes the eight phases of the HIQM methodology, with a special emphasis on the warning phase and the required architecture for warning management. The proposed methodology is illustrated by means of a bookshop scenario employing web services.

2.2 System Quality

The papers of Cachero *et al.* and Condori-Fernandez *et al.* relate to system quality rather than information quality. In *Quality-driven Automatic Transformation of Object-Oriented Navigational Models* Cachero *et al.* present a novel approach to define quality measures for conceptual models of web applications. A Software Measurement Ontology (SMO)-aware meta-model of Web Engineering methods is proposed that can be instantiated to define measures in terms of constructs that are independent of specific Web Engineering approaches. This approach to measure definition improves the reusability and applicability of proposed quality measures. Further, the authors demonstrate the use of QVT model transformation rules (in an MDA context) that show great potential of refactoring web application conceptual models in order to improve their quality characteristics, based on automated quality measurements. The proposed approach and its application are illustrated for the navigability quality property of OO-H navigational models and the Number of Navigational Targets metric. The work of Cachero *et al.* clearly fits the System Quality construct of the D&M Model as the quality of a web application conceptual model is expected to impact the quality of the resulting web application.

Condori-Fernandez *et al.* present in *Evaluating the Productivity and Reproducibility of a Measurement Procedure* an empirical study that evaluates RmFFP, which is a new measurement procedure for the COSMIC-FFP functional size

measurement method. RmFFP is intended to be used on functional requirements specifications that are obtained using the Requirements Model of the OO-Method approach to object-oriented systems development. The RmFFP measurement procedure can be considered as an information system that takes data from the functional requirements specification as an input and produces a functional size value (similar to function points) as an information output. In their paper, Condori-Fernandez *et al.* describe the planning, operation and results of a quasi-experiment with computer science students in which the reproducibility of the RmFFP measurement results and the productivity of the measurers were assessed. The authors thus measured the quality of the RmFFP ‘system’ with the aim of evaluating the usefulness of their measurement procedure.

3 Conclusion

All papers reviewed in the previous section focus on quality aspects of information systems, either information (or (meta)-data) quality or system (or model) quality. In line with the D&M Model, in many papers information system quality is not considered the end-goal of the research, but a means to evaluate and/or achieve other system benefits, in particular the usefulness of the information system. In times of IT budget constraints (or even cuts) and fierce competition on a global market, the ability to demonstrate system benefits becomes even more important. Research on quality of information systems may contribute to the justification of system development costs and IT investments by identifying the most appropriate ways to use the scarce resources available with maximum returns in terms of quality and benefits.

We hope that you will find the papers accepted to QoIS 2006 interesting reading material and useful in the application of their results. We also hope that these papers may inspire you in your own research on quality aspects of information systems.

References

- [1] DeLone, W.H., McLean, E.R., 1992. Information System Success: The Quest for the Dependent Variable. *Information Systems Journal* 3(1), 60-95.
- [2] DeLone, W.H., McLean, E.R., 2003. The DeLone and McLean Model of Information System Success: A Ten-Year Update. *Journal of Management Information Systems* 19(4), 9-30.
- [3] Nelson, R.R., Todd, P.A., Wixom, B.H., 2005. Antecedents of Information and System Quality: An Empirical Examination Within the Context of Data Warehousing. *Journal of Management Information Systems* 21(4), 199-235.
- [4] Rai, A., Lang, S.S., Welker, R.B., 2002. Assessing the Validity of IS Success Models: An Empirical Test and Theoretical Analysis. *Information Systems Research* 13(1), 50-69.

Quality-Driven Automatic Transformation of Object-Oriented Navigational Models*

Cristina Cachero¹, Marcela Genero², Coral Calero², and Santiago Meliá¹

¹ Web Engineering and Databases Research Group. University of Alicante. Spain
{ccachero, smelia}@dlsi.ua.es

² ALARCOS Research Group. University of Castilla La Mancha. Spain
{Marcela.Genero, Coral.Calero}@uclm.es

Abstract. Navigability is a main concern in the design of Web applications. In order to assess such navigability a number of measures has been proposed. From them, measures defined on conceptual models are specially relevant, as it is well known that high quality conceptual models are critical to the success of the deployed system. However, measurement methods associated to such measures, as well as the design modifications that need to be performed on the models in order to improve their values, are usually tightly coupled with particular Web Engineering approaches. This fact compromises their effectiveness and their propagation capacity to different environments and/or methodologies.

Our aim in this paper is to illustrate how navigability measures can be captured in a general manner. In this way, not only is it possible to define a reusable set of relevant measures for a given family of applications, but also such measures can be queried in the context of MDA transformation rules. These rules capture both the measure decision criteria and the design modifications that should take place if the measure value for a given navigational model does not match such criteria.

1 Introduction

The ever increasing complexity of Web applications has caused the Web Engineering field to evolve at an extremely fast pace. This discipline intertwines sound Software Engineering principles with a suitable set of abstractions, particular to the idiosyncrasy of the Web. However, the inclusion in the different Web Engineering proposals of guiding principles that contribute to guarantee that the resulting applications comply with a set of desirable quality characteristics is still a challenge for the field. In fact, up to now no significant relationship has

* This paper has been supported by the Spain Ministry of Science and Technology, project numbers TIN2004-00779, TIN2005-25866-E and TIC2003-07804-C05-03. Also, this research is part of the DADASMECA project (GV05/220), financed by the Valencia Government and the DADS (PBC-05-012-2) and the DIMENSIONS (PBC-05-012-1) projects, financed by the Regional Science and Technology Ministry of Castilla-La Mancha (Spain).

been empirically demonstrated between the quality of the Web application and the fact that the designer has followed or not a given Web methodology.

In fact common practice in the field consists on the use of common-sense practices that are usually aimed at improving the interface usability. Although assessing quality at such a late stage of development can be useful, it is commonly avowed that early detection of problems in the artifacts produced in the initial phases of the life-cycle can save time, cut production costs, and raise the final desired product quality [5]. Aware of such potential benefits, the Web Engineering community has recently started to invest efforts in the inclusion of early measures to guide the construction of the different Web models, with special emphasis on domain and navigational models.

From them, domain models for Web applications hoard the highest number of early measure proposals [4]. Regarding navigational models, which are a distinctive feature of Web Engineering proposals, most efforts have been directed towards the promotion of a good use of the conceptual navigation constructs provided by each approach. For example, WebML defines a WebML Quality Analyzer [6] that is able to automatically check the XML specification of WebML conceptual schemas, and verify and measure some internal attributes, such as the consistency or completeness of the models. Similarly, UWE [3] pays special attention to usability, and recommends the integration of guidelines into both the design process and the CAWE tools that usually accompany Web proposals. Going one step further, the work of Abrahao *et al.* [1] proposes a set of measures that capture some available heuristics [9].

In fact, navigability is at the core of quality aspects such as usability or maintainability [1, 12, 2], as it is widely recognized that a good navigation design let users acquire the information they are seeking quickly and efficiently.

However, much work remains to be done. To our knowledge, all the proposed measures for Web models are tightly coupled with the Web Engineering approach for which they were defined. Unfortunately Web approaches usually greatly differ not only in notation but also in the semantics associated to the constructs, hampering the measure propagation to different environments and/or methodologies. Also, measures are usually defined and calculated in an ad-hoc manner, and the effects of the measure values on the models (that is, the actions the designer could perform in order to improve the model) are usually not specified.

Therefore, in this paper we aim at demonstrating how the definition of a navigational meta-model not only facilitates the understanding of the different Web approaches but also serves as a basis on which early measures for navigational models can be defined and calculated. For this purpose, in Section 2 we present a partial view of the OO-H navigational meta-model, as well as how navigability measures can be formally expressed over this meta-model by means of OCL expressions [14]. Also, we believe that the Web Engineering field can benefit from defining loosely coupled Web measuring models, that is, models that are applicable to a whole family of Web applications regardless of the chosen Web approach. Fortunately, we have at our disposal a Software Measurement Ontology (SMO) [7], and its corresponding meta-model that allows to derive concrete

measurement models. Our proposal in this sense is to instantiate a subset of this meta-model to express in an independent way a set of navigability measures. In Section 3 we illustrate this approach, and provide as an example a meta-model instantiation that reflects the navigability measure presented in Section 2. Last, we need a way to automate the evaluation/evolution of the navigational model depending on the chosen measures. For this purpose in Section 4 we use a QVT transformation rule [13] that, departing from the OO-H navigational meta-model and the SMO meta-model, generates a new OO-H navigational model that is checked against the quality decision criteria, which are also expressed in the meta-model instantiation. Section 5 concludes this paper with a summary of the main contributions and some future lines of research.

2 The OO-H Navigation Model

As we have stated above, it is well known that the improvement of the early artifacts produced in a development process has a great impact on the final product cost and quality [5]. Also, it is commonly avowed that, when developing Web applications, one of the most important early artifacts that must be produced is the navigational model. The Navigational model reflects the paths the user must follow through the information domain in order to achieve her goals. It is usually organized around navigational packages (also named contexts or targets in different approaches), each one encapsulating views defined over domain objects and paths to connect them. The names of the constructs, as well as their corresponding decorators, usually diverge from proposal to proposal, reflecting their different roots and degree of evolution. Due to this lack of a common Web conceptual ontology, it is highly recommendable for proposals to provide a navigational meta-model that systematically and unambiguously defines the concepts involved and their relationships, and facilitates in this way the communication among researchers and/or practitioners.

Such has been the approach taken by OO-H [8], a well known Web Engineering method whose navigational meta-model is partially presented in Fig. 1. Due to space reasons, we have intentionally left out details concerning service invocation, as well as the existing relationships with UML meta-classes.

In Fig. 1 we can observe how, as in most Web Engineering approaches, the main constructs of the OO-H Navigational Model are *Navigational Targets* (NT), *Navigational Classes* (NC), *Navigational Associations* (NA) and *Navigational Collections* (C). A Navigational Target is a packaging mechanism that serves to structure the navigation through the application subsystems. Navigational Classes are views over conceptual classes, and reflect the information that makes up the current view together with the operations that can be invoked from each view. Navigational Associations reflect navigation steps through the information. Last, Navigational Collections are access mechanisms (menus) that group together navigation paths.

To illustrate their use, let's imagine that we want to model a Ticket Sales system. A Navigation Model corresponding to this system is presented in Fig. 2.

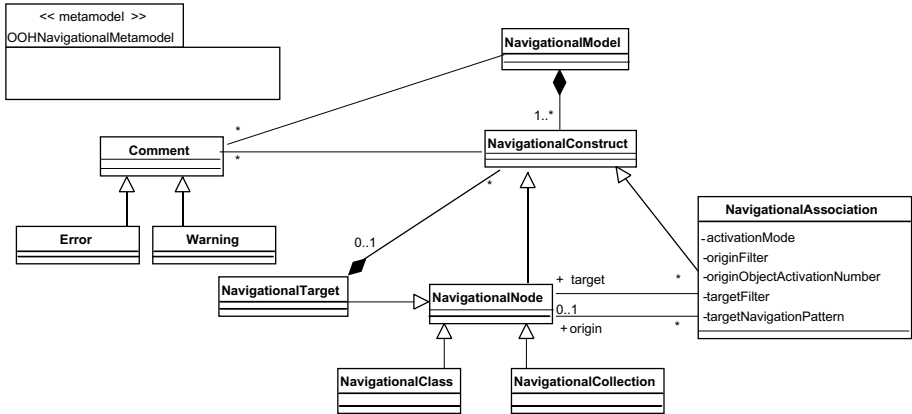


Fig. 1. Partial View of the OO-H navigational meta-model

The partial navigation model of Fig. 2 reflects a system whose navigation is organized around many NT (e.g. *User Registration*, *Shopping Cart*, *Show Details*, and so on), each one encapsulating the navigation paths needed to fulfill a subset of system requirements.

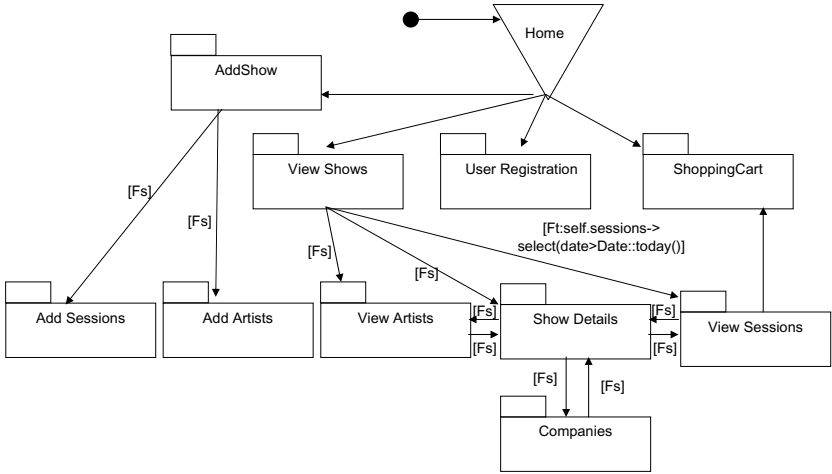


Fig. 2. OO-H Navigational Model corresponding to a Ticket Sales system

The different NT are accessed via a *Home* Collection that, depicted as an inverted triangle, represents the application main menu. The NT and the Home collection are connected by means of NA which may have one or more *Filters* associated. Filters are OCL expressions that constrain navigation. In OO-H, filters can be either user-defined or automatically generated, based on the domain

conceptual associations. In our example, the NA between *View Shows* and *View Artists* is an example of NA adorned with a predefined (structural) filter (Fs): the filter extracted from the relationship between shows and their corresponding artists. On the other hand, the filter *self.sessions->select(date>= Date::today())* that adorns the NA between *View Shows* and *View Sessions* is an example of target filter (Ft) that restricts the set of target objects to those sessions that have not yet taken place. In fact Fig. 2 represents one of the most evident mistakes novice web designers make when first designing Web applications using OO-H: they tend to define a new NT for each requirement, instead of performing a previous grouping task. This causes a poorly structured navigational tree and augments the model complexity. In fact, the impact of the number of NT on the model complexity has already been assessed in [1], through the Number of Navigational Contexts measure. For the sake of coherence with OO-H, in the remaining of the paper we have renamed this measure *Number of Navigational Targets* (NNT). Up to now, this measure could be defined in the context of the VisualWADE tool, the Computer Aided Web Engineering (CAWE) tool that supports OO-H, as follows:

```
context NavigationalModel
def NNT:Integer=self.navigationalConstruct->select(oclIsTypeOf
                                                    (NavigationalTarget))->size()
```

At this point we face a problem: if little work has been done on validating the impact of measures on navigability, much less has been done on assessing threshold values for such measures. However, for the sake of the example, let's assume that, for this kind of application, the famous Miller's 7+-2 rule [11] is applicable, that is, we do want a navigational model that neither is too scattered (more than nine NT) nor too compressed (less than four NT). This restriction may be expressed as follows:

```
context NavigationalModel
inv notTooManyNT: NNT<10
inv notTooFewNT: NNT>4
```

If we look back to Fig. 2 we can observe that the **notTooManyNT** invariant is not fulfilled, what in VisualWADE would cause a warning to be raised. Although this approach works well for prototypical development, it is tightly coupled with OO-H. This causes that OCL rules associated with the OO-H meta-model may need to be changed whenever we change the kind of application we are modelling (as measures that are relevant for a certain family of applications may not be applicable to others). Therefore we propose to go one step further, and provide a way to define measures that is independent from the chosen methodology and its corresponding meta-model, facilitating reusability.

Next we explain how we can achieve this goal.

3 Adaptation of the SMO to Navigational Models

The homogenization and systematization of the concepts that are relevant for a given domain is a goal that has been achieved through a number of ontologies,

3.1 Ontology-Aware Measuring Meta-model Instantiation

As the reader may have already inferred, the definition of the NNT measure (see Fig. 4) corresponds to the need to *assess the navigability of the Navigational Model*. In order to fulfill this **Information Need**, our measure is part of a **OO-H Quality Model** that gathers all the measures, decision criteria etc. that are applicable to any OO-H entity. This model is aimed at evaluating the *navigability* **Measurable Concept**. The **OO-H Quality Model** is made up of a set of **Attributes**, among which the *Navigational Complexity* is the one related with the NNT measure. All these concepts correspond to the Characterization and Objectives package that reflects the namesake sub-ontology that we presented in Fig. 3.

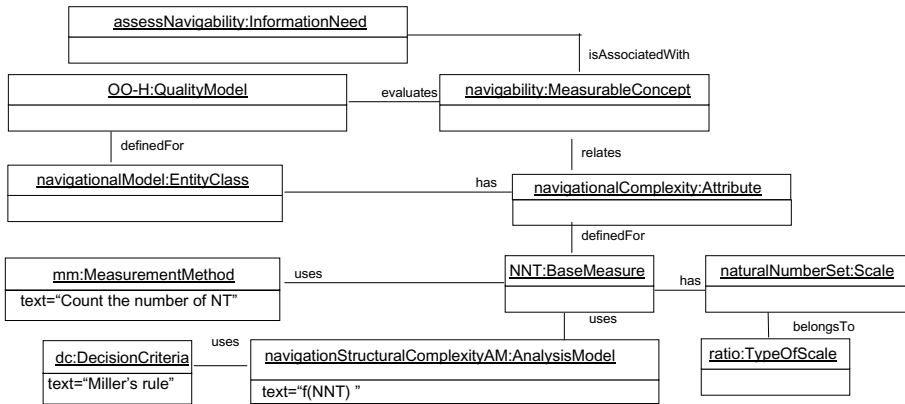


Fig. 4. Ontology-aware measuring model that reflects the NNT measure

Also, we need to express that the afore mentioned *Navigational Complexity* attribute is going to be measured, among others, with the aid of the **NNT Base measure**. The fact that the measure is of subtype Base implies that it does not depend on any other measure to calculate its value. Also, we want to express that the value of the NNT measure belongs to a *natural number* **Scale**, of **ratio Type of scale**. The **Measurement Unit** would be the *Navigational Target*.

Last, we need to express that the **Measurement Method** used to calculate the NNT measure consists on counting the number of NT that a given navigational view includes.

Also, we need to specify the **Analysis Model**, that includes a set of **Decision Criteria**. In our example, as we have just defined one measure, our Analysis Model assesses navigability just depending on the value of the NNT measure. The Decision Criteria on its turn establishes that a good navigability value has to comply with the Miller's rule, what implies that, in order for the model to be not too trivial nor too complex, the NNT must be between five and nine. Once the NNT measure has been defined, it is time to see how we can apply it to a given navigational model, such as the one presented in Fig. 2.

4 Automation of Measures

One of the main advantages of the measuring model presented in Fig. 4 is its capacity of reuse, as it does not assume any particular Web Engineering navigational meta-model. However, in order to be able to apply a given measure to a concrete navigational model, such connection needs to be established. This can be easily done if we regard a navigational model as a subtype of the concept **Entity Class**. This connection opens the path to the application of a Model-Driven Engineering approach [10] to automate the navigability assessment.

Back to our example, in Fig. 2 we can intuitively observe how the OO-H navigational model does not fulfill the rule of 7+-2 navigational targets. Therefore an expert designer would manually restructure the application to fulfill such rule. The automation of this process can be achieved by means of a set of transformation rules that, expressed in QVT [13], allows to encapsulate all the knowledge particular to a given Web Engineering approach (in our case OO-H).

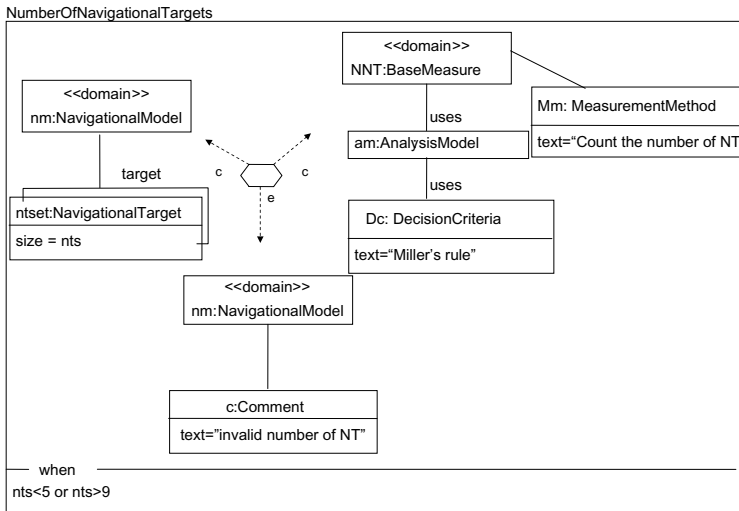


Fig. 5. QVT Transformation rule that checks OO-H navigational models for Miller's rule violations

Let's illustrate this approach by depicting a possible transformation rule that counts the number of NT for the navigational model of Fig. 2 and annotates the model if the decision criteria is not fulfilled (see Fig. 5).

In Fig. 5 the QVT graphical notation for the *NumberOfNavigationalTargets* relation is presented. This transformation rule involves two checkonly (c) domains: the *NavigationalModel* domain (root for the OO-H meta-model) and the *BaseMeasure* domain (defined in the context of the measuring meta-model). First, the transformation rule checks whether the NNT measure is relevant

for our navigational model, and whether Miller's decision criteria is applicable. This will be evaluated to true if the corresponding objects are present in the measurement model (meta-model instantiation) that we have previously defined (see Fig. 4). Then we must calculate the actual number of NT included in the navigational model under consideration. In OO-H this value can be established by simply counting the number of NT associated to our Navigational Model. The transformation rule stores this value in the *nts* variable. The adaptation of Miller's rule to the OO-H meta-model is established in the *when* clause of the transformation. This clause indicates, also in OCL-like syntax, that the transformation rule must be activated only if the number of NT is erroneous. In this case, the desired action is to enrich the OO-H model with a comment, associated with the whole model, that warns the designer about the violation of the rule. We specify such action on a third enforceable (e) domain (again the OO-H *NavigationalModel*). More transformation rules can be defined in this general way, making up a repository of measuring transformation rules.

We would like to stress the fact that the decision whether or not a particular rule is relevant for a given application will be taken once the measuring meta-model has been instantiated for such application. If the measuring structure reflected in the transformation rule is present in the meta-model instantiation, the rule will check whether the decision criteria is met, and will take any desired action if this is not the case. Although in our example the action has consisted on simply annotating the model, more sophisticated transformation rules could be defined to automatically generate a new model that does comply with the measuring criteria.

5 Conclusions and Further Work

This paper has presented a way to define a reusable measuring model that, based on the SMO meta-model, can be integrated into any particular Web Engineering approach. Also, it has demonstrated how the instantiation of this meta-model can participate in the quality assessment of particular navigational models, becoming a discriminator to decide whether or not a given QVT transformation rule, defined as part of a transformation rule repository, is applicable. Such transformation rules are the only elements that are aware of the specific Web Engineering meta-model that is being used, encapsulating in this way the specific knowledge. Besides, these transformation rules automate the measurement process, as well as the annotation/modification of the corresponding navigational models (if necessary) depending on the established measuring decision criteria. We would also like to stress the fact that the explicit consideration of ontologies (SMO) and standards (UML, OCL, QVT) whenever possible improves understandability and reusability of the approach. At this moment efforts are being made towards the definition of transformation rules that not only annotate but also modify in a sound way the OO-H navigational models. Also, intensive work is being performed on the OO-H CAWE tool to provide full support to this proposal.

Last, we would like to stress how, as soon as the Web Engineering community reaches an agreement regarding a common meta-model for Web application development, not only the meta-model instantiation but also the set of defined transformation rules will be able to be seamlessly reused among approaches.

References

- [1] S. Abrahao, N. Condory-Fernandez, L. Olsina, and O. Pastor. A Defining and Validating Metrics for Navigation Models. In *Proceedings of the 9th International Software Metrics Symposium*, pages 200–210, 2003.
- [2] J. Almer. Designing for Web Site Usability. *IEEE Computer*, 35(7):102–103, 2002.
- [3] R. Atterer, A. Schmidt, and H. Hussmann. Extending Web Engineering Models and Tools for Automatic Usability Validation. *Journal of Web Engineering*, 2005.
- [4] L. Baresi, S. Morasca, and P. Paolini. Estimating the Design Effort of Web Applications. In *Proceedings of the 9th International Software Metrics Symposium (METRICS'03)*. Springer, 2003.
- [5] L. Briand, S. Morasca, and V. R. Basili. Defining and Validating Measures for Object-Based High-Level Design. *IEEE Transactions on Software Engineering*, 25(5):722–743, 10 1999.
- [6] S. Comai, M. Matera, and A. Maurino. A Model and an XSL Framework for Analyzing the Quality of WebML Conceptual Schemas. In *Proceedings of the ER'02 International Workshop on Conceptual Modeling Quality*, pages 339 – 350. Springer, 10 2002.
- [7] F. García, M.F. Bertoa, C. Calero, A. Vallecillo, F. Ruiz, M. Piattini, and M. Genero. Towards a consistent terminology for software measurement. *Information and Software Technology*, pages 1–14, 07 2005.
- [8] J. Gómez, C. Cachero, and O. Pastor. Conceptual Modelling of Device-Independent Web Applications. *IEEE Multimedia Special Issue on Web Engineering*, 8(2):20–32, 04 2001.
- [9] M. Ivory. *Automated Web Site Evaluation*. Kluwer Academic Publishers, 2004.
- [10] S. Kent. The Expressive Power of UML-based Engineering. In *Proceedings of the 3rd International Conference on Integrated Formal Methods*, volume 2335, page 286. Springer, 06 2002.
- [11] G. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. In *The Psychological Review*, volume 63, pages 81–97, 1956.
- [12] J. Nielsen. *Designing Web Usability: The Practice of Simplicity*. New Riders, 2000.
- [13] Mof query/views/transformations final adopted specification. omg doc. ptc/05-11-01. www.omg.org/docs/ptc/05-11-01.pdf.
- [14] J. Warmer and A. Kleppe. *The Object Constraint Language. 2nd Edition. Getting your models ready for MDA*. Addison Wesley, 2003.

HIQM: A Methodology for Information Quality Monitoring, Measurement, and Improvement

Cinzia Cappiello, Paolo Ficiaro, and Barbara Pernici

Politecnico di Milano,
Dipartimento di Elettronica e Informazione
{cappiell, ficiaro, pernici}@elet.polimi.it

Abstract. Hybrid Information Quality Management (HIQM) methodology is conceived to be a support to solve run-time data quality problems. The analysis of the business processes and context in the design phase allows identifying critical points in the business tasks in which information quality might be improved. In these points, information quality blocks have to be inserted in order to continuously monitor the information flows. Through suitable checks, failures due to information quality problems can be detected. Furthermore, failures and warnings in service execution may depend on a wide variety of causes. Along the causes, the methodology also produces a list of the suitable recovery actions for a timely intervention and quality improvement. The methodology is explained by means of a running example.

1 Introduction

Nowadays, in the information era, data quality is a crucial requirement for obtaining a competitive advantage in modern enterprises. Actual business strategies focus on customers' satisfaction, or more in general, on stakeholders' satisfaction, considering it the key to obtain a leader position in the market and to maintain it over time. Stakeholders' satisfaction is usually reached by satisfying users' functional and non-functional requirements with personalized services offerings. Functional requirements are related to the specific execution of the service, while non functional requirements refer to the quality of service dimensions. Data quality is a considerable aspect of the quality of service and in the last few years its importance is constantly growing. This phenomenon is attributable to the development of the information and communication technology, the implementation of data warehouses and the direct usage of data by managers and customers. More than ever before, businesses, governments, and research organizations rely on the exchange and sharing of data.

In order to achieve a suitable data quality level, organizations should implement a complete data quality management program, that should support the four fundamental main phases: *data quality definition*, *measurement*, *analysis*, and *improvement*. In the data quality definition phase, the set of data quality dimensions to measure is identified. Then, after the measurement process, the

results of the evaluation are analyzed and poor quality data situations are detected and sent as input to the improvement phase. In the improvement phase, the collection of poor quality data cases is deeply investigated and an improvement action is suggested. The four phases are iterated in this order over time. In fact, the primary goal of data quality assurance is the continuous control of data values and, possibly, their improvement

In the literature different methodologies have been proposed in order to provide guidelines, and thus support data quality management. Most famous complete and general methodologies are the methodology developed at the MIT (Massachusetts Institute of Technology), called the Total Data Quality Management (TDQM) [10] and the TIQM (Total Information Quality Management) methodology [5]. These data quality programs in general require a considerable personalization effort before application. In fact, a big limit in the application of these methodologies is that the authors provide a series of guidelines without providing tools or specific algorithms that support users in the assessment, monitoring, and improvement phases.

The basis of the implementation of such programs should be a thorough knowledge of the business context and the quality of the data stored and used in the organizational processes. The analysis of the business processes and context allows identifying critical points in the business tasks in which information quality might be improved. In these points, information quality blocks have to be inserted in order to continuously monitor the information flows. In fact, run time recovery actions in data quality require the identification of the causes of data errors and their permanent elimination through an observation of the whole process where data are involved. Data tracking methods are required in order to determine the exact stage or steps in information process where the causes of data quality decreasing occur [2]. In the literature, it is possible to find several methods that allow the representation of a process and the associated information flow in order to detect errors and facilitate data quality improvement. An example is the IP-MAP methodology [8]. Once the information product has been selected and its quality specifications are well understood, in this phase it is possible to construct the Information Product Map (IP-MAP). The information product map graphically describes the process by which the information product is manufactured. There are different types of construct blocks that form the IP-MAP but the most important innovative feature is the introduction of “data quality block” that is used to represent the checks for data quality on those data items that are essential in producing a “defect-free” information product. The representation of the process by using IP-MAP enables the detection of different anomalies in data flows.

After the assessment process, the classical strategy of the data quality methodologies proposed in the literature is based on fixing a unique data quality goal. Data quality goal consists of a set of acceptable minimum values associated with the set of data quality dimensions chosen in the definition phase. Note that existing methodologies often use the Zero Defects approach by setting targets to highest quality values. This approach is sometimes excessive, since it would be

better to fix data quality objectives by considering the processes in which data are involved and the users' perspective.

In order to achieve the fixed data quality goal, data-oriented and process-oriented techniques are used in the improvement phase. The most straightforward solution suggests the adoption of data-oriented inspection and rework techniques, such as data bashing or data cleaning, to solve problems related to data accuracy and data consistency. A fundamental limitation of these techniques is that they do not prevent future errors, so they are considered appropriate only when data are not modified frequently [6]. On the other hand, a more frequent use of data bashing and data cleaning algorithms involves high costs that can be difficult to justify. To overcome these issues, several experts recommend the use of process-oriented methods. These methods allow the identification of the causes of data errors and their permanent elimination through changes in data access and update activities. These methods are more appropriate when data are frequently created and modified.

However, the improvement activities in case of poor quality data are defined as results of a periodical data quality analysis. In the literature, methodologies that manage the run time errors detection and correction are missing. In this paper, the Hybrid Information Quality Management (HIQM) methodology is proposed. This methodology is a first attempt to provide a methodological approach at the run time error detection and correction management. In fact, detection of errors in the run-time phase of the process is supported and suitable improvement actions are enforced. Finally, HIQM also introduces some changes in the traditional data quality management cycle. In fact, the concept of user perspective is introduced. In the data quality definition phase, the main stakeholders of the business processes are identified and the set of relevant dimensions is defined for each class of stakeholders. The importance and the target associated with each quality dimension by each users' class define the user perspective. This concept is used to analyze the data quality problem under the suitable point of view.

This paper is organized as follows: Section 2 describes the methodology from a general point of view, by illustrating the phases that compose it and focusing on innovative aspects as regards the literature. Section 3 focuses on the Warning Management module and Section 4 provides the description of the tool that has been implemented to support the HIQM methodology.

2 HIQM: A Methodology for Quality Management

The HIQM methodology is a complete methodology for IQ management and improvement and it is composed of eight steps (see Figure 1).

Considering the classical four phases of the data quality management program discussed in Section 1, it is possible to identify the following mappings: the *IQ Environment Analysis*, *Resource Management*, and *Quality Requirements Definition* phases are all referable to the *Definition* phase; *Quality Measurement* and *Analysis & Monitoring* phases have the same functions of the classical approach, while the *Improvement* phase in the HIQM methodology includes the classical improvement functions and the *Strategy Correction* phase. The following sections

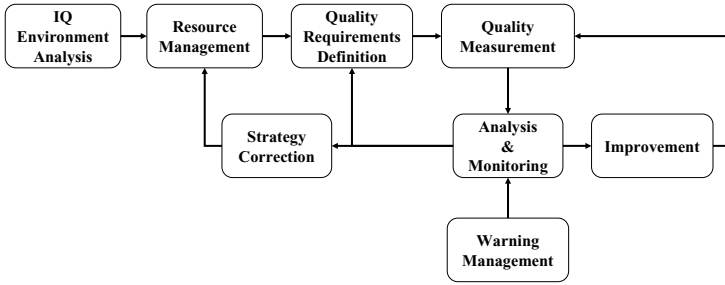


Fig. 1. HIQM methodology

provide details about each phase. The Warning phase is an innovative contribution of the HIQM methodology for the realtime warning management and it is described in Section 3.

2.1 Data Quality Definition Phase

As stated in Section 1, data quality program should be based on a thorough knowledge of business context and organizational data and processes. In the HIQM methodology, this knowledge is acquired in the *IQ Environment Analysis* and *Resource Management* steps. The former step is composed of different tasks in which data sources, processes, and stakeholders of the organizations are identified, and strategic issues are considered. In fact, in this phase, there is the analysis of the feasibility of the implementation of the data quality management program and thus the definition of the management plan together with the identification of potential critical issues, risks critical success factors, and performance key indexes. The management plan is the input for the *Resource Management* step in which there is the definition of all the resources that have to be involved in the data quality program. Thus, starting from the business processes that have to be monitored, it is possible to identify in this phase the data sources, the applications and the human resources that are related to each business process. The *Data Quality Requirements definition* step is the most important part of the Definition phase since it is based on the concept of the user perspective. HIQM is different from the other methodologies proposed in the literature since it considers the different information stakeholders and models their preferences and needs. In an organization, different types of users access and manipulate data. We can identify three principal categories:

- *Enterprise consumers*: users that work in the organization and access data. Users belonging to all the different roles in the organizational pyramid are considered.
- *Supplier consumers*: users and organizations that collaborate with the organizations or cooperate in common activities in reaching a common goal.
- *User-end consumers*: final users that are interested to the quality of data. This category includes both customers that are interested to the value that

quality can add to the purchased product or service and users that receive a generic output of the business processes.

Clearly, each users' category is characterized by different needs and thus different requirements as regards the set of relevant data quality dimensions and the associated minimum acceptable value. In order to formally model the users' perspective and its impact on the organization, HIQM combines the Fuzzy Linguistic Approach that is a technique for Multiple Attribute Decision Making and the Analytical Hierarchy Process method [7]. The proposed approach is composed of the following steps: (i) Definition of the criteria (i.e. quality dimensions) that have to be evaluated and related relevance weights along the user perspective and the process objectives; (ii) Evaluation and comparison of the different alternatives (i.e. sets defined by the different users) (iii) Ranking of the different alternatives in relation to a specific business process.

The utilization of Fuzzy numbers is useful in phase (i) for the identification of relevance that quality dimensions have for the different users. In fact, users are not able to provide a numerical evaluation of the importance of a data quality dimension. Fuzzy numbers provide the logical-mathematical instrumentation that combines the capabilities of the natural language with the advantages that characterize the algebraic formalization and numerical representation [3]. In this way, the importance of the dimensions is established by the users that express their opinion by using natural language expressions, e.g. very low, low, more or less low, medium. HIQM uses the conversion scales introduced in [4] to convert the linguistic expressions in Fuzzy numbers. Note that the conversion scales also consider the risk aversion and propensity of the users. At the end of this process we obtain a decisional matrix D in which each element d_{ij} represents the relevance of the i -th quality dimension along the j -th user perspective.

In a parallel way, for each business process, with the Analytical Hierarchy Process method, a hierarchy of data quality dimensions is defined in order to identify the role of the quality dimensions in the success of the business process. The weight w_{ik} identifies the relevance of the i -th quality dimension in the success of the k -th business process. Finally, the involvement of the user classes in the different business process has to be considered. The coefficient r_{jk} has to be identified in order to express the participation of the j -th user class in the k -th business process. At the end of the Quality Requirements Definition step, considering a k -th business process, the evaluation of a j -th user perspective (E_{jk}) is calculated as following:

$$E_{jk} = r_{jk} \cdot \sum_{i=1}^I d_{ij} \cdot w_{ik}$$

For each business process, the maximum among the total scores (i.e. $Max(E_{jk})$) identifies the j -th user perspective that has to be considered to evaluate the quality of data in the k -th process. The identification of the most important user perspective for each business process defines the list of data quality dimensions that have to be assessed, analyzed and improved along the specified requirements in the last three phases referred as the Evaluation phases.

2.2 Data Quality Evaluation Phases

Data Quality Evaluation phases include the *Measurement*, *Analysis*, and *Improvement* phases. The HIQM approach here is similar to the existing methodologies one. In the *Measurement* phase, for each quality dimensions, a measurement algorithm has to be identified. The results of the evaluation are sent as input to the *Analysis* phase that compares the values associated with each quality dimension to the quality requirements. If the requirements are not satisfied, this phase is in charge to find out the causes of the poor quality and along these causes, suitable improvement actions have to be identified. As discussed in Section 1 both data-oriented and process-oriented techniques are considered in the *Improvement* phase. In critical situations, improvement actions also imply modifications at a strategic level, that is changes in data quality planning system and requirements. This structural changes are planned and performed in the *Strategy Correction* step.

3 HIQM Data Quality Monitoring and Recovery Support

The *Warning* phase represents an innovative contribution of the HIQM methodology. In fact, no other methodologies have proposed an interactive analysis of warning and failure messages to assess and improve data quality. Essentially, the warning management represents an internal system for real-time data and processes monitoring. As regards the Warning phase, it is managed by different components as represented in Figure 2.

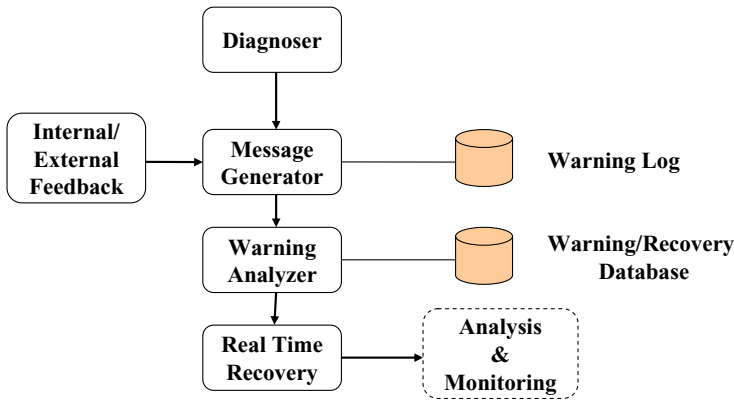


Fig. 2. Architecture of the Warning management

The *Message Generator* is the core module of the architecture and it works on the basis of the messages sent by the *Diagnoser* and the *Internal/External Feedback* modules. These modules are in charge to monitor the system and detect all the anomalies that occur in data management. In details, the *Diagnoser* represents the most important component of the Warning Management system. It

identifies and manages fault and warning events by monitoring the internal system and external context. After the warning identification, *Diagnoser* sends the information to the *Message Generator* that produces the warning message that is processed by the *Warning Analyzer*. Three different monitoring submodules supports the *Diagnoser* in faults detection:

- *System Discrepancy Analyzer*: this module detects all the discrepancies that occur between internal data and external data that are contained in external sources (e.g. Web data, suppliers' data);
- *System Inconsistency Analyzer*: this module focuses on quality dimensions values. In particular, it identifies inconsistencies between actual data quality values and requirements;
- *System Monitoring*: this module monitors the system and its behavior in order to detect anomalies in data management such as outlier detection or peak values in data accesses. These anomalies have to be analyzed since they may be the indicators for data quality problems.

Data quality problems can be also identified on the basis of received feedbacks. Feedbacks can be classified as (i) *External feedbacks* that are generated by external applications or sources that interact or cooperate with the organization platform; (ii) *Internal feedbacks* that are generated by the internal system; (iii) *Subjective feedbacks* that are based on surveys of the stakeholders' satisfaction. The *Internal/External Feedback* module receives the different types of feedbacks, analyzes them and selects the ones that are the indicators of significant data quality problems. Once the fault has been detected, a warning message is created, stored in the *Warning Log Database*, and sent to the *Warning Analyzer*. This module, on the basis of the rules contained in the *Warning/Recovery Database*, identifies the most suitable recovery action for the detected problem. In fact, HIQM defines a priori rules for problem solving and recovery selection in order to make the resolution of data quality issues easier and automated. Thus, *Warning/Recovery Database* contains all the associations between possible poor data quality situation and related recovery actions. Associations are based on considered data, data quality dimensions, access frequencies and so on. For example, if the problem regards the accuracy of a data value that is scarcely accessed, the system will only suggest the immediate correction of the problem; while if data value is frequently accessed, besides the correction, a process-oriented method is required. Finally, *Real Time Recovery Module* applies the appropriate recovery actions and monitors their impact to the system during implementation and execution. Note that each time that a warning message arrives and recovery actions are performed, the system updates data quality values and traditional monitoring operations are performed.

4 An Example: Bookshop Web Service

In this chapter, an example of cooperative business process is described in order to explain how the methodology works. Figure 3 describes the architecture

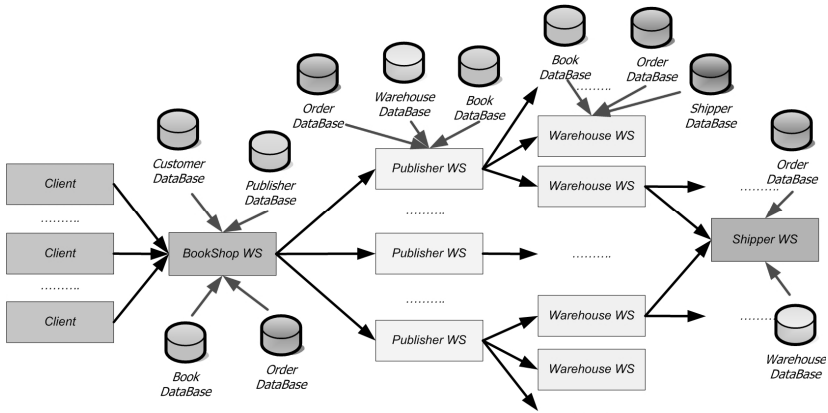


Fig. 3. The bookshop example

underlying a sample process composed by Web services managed by different organizations **BookShop WS**, **Publisher WS**, **Warehouse WS** and **Shipping WS** and that aims at supporting a customer in buying a book. According to the diagram presented in the figure, a **customer** asks for a book invoking the **BookShop WS**, and submitting the title of the book.

We suppose the bookshop owns a set of books catalogs obtained by the associated publishers. So, after receiving the customer request, the **BookShop WS** retrieves from these catalogs the most suitable book edition with respect to the title, and sends the order request invoking the Web service of the selected publisher, sending order details containing the ISBN of the book, and the set of customer information relevant for the **Publisher WS** (e.g., the user's address). In the same way, once the order from the **BookShop WS** arrives, the **Publisher WS** identifies the warehouse where the book is available and then selects the one (i.e., **Warehouse WS**) closest to the customer. Each warehouse usually relies on a set of shippers and, once an order arrives, the most suitable one (i.e., **Shipper WS**) is selected according to the final destination. Notice that, once the book is received by the customer, the shipper sends a notification directly to the publisher in which the reference to the concluded order is included. After receiving the notification, the bookshop sends the final receipt to the customer.

In the proposed example, the goal of the set of interacting services is to send the right book to the user according to his preferences. A number of faults may occur during the execution of the process illustrated above, which may cause the wrong book to be delivered [1], a book to be indefinitely reserved for a user who is not going to buy it, an order to be indefinitely delayed, and one or more of the involved services to be indefinitely running waiting for a reply.

In this example, the most important dimensions are accuracy, completeness, consistency, and timeliness. Accuracy, completeness, and consistency assess data

along their numerical extension and correctness [6,9]. Timeliness evaluates the validity of data along time [2]. These dimensions are objective dimensions and, therefore, are suitable for a quantitative evaluation and constitute a minimal set that provides sufficient information to evaluate the data quality level. Along the presented data quality dimensions, it is possible to classify faults along two categories: value mismatch and missing data. The value mismatch can be derived by:

- Typos: e.g., Jhon instead of John
- Different format: e.g., date expressed as dd/mm/yyyy instead of mm/dd/yyyy
- Conflict in data values: e.g., Residence city: London Country: Italy
- Delay in update operations

Typos are related to data accuracy while conflicts in data values are related to data consistency. A case of different format can be related to both accuracy, because the value is not a right representation of real-world value, and to representation consistency. Delay in update operations between two databases that contain the same values can be related to both timeliness, since the out-of-date value is not valid anymore, and accuracy since that the out-of-date value is a incorrect representation of the real world value. Missing data can be caused by value unavailability or by a delay in update operations. The former is related to the completeness dimension. Delay in update operations between two databases that contain the same values can be related to the timeliness dimension but in this case the update operation would create new tuples in a database.

Run time recovery actions in data quality can be performed using different data-oriented methods:

- *Data cleaning by manual identification*: comparison between value stored in the database and value in the real world;
- *Data bashing (or Multiple sources identification)*: comparison of the values stored in different databases;
- *Cleaning using Data edits*: automatic procedures that verify that data representation satisfies specific requirements;

The Warning log database can be used to adopt process-oriented methods and to establish which part of the process has to be modified in order to eliminate the causes of poor quality.

The bookshop example has been used as the basis of the development of an application that supports the HIQM methodology. The developed tool supports both the Quality Requirements Definition phase and the Warning phase. As regards the Quality Requirements definition phase, a series of interfaces have been designed to set the users preferences and to calculate and show the most important user perspective. Figure 4 shows the interface through which users assign the degree of relevance to the different quality dimensions along their perspective.

Figure 5 contains the interface through which it is possible to define the hierarchy of data quality dimensions and thus the role that each quality dimension has in the success of the business process. Once that the system has acquired all

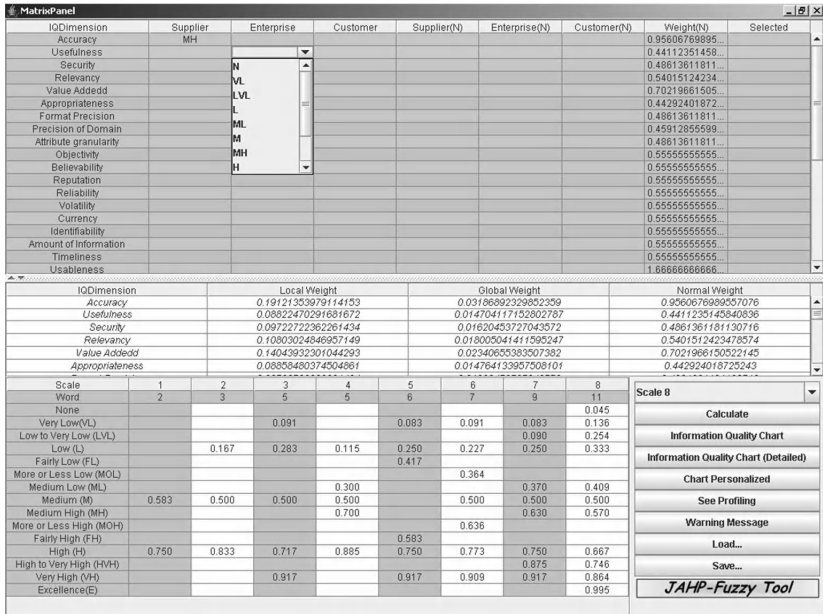


Fig. 4. The Fuzzy Interface



Fig. 5. The Analytical Hierarchy Process Interface

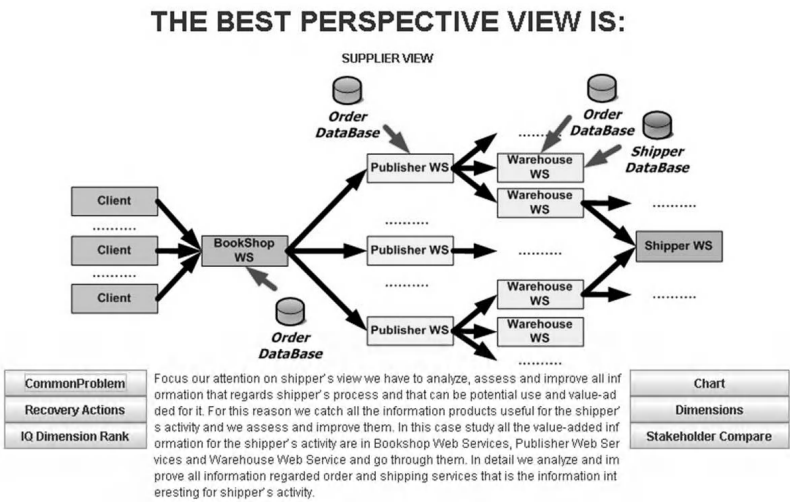


Fig. 6. Identification of the best user perspective

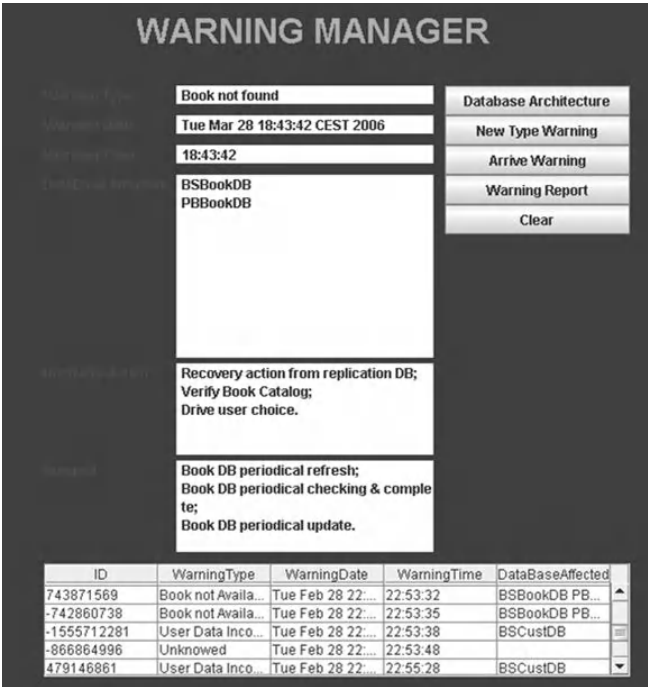


Fig. 7. The Warning module interface

the required parameters, it performs the calculations described in Section 2.1 and displays the relevant user perspective (see Figure 6).

The Warning Interface (see Figure 7) represents a valid approach to analyze and manage problems and warnings occurred on information system.

Obviously this software and in particularity this interface is not supposed to be a complete tool to support warning management but it is a first attempt to automatically manage warning and alert message in order to improve data quality.

This part of tools use the Warning log and the Warning/Recovery database to support warning analysis, recovery action selection and historical problem management. The Warning Interface performs the following activities: (i) the process monitoring and warning generation and analysis and (ii) the management of the databases with insertion, update and delete operations.

As regards the activity (i) the tool manages warning reception and analysis. For each received warning message it determines the data involved, time and date of warning, recovery actions suggested by considering both data-oriented and process-oriented methods.

5 Conclusions and Future Work

The paper proposes an approach for data quality management that combines a traditional warning management of data quality problem on an information systems. A demonstration tool is used to support HIQM methodology, warning analysis and recovery implementation.

Future work will focus on the definition of automatic reasoning techniques that use the Warning Log database for the automatic classification of warnings and the association of the most suitable recovery action.

Acknowledgements

This work has been partially supported by the European Project Ws-Diamond.

References

1. L. Ardissono, L. Console, A. Goy, G. Petrone, C. Picardi, M. Segnan, and D. Theiseider Dupre, *Advanced fault analysis in web service composition*, Proceedings of the International World Wide Web Conference, Chiba, Japan, poster session, 2005, pp. 1090–1091.
2. D.P. Ballou, R.Y. Wang, H.L. Pazer, and G.K. Tayi, *Modelling information manufacturing systems to determine information product quality*, Management Science **44** (1998), no. 4, 462–533.
3. C.H. Chen, *Fuzzy logic and neural network handbook*, McGraw Hill, 1996.
4. S.J. Chen and C.L. Hwang, *Fuzzy multiple attribute decision making: Methods and applications*, Springer-Verlag, 1992.
5. L.P. English, *Improving data warehouse and business information quality*, John Wiley & Sons, 1999.
6. T.C. Redman, *Data quality for the information age*, Artech House, 1996.

7. T.L. Saaty, *The analytical hierarchy process*, McGraw Hill, 1980.
8. G. Shankaranarayan, R. Y. Wang, and M. Ziad, *Modeling the manufacture of an information product with ip-map*, Proceedings of the 6th International Conference on Information Quality, 2000.
9. Y. Wand and R.Y. Wang, *Anchoring data quality dimensions in ontological foundations*, Communication of the ACM **39** (1996), no. 11.
10. R.Y. Wang, *A product perspective on total data quality management*, Communications of the ACM **41** (1998), no. 2.

Evaluating the Productivity and Reproducibility of a Measurement Procedure*

Nelly Condori-Fernández and Óscar Pastor

Department of Information Systems and Computation
Valencia University of Technology
Camino de Vera, s/n, Valencia, Spain
{nelly, opastor}@dsic.upv.es

Abstract. This paper reports an empirical study that used computer science major students as experimental subjects to evaluate the productivity and the reproducibility of RmFFP. This is a functional size measurement procedure designed according to the COSMIC-FFP method for object-oriented systems that are specified using the OO-Method approach. The results show that the productivity of RmFFP is acceptable when compared to other procedures found in the literature. Furthermore, RmFFP produces reproducible functional size assessments.

Keywords: Measurement procedure, COSMIC-FFP, Functional Size, experiment.

1 Introduction

Nowadays, Functional Size Measurement (FSM) plays an important role in software project management due to its extensive use in industry to monitor progress and performance, determine overall productivity, better manage software portfolios, assist in planning, etc. However, despite the fact that measurement method evaluation is crucial in ensuring high-quality size measures, very few of the proposed functional size measures have been evaluated in an experimental way [1], [2].

A new FSM method can be evaluated using the ISO/IEC 14143-3 standard [3], where performance properties (e.g. reproducibility, repeatability, accuracy, convertibility) are given.

In the last few years a number of proposals that measure object-oriented systems based on the COSMIC-FFP standard method have been proposed, such as Bévo et al. [4], Jenner [5], Poels [6], Diab et al. [7], Nagano et al. [8], Azzouz et al. [9], and Habela et al. [10]. However we have found very few empirical studies on the performance properties of COSMIC-FFP. With respect to the productivity of COSMIC-FFP, Nagano carried out an initial analysis to evaluate whether COSMIC-FFP is ease to use [1]. Diab evaluated the accuracy and repeatability of COSMIC-FFP in the measurement of real-time systems by means of the automation of the proposed procedure [7].

* Supported by the Ministry of Science and Technology under the DESTINO project, ref. TIN2004-03534.

This paper focuses on the evaluation of a functional size measurement procedure for object-oriented systems called RmFFP [11] that was designed by mapping the concepts of the COSMIC-FFP method [12] onto the primitives of the OO-Method approach [13]. OO-Method is an automatic software production method that follows the model-driven architecture paradigm. The evaluation of RmFFP was carried out by means of an empirical study that investigates the productivity and reproducibility of RmFFP in estimating functional size from requirement specifications. Productivity is the quantity of size units that can be measured per unit of time. Reproducibility is the closeness of the agreement between results of measurements of the same measurand carried out under changed conditions of measurement [3].

This paper is organized as follows: Section 2 presents an overview of the RmFFP measurement procedure. Section 3 describes the empirical study used to evaluate RmFFP in terms of productivity and reproducibility. Section 4 discusses the analysis and interpretation of the results. Finally, Section 5 sets out our conclusions and indicates further work to be carried out.

2 A Size Measurement Procedure

The RmFFP procedure [11] was proposed in order to estimate the functional size of object-oriented systems from functional requirements specifications obtained using the Requirements Model [14]. This model includes the *Functions Refinement Tree* (FRT), which is a hierarchical decomposition of the business functions of the system. The leaves of this tree represent the functions of the desired system and are the entry point to be considered as *Primary Use Cases*. *Secondary Use Cases* are also possible, which are important for organizing and managing complexity through relationships among use cases stereotyped as EXTEND and INCLUDE. The *Sequence Diagrams* are built semi-automatically from use cases. A Sequence Diagram is represented by means of a set of messages between the required classes to perform the system behaviour. These messages are labelled with different stereotypes (signal, service, query, connect), which allows subsequent identification of the different elements of the OO-Method Conceptual Schema.

RmFFP starts with the definition of the measurement context, which includes purpose, scope, and measurement viewpoint. The scope of RmFFP comprises the functionality to be included in a particular measurement. The measurement viewpoint corresponds to the ‘end-user’ viewpoint, which will focus on an OO-Method requirements specification.

At this point, RmFFP starts a mapping phase to identify the significant primitives of the Requirements Model that contribute to the system’s functional size according to the concepts of the COSMIC-FFP metamodel. For this purpose we defined sixteen mapping rules whose principal purpose is to reduce misinterpretation of COSMIC-FFP generic concepts and facilitate automating of the RmFFP procedure. For instance, each use case is identified as a functional process, each message of the sequence diagram is identified as a data movement type, etc.

As the data movement is the fundamental component of the COSMIC-FFP method, we also defined four additional rules for eliminating duplicated data movements, which are explained with more detail in [15].

Once the data movements have been correctly identified, we proceed with the measurement phase, whose purpose is to produce a quantitative value that represents the functional size of software of a requirements specification. To do this we apply the measurement function, which consists of assigning a numerical value of 1 Cfsu (Cosmic Functional Size Unit) to each data movement. We defined four rules to add these quantified data movements, considering the relationships type between use cases in order to calculate the size of a (use case) functional process and the size of the entire system.

3 Evaluation of the Application of RmFFP

In this section, we describe an empirical study carried out to evaluate the application of RmFFP in terms of productivity and reproducibility. In designing the experiment we used the experimental process provided by Wohlin et al. [16].

3.1 Experiment Planning

In order to define the goal of our empirical study we used the Goal/Question/Metric (GQM) template [17], which is described as follows:

“To analyze RmFFP for the purpose of evaluating its productivity and reproducibility from the viewpoint of the researcher in the context of Computer Science students measuring OO-Method requirements specifications”.

Two research questions were addressed by this empirical study, the first being whether RmFFP is efficient, and the second being whether RmFFP is reproducible.

Selection of subjects. The subjects were computer science students at the Valencia University of Technology with similar backgrounds in the use of the OO-Method Requirements Model. These subjects were students enrolled in the “Software Development Environments” course from February to June of 2005. The experiment was organized as a mandatory part of this course. Two groups of students were formed because some students could not regularly attend class due to work commitments in companies. The first group was made up of 18 students who had no links to companies (e.g. work experience) and the second group was made up of 17 students who had some connection with companies.

Selection of variables. The independent variable is the variable for which the effects should be evaluated. In our study, this variable corresponds to the FSM procedure and as single treatment: RmFFP.

The dependent variables selected to evaluate RmFFP are as follows:

- *Measurement productivity*: this is obtained by calculating the number of size units, Cfsu, that can be measured per unit of time (e.g. per hour). The time recorded was the time required to apply the mapping rules and measurement rules of RmFFP.
- *Reproducibility* this is obtained by calculating the extent of variability existing in the measures obtained by different subjects and the same measurand.

Hypotheses formulation. The following hypotheses regarding the research questions were defined:

- Hypothesis 1: RmFFP is efficient when compared to reports found in the literature.
- Hypothesis 2: The functional size measures are reproducible applying RmFFP under changed conditions of measurement.

Experimental Tasks. Two groups of experimental tasks were carried out during the training task and measurement task.

The purpose of the *training* was for the subjects to develop the expertise required to measure using RmFFP. To do this, we carried out a training procedure with both groups. The training method used was demonstration/practice [18].

For the *demonstration part*, we considered the following tasks: (a) Presentation of the OO-Method Requirements Model, (b) use of the RETO tool (that supports the OO-Method Requirements Model), (c) presentation of the RmFFP measurement procedure, and (d) illustration of the use of RmFFP with an illustrative example of a case study.

For the *practice part*, we considered the following tasks: (a) Application of the theory presented in the case study and guided by the instructor, (b) guided application of RmFFP to a case study (the students could clarify their doubts), and (c) verification of knowledge learned by the student by working out an assigned case study. The time used for all the tasks included in this first session was eight hours distributed over four days.

With respect to the *measurement task*, each subject used the RmFFP guide to measure an OO-Method requirements specification (rent a car). This task was used to collect functional size and measurement time.

Before the subjects took the test, the experiment was conducted with another small group of people in order to improve it and ensure that the documentation was well designed. No changes were necessary as a result of this pre-test.

Instrumentation. The instrumentation used in this experiment included the experimental object and training materials. The experimental object was an OO-Method requirements specification of the Car Rental application. The functional specifications of this case study were developed by the students; however, for this experiment, we used the specification proposed by the course teacher. The training materials¹ were the following: a set of instructional slides on the OO-Method Requirements Model and the RmFFP procedure; a case-study that describes an example of the application of RmFFP, a measurement guide, and another case study to verify the training carried out.

3.2 Experimentation Operation

Execution. The experiment took place in a classroom. The interaction among subjects was controlled to avoid plagiarism. There was no time limit set for measuring the OO-Method requirements specification. We also allowed the use of the material used in the training sections.

¹ <http://www.dsic.upv.es/relly/materials>

Data Recording and Validation. The data recorded by the students was the functional size of the specification given and the time used to carry out this measurement. Once the data were collected, we verified whether the results were valid. We noted that two students made serious mistakes during the application of RmFFP (outliers). For instance, they confused the concepts of INCLUDE and EXTEND (relationships between use cases) for the application of the aggregation functions. Therefore, we did not take into account these two tests in the analysis of the results.

4 Analysis and Interpretation

The analysis and interpretation of the results are divided in four phases according to the research questions previously stated.

4.1 Analysis of the Actual Productivity

To calculate the measurement productivity of a subject, we divided the subject assessment by the measurement time. The obtained productivity rate for each subject measuring the experimental object is shown in Figure 1. These values oscillate between 90-190Cfsu/hour; the average measurement productivity was 131.48 Cfsu/hour.

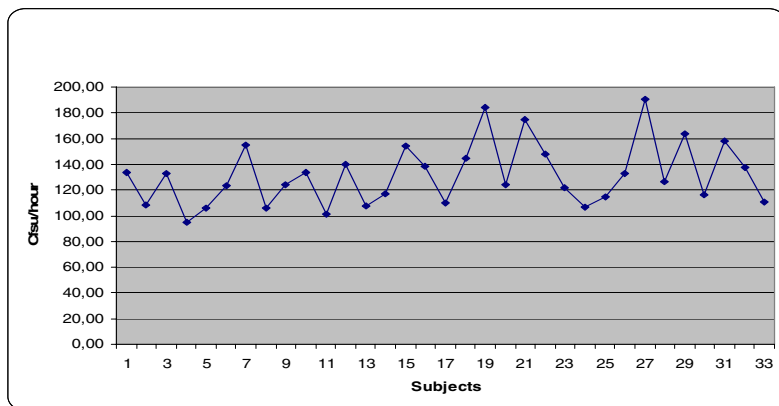


Fig. 1. Rate measurement productivity for each subject using RmFFP procedure

To evaluate the productivity of RmFFP², we considered the analysis carried out by Nagano about measurement productivity³ applying COSMIC-FFP to 13 requirements specifications [1]. The average measurement productivity obtained was 45 Cfsu/hour. This value was about three times less compared to the Productivity of RmFFP. However, several factors could affect this result, such as:

- The functional specification provided to the subjects in order to size the system requirements. We used the OO-Method Requirements Model, which is based on UML notation with some stereotypes. Nagano used the natural language for the functional specification of the switching systems.

² Nagano used the term speed for productivity.

- The mapping rules defined specifically for the OO-Method context that allowed the reduction of the generality of COSMIC-FFP. Nagano applied directly the generic rules of COSMIC-FFP.
- The subjects were well-versed in the OO-Method Requirements Model and the RmFFP measurement procedure.

We also considered the different counting levels for Function Point Analysis (FPA) published by the company Total Metrics [19]. According to these levels, the productivity of an estimator can vary between 200-750 FP/day. As a day is assumed to have 8 working hours, the obtained productivity rate is approximately between 25-93.75 FP/hour, which will depend on the experience of the measurer.

Given that the correlation between function points and Cfsu is linear, and convertibility factor between FPA [20] and COSMIC-FFP is close to one [12], the obtained productivity using RmFFP (131.48 Cfsu/hour) was also higher than the rate reported by Total Metrics. However, we cannot yet draw definite conclusions about the measurement productivity obtained with RmFFP due to external conditions such as the complexity of the experimental object (functional specification), level of experience, etc.

4.2 Analysis of the Actual Reproducibility

To measure the degree of variation between assessments produced by different subjects using RmFFP, we used a practical statistical equation similar to the one proposed by Kemerer [21]. This equation is calculated by taking the difference in absolute value between the size value produced by a subject i and the average value produced by the other $n-1$ subjects in the sample, divided by this average value. The scores (REP_i) closest to zero indicate least variability in the measurement or most reproducibility. These scores were thus obtained for each observation by applying the following equation:

$$REP_i = \left| \frac{\sum_{k=1, k \neq i}^n \frac{Values_k}{n-1} - Value_i}{\sum_{k=1, k \neq i}^n \frac{Values_k}{n-1}} \right| \quad (1)$$

These REP_i values obtained for each subject are presented in the Appendix. The mean of variability (REP_i) for thirty three observations was 5.1 % (See Table 1). This value was compared to the result reported by Kemerer (26.53%) in a study about IFPUG FPA reliability [21], where the reproducibility obtained using RmFFP was higher.

Table 1. Descriptive statistics for RmFFP reproducibility

Statistics	Reproducibility
Mean	0.051
Standard deviation	0.04164
Min	0.000
Max	0.15

To evaluate the actual reproducibility of RmFFP, we checked the normality of the score obtained (REP_i). To do this, we used the Shapiro-Wilk test. This test was applied to our data and we found that they were normal.

Next, we tested the hypothesis (H2). To do this, we used the t-student statistic since we were working with normal data; we considered the value zero as reference value to compare with the scores assigned by the subjects.

The results of the t-test, shown in Table 2, allow for the rejection of the null hypothesis because the value t obtained was outside the interval. Furthermore, the level of significance obtained was very high ($p < 0.001$). Therefore, we can claim with 95% confidence that the data obtained would satisfy the hypothesis that RmFFP is reproducible.

Table 2. One Sample t-test for the reproducibility

Statistic	Reproducibility
Mean Difference	0.05091
95% Conf. Interval for the diff.	0.0361 (lower) 0.0657 (upper)
t	7.024
1-tailed p-value	.000

4.3 Validity of Results

It is important to ensure that the experimental results are valid for the target population. In this section, we discuss the more important threats [16] related to conclusion validity, internal validity, and external validity of our empirical study.

Conclusion validity. Threats to conclusion validity are concerned with issues that affect the ability to draw the correct conclusion about relations between the treatment and the outcome of the experiment. The following threats were considered:

- Reliability of the application of treatments to subjects. There is a risk that the application is not similar for different persons applying the treatment on different occasions. In our experiment, RmFFP (treatment) was applied following a prescribed procedure for the two defined groups. Hence, the risk of obtaining dissimilar applications for different subjects and occasions was low.
- Random heterogeneity of subjects. All the subjects in the experiment had approximately the same level of experience working with the OO-Method Requirements Model. However, this homogeneity reduces the external validity of the experiment.

Internal validity. Threats to internal validity are influences that can affect the independent variable with respect to causality, without the researcher’s knowledge. The following threats were considered:

- Selection. Depending on how the subjects are selected from a larger group, selection effects can vary. In the experiment, the subjects were selected for convenience, i.e., they were students enrolled in the “Software Development Environments” course. This course was selected because it was a specialized

teaching unit. Furthermore, the students had the necessary preparation and training, and the experimental task itself fitted well into the scope of this course.

- **Instrumentation.** This is the effect caused by the artefacts used for experimentation execution. The instruments used for the experiment were verified. First, the specifications of the case studies (objects) were reviewed by an expert of the OO-Method Requirements Model. Second, the measurement guide (guideline) and the survey were verified in advance with a small group of people in order to improve its understandability.

External validity. It is concerned with generalization of the results to industrial practice. Here the following threats were considered:

- **Interaction of selection and treatment.** This is the effect of not having a representative population in the experiment with which to generalize. In our case, we accept that more experiments with a larger number of subjects (e.g., professionals) will be necessary.
- **Interaction of setting and treatment.** This is the effect of not having representative material. In the experiment, we tried to use a representative OO-Method requirement specification of a real case in the MIS functional domain. This specification differs to some extent from “classical” specifications because RmFFP uses sequence diagrams with stereotyped messages as artefacts. However, we believe that RmFFP could also be used in UML sequence diagrams on a manual basis.

5 Conclusions and Future Works

This paper describes an empirical study that evaluates the productivity and reproducibility of RmFFP for estimating the functional size of object-oriented systems from software requirements specifications.

With respect to the efficiency analysis, the data collected indicate that the productivity of the subjects using RmFFP is several times higher than the productivity rate obtained by Nagano [1]. This way, we confirm that a measurement method due to its generic character is least efficient than a measurement procedure, such as RmFFP. We also found that the RmFFP productivity is higher than the industry rates found with IFPUG FPA, which were reported by the company Total Metrics [19].

With respect to the reproducibility analysis, we have corroborated that users of RmFFP produce reproducible assessments. This result can be explained by the training carried out with the subjects. Moreover, the complementary rules defined to control the duplicity of data movements allowed to reduce some interpretation problems of the RmFFP guidelines that could appear during the measurement procedure application.

Finally, we are aware that it is necessary carry out more empirical studies with industry professionals in order to confirm our initial results. In addition, our measurement procedure is currently being automated to be incorporated in the RETO tool, which captures user requirements and generates elements of an OO-Method conceptual schema.

References

1. Nagano, S.; Mase, K., Watanabe Y., Watahiki T., Nishiyama S., Validation of Application Results of COSMIC-FFP, in Australian Software Conference on Measurement (ASCOM), Australia, 2001.
2. Abrahao S., Poels G., Pastor O. A Functional Size Measurement Method for Object-Oriented Conceptual Schemas: Design and Evaluation Issues. Software & System Modelling, Volume 5, Issue 1, Springer Verlag, 2005.
3. ISO, "ISO/IEC 14143-3: Information technology - Functional size measurement - Part 3: Verification of functional size measurement methods", 2003.
4. Bévo V., Lévesque G., and Abran A. UML Notation for Functional Size Measurement Method. In Proc. 9th International Workshop on Software Measurement, Canada, September 8-10, 1999, pp. 230-242.
5. Jenner M.S., "Automation of Counting of Functional Size Using COSMIC-FFP in UML," Proc. 12th Int'l Workshop Soft. Measurement, pp. 43-51, 2002.
6. Poels G., 2003. Functional Size Measurement of Multi-Layer Object-Oriented Conceptual Models, Working Paper, Gent University.
7. Diab H., Koukane F., Frappier M., St-Denis R., 2004, µROSE: Automated Measurement of COSMIC-FFP for Rational Rose Real Time, Information and Software Technology, Volume 47, Issue 3, 1 March 2005, Pages 151-166.
8. Nagano S., Ajisaka T., Functional metrics using COSMIC-FFP for object-oriented real-time systems, in 13th International Workshop on Software Measurement (IWSM), Montreal, Canada, 2003.
9. Azzouz S., Abran A., "A Proposed Measurement Role in the Rational Unified Process and its Implementation with ISO 19761: COSMIC-FFP" in Software Measurement European Forum, Rome, Italy, 2004.
10. Habela P., Glowacki E., Serafinski T., Adapting Use Case Model for COSMIC-FFP based Measurement, in the 15th International Workshop on Software Measurement, Montreal, Canada, Shaker-Verlag, 2005.
11. Condori-Fernández N., Abrahao S., Pastor O., Towards a Functional Size Measure for Object-Oriented Systems from Requirements Specifications Abstract Functional Size Measurement. IEEE Quality Software International Conference, Germany, 2004.
12. Measurement Manual: The COSMIC Implementation Guide for ISO/IEC 19761: 2003, version 2.2.
13. Pastor O., Gomez J., Insfran E., Pelechano V., 2001. "The OO-Method approach for information systems modelling: from object-oriented conceptual modelling to automated programming", Information Systems 26, pp. 507-534.
14. Insfran E., Pastor O. and Wieringa R., 2002. Requirements Engineering-Based Conceptual Modelling. Journal Requirements Engineering (RE), Springer-Verlag, pp. 61-72.
15. Condori-Fernández N., Abrahao S., Pastor O., The Problem of Data Movements Duplicity in a Measurement Procedure, In Proc. of 9th Iberoamerican Workshop on Requirements Engineering and Software Environments, La Plata-Argentina, 2006 (in Spanish).
16. Wohlin C., Runeson P., Höst M., M. C. Ohlsson, B. Regnell, and A. Wesslén, Experimentation in Software Engineering: An Introduction, 2000.
17. Basili V. R. and Rombach H. D., 1988. The TAME Project: Towards Improvement-Oriented Software Environments, IEEE Transactions on Software Engineering, pp. 758-773.
18. DOE Handbook, Alternative Systematic Approaches to Training, 1074-95, January 1995.
19. Total Metrics, "Levels of Counting", Australia, August 2001.

20. IFPUG, 1999. Function Point Counting Practices Manual, Release 4.1, International Function Points Users Group, Mequon, Wisconsin, USA.
21. Kemerer C. F., "Reliability of Function Points Measurement", Communications of the ACM, vol. 36, pp. 85-97, 1993.

Appendix

Table A1. Data set used in the experiment

Subject	Size (Cfsu)	Measurement time (Hour)	Productivity (Cfsu/hour)	REP
1	144	1,08	133,33	0,11
2	160	1,48	108,11	0,01
3	153	1,15	133,04	0,05
4	142	1,50	94,67	0,12
5	159	1,50	106,00	0,01
6	154	1,25	123,20	0,05
7	155	1,00	155,00	0,04
8	154	1,45	106,21	0,05
9	149	1,20	124,17	0,08
10	156	1,17	133,33	0,03
11	160	1,58	101,27	0,01
12	168	1,20	140,00	0,04
13	161	1,50	107,33	0,00
14	181	1,55	116,77	0,13
15	162	1,05	154,29	0,01
16	159	1,15	138,26	0,01
17	176	1,60	110,00	0,10
18	166	1,15	144,35	0,03
19	184	1,00	184,00	0,15
20	180	1,45	124,14	0,12
21	175	1,00	175,00	0,09
22	148	1,00	148,00	0,08
23	152	1,25	121,60	0,06
24	160	1,50	106,67	0,01
25	149	1,30	114,62	0,08
26	166	1,25	132,80	0,03
27	158	0,83	190,36	0,02
28	168	1,33	126,32	0,04
29	164	1,00	164,00	0,02
30	168	1,45	115,86	0,04
31	166	1,05	158,10	0,03
32	161	1,17	137,61	0,00
33	157	1,42	110,56	0,03

Quality of Material Master Data and Its Effect on the Usefulness of Distributed ERP Systems

Gerhard F. Knolmayer¹ and Michael Röthlin²

¹ Institute of Information Systems, University of Bern, CH-3012 Bern, Switzerland

² Department of Engineering and Information Technology,
Bern University of Applied Sciences, CH-2501 Biel/Bienne, Switzerland
gerhard.knolmayer@iwi.unibe.ch, michael.roethlin@bfh.ch

Abstract. Master data is a main component of most information systems. In distributed and heterogeneous organizations, problems of data quality may arise if several Enterprise Resource Planning (ERP) systems, customized with respect to local business needs and objectives, use subsets of common master data. In this paper we describe data management issues in a large organization, running 10 instances of the SAP R/3 system. For coordinating purposes, common elements of materials master data are entered via a centralized application and subsequently distributed to the affected instances. However, this master data management approach did not avoid massive data quality problems, which are, for instance, hampering the computation of informative key performance values and the effective realization of inventory reduction programs. The paper discusses possible approaches for improving data quality in this situation and in other cases of distributed ERP systems.

Keywords: Data Quality, Information Quality, Material Master Data, Enterprise Resource Planning Systems, Master Data Management.

1 Introduction

For most business activities, information systems (IS) are key enablers for operations and decision making. Considered as the central component of today's IS landscape, ERP systems provide organizations with large application functionality, supporting a major part of all business activities. Their design promises to solve the problem of fragmented information in large organizations [7], bringing together various types of data from very different business activities such as sales, human resources, or finance in a consistent model of the business.

Large companies typically do not operate a single ERP instance (as often assumed in the early literature) but several instances, each being configured more or less differently [13]. As a consequence, need has emerged to coordinate data between different information systems – not only between ERP, CAD, PDM, CRM, and SCM systems, but also between different instances of ERP systems.

Many business transactions rely on the availability of master data, representing, for instance, customers, suppliers, or products. Master data is typically created once and re-used many times, and does not change too frequently. Its distribution is often controlled by a process which should ensure that all data is entered and approved with

respect to business rules, and that every user and every system should receive new or updated master data as soon as needed.

Master data capture and processing are error-prone activities. Inappropriate IS architectures, insufficient coordination with business processes, inadequate software implementations, or inattentive user behavior may lead to disparate master data. For instance, if an employee does not find required master data within reasonable time, he will be tempted to swiftly create duplicate data, in order to continue his business tasks. Mistyped names, addresses, and other data fields contribute to data redundancy as well. As a consequence, deficiencies in master data may affect an organization at operational, tactical, and strategic levels [22].

The remainder of the paper is organized as follows. In Section 2, we refer to the most closely related publications about data quality in ERP systems. Section 3 describes a case study about information systems in a large corporate group, focusing on data quality issues in logistics. Section 4 discusses options for better managing ERP master data and, consequently, improving data quality by focusing on people, processes, and technology. Section 5 puts the situation described in the case study in a broader master data management context.

2 Background and Related Research

The discussion of data quality aspects in information systems is almost as old as information technology (IT) itself, considering the causes and the impact of “dirty data” on operations and decision-making [4]. A recent survey on data quality mentions that more than 80 percent of the respondents still feel that improved information would allow them to make better operating decisions and reduce costs [6]. In another survey on ERP environments, IT managers argued that state-of-the-art database and middleware technologies do not guarantee high-quality data [23].

In this paper we focus on the quality of master data. Many statements from practitioners’ perspectives stress the importance of master data management (MDM) and the quality issues related to it (e.g., [8, 25, 29]). Problems have been described for many classes of master data. Galway and Hanks [10] analyzed information management in logistics of the U.S. Army. They report numerous consequences arising from definition and usage problems related to product identification codes. The main reason for missing consensus on basic definitions and data processing practices is seen in a “deep divide” between organizational units and the fragmented responsibility for logistics information systems. Cross-border organizational issues can be the very source of information quality problems. Organizational fragmentation is a key driver for data quality problems [10], besides the often cited aspects of software design and operational deficiencies [26]. Kirsche et al. [15] describe problems of harmonizing product and sales data obtained from retailers; search algorithms are used to generate suggestions for identifying products and storing the associated data in a reference database.

Gattiker and Goodhue [11] discussed the adoption of ERP systems in a distributed organization, consisting of 20 units. In a first attempt to implement ERP, much autonomy was given to the units in configuring the ERP instances. This strategy failed as the project ran quickly out of budget and it became clear that differences

between the implementations made it almost impossible to obtain meaningful data for group management and controlling tasks. In a second effort, management enforced standard settings for all ERP instances. This approach worked well for most units, but not at all for a major part of one unit, characterized by very specific manufacturing and order management processes.

ERP systems have been marketed as a panacea: If the inadequately coordinated legacy systems were replaced by an ERP system, most data quality problems should disappear. A powerful argument was that master data has to be entered only once and can be reused in different contexts, on a system-wide basis. However, errors made during master data capture may proliferate through the system and also affect the quality of transactional data in the database of the ERP system [28]. Decision making processes based on this data may be adversely affected [2].

3 Case Study

3.1 The Organization

The corporate group described in this case has a long-standing tradition of industrial excellence. Its 13 operating units (below referred to as “divisions”) carry out various tasks in procuring, maintaining, and disposing complex high-tech systems that cause considerable expenditures. The authors conducted more than a dozen interviews with employees from all major divisions, including senior managers, controllers, material managers, process owners, IT specialists, and public relations managers. The interviews were mostly held with groups of employees, lasting typically about two hours.

In the following we focus on the main task of this group, i.e. providing logistics services to external and internal customers. From the group perspective, logistics is understood as a set of distributed, cross-organizational tasks, in which many players contribute to overall success. A Logistics Services Core Unit (LSCU) has been installed for ensuring efficient logistics and economically sound management of materials handling. This includes tasks like planning, central coordination, and controlling of logistical processes, the definition of policies for inventory and supplier management, the organization of educational programs for all employees concerned with logistical tasks, and coordinating the use of the group's logistics IS. Data Warehouse (DWH) technology for consolidating logistics data stemming from divisional ERP and other information systems is applied to generate reports for divisional and group management.

A comprehensive process handbook was developed for harmonizing and formalizing logistical procedures, defining activities, roles, and responsibilities. All activities are related to a high-level reference model, covering the life cycle from the abstract definition of a technical system to its withdrawal and disposal. An item can be used (e.g., purchased, stored, sold or moved) by several divisions. It sometimes happens that items are defined (in terms of product specification, commercial terms, or quantities) in one, purchased by a second, stored by a third, transformed in a fourth, and maintained by a fifth division. The process documentation describes such distributed activities on a conceptual level, in the group's business language. Responsibility for all aspects of an item has been assigned to an “item manager”. This person is, among others, responsible for capturing and maintaining all data, including material master data, over the life cycle of the components of technical systems.

3.2 Evolution of the ERP Landscape and Material Master Coordination

ERP systems and their MRP II predecessors have been used in the group since about two decades. In the mid 1990s, more than 60 fragmented legacy applications existed, which communicated via batch interfaces. By the end of the 1990s, mainly under the pressure of the “Year 2000” threat [16], the key legacy applications had been migrated and consolidated to ten SAP R/3 system instances. The process handbook was used as central guideline for implementing the systems. However, divisions were allowed to implement, adapt, and extend the R/3 system to their individual business needs and to handle implementation projects with their own resources, consultancies, and methodologies.

The need for coordinating several instances in distributed information systems is well known and has led, in the case of SAP R/3, to concepts for data exchange such as Application Link Enabling (ALE) [1]. Based on this technology, LSCU developed a Material Master Coordination System (MMCS) as a special SAP R/3 instance, ensuring the centralized capturing and “one-to-many” distribution of master data values to the divisional ERP instances, where they are completed and adapted to divisional needs. MMCS can be regarded as an early representative of MDM tools. Similar approaches of centralized creation and maintenance of master data have been described recently [14, 19].

LSCU also created a comprehensive MMCS documentation which specifies, in particular, the (division-specific) domains of material master data attributes. Today, over 1.5 million records of material master data exist in the MMCS.

3.3 Overall Evaluation of the ERP System

The comprehensive functionality of ERP systems employed in the divisions supports the activities of logisticians in the different units. Broad process support is offered; almost all activities described by the process framework are covered by the software. Low-level policies have been established for keeping the data accurate enough for operational work with the ERP systems.

However, the current ERP system landscape causes several problems that will be analyzed in more detail below:

1. Data records stemming from different SAP R/3 instances cannot be consolidated without further logic, as they have been generated by differently customized systems. Thus, consolidated views on a group level are difficult to obtain.
2. Transaction data exported from divisional SAP R/3 systems to the DWH generally suffers from low quality: Out-dated, duplicate, and inaccurate records complicate reporting. The reports generated by LSCU in the DWH are not in line with divisional reports, and managers do not trust LSCU's figures.
3. Lacking part lists make it impossible to use state-of-the-art materials requirements planning algorithms of ERP systems. This may result in sub-optimal procurement and inventory management processes.

3.3.1 Division Specific Customization of ERP Instances

Logistics is typically seen as a cross-sectional activity. In smaller businesses, ERP technology supporting logistics elegantly bridges the organizational gaps between

units as only one ERP instance exists. In the case of the group described, however, cross-divisional processes have to be mapped explicitly to different ERP instances. Common master data elements created in the MMCS have to be completed to cope with divisional structures and inventory policies. As divisions customized the ERP systems to their specific needs, the parameters of the ERP instances and the selection of transactions to be used are not consistent. For instance, one division uses paired “goods entry”/“goods issue” transactions for maintenance operations, showing the maintained item temporarily in “maintenance stock,” whereas others do not. In such cases it is difficult to determine the overall stock available. As soon as consolidated reports are needed on a group level, master data as well as transactional data have to be reconciled using additional logic, manpower, and guesswork.

3.3.2 Availability and Quality of Master Data

Due to the omnipresent time pressure in ERP implementation projects, data from the legacy systems was transferred to the SAP systems without too much cleansing and supplementing activities. Because of the long life span of many technical systems this data is still relevant in various tasks and analyses.

Many users feel that the quality of data provided by their “own” ERP instance is sufficient for their operational work, even if they recognize it is not perfect – in accordance with the “high tolerance for errors in primary systems” found by Olson [20]. On the other hand, managers and IT users often mistrust data received from other units. Credibility of controlling reports is often maligned, since stock values reported by LSCU are incommensurable with inventory statements obtained from the divisional ERP systems. Determining present values of inventories is even more critical: Prices are only updated for a minority of (recently purchased) items, making a value-based consolidation of inventories difficult. Furthermore, many items are not offered any longer and, thus, market prices do not exist for them or are hard to determine.

Item data like weight or volume has not been entered, because no immediate needs existed. However, in a current relocation project this data was highly relevant and had to be supplemented retroactively. The same happened with environmental data, which was neglected in earlier times but became relevant recently.

3.3.3 Missing Bill of Materials Information

Manufacturing companies typically use bills of materials (BOM) to specify relationships between parts for design, production, and maintenance processes. In the case of the analyzed group, BOM data is available only in some ERP instances and for some items. Traditionally assembly information was kept for documentation purposes in maintenance manuals. One reason why some divisions are reluctant to the use of BOM is that the technical systems purchased sometimes consist of ten thousands of parts. For the maintenance of these systems, BOM would be very helpful, and some suppliers offer to sell BOM data of their systems. However, life cycle aspects are not always given high priority in the purchasing process and the purchasing unit may primarily see the additional effort of transferring the data and adjusting it to internal standards and views.

As structures between the parts of some physical systems are not formally known to the ERP system, material requirements planning algorithms based on BOM explosion cannot be applied. In this case, future demand is forecast roughly by analyzing the time series of the demand values. Missing BOM data may also result in problems

if the supplier does not offer subassemblies for maintenance any longer, due to the long life span of some technical systems. In this case, maintenance shops have to re-design these subassemblies and to reinvent their BOM data.

3.4 Review of the Material Master Coordination Process

As mentioned above, MMCS is implemented as a special SAP R/3 instance. No organization specific assistance is provided for data entering, as the rules and individual domains documented in the handbook have not been made available to MMCS and the ERP instances. Users may enter data values that are forbidden by the handbook and no division specific input validation process is implemented. There is also no support for avoiding the entry of duplicate master data.

Drawbacks of ERP standard user interfaces, such as an overload of data fields and unexpected reactions to certain entries, have been asserted [13, 21]. However, corporate IT management is interested to have users working with the standard GUI instead of adapting it to user needs, to reduce implementation and maintenance costs.

Often only data needed for daily operations is entered, focusing on immediate needs, and acting in the myopic interest of individuals or the division, without taking the goals of the group into account. Data management responsibilities of item managers occasionally conflict with short-term needs, for instance in a time-critical purchasing process: If purchased items are needed urgently, they may be stored or even be used before the associated master data has been fully defined, leading to differences between physical stock and inventory reports.

3.5 Impact of Data Quality Problems and Reengineering Strategy

For corporate management, the situation of today's ERP landscape seems intolerable. From a business viewpoint, the group suffers from a low transparency of operations and the subsequent mistrust between the managers of different divisions. The problems arising in target setting and scrutinizing the achievement of goals even affected negatively the relationships with external stakeholders. A lack of accepted management instruments tampers high-level initiatives such as the implementation of a uniform group cost accounting model or a massive inventory reduction program. From an IT perspective, maintaining a multitude of ERP instances is complex and expensive, since, in addition to license fees, each division still employs its own team of consultants and internal software specialists. Apparently, risks of misusing company resources exist, and corporate governance is at stake.

The group decided to implement SAP's industry solution for a better support of new, unified processes, to be adopted in a big bang conversion by 2008. However, due to the heterogeneity of tasks and legal and organizational idiosyncrasies, a reduction to one single ERP instance is still not achievable. Therefore, some type of MDM solution will be necessary also after reengineering the ERP environment.

4 Approaches for Improving Material Master Data Quality

The organization described above faces a severe data quality problem: Adopting ERP technology was apparently not sufficient to avoid disparate data. ERP users commit

errors when entering and using data, without being warned by the system. Process guidelines exist but are implemented differently or not at all in the ERP systems. Technology assisting ERP users supports the basic business transactions, but is not able to provide consolidated information and decision support.

Problem solving efforts can be based on data quality procedure models [9, 18, 20, 22] and have to take people, processes, and technology [21] into account.

People. With regard to users of ERP systems, management should make sure that the mutual relationship between business processes and data becomes evident to each and every user. Caring for data and for data quality should not be considered as ancillary elements of the “really” important business activities. The goal of improving data quality should be a part of the performance standards and incentives for providing good data quality should be defined. In pursuing this goal, metrics should be defined and monitoring tools for data quality and adequate system use should be applied. As formulated in [21], all employees need to understand that they share responsibility for the data they generate and that they will be held accountable for its quality.

Processes. In contrast to promises of vendors and consultants, ERP systems require the design and enforcement of formal processes for managing master data, in order to ensure long-term data quality [3]. If a business unit uses one process variant in a consistent way, it may be quite satisfied with its data base. However, if different units apply different system options, the data aggregation process may be considerably hampered. At the operational level ERP systems offer different ways of mapping real-world processes into information systems. For instance, backflushing decisions [21] related to work in progress may differ and the shipment of parts to a subcontractor may be booked differently. Similar ambiguities may exist in connection with defective inventory, returned material, and in-transit inventory [5]. Processes and the corresponding requirements should be defined and system transactions be implemented with respect to the goal of improving the quality of aggregate inventory data. Management should limit the number of ERP system transactions allowed, refine business process guidelines down to the transaction level, ban missing or dummy values in mandatory data elements, and install coordinated and automated checks for enforcement of guidelines.

On the organizational level, the value of information should be critically considered, especially at interfaces between organizational units. Often costs for improving data quality originate in one division and benefits of using the data are reaped by another division. Galway and Hanks [10] suggest that in such cases the costs for creating and maintaining high-quality data should be charged to the organizational unit which benefits from the data. In addition, a formal data management function (“data stewardship”) could help to improve data quality – “when you manage data you manage data quality” [24]. Specific management instruments such as data quality scorecards [18] or information quality report cards [9] for visualizing data quality may also be applied.

Technology. An obvious recommendation is to reduce the number of ERP instances to enforce standardization of business processes and lower license, maintenance, and upgrade costs. Many software features originally developed as add-ons now belong to the ERP core functionality or an associated “industry solution.” However, major differences in business processes may legitimate the use of several ERP

instances and ERP consolidation projects are as risky, costly, and time-consuming as other ERP migrations - and often profoundly disliked by many ERP users [17].

ERP system vendors have recognized the reality of multi-instance installations and the widespread need for master data consolidation and offer MDM functionality fitting to their product suites. For instance, SAP offers an MDM solution which includes master data storage and forwarding across different systems and locations, decentralized augmentation of data elements with local context information, and portal-based input screens [27]. However, similar to ERP technology, MDM solutions will not be a panacea for solving all data quality issues: Reworking an existing base of master data will be quite difficult in an integrated IS context and organizational barriers cannot be removed by a simple adoption of technology. The business value of costly MDM projects is not easily computed and communicated [12], although the "high costs of low-quality data" have been described in much detail [9].

The group could also consider improving its master data gathering process by incorporating (division specific) rules in the existing MMCS concept. In addition, commercially available tools for locating incomplete documents and fixing broken processes could be implemented in order to maintain an acceptable quality level of ERP data.

5 Conclusion

In the organization described in the case study, almost every division implemented its "own" ERP system instance, and configured and adapted it to best support divisional business processes. Despite guidelines issued by corporate management and the IT department, today's ERP landscape suffers from several weaknesses. The existing legacy databases had not been sufficiently cleansed before migrating them to the SAP systems and BOM information was not entered for relevant technical systems, preventing the use of material requirements planning algorithms and resulting in the reinvention of missing data. No formal data management functions had been installed, leaving the evolution of the ERP instances and of their master data uncontrolled.

Aggregate transaction data does not provide a consistent view of business reality, and reports generated by corporate controlling are not in line with divisional figures, creating mistrust and driving division managers to decline measures based on aggregate data. Although many ERP users consider data quality of the whole ERP environment as insufficient, they are often quite satisfied with their local ERP systems. Users normally accept data stemming from their own division or from a "trusted" co-worker of another division. For corporate management, however, the situation of today's ERP landscape is clearly intolerable, as it tampers the implementation of strategic initiatives.

Overall, a master data process must be put in place which is in accordance with corporate goals, respects structural constraints, and assists ERP users to fulfill their data management tasks. As shown in the case study, even a detailed common process handbook does not ensure consistent settings in distributed ERP systems, especially if customizing per instance is driven by distinct implementation forces. Requirements from the corporate level, calling, for instance, for aggregated information about overall inventories, must be available for the local implementation processes. MDM solutions are a technical remedy in some cases, but cannot substitute a careful design

of IS architecture and processes as well as user awareness and commitment for data management issues.

Maintaining information quality demands a continuous effort, in which each organization using ERP systems has to define and apply structures, processes, and tools to foster a consistent commitment. This is especially true if ERP data is to be used beyond divisional boundaries, in post-merger activities, or is exchanged beyond the borders of the organization with business partners in a supply chain.

References

1. Angeli, A., Streit, U., Gonfalonieri, R.: *The SAP R/3 Guide to EDI and Interfaces*. 2nd ed., Vieweg, Braunschweig/Wiesbaden (2001)
2. Ballou, D.P., Pazer, H.L.: Modeling Data and Process Quality in Multi-Input, Multi-Output Information Systems. *Management Science* 31 (1985) 2, 150–162
3. Brackett, M.H.: *Data Resource Quality. Turning Bad Habits into Good Practices*. Addison-Wesley, Boston et al. (2000)
4. Brodie, M.L.: Data Quality in Information Systems. *Information and Management* 3 (1980) 6, 245–258
5. Broeckelmann, R.G.: *Inventory Classification Innovation*. St. Lucie Press/APICS, Boca Raton/Falls Church (1999)
6. CFO Research Services/Deloitte Consulting: *IQ Matters: Senior Finance and IT Executives Seek to Boost Information Quality*. CFO Publishing, Boston (2005) http://www.deloitte.com/dtt/cda/doc/content/Final%20Deliverable_IQ%20Matters.pdf
7. Davenport, T.H.: Putting the Enterprise into the Enterprise System. *Harvard Business Review* 76 (1998) 4, 121–131
8. Druker, D., Rich, R.: Master Data Management. *DB2 Magazine* 10 (2005) 3, 33–36
9. English, L.P.: *Improving Data Warehouse and Business Information Quality*. Wiley, New York (1999)
10. Galway, L.A., Hanks, C.H.: *Data Quality Problems in Army Logistics: Classification, Examples, and Solutions*. RAND, Santa Monica (1996)
11. Gattiker, T.F., Goodhue, D.L.: Understanding the local-level costs and benefits of ERP through organizational information processing theory. *Information & Management* 41 (2004) 4, 431–443
12. Griffin, J.: Overcoming Challenges to Master Data Management Implementation. *DM Review Magazine*, April 2006. <http://www.dmreview.com/portals/portalarticle.cfm?articleId=1051113&topicId=1031394>
13. Hayler, A.: Are You Master of Your Data or Its Slave? *DM Direct Newsletter*, March 4, 2005. http://www.dmreview.com/article_sub.cfm?articleID=1020858
14. Khanduja, D.: Master Data Management: Three Approaches to Consolidating Master Data. *DM Direct Newsletter*, October 14, 2005. <http://www.dmreview.com/portals/portalarticle.cfm?articleId=1038907&topicId=1031394>
15. Kirsche, T., Baumann, G., Schanzenberger, A.: Alignment of Product Master Data. In: Cremers et al. (eds.): *INFORMATIK 2005: Informatik LIVE!, Proceedings of the 35th Annual Conference of the Gesellschaft für Informatik*, Vol. 2, Bonn 2005, 449–453
16. Klaus, H., Rosemann, M., Gable, G.G.: What is ERP? *Information Systems Frontiers* 2 (2000) 2, 141–162
17. Kremers, M., van Dissel, H.: ERP System Migrations. *Comm. ACM* 43 (2000) 4, 53–56

18. Loshin, D.: Enterprise Knowledge Management – The Data Quality Approach. Morgan Kaufmann, San Diego et al. (2001)
19. McKnight, W.: Justifying and Implementing Master Data Management. DM Review Magazine, April 2006. http://www.dmreview.com/article_sub.cfm?articleID=1051165
20. Olson, J.E.: Data Quality – The Accuracy Dimension. Elsevier, Amsterdam et al. (2003)
21. Piasecki, D.J.: Inventory Accuracy: People, Processes, & Technology. OPS Publishing, Kenosha (2003)
22. Redman, T.C.: Data Quality for the Information Age. Artech House, Norwood et al. (1996)
23. Röthlin, M.: An Exploratory Study of Data Quality Management Practices in the ERP Software Systems Context. In: Dadam, P., Reichert, M. (eds.): INFORMATIK 2004, Proceedings of the 34th Annual Conference of the Gesellschaft für Informatik, Vol. 1, Bonn 2004, 254–258
24. Sherman, R.: Seven misconceptions about data quality. Software World 35 (2004) 6, 13–14
25. Walsh, T.: Master Data Management: Cross-System Assessment. DM Review Magazine, January 2004. http://www.dmreview.com/article_sub.cfm?articleID=7911
26. Wand, Y., Wang, R.Y.: Anchoring data quality dimensions in ontological foundations. Comm. ACM 39 (1996) 11, 86–95
27. Wittebrock, T.: Master data – Everyone Needs it, but No-one Wants to Maintain it. SAP INFO international, September 15, 2003. <http://www.sap.info/goto/en/go/21299/>
28. Xu, H., Nord, J.H., Brown, N., Nord, G.D.: Data quality issues in implementing an ERP. Industrial Management & Data Systems 102 (2002) 1, 47–58
29. Zornes, A.: Dispelling Master Data Management Myths. DM Direct Special Report, October 20, 2005. http://www.dmreview.com/article_sub.cfm?articleId=1039495

Towards Automatic Evaluation of Learning Object Metadata Quality

Xavier Ochoa¹ and Erik Duval²

¹ Escuela Superior Politécnica del Litoral (ESPOL), Via Perimetral Km. 30.5,
Guayaquil, Ecuador
xavier@cti.espol.edu.ec

² Computerwetenschappen Dept., Katholieke Universiteit Leuven, Celestijnenlaan 200 A,
B-3001 Leuven, Belgium
erik.duval@cs.kuleuven.be

Abstract. Thanks to recent developments on automatic generation of metadata and interoperability between repositories, the production, management and consumption of learning object metadata is vastly surpassing the human capacity to review or process these metadata. However, we need to make sure that the presence of some low quality metadata does not compromise the performance of services that rely on that information. Consequently, there is a need for automatic assessment of the quality of metadata, so that tools or users can be alerted about low quality instances. In this paper, we present several quality metrics for learning object metadata. We applied these metrics to a sample of records from a real repository and compared the results with the quality assessment given to the same records by a group of human reviewers. Through correlation and regression analysis, we found that one of the metrics, the text information content, could be used as a predictor of the human evaluation. While this metric is not a definitive measurement of the “real” quality of the metadata record, we present several ways in which it can be used. We also propose new research in other quality dimensions of the learning object metadata.

1 Introduction

In the first years of learning object metadata [1] production, most of the metadata were created manually by the author of the learning object, a field expert or a librarian and inserted it in a repository where other users could search for it. This wasn't that much of a problem as long as the number of objects present in most repositories was small (less than 10000). The main research effort among the learning object community was to find ways to increase the number of available objects.

As a result of these efforts, several approaches have been developed to deal with the small number of learning objects indexed in a single repository. Information extraction techniques are now used to (semi-) automatically extract the metadata values from the learning object itself or from the context where it is created or published. Examples of systems that employ these techniques could be found at [2] and [3]. Moreover, federated search facilities across learning object repositories have been standardized [4]. In federated searches, the aggregated content of all participant

repositories can be accessed from any one of them. The GLOBE partnership of repositories is a clear example of this tendency [5]. A third approach, disaggregating existing learning objects in their components, has been successfully researched and workable prototypes are available [6]. These new technologies have led to an exponential growth in the number of learning objects with metadata. While this increase is solving the lack of available learning objects, it creates a new and different problem: there exists no feasible way to assure the quality of metadata, either newly produced or accessed from a repository.

The quality of the metadata record that describes a learning object affects directly the chances of the object to be found, reviewed or reused [7]. For example, a learning object automatically indexed from a Learning Management System (LMS) with the title “Lesson 1 - Course CS201”, without any description or keywords will hardly appear in a search for materials about “Introduction to Java”, even if the described object is, indeed, a good introductory text to Java. The object will just be part of the repository but will never be retrieved in relevant searches.

The traditional approach to evaluate the quality of learning object metadata has been the same as in the digital library world [8][9]: Manually review a statistical significant sample of records by comparing the values with those provided by metadata experts. While this human review could be useful for small-sized and slow-growing repositories, it becomes impractical for large or federated repositories or in the case of automatic indexers as humans “do not scale” [10]. Also learning object metadata is different from a library record as the first is not unique (there could be several instances for the same learning object) and evolves as the object itself is reused in different contexts.

To deal with the exponential grow of metadata records available and at the same time to be able to retain some sort of quality assurance for the information contained in the metadata record, we propose automating the quality assessment of learning object metadata. This automated evaluator will assess intrinsic characteristics of the metadata itself, measured through the use of one or more synthetic metrics. This paper builds on a previous work [11] where several quality metrics for learning object metadata are presented. In the following sections, we select and describe a group of those metrics. The values obtained from the metrics are contrasted with evaluations by human reviewers to a sample of learning object metadata from a real repository, and the results are evaluated. We propose some applications for the metrics that correlate highly with the human evaluation. Finally, in the last sections, we propose more experiments to evaluate metrics for different functions of the metadata and present related works.

2 Quality Metrics

The quality metrics are small calculation performed over the values of the different fields of the metadata record in order to gain insight in a quality characteristic. The quality characteristics in which these metrics are loosely based are the ones proposed by [12]: completeness, accuracy, provenance, conformance to expectations, consistency & coherence, timeliness and accessibility. For example, we can count the number of fields that have been filled with information (metric) to assess the completeness

of the metadata record (quality characteristic). These measurements are only concern with the quality of the learning object metadata record, not the quality of the learning object itself. [13] gives a more holistic view of the quality of the learning object as a whole (metadata information included). A more formal description of the metrics that we will use and their *rationale* has been discussed in detail in [11]. We have selected the metrics that are feasible to implement with the information available in real repositories. We present here a brief description and an example of calculation of each one of the selected metrics.

- **Simple Completeness:** It tries to measure the Completeness of the metadata record. This metric counts the number of fields that content a no-null value. In the case of multi-valued fields, the field is considered complete if at least one instance exists. The score could be calculated as a percentage of possible fields and divided by 10 to be in a scale from 0 to 10. For example, according to this metric, a record with 80% of its fields filled has a higher score (higher quality) ($q=8$) than one in which only the 40% has been filled ($q=4$).
- **Weighted Completeness:** It tries to measure the Completeness in a more meaningful way than the “Simple Completeness” metric as not all the fields are equally important for a given application. This metric not only counts the number of filled fields, but assigns a weight value to each of the fields. This weight value should reflect the importance of that field for the application. To be comparable with the Simple Completeness the obtained value should be divided by sum of all the weights and multiplied by 10. For example, if the main application of the metadata will be to provide information about the object to a human user, the title, description and annotation fields are more important than the identifier or metadata’s author fields. The more important fields could have a weight of 1 while the not important records could have a weight of 0.2. In this case, a record with information for title and description only will receive a higher score (higher quality) ($q=2/2.4*10=9$) than a record with information for title, identifier and metadata author ($q=1.4/2.4*10=6$).
- **Nominal Information Content:** One of the main requirements of a metadata record is that it contains enough information to describe uniquely its referred learning object. Unique information helps the user to distinguish and evaluate different objects. This metric tries to measure the Amount of Information that the metadata posses in its nominal fields (fields that can only be filled with values taken from a fixed vocabulary). For this kind of field, the Information Content can be calculated as 1 minus the entropy of the value (the entropy is the negative log of the probability of the value in a given repository) [14]. This metric sums the information content for each categorical field of the metadata record. For example, if the difficulty level of a metadata record is set to “high”, where the majority of the repository is set to “medium”, it will provide more unique information to the record and, thus, a higher score (high quality). On the other hand, if the record’s nominal fields only content the “default values” used in the repository, they will provide less unique information to the record and a lower score.
- **Textual Information Content:** This metric also tries to measure the Amount of Information contained in the record. It measures how much relevant and unique words are contained in the record’s text fields (the fields that can be filled with free text). The “relevance” and “uniqueness” of a word is directly proportional to how

often that word appears in the record and inversely proportional to how many records contain that word. This relation is handled by the TF-IDF (Term Frequency-Inverse Document Frequency) calculation [15]. The number of times that the word appears in the document is multiplied by the negative log of the number of documents that contain that word (could be considered as a weighted entropy measurement). The log of the sum of all the TF-IDF value of all the words in textual fields is the result of the metric. For example, if the title field of a record is “Lecture in Java”, given that “lecture” and “java” are common words in the repository, will have lower score (lower quality) than a record in which the title is “Introduction to Java objects and classes”, not only because “objects” and “classes” are less frequent in the repository, but also because the latter title contains more words.

- **Readability:** This metric tries to measure how accessible the text in metadata is. This metric applies a readability index, for example the Flesch Index [16], to assess how easy is to read the description of the learning object. The readability indexes in general count the number of words per sentence and the length of the words to provide a value that suggest how easy is to read a text. For example, a description where only acronyms or complex sentences are used will receive a higher score (lower quality) than a description where normal words and simple sentences are used.

2.1 Evaluation of the Metrics

We designed an experiment to evaluate how the quality metrics presented above correlate with quality assessment by human reviewers. During the experiment, several human reviewers graded the quality of a set of records sampled from the ARIADNE repository [17]. We selected metadata records about objects on Information Technologies objects that were available in English in the repository. From this universe (425 records), we randomly selected 10 with metadata generated manually and 10 with metadata generated by an automated indexer. Following a common practice to reduce the subjectivity in the evaluation of the quality of metadata, we used an evaluation framework. The selected framework was the same 7 quality characteristics proposed by [12] on which the metrics are loosely based. The experiment was carried out online using a web application. After reading the instructions, the user was presented with a list of the 20 selected objects in no specific order. When the user selected an object, a representation of its LOM record was displayed. The user then downloaded the referred object for inspection. Once the user had reviewed the metadata and the object, he was asked to give grades in a 7-point scale (From “Extremely low quality” to “Extremely high quality”) for each one of the 7 parameters. Only participants that graded all the objects were considered in the experiment. The experiment was available for 2 weeks. During that time, 22 participants completed successfully the review of all the 20 objects. From those 22, 17 (77%) work with metadata as part of their study/research activities; 11 (50%) were undergraduate students in their last years, 9 (41%) were postgraduate students and 2 (9%) had a Ph.D. degree. All of them had a full understanding of the nature and meaning of the examined objects and their metadata, and were trained in the evaluation framework. Parallel to the human evaluation, an implementation of the quality metrics described before was applied to the same set of data that was presented to the reviewers. For the weighted completeness calculation, the weights were obtained from the frequency of use of the fields in the ARIADNE searches [18].

2.2 Data Analysis

Because of the inherent subjectiveness in measuring quality, the first step in the analysis of the data was to estimate the reliability of the human evaluation. In this kind of experiment, the evaluation could be considered reliable if the variability between the grades given by different reviewers to a record is significantly smaller than the variability between the average grades given to different objects. To estimate this difference we use the Intra-Class Correlation (ICC) coefficient [19] over the average quality grade (the sum of the value given to each of the seven quality characteristics, divided by 7). We calculated the measure of ICC using the two-way mixed model, given that all the reviewers grade the same sample of objects. In this configuration, the ICC is equivalent to another widely used reliability measure, the Cronbach’s alpha. The result obtained was 0.909, much higher than the 0.7 threshold needed to be considered acceptable. In other words, the ICC suggests that the reviewers provided similar quality scores and that further statistical analysis can be performed.

The next step was to average the value of all the human reviewers for each record and correlate it with the values obtained from the calculation of the quality metrics over the same records. The results are presented in the Table 1.

Table 1. Correlation between the Human Evaluation Score and the Metrics Scores

		Simple- Comple.	Weighted Comple.	Nominal Info. Content	Textual Info. Content	Read- ability
Human Evaluation	Pearson	-.395	-.457	-.182	.842	.257
	Sign.	.085	.043	.443	.000	.274

The Textual Information Content Metric correlates in a high degree (0,842) with the average quality value given by the human reviewers. The significance of that correlation is very high (<0,01), that means that the correlation is real, and that it is not produced by chance. The correlation is even visible if we plot both scores (textual information content and the average score for quality) for the 20 examined objects (Figure 1). The line on top is the Average Quality as calculated from the scores given by the human reviewers. The bottom line is the value that the metric returned. While the lines do not follow the same exact pattern, much of the first line behavior could be explained by the second one.

Digging deeper, a multivariate regression analysis shows that roughly the 70% of the score given by humans could be explained by the Textual Information Content metric. Another 10% of the variation could be explained by the origin of the metadata (automatic or manual). Other tested metrics do not contribute to explain the grade. The results of the regression can be seen in Table 2.

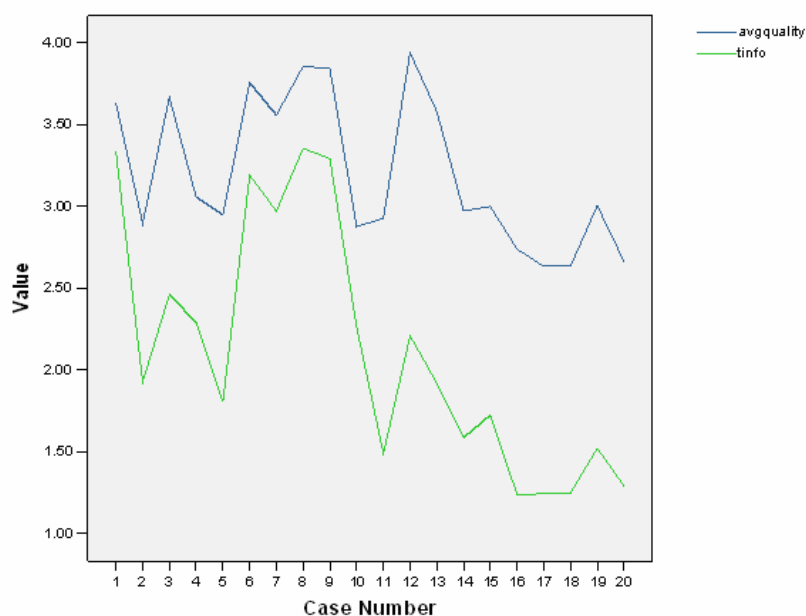


Fig. 1. Average Quality Score and the Textual Information Content Metric values

Table 2. Multivariate Regression Result

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
Textual Information	.842	.710	.694	.25406
Textual Information + Origin of Metadata	.905	.819	.798	.20623

We can conclude from the experiment that the Textual Information Content can be used as a good predictor of the human evaluation of the metadata record. This suggests that human reviewers mainly focus in the free text fields when they are examining the record. This result could have implications in the way that the metadata is presented to final users. For example: only textual fields should be shown by default and non-textual fields should only be presented by user's request. The other metrics, while could be useful to assess the fitness of the record for other applications, do not seem to be useful to predict the quality score of the record given by human reviewers.

3 Applications

Once we have identified good metrics to predict the value of some kind of quality of a metadata record we can use it in several applications:

- **Automatic Record Improvement:** We can compute the metric of all the records present in the repository and process those with a low score by an automatic extractor of metadata, such as [3]. Then we merge the newly generated record with the existing one and evaluate if the score improved. This could be used in legacy repositories, especially when the quality of the original metadata is low.
- **Quality Visualization:** The metrics values can be used to create visualizations of the repository in order to gain a better understanding of the distribution of the quality problems. For example, a treemap visualization [20] could be used to find answers to different questions: Which authors or sources of metadata cause quality problems? How has the quality of the repository evolved over time? Which is the most critical problem of the metadata in the repository?, etc. An example of such visualization is shown on Figure 2. The treemap represent the structure of the ARIADNE repository. The global repository contains several Local repositories (BLKL, CS_L, UGAL, etc). The different authors publish metadata in the local repositories. The boxes represent the set of learning objects metadata records published by a given author. The color of the boxes represents the average of the Textual Information Content metric score of that set of records. The color scale goes from red (low quality) to yellow (medium quality), to green (high quality). This visualization helps us to easily spot authors that provide good textual descriptions to their objects.

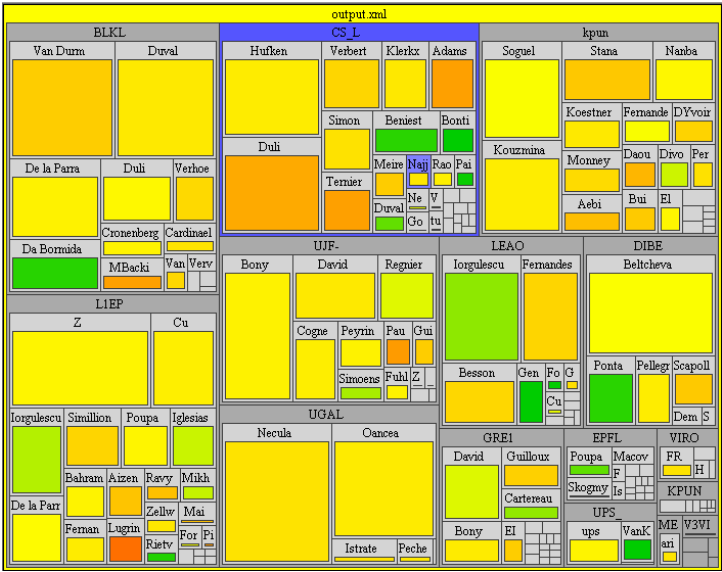


Fig. 2. Visualization of the ARIADNE Repository

- **Repository Interoperability:** One of the problems of the federation of different learning object repositories is the different quality standards for the metadata in the different communities of practice [21]. To avoid degrading the experience of a user in a given repository, the quality metrics could be integrated into a SQI Query Language [4] to restrict the search of metadata records that have a similar quality level as the destination repository.

4 Conclusions

We have proposed 5 of quality metrics. Based in a small experiment, one of the metrics proposed, the Textual Information Content, seems to be useful to predict the quality score assigned by human reviewers to learning object metadata. While a bigger experiment with more records and different repositories is necessary in order to corroborate this result, it suggests that simple metrics could be use to gain insight in something as subjective and complex as the perception of metadata quality. If the correlation persists in further evaluations, it can be concluded that instead of showing the full metadata fields to the user, as several learning object applications do, it will be enough to present them the textual fields of title and description, much like in modern web searches provide only the title and excerpt of web pages.

While the quality metrics has been developed for learning object metadata, the calculations are “standard-agnostic”. The same metrics could be easily extended to most kind of metadata records. For example, they could be applied to evaluate the quality of a digital library using, for example, Dublin Core or MARC metadata standards.

A lot more research and experimentation in quality metrics is needed, but it is clear that if we want to make the transition from small and isolated learning object repositories, to a fully integrated learning object ecosystem where millions of objects are created, indexed and changed daily, we need to have some kind of automatic evaluation of quality to avoid the system to break apart. This work is a first step that contributes to enabling such automated quality assessment.

5 Further Work

The quality value, assessed by the reviewers and predicted by the Textual Information Content metric, can be considered as “synthetic” quality. No real application present the learning object metadata record to the user in order to obtain a quality evaluation. The “real” quality of the metadata can be defined as the fitness of the metadata to fulfill a given purpose [9]. According to [1], the main purposes of the learning object metadata are to facilitate the search, evaluation, acquisition, and use of the learning object. Under these premises, there should be 4 different dimensions of quality to be measured: Retrieval Quality, Evaluation Quality, Accessibility Quality and Re-use Quality. As further work to be done in order to obtain the desired automatic evaluator of learning object metadata quality, we present some possible ways to calculate quality sources for the different quality dimensions. Those scores should be correlated with new quality metrics to find, among those metrics, useful predictors that can be used to develop the automatic evaluator of quality.

- **Retrieval Quality:** The logs of the searching tools for learning objects can be analyzed in order to calculate the number of times that a given object appear in the result list when a relevant search has been performed. High quality metadata should allow the object to appear in high percentage of relevant queries. The position of the object inside the result list could also be taken in account for the retrieval quality score.
- **Evaluation Quality:** The tools that present the metadata record to the user for evaluation can include a small survey asking if the information shown has been

useful in order to decide if the object is relevant for him/her. The percentage of times that the metadata information has been found useful could be considered evaluation quality score.

- **Accessibility Quality:** Learning object tools could log the times that a learning object could not be accessed or retrieved because of errors in the metadata information. For example when there are errors in the identifier, URI, or cost of the learning object. The number of errors detected should decrease the accessibility quality score.
- **Re-use Quality:** The re-use of the learning object is not only determined by the characteristics of the metadata, but also for the characteristics of the learning object itself. Nonetheless, objects that are re-used more often, most probably, have a suitable metadata that allow users to integrate it in their learning systems. This, the re-use quality score could be determined by the number of times that the object has been repurposed in different learning contexts.

6 Related Work

While, to the authors' knowledge, there are no currently other initiatives to build an automatic quality assessment system for learning object metadata, in the Semantic Web area, [22] propose an automated system to assess the quality of the automatic extraction of semantic information from web pages. While the core idea, to have an automated way to ensure quality of an automated system, is similar to ours, it focus mainly in the correction of spelling errors and concept disambiguation through the use of ontologies and Semantic Web tools.

References

1. IEEE (2002). IEEE Standard for Learning Object Metadata. <http://ltsc.ieee.org/doc/wg12/>
2. Singh, A., Boley, H., & Bhavsar, V. C. (2004). A learning object metadata generator applied to computer science terminology. In Presented at the Learning Objects Summit, March 29-30, 2004.
3. Cardinels, K., Meire, M., & Duval, E. (2005). Automating metadata generation: the simple indexing interface. In Proceedings of the 14th WWW conference (pp. 548-556). New York, NY, USA: ACM Press.
4. Simon, B., Massart, D., van Assche, F., Ternier, S., Duval, E., Brantner, S. et al. (2006). A Simple Query Interface for Interoperable Learning Repositories. In B. Simon, D. Olmedilla, & N. Saito (Eds.), (pp. 11-18).
5. GLOBE (2006). Global Learning Objects Brokered Exchange. <http://taste.merlot.org/initiatives/globe.htm>
6. Verbert, K., Jovanovic, J., Gašević, D., & Duval, E. (2005). Repurposing Learning Object Components. In OTM 2005 Workshop on Ontologies, Semantics and E-Learning
7. Currier, S., Barton, J., O'Beirne, R., & Ryan, B. (2004). Quality assurance for digital learning object repositories: issues for the metadata creation process. ALT-J, Research in Learning Technology, 12, 5-20.

8. Barton, J., Currier, S., & Hey, J. (2003). Building Quality Assurance into Metadata Creation: an Analysis based on the Learning Objects and e-Prints Communities of Practice. In *Proceedings 2003 Dublin Core Conference*: (pp. 39-48). Seattle, Washington.
9. Guy, M., Powell, A., & Day, M. (2004). Improving the quality of metadata in Eprint archives. *Ariadne*.
10. Weibel, S. (2005). Border Crossings: Reflections on a Decade of Metadata Consensus Building. *D-Lib Magazine*, 11.
11. Ochoa, X. & Duval, E. (2006). Quality Metrics for Learning Object Metadata. In *Proceedings ED-Media 2006*. (pp. 1004 -1011)
12. Bruce, T. & Hillman, D. (2004). The continuum of metadata quality: defining, expressing, exploiting. In D.Hillman & L. Westbrooks (Eds.), *Metadata in Practice* (Chicago: American Library Association.
13. Duval, E. (2005). LearnRank: the real quality measure for learning materials. In A.McCluskey (Ed.), *Policy and Innovation in Education - Quality Criteria* (pp. 26-29). European Schoolnet.
14. Resnik, P. (1995). Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *Proceedings of International Joint Conferences on Artificial Intelligence* (pp. 448-453).
15. Aizawa, A. (2003). An information-theoretic perspective of tf-idf measures. *Information Processing and Management*, 39, 45-65.
16. Foltz, P., Kintsch, W., & Landauer, T. (1998). The measurement of textual coherence with Latent Semantic Analysis. *Discourse Processes*, 25, 285-307.
17. Ariadne Foundation (2005). Ariadne Foundation. <http://www.ariadne-eu.org>
18. Najjar, J., Ternier, S., & Duval, E. (2004). User Behavior in Learning Objects Repositories: An Empirical Analysis. In *Proceedings of ED-MEDIA 2004*
19. Shrout, P. & Fleiss, J. (1979). Intraclass Correlations: Uses in Assessing Rater Reliability. *Psychological Bulletin*, 2, 420-428.
20. Bederson, B., Shneiderman, B., & Wattenberg, M. (2002). Ordered and Quantum Tree-maps: Making Effective Use of 2D Space to Display Hierarchies. *ACM Transactions on Graphics (TOG)*, 21, 833-854.
21. Shreeves, S., Knutson, E., Stvilia, B., Palmer, C., Twidale, M., & Cole, T. (2005). Is "Quality" Metadata "Shareable" Metadata? The Implications of Local Metadata Practices for Federated Collections. In *Proceedings of the Twelfth National Conference of the Association of College and Research Libraries* Chicago, IL: Association of College and Research Libraries.
22. Lei, Y., Sabou, M., Lopez, V., Zhu, J., Uren, V., & Motta, E. (2006). An Infrastructure for Acquiring High Quality Semantic Metadata. In *Proceedings of the 3rd European Semantic Web Conference (ESWC 2006)*

Expressing and Processing Timeliness Quality Aware Queries: The DQ²L Approach

Chao Dong, Sandra de F. Mendes Sampaio, and Pedro R. Falcone Sampaio

School of Informatics, University of Manchester,
Manchester M60 1QD
Chao.Dong@Postgrad.Manchester.ac.uk,
{S.Sampaio, P.Sampaio}@Manchester.ac.uk

Abstract. With the growing need for querying and combining data from multiple data sources, data analysts, database application programmers and advanced database users are increasingly facing the problem of filtering out low quality data with regard to the intended use. This paper investigates the problem of expressing and processing data quality requests during quality-aware query formulation. The paper proposes the Data Quality Query Language (DQ²L), an extension of SQL aimed at enabling query language users to express data quality requests and a query processing framework (architecture, query processing stages, metadata support and quality model) aimed at extending relational query processing with quality-aware query processing structures and techniques. The paper focuses on the timeliness data quality dimension.

1 Introduction

With the growing need for querying and combining data from multiple data sources, data analysts, application programmers and advanced database users are increasingly facing the problem of filtering out low quality data with regard to the intended use.

This difficulty is common in applications such as customer relationship management and health care management, which rely on data available from multiple data sources originally designed for other intended uses, e.g., a Telecom phone directory stored in a relational table that can be queried to find potential customers for cross-selling broadband services in a CRM call centre campaign; Mining and collating information in patient records for disease trend analysis. Frequently, when faced with the need for rapidly obtaining, merging, collating and formatting data relating to a new information request, advanced database users are not able to wait for the completion of a new data warehouse project that will provide integrated and clean data to address the information request.

A more likely scenario is that the new information request will be addressed by a series of SQL queries, cut and paste operations, and possibly the development of application programs to help filtering out low quality data, e.g., filter data items that have not been updated in the database for more than 2 years to avoid decayed data, and also to filter tuples that have low information completeness such as three or more attributes with null values, etc.

Some of these costly and partially automated operations can be simplified if advanced database users are supported by a query language that simplifies the

formulation of data quality requests and a database environment that incorporates quality aspects into the query processing framework, e.g., logical rewriting and cost/quality based optimization.

This paper investigates the problem of expressing and processing data-quality requests during quality-aware query formulation. The paper proposes the Data Quality Query Language (DQ²L), an extension of SQL aimed at enabling query language users to express data quality requests and a query processing framework (architecture, query processing stages, metadata support and quality model) aimed at extending relational query processing with quality-aware query processing structures and techniques.

The remainder of the paper is structured as follows. Section 2 explains the background and related work in expressing data quality constructs in queries. Section 3 presents the high-level design and syntax of DQ²L. Section 4 illustrates the framework, architecture and logical algebra for supporting quality aware query processing. Section 5 introduces the data model and metadata model used in the quality assessments performed during query processing. Section 6 presents conclusions and addresses future work.

2 Expressing Quality Constraints in Queries: Related Work

Despite quality constraints typically refer to application specific requirements, there are data quality dimensions that can be regarded as applicable to all kinds of database related applications. The classical examples described in the data quality literature are [3]:

- *Accuracy of data:* Data can have errors or inconsistencies in its representation. For example, the data values “ST Louis” and “Saint Louis” may not be matched in a Join between different relational tables despite referring to the same address instance, due to the different representation formats.
- *Timeliness of data:* Data can be of poor quality when it is not timely enough for the intended use. For example, a data source containing sales information is used to compute company wide sales bonus due on the 15th of the following month. However, some units of the company may take several days to send in updated regional sales information reducing the quality of the computed results.
- *Completeness of data:* Data can be incomplete. For instance, a null value for a mandatory attribute is associated with lower completeness quality.

Along with the development of data quality research, how to simply and concisely express quality constraints as database queries has become an important issue in the literature. [1] proposed a model for expressing currency and consistency (C&C) constraints in queries for replicated and cached data by using a new currency clause that extends SQL. A C&C constraint is composed by triples, where each triple consists of 1) a currency bound, 2) a consistency class formed by tables, and 3) grouping rules from consistency class into consistency groups. In addition, both single-block and multi-block queries are supported in this approach.

```

SELECT  NAME
FROM    BOOKS B, REVIEWS R
WHERE   B.ISBN=R.ISBN
CURRENCY BOUND 10 MIN ON(B, R)

```

Fig. 2.1. An single-block C&C constraint query example

In the above query, the **CURRENCY BOUND** quality clause indicates that (1) the inputs tables B and R cannot be more than 10 minutes out of date and (2) the two input tables B and R must be consistent.

In [5], a query language extension “with QUALITY” was first proposed to address quality concerns. Followed by “with QUALITY”, quality requirements for specific attributes can be specified, as shown in Fig 2.2. The primary goal is to assist users in distinguishing the quality requirements regarding the attributes referred in SQL queries. Moreover, a set of quality extensions to algebraic operations extensions including *Selection*, *Projection*, and *Union* have been also proposed.

```

SELECT  ACTOR
FROM    MOVIES
WHERE   TITLE = “GOD FATHER”
with QUALITY DBA = “SMITH”

```

Fig. 2.2. an example query with clause “with QUALITY”

In the above query, a quality assertion is placed associating a quality constraint with regard to the person that was responsible for data input.

The Trio system [4] is an ongoing project which aims at extending traditional data management by introducing components such as uncertainty and lineage into both data and queries. As its query language, TriQL is designed as an extension to SQL, for querying and modifying data in the Trio data model. In TriQL, extended semantics of SQL queries and new features enable the capability of managing data with uncertainty and lineage. Some potential ideas for core functionality for TriQL have been discussed in the paper, although no specific syntax for timeliness is given.

3 The Data Quality Query Language (DQ²L)

DQ²L was designed to query relational data supporting a data quality aware query processing framework. It extends SQL with a number of simple constructs that are used within the framework to establish quality requirements relating to a query, to measure quality, and to assign quality scores to individual data units in query results.

The Data Quality Query Language (DQ²L) adopts the feature “with QUALITY” which was proposed in [5], however, in contrast to [5], we also support the expression of quality related functions and we substantially extend the clause to cater for detailed specification of quality assertions:

1. DQ²L extends the **WHERE** clause of a **SELECT-FROM-WHERE** SQL statement with the **WITH QUALITY AS** clause, which is optionally used when the user

wishes to specify data quality requirements or constraints in a query, as shown in the *Query 1* below. In this example, the **WITH QUALITY AS** clause is used to indicate to the query processor that tuple results containing information about the amount information of a medicine that has been last updated before or on '2006-04-01 12:00:00+00:00' are not acceptable. Note that the last update time for a particular attribute, in this case attribute **Amount**, is stored as a data quality attribute **LastUpdateTime(attributeName)** in an Quality Information Repository (QIR), that is, in turn, associated with relation **Medicine** in the database. Other constructors include **ExpiryTime(attributeName)** and **Age(attributeName)**, which correspond to the expiry time and the age of an attribute. These constructors can also be used in the **SELECT** clause, to retrieve the corresponding quality information for specific attributes. More details about the Quality Information Repository and data quality indicators are provided in Section 5.

Query 1: "Select all information about medicines for 'headache', in which amount information was last updated after '2006-04-01 12:00:00+00:00', from relation **Medicine**."

```
SELECT *
FROM Medicine
WHERE Category='headache'
WITH QUALITY AS
LastUpdateTime(Amount)>'2006-04-01 12:00:00+00:00'
```

2. DQ²L also extends SQL with functions such as **TIMELINESS(attributeName)**, which can be used in the **SELECT** clause or **WITH QUALITY AS** clause to request the timeliness measure for a particular attribute, as shown in *Query 2* and *Query 3*. Details about the timeliness measurement for attributes can be found in Section 5.

Query 2: "Select the medicine's name, price and the timeliness score of price from relation **Medicine**, for medicines for 'headache', with the timeliness score for amount information greater than 0.60."

```
SELECT Name, Price, TIMELINESS(Price)
FROM Medicine
WHERE Category='headache'
WITH QUALITY AS
TIMELINESS(Amount)>'0.60'
```

Query 3: "Select the medicine's name from relation **Medicine**, grouped by the manufacturer, for medicines for 'inflammation', with last update time of amount no later than '2006-04-01 12:00:00+00:00', and the timeliness score of price no less than '0.50'."

```
SELECT Name
FROM Medicine
WHERE Category='inflammation'
WITH QUALITY AS
LastUpdateTime(Amount)>'2006-04-01 12:00:00+00:00' AND
TIMELINESS(Price)>='0.50'
GROUP BY Manufacturer
```

The Front-End Query Parser is currently implemented using the JavaCC compiler generator framework. Query processing is discussed in the following section. The complete BNF syntax definition of DQ²L is also available by contacting the authors.

4 Quality Aware Query Processing Framework

The goal of data quality aware query processing is to find the query plans which maximize the quality of the query results. Moreover, since data quality is a subjective issue, users can specify different data quality constraints, that should be taken into account during query processing.

To enable data quality aware query processing, we propose to extend the mainstream query processing framework with data quality processing capabilities. For example, information such as the last update time of the data, the validity period of the data, etc. can be tagged onto the data and used during query processing to inform the user of the timeliness measure of the results of a query, as described in [6].

In data quality aware query processing, users may be concerned with quality issues more than with traditional cost model issues, such as response time, number of disk I/O, etc. However, the traditional cost-based query optimization should not be ignored. For example, a user should not wait for a result with 1% higher quality but 200% longer response time. Therefore, in the data quality aware query processing, the traditional cost model is still an indispensable component to guarantee execution performance.

Figure 4.1 illustrates the data quality aware query processing framework.

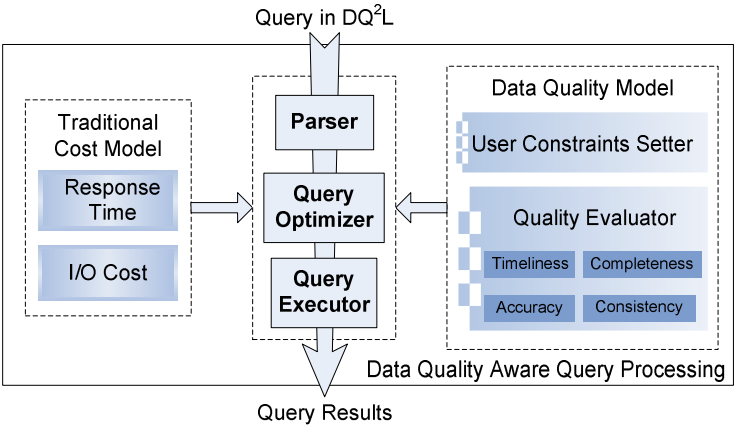


Fig. 4.1. Data Quality Aware Query Processing Framework

In this framework, typical components in traditional query processing are included. The user's query is written in DQ²L and sent to the Parser for syntax checking and validation against the schema. The user explicitly states the quality thresholds via the DQ²L syntax and the quality constraints are registered at the User Constraints Setter for use during query optimization and evaluation.

The output from the parser is then delivered to the Query Optimizer, whose main responsibility is to choose the 'best cost' query plan that complies with the specific set of quality conditions specified in the WITH QUALITY AS clause. Therefore, the aim of the Data Quality Model is to guarantee that quality compliance is achieved through the matching of quality constraints described in the User Constraints Setter

and the query plans generated by the query optimizer. Finally, the optimal query plan will be executed by the Query Executor with higher quality query results sent to the user. Fig 4.2 illustrates the DQ²L Query Processor Architecture. The main components of the architecture are:

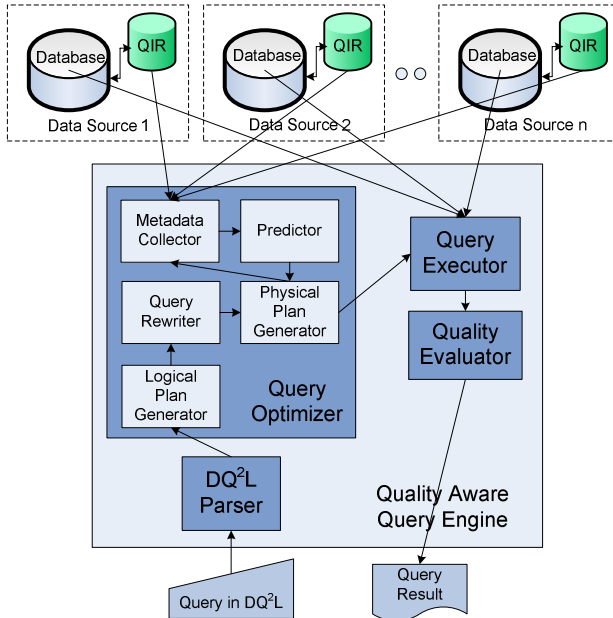


Fig. 4.2. DQ²L Query Processor Architecture

Data Sources: Due to the quality evaluation module in our framework, it is necessary to support a Quality Information Repository (QIR) providing quality metadata relating to each data source. QIR is responsible for storing the quality information for attributes in the database, used during processing of each query plan and intermediate query results. QIR is divided into two categories, (1) a summary of quality information for each attribute in a relation, which is aimed at predicting the quality level for query plans; (2) individual quality information for each tuple, which is responsible for producing precise quality scores later in the Quality Evaluator. More information can be found in the next section.

Quality Aware Query Engine: This is the core of the DQ²L framework. In this layer, queries from users are parsed, translated, estimated, and query results are evaluated, filtered, marked, and ranked. A query written in DQ²L is mapped into a logical algebraic query plan extended with algebraic operations for dealing with quality assessments and filtering of low quality data. To construct query plans, the DQ²L query processing framework extends relational algebra operators with operations to compute quality measures. Some of the key operators used in DQ²L query plans are described in Table 4.1.

User Front-End: The interface used to formulate and submit DQ²L queries and browse over query results.

Table 4.1. DQ²L Timeliness Algebra

Logical Operators	Description
Scan (inputData)	Receives the data coming from the a data source (inputData), builds a data structure for each unit of data (input) and passes each structure to the next operator. Data containing Timeliness information (last update time, expiry time and age) are treated as relational metadata attributes, which are associated with a specific attribute in the structure by being treated as relational attributes to this attribute.
Select (input, pred)	Receives an input containing a number of values for relational attributes, applies a predicate (pred) over the input and either discards or retains the input depending on whether the predicate evaluates to false or true.
Project (input, list)	Receives an input containing a number of values for relational attributes, and discards all the attributes that are not specified in list. For each discarded attribute, all the Timeliness attributes are also discarded.
Join (inputLeft, inputRight, pred)	Receives two inputs, each containing a number of values for relational attributes, "concatenates" both inputs, retaining all their attributes, applies a predicate (pred) over the result and either discards or retains the result depending on whether the predicate evaluates to false or true.
Timeliness (input)	Receives an input containing a number of values for relational attributes, calculates the Timeliness measure for attributes in the input, attach each measure to the corresponding attribute.

An example of how the DQ²L *Query 2* described in section 3 is mapped into the above logical algebra is provided in Fig 4.3:

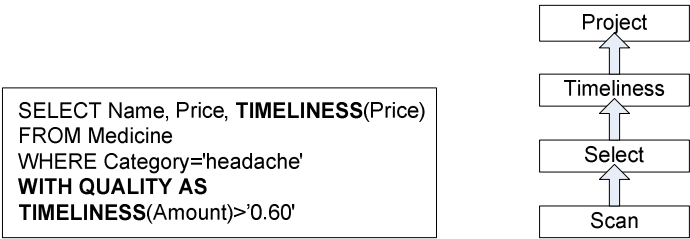


Fig. 4.3. Logical Query Plan for DQ²L *Query 2*

The main difference between the query processing framework described in this paper and mainstream query processing is the use of data quality models: (1) quality prediction for logical query plans during logical plan generation; (2) quality evaluation when quality indicators information is gathered. Other data quality query processing frameworks that extend mainstream query processing with quality models can be found in [7, 8, 9, 10]. The main difference between our approach and the other quality aware query processing frameworks described in [7, 8, 9, 10] is that we

develop a framework based on algebraic query processing techniques and we also adopt SQL as the baseline language for the data quality extension.

5 Quality Related Metadata Models

As discussed before, in Quality Aware Query Processing, quality information is indispensable for query optimization and evaluation. However, due to the potential large scale of databases, it is unfeasible and inefficient to retrieve quality information for each attribute during optimization. Therefore, we introduce two categories of metadata models into our DQ²L to address this problem: (1) a metadata relation to summarize the quality information for each attribute of relations in the database, illustrated in Fig 5.1 as Table 1; (2) relations which store individual quality information for each instance of the attributes that are being controlled for quality purposes, as shown in Fig 5.1 as Tables 2, 3, and 4. The metadata is updated every time any tuple related to the quality metadata relations changes. Thus, when the query optimization takes place, the system only needs to search the metadata in each data source for the necessary quality information. This approach enables the possibility of conducting query optimization effectively. In the remainder of this section, we will explain the quality data and metadata models to support the timeliness dimension.

Timeliness is usually defined via two concepts: *Currency* and *Volatility* [2].

- *Currency*: is defined as the age of the data when it is delivered to the user. It is dependent upon three key factors: (i) the time when the data is delivered to the user (Delivery Time), (ii) the time when the data was entered/modified in the database (Last Update Time), and (iii) how old the data was when entered into the database (Age). Based on the measurement formula proposed in [2], *Currency* can be described as:

$$Currency = Delivery\ Time - Last\ Update\ Time + Age \quad (1)$$

- *Volatility*: is defined as the length of time during which the data remains valid. It is also dependent upon three key factors: (i) the time when the data expires (Expiry Time), (ii) the last update time of data and (iii) the age of data, which are described above. *Volatility* can be expressed as:

$$Volatility = Expiry\ Time - Last\ Update\ Time + Age \quad (2)$$

From the above definitions, the timeliness measurement formula can be described as follows [2]:

$$Timeliness = \{max[(1 - Currency/Volatility), 0]\}^s \quad (3)$$

The exponent s is a parameter that allows control of the sensitivity of Timeliness to the Currency-Volatility ratio, and its value should be chosen depending upon the context in which data is being judged.

In order to evaluate the timeliness quality of data, the following quality indicators should be provided by the data sources: Last Update Time, Expiry Time, and Age. The quality indicators should be tagged onto the tuples in the data sources and

retrieved and processed by the Quality Aware Query Engine so that quality evaluation can take place. We reused the mechanism proposed in [5] to incorporate quality tags to relational databases, which is done at *attribute* level.

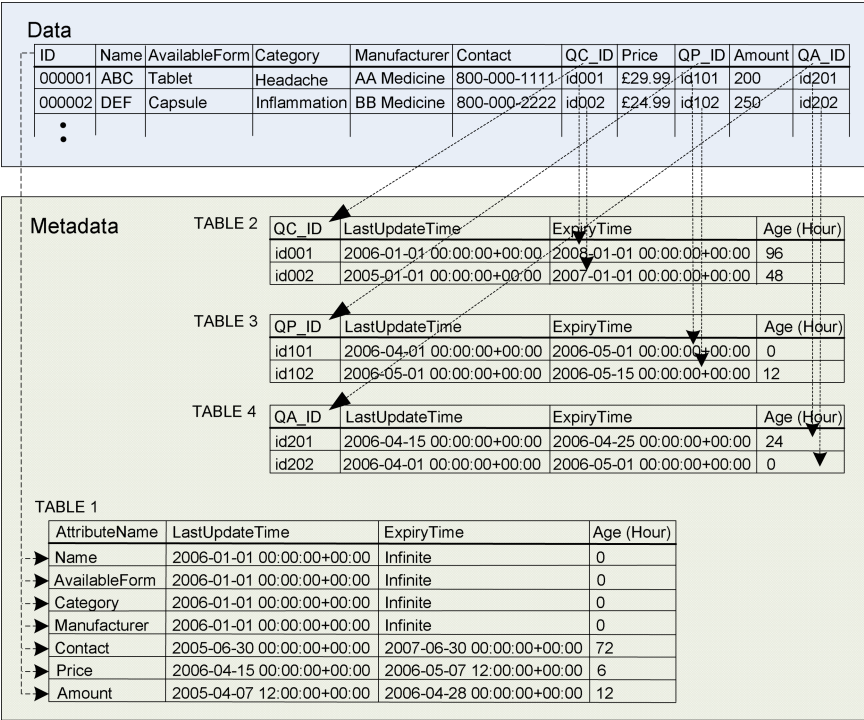


Fig. 5.1. DQ²L Data Models

The quality information is stored in the Quality Information Repository (QIR), where each tuple refers to a particular instance of an attribute in a database relation. As shown in Fig. 5.1, the QIR contains four tables, Tables 1, 2, 3 and 4, which are associated with relation **Medicine** (**ID**, **Name**, **AvailableForm**, **Category**, **Manufacturer**, **Contact**, **QC_ID**, **Price**, **QP_ID**, **Amount**, **QA_ID**). Table 1 stores a summary for the detailed quality information stored in Tables 2, 3, and 4. The quality information in Tables 2, 3 and 4 represent individual quality information for each instance of an attribute in the database, through the quality information pointers **QC_ID**, **QP_ID**, and **QA_ID**. More detailed information about the quality data model used in our implementation can be found in [5].

6 Conclusions and Ongoing Work

This paper proposes the Data Quality Query Language (DQ²L), an extension of SQL aimed at enabling query language users to express data quality requests and a query processing framework (architecture, query processing stages, metadata support and

quality model) aimed at extending relational query processing with quality-aware query processing structures and techniques. The paper focuses on the timeliness data quality dimension. We have developed a feasibility study of incorporating the timeliness data quality query processing framework using the Niagara Internet Query system [6]. In the Niagara extension, DQ²L was not used as the front-end language, and quality constructs were hardwired in the XML-QL statements of Niagara. We are now extending SQL constructs to express completeness data quality constraints and developing a relational implementation of the data quality aware framework based on mapping DQ²L statements into a relational query processing engine. The prototype will be used in the context of data filtering and cleaning for health care management applications [11]. We are also planning to benchmark the impact of quality and cost-based query optimization for queries involving data quality constructs.

References

- [1] Hongfei Guo; Per-Ake Larson; Raghu Ramakrishnan; Jonathan Goldstein, Relaxed Currency and Consistency: How to Say “Good Enough” in SQL, SIGMOD 2004, France, 2004
- [2] Ronald Ballou; Richard Wang; Harold Pazer; Giri Kumar Tayi, Modeling Information Manufacturing Systems to Determine Information Product Quality, Management Science, 44, 4; ABI/INFORM Global, Pages.462-484, 1998
- [3] M. Gertz; T. Ozsu; G. Saake; K. Sattler, Data Quality on the Web, Dagstuhl Seminar, Germany, 2003
- [4] P. Agrawal; O. Benjelloun; A. Das Sarma; C. Hayworth; S. Nabar; T. Sugihara; J. Widom, Trio: A System for Data, Uncertainty, and Lineage, VLDB Conference, Seoul, Korea, 2006
- [5] Richard Y. Wang; M.P. Reddy; Henry B. Kon, Toward Quality data: An attribute-based approach, Decision Support Systems 13 (1995), 349-372, 1995
- [6] Sandra de F. Mendes Sampaio; Chao Dong; Pedro R. Falcone Sampaio, Incorporating the Timeliness Quality Dimension in Internet Query Systems, WISE 2005 Workshop on Web Information Systems Quality - WISQ, LNCS 3807, pp. 53-62, New York, USA, 2005
- [7] V. Peralta; R. Ruggia; Z. Kedad; M. Bouzeghoub, A Framework for Data Quality Evaluation in a Data Integration System, Anais do SBBD-SBES'2004, Brazil, 2004
- [8] Felix Naumann, Quality-driven Query Answering for Integrated Information Systems, Lecture Notes in Computer Science, vol. 2261, Springer, 2002
- [9] M. Mecella, M. Scannapieco, A. Virgillito, R. Baldoni, T. Catarci, C. Batini, The DaQuinCIS Broker: Querying Data and Their Quality in Cooperative Information Systems, J. Data Semantics 1: 208-232, 2003
- [10] Zoran Majkic, A General Framework for Query Answering in Data Quality-based Cooperative Information Systems, in Proc. of the Intl. Workshop on Information Quality in Information Systems, pages 44-50, 2004.
- [11] Sandra de F. Mendes Sampaio, Chao Dong; Pedro R. Falcone Sampaio, Building a data quality aware internet query system for health care applications, in Proc. Of Information Resources Management Association International Conference - Databases Track, San Diego, USA, 2005

Preface for SemWAT 2006

Hyoil Han¹ and Ramez Elmasri²

¹ Drexel University, USA

² University of Texas at Arlington, USA

The Semantic Web has been proposed as the next generation of the existing Web. Semantic Web-enabled applications can potentially produce better results for semantic integration, interoperability and search. Ontologies have been utilized for interoperability among various data sources. The role of ontology is one of the central points in Semantic Web-enabled applications. Semantics in ontologies and conceptual modeling play central roles in achieving interoperability among Semantic Web applications.

The emergence of the World Wide Web made massive amounts of data available. Data exist in many scattered electronic data sources (e-sources) over the Web. Even though some of the data are in well-organized data sources, when they need to be integrated with or be interoperable with data from other sources, semantic coordination and conflict resolution are required.

The focus of this workshop was to present research concerning issues such as: learning/constructing ontologies and conceptual modelling for Semantic Web-enabled e-sources and applications; utilizing ontologies and conceptual modelling for data management, integration and interoperability in Semantic Web applications; and building architectures, models, and languages for achieving Semantic Web goals.

Combining Declarative and Procedural Knowledge to Automate and Represent Ontology Mapping

Li Xu¹, David W. Embley², and Yihong Ding²

¹ Department of Computer Science,
University of Arizona South, Sierra Vista, Arizona 85635, U.S.A.
lxu@email.arizona.edu

² Department of Computer Science,
Brigham Young University, Provo, Utah 84602, U.S.A.
{embley, ding}@cs.byu.edu

Abstract. Ontologies on the Semantic Web are by nature decentralized. From the body of ontology mapping approaches, we can draw a conclusion that an effective approach to automate ontology mapping requires both data and metadata in application domains. Most existing approaches usually represent data and metadata by ad-hoc data structures, which lack formalisms to capture the underlying semantics. Furthermore, to approach semantic interoperability, there is a need to represent mappings between ontologies with well-defined semantics that guarantee accurate exchange of information. To address these problems, we propose that domain ontologies attached with extraction procedures are capable of representing knowledge required to find direct and indirect matches between ontologies. And mapping ontologies attached with query procedures not only support equivalent inferences and computations on equivalent concepts and relations but also improve query performance by applying query procedures to derive target-specific views. We conclude that a combination of declarative and procedural representation based on ontologies favors the analysis and implementation for ontology mapping that promises accurate and efficient semantic interoperability.

1 Introduction

Ontologies on the Semantic Web, by nature, are decentralized and built independently by distinct groups. The research on *ontology mapping* is to compare ontological descriptions for finding and representing semantic affinities between two ontologies. By analyzing the body of ontology mapping approaches [2,6,7,8,12,13,15,16,18], a key conclusion is that an effective ontology mapping approach requires a principled combination of several base techniques such as linguistic matching of vocabulary terms of ontologies, detecting overlap in the choice of data types and representation of data values populated in ontologies, considering patterns of relationships between ontology concepts, and using domain knowledge [13].

To support knowledge sharing between base ontology-mapping techniques, a knowledge base that describes domain models is of great value. The knowledge bases in most existing approaches, however, are represented informally by ad-hoc data structures, in which it is difficult to effectively capture well defined semantics. To further facilitate

interoperability between ontologies, there is a need to represent mappings such that the mapping representation guarantees successful exchange of information. The research work that addressed this ontology-mapping representation problem is usually done separately from the research that focuses on finding semantic affinities [3,10,11,14]. The separation results in lack of support for an efficient approach to achieve interoperability on the Semantic Web. To approach these problems within one knowledge-representation framework, we argue that a combination of declarative and procedural representation based on ontologies favors the analysis and implementation for ontology mapping and promises accurate and efficient semantic interoperability.

Our declarative representation for ontology mapping includes (1) domain ontologies that provide semantic bridges to establish communications among base techniques in order to find semantic affinities between ontologies; and (2) mapping ontologies that provide means to correctly exchange information. Declaratively, ontologies are usually expressed in a logic-based language so that detailed, accurate, consistent, sound, and meaningful distinctions are possible among concepts and relations. Their logic base promises proper reasoning and inference on ontologies.

Ontology mapping requires more than the declarative expressivity of ontologies. One reason is that ontologies have difficulties to effectively express exceptions and uncertainties [17]. Semantic heterogeneity among ontologies is caused by exceptions and uncertainties. Hence, the capability of handling exceptions and uncertainties is extremely important for ontology mapping since the goal of ontology mapping is to find and represent semantic affinities between semantically heterogeneous ontologies. Moreover, to support interoperability across ontologies, based on a debate on the mailing list of the IEEE Standard Upper Ontology working group,¹ semantic interoperability is to use logic in order to guarantee that, after data are transmitted from a sender system to a receiver, all implications made by one system must hold and must be provable by the other, and that there should be a logical equivalence between those implications [11]. To express equivalent concepts and relations between two ontologies, queries have to be issued to compute views over ontologies since ontologies rarely match directly [18]. The associated set of logic inference rules with ontologies, however, support neither expressing complex queries nor reasoning about queries efficiently.

Procedural attachment is a common used technique to enforce the expressive power in cases where expression power is limited [17]. A *procedural attachment* is a method that is implemented by an external procedure. We employ two types of procedural attachments in our approach. A domain ontology shared by base ontology-mapping techniques is attached with *extraction procedures*. An extraction procedure is an encoded method with extraction patterns, which express both vocabulary terms and lexical instantiations of ontology concepts. A mapping ontology, on the other hand, is attached with *query procedures* to establish a communication across ontologies. Each mapping instance maps a *source* ontology to a *target* ontology, which is guaranteed to correctly transmit the source data to the target. A *query procedure* computes a target-specific view over the source so that the view data satisfies all implications made by the target.

In this paper, we offer the following contributions: (1) attaching extraction procedures with domain ontologies to represent knowledge shared by base techniques to find

¹ Message thread on the SUO mailing list initiated at <http://suo.ieee.org/email/msg07542html>.

semantic affinities between ontologies; and (2) attaching query procedures with mapping ontologies to efficiently interoperate among heterogeneous ontologies based on mapping results produced by base techniques. We present the details of our contribution as follows. Section 2 describes elements in input and domain ontologies and how to apply domain ontologies to find semantic affinities between ontologies. Section 3 describes source-to-target mappings as mapping ontologies and how the representation supports accurate and efficient semantic interoperability. Section 4 gives an experimental result to demonstrate the contribution of applying domain ontologies to ontology mapping. Finally, we summarize and draw conclusions in Section 5.

2 Domain Model Representations

2.1 Input Ontology

An ontology include classes, slots, slot restrictions, and instances [4]. A class is a collection of entities. Each entity of the class is said to be an instance of that class. With “IS-A” and “PART-OF” relationships, classes constitute a hierarchy. Slots attached to a class describe properties of objects in the class. Each slot has a set of restrictions on its values, such as cardinalities and ranges. By adopting an algebra approach to represent ontologies as logical theories [11], we use the following definition for input ontologies.

Definition 1. An *input ontology* $O = (S, A, F)$, where S is the *signature* that describes the vocabulary for classes and slots, A is a set of *axioms* that specify the intended interpretation of the vocabulary in some domain of discourse, and F is a set of *ground facts* that classifying instances with class and slot symbols in the signature S . $\square$

For discussion convenience, in this paper we use rooted hypergraphs graphs to illustrate structure properties between classes and slots in ontological signatures. A hypergraph includes a set of nodes modeling classes and slots and a set of edges modeling relations between them. The root node is representing a designated class of primary interest. Figure 1, for example, shows two ontology hypergraphs (whose roots are *house* and *House*). In hypergraphs, we present a class or slot using either a solid box or a dashed one. A solid box indicates that there are object identifiers populated for the concept. And a dashed box indicates that there is lexical data populated for the concept. A functional relation using a line with an arrow from its domain to its range, and a nonfunctional relation using a line without arrowhead.

2.2 Domain Ontology

To accommodate knowledge requirements of ontology mapping, we define domain ontologies as follows.

Definition 2. A *domain ontology* $O = (S, A, P)$, where S is the *ontological signature*, A is a set of *ontological axioms*, and P is a set of *extraction procedures* that extract metadata and data from vocabulary terms and populated instances of input ontologies based on extraction rules². $\square$

² Ground facts are not part of a domain ontology since a domain ontology is not populated with instances.

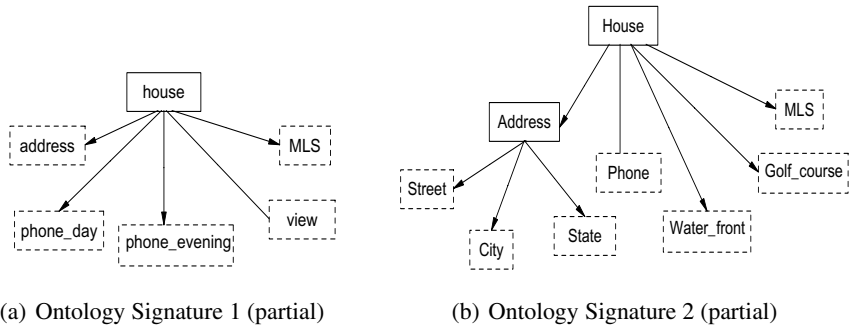


Fig. 1. Signatures of Input Ontologies

A domain ontology is used to establish a semantic bridge in order to find semantic affinities between ontologies. Extraction procedures attached with domain ontologies apply data extraction techniques [9] to retrieve data and metadata when matching two ontologies. Each extraction procedure is designed for either a class or slot in a domain ontology. When an extraction procedure is invoked, a recognizer does the extraction by applying a set of extraction rules specified using regular expressions. Figure 2 shows the regular expressions using the Perl syntax for slot *View* and *Phone* in a real-estate domain.

Each list of regular expressions include declarations for data values that can potentially populate a class or slot and keywords that can be used as vocabulary terms to name classes and slots. We describe data values using *extract* clauses and the keywords using *keyword* clauses. When applied to an input ontology, both the *extract* and *keyword* clauses causes a string matching a regular expression to be extracted, where the string can be a vocabulary term in the ontological signature or a data values classified by the ontological ground facts.

2.3 Application of Domain Ontology

Figure 3 shows three components in a real-estate domain ontology, which we used to automate the mapping between two ontologies in Figure 1 and also for mapping real-world ontologies in the real-estate domain in general. Each box in Figure 3 associates with an extraction procedure. Filled-in (black) triangles denote aggregation (“PART-OF” relationships). And open (white) triangles denote generalization/specialization (“IS-A” superclasses and subclasses).

Provided with the domain ontology described in Figure 3, we can discover many semantic affinities between Ontology 1 in Figure 1(a) and Ontology 2 in Figure 1(b) as follows.

1. *Merged/Split Values.* Based on the *Address* declared in the ontology in Figure 3(a), the attached extraction procedure detects that (1) the values of *address* in Ontology 1 match with extraction patterns for concept *Address*, and (2) the values of *Street*, *City*, and *State* in Ontology 2 match with extraction patterns for concepts *Street*, *City*, and *State* respectively. Based on “PART-OF” relationships in Figure 3(a),

```

View matches [15] case insensitive
constant
    { extract "\bmountain\sview\b"; },
    { extract "\bwater\sfront\b"; },
    { extract "\briver\sview\b"; },
    { extract "\bpool\sview\b"; },
    { extract "\bgolf\s*course\b"; },
    { extract "\bcoastline\sview\b"; },
    ...
    { extract "\bgreenbelt\sview\b"; };
keyword
    "\bview(s)?\b";
End;
Phone matches [15] case insensitive
constant
    { extract "\b\d{3}-\d{4}\b"; }, - nnn-nnnn
    { extract "\b(\d{3})\s*\d{3}-\d{4}\b"; }, - (nnn) nnn-nnnn
    { extract "\b\d{3}-\d{3}-\d{4}\b"; }, - nnn-nnn-nnnn
    { extract "\b\d{3}\\\d{3}-\d{4}\b"; }, -nnn\nnn-nnnn
    { extract "\b1-\d{3}-\d{3}-\d{4}\b"; }, - 1-nnn-nnn-nnnn
Keyword
    "\bcall\b","bphone\b";
End;

```

Fig. 2. Example of regular expressions in a real-estate domain

we can find the “PART-OF” relationships between *Street*, *City*, and *State* in Ontology 2 and *address* in Ontology 1.

2. *Superset/Subset*. By calling extraction procedures attached with classes in Figure 3(b), *phone_day* in Ontology 1 matches both keywords and data value patterns for *Day Phone* and *phone* in Ontology 2 matches with *Phone*. In Figure 3(b) the ontology explicitly declares *Phone* is a superset of *Day Phone* based on the “IS-A” relationship between *Day Phone* and *Phone*. Thus we can find the semantic affinity between *phone_day* in Ontology 1 and *Phone* in Ontology 2.
3. *Vocabulary Terms and Data Instances*. Extraction procedures apply extraction patterns to recognize keywords and value patterns over both ontology terms and populated data instances since it is difficult to distinguish boundaries between meta-data and populated data instances in complex knowledge representation systems. In Ontology 1, “*water front*” is instance data populated for *view*. In Ontology 2, *Water-front* is a vocabulary term. Boolean values “Yes” and “No” for *Water-front* in Ontology 2 indicate whether the values *Water-front* should be included as description values for *view* of *House* in Ontology 1 if we map the two ontologies. The extraction procedure for concept *View* in Figure 3(c) recognizes terms such as *Water-front* in Ontology 2 as values and the procedure for concept *Water Front* can also recognize keyword “*water front*” associated with *view* in Ontology 1. Since *Water Front* “IS-A” *View* in Figure 3(c), by derivation, we can detect that *view* in Ontology 1 has a semantic affinity with *Water-front* in Ontology 2.

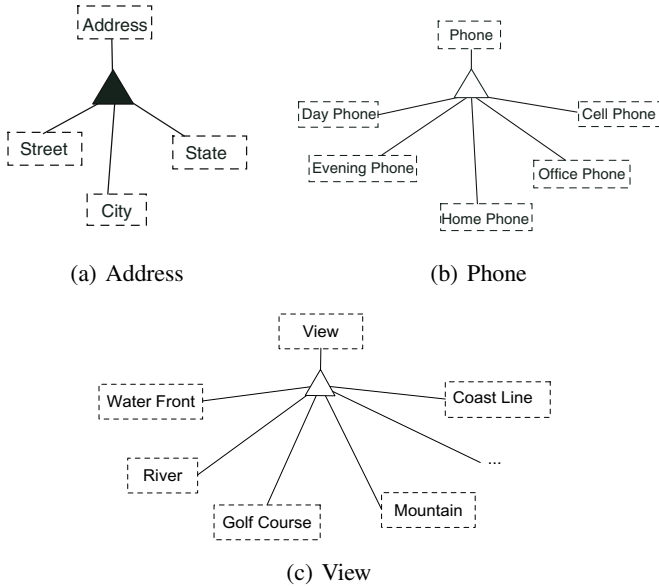


Fig. 3. Real-estate domain ontology (partial)

3 Mapping Result Representation

3.1 Source-to-Target Mapping

We adopt an ontology mapping definition [11] as follows.

Definition 3. A *source-to-target mapping* M_{ST} from $O_S = (S_S, A_S, F_S)$ to $O_T = (S_T, A_T, F_T)$ is a morphism $f(S'_S) = S'_T$ such that $A'_T \models f(A'_S)$, i.e. all interpretations that satisfy O'_T axioms also satisfy O'_S translated axioms if there exist two sub-ontologies $O'_S = (S'_S, A'_S, F'_S)$ ($S'_S \subseteq S_S, A'_S \subseteq A_S, F'_S \subseteq F_S$) and $O'_T = (S'_T, A'_T, F'_T)$ ($S'_T \subseteq S_T, A'_T \subseteq A_T, F'_T \subseteq F_T$). $\square$

Our representation solution for source-to-target mapping allows a variety of source derived data based on the discovered semantic affinities between two input ontologies. These source derive data include generalizations and specializations, merged and split values, translations between vocabulary terms and data instances, and etc. Therefore, our solution “extends” elements in an ontological signature S_S of a source ontology O_S by including views computed via queries, each of which we call a *view* element. We let V_S denote the extension of S_S with derived, source view elements.

Every source-to-target mapping M_{ST} is composed of a set of triples. Each triple $t = (e_t, e_s, q_e)$ is a *mapping element*, where $e_t \in S_T$, $e_s \in V_S$, q_e is either empty or a *mapping expression*. We call a triple $t = (e_t, e_s, q_e)$ a *direct match* which binds $e_s \in S_S$ to $e_t \in S_T$, or an *indirect match* which binds a view element $e_s \in V_S - S_S$ to $e_t \in S_T$. When a mapping element t is an indirect match, q_e is a *mapping expression* to illustrate how to compute the view element e_s over the source ontology O_S .

To represent source-to-target mapping as logic theories, we specify source-to-target mappings as populated instances of a mapping ontology, which we define as follows.

Definition 4. A mapping ontology $O = (S, A, F, P)$, where S is the ontological signature, A is the set of ontological axioms, F is a set of ground facts presenting source-to-target mappings, and P is a set of query procedures that describe designed query behaviors to compute views over ontologies. $\square$

If a mapping element $t = (e_t, e_s, q_e)$ in a source-to-target mapping M_{ST} is an indirect match, i.e. e_s is a source view element, a query procedure is attached with t to compute e_s by applying the mapping expression q_e .

3.2 Mapping Expressions

We can view each class and class slot (including view elements corresponding to either classes or slots) in ontologies as single-attribute or multiple-attribute relations. Relational algebra is therefore ready to be applied to describe procedural behaviors for query procedures. Since traditional operators in relational algebra do not cover the ones required to address problems such as *Merged/Split values* and *Vocabulary Terms and Data Instances*, we present mapping expressions by an extended relational algebra.

For example, to address *Merged/Split Values*, we design two operations *Composition* and *Decomposition* in the extended relational algebra, which we describe as follows. In the notation, a relation r has a set of attributes; $attr(r)$ denotes the set of attributes in r ; and $|r|$ denotes the number of tuples in r .

- *Composition* λ . The λ operator has the form $\lambda_{(A_1, \dots, A_n), A} r$ where each A_i , $1 \leq i \leq n$, is either an attribute of r or a string, and A is a new attribute. Applying this operation forms a new relation r' , where $attr(r') = attr(r) \cup \{A\}$ and $|r'| = |r|$. The value of A for tuple t on row l in r' is the concatenation, in the order specified, of the strings among the A_i 's and the string values for attributes among the A_i 's for tuple t' on row l in r .
- *Decomposition* γ . The γ operator has the form $\gamma_{A, A'}^R r$ where A is an attribute of r , and A' is a new attribute whose values are obtained from A values by applying a routine R . Applying this operation forms a new relation r' , where $attr(r') = attr(r) \cup \{A'\}$ and $|r'| = |r|$. The value of A' for tuple t on row l in r' is obtained by applying the routine R on the value of A for tuple t' on row l in r .

Assuming that Ontology 1 in Figure 1(a) is the target and Ontology 2 in Figure 1(b) is the source, the following lists the derivation of a view element *House – address'* in Ontology 2 that matches with *house – address* in Ontology 1.

$$\begin{aligned}
 Address - Address' &\Leftarrow \pi_{Address, Address'} \lambda_{(Street, ", ", City, ", ", State), Address'} (\\
 &\quad Address - Street \bowtie Address - City \bowtie Address - State) \\
 House - address' &\Leftarrow \rho_{Address' \leftarrow address'} \pi_{House, Address'} (House - Address \\
 &\quad \bowtie Address - Address')
 \end{aligned}$$

The λ operator denotes the *Composition* operation in the relational algebra. The Composition operation merges values in *Street*, *City* and *State* for a new concept *Address'*. The derivation also applies standard operations including *Natural Join* $\bowtie$, *Projection* π , and *Rename* ρ .

3.3 Semantic Interoperability

We define a semantic interoperable system as follows.

Definition 5. A *semantic interoperable system* $I = (O_T, \{O_{S_i}\}, \{M_{S_iT}\})$, where O_T is a target ontology, $\{O_{S_i}\}$ is a set of n source ontologies, and $\{M_{S_iT}\}$ is a set of n source-to-target mappings, such that for each source ontology O_{S_i} there is a mapping M_{S_iT} from O_{S_i} to O_T , $1 \leq i \leq n$. $\square$

Note that data instances $F_{O_{S_i} \rightarrow O_T}$ flowing from any source O_{S_i} to the target O_T based on M_{S_iT} hold classifications to either signature or view elements in O_{S_i} . Since a source-to-target mapping defines a morphism $f(S'_{O_{S_i}}) = S'_{O_T}$, the data instances $F_{O_{S_i} \rightarrow O_T}$ hence hold the classifications to the signature elements in O_T that correspond source elements in O_{S_i} .

The following theorem provides that accurate information exchange between ontologies is guaranteed by derived source-to-target mappings.

Theorem 1. Given a semantic interoperable system $I = (O_T, \{O_{S_i}\}, \{M_{S_iT}\})$ where $1 \leq i \leq n$, data instances $F_{O_{S_i} \rightarrow O_T}$ flowing from O_{S_i} to O_T by M_{S_iT} hold and are provable by O_T . $\square$

Assume that user queries issued over I are Select-Project-Join queries and we also assume that they do not contain comparison predicates such as $\leq$ and $\neq$. We use the following standard notation for conjunctive queries.

$$Q(\overline{X}) : -P_1(\overline{X_1}), \dots, P_n(\overline{X_n})$$

Where $\overline{X}$, $\overline{X_1}, \dots, \overline{X_n}$ are tuples of variables, $\overline{X} \subseteq \overline{X_1} \cup \dots \cup \overline{X_n}$, and predicate P_i ($1 \leq i \leq n$) is a target signature element.

When evaluating query answers for a user query Q , the semantic interoperable system I transparently reformulates Q as Q^{ext} , a query over the target and source ontologies in I . The source-to-target mapping instances lead automatically to a rewriting of every target element as a union of the target element and corresponding source elements. Query reformulation thus reduces to rule unfolding by applying the view definition expressions for the target elements in the same way database systems apply view definitions.

With query reformulation in place, we can now prove that query answers are *sound*—every answer to a user query Q is an entailed fact according to the source(s) and the target—and that query answers contain all the entailed facts for Q that the sources and the target have to offer—*maximal* for the query reformulation.

Theorem 2. Let Q_I^{ext} be the query answers obtained by evaluating Q^{Ext} over I . Given a user query Q over I , a tuple $\langle a_1, a_2, \dots, a_M \rangle$ in Q_I^{Ext} is a sound answer for Q . $\square$

Theorem 3. Let Q_I^{ext} be the query answers obtained by evaluating Q^{Ext} over I . Given a user query Q over I , Q_I^{Ext} is maximal for Q with respect to I . $\square$

4 Experimental Result

We used a real-world application, *Real Estate*, to evaluate applications of a domain ontology shared by a set of matching technique [18]. The *Real Estate* application has five ontologies. We decided to let any one of the ontologies be the target and let any other ontology be the source. In summary, we tested 20 pairs of ontologies for the *Real Estate* application. In the test, *Merged/Split Values* appear four times, *Superset/Subset* appear 48 times, and *Vocabulary Terms and Data Instances* appear 10 times. With all other indirect and direct matches, there are a total of 876 matches. We evaluated the performance of our approach based on three measures: precision, recall and the F-measure, a standard measure for recall and precision together [1]. By exploiting knowledge specified in the domain ontologies attached with extraction procedures, the performance reached 94% recall, 90% precision, and an F-measure of 92%³.

One obvious limitation to our approach is the need to manually construct an application-specific domain ontology with extraction procedures⁴. To facilitate the knowledge acquiring process to build domain ontologies, we can reuse existing ontologies. Machine learning techniques can also be applied to facilitate the construction of extraction patterns for extraction procedures. Since we predefine a domain ontology for a particular application, we can compare any two ontologies for the application using the same domain ontology. Therefore, the work of creating a domain ontology is amortized over repeated usage.

5 Conclusions

We have proposed an approach to automate and represent ontology mappings by combining both declarative and procedural representations. With experiments we have shown that a set of base techniques is able to establish communications via domain ontologies attached with extraction procedures. By sharing domain ontologies, the base techniques detected indirect matches related to problems such as *Superset/Subset*, *Merged/Split values*, as well as *Vocabulary Terms and Data Instances*. To approach semantic interoperability across ontologies, we present source-to-target mappings as mapping ontologies attached with query procedures, which not only support equivalent inferences and computations on equivalent concepts and relations but also improve query performance by applying query procedures.

References

1. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, Menlo Park, California, 1999.
2. J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *Proceedings of the International Conference on Advanced Information Systems Engineering (CAISE 2002)*, pages 452–466, Toronto Canada, 2002.

³ See a detailed explanation about the experiment in [18].

⁴ Experience has shown that computer science students can build a domain ontology with extraction rules for a data-rich, narrow domain of interest in a few dozen person hours. Students have built ontologies for a wide variety of applications including such diverse areas such as digital cameras, prescription drugs, campgrounds, and computer jobs [5].

3. D. Calvanese, G. De Giacomo, and M. Lenzerini. A framework for ontology integration. In *Proceedings of the 1st Internationally Semantic Web Working Symposium (SWWS)*, pages 303–317, 2001.
4. V.K. Chaudhri, A. Farquhar, R. Fikes, P.D. Karp, and J.P. Rice. OKBC: a programmatic foundation for knowledge base interoperability. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, Madison, Wisconsin, 1998.
5. Demos page for BYU data extraction group. <http://www.deg.byu.edu/multidemos.html>.
6. R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. iMAP: Discovering complex semantic matches between database schemas. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data (SIGMOD 2004)*, pages 283–294, Paris, France, June 2004.
7. H. Do and E. Rahm. COMA - a system for flexible combination of schema matching approaches. In *Proceedings of the 28th International Conference on Very Large Databases (VLDB)*, pages 610–621, Hong Kong, China, August 2002.
8. A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy. Learning to match ontologies on the semantic web. *VLDB Journal*, 12:303–319, 2003.
9. D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith. Conceptual-model-based data extraction from multiple-record Web pages. *Data & Knowledge Engineering*, 31(3):227–251, November 1999.
10. A.Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, December 2001.
11. Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.
12. W. Li and C. Clifton. SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data & Knowledge Engineering*, 33(1):49–84, April 2000.
13. J. Madhavan, P.A. Bernstein, A. Doan, and A. Halevy. Corpus-based schema matching. In *ICDT'05*, y, 2005.
14. J. Madhavan, P.A. Bernstein, P. Domingos, and A. Halevy. Representing and reasoning about mappings between domain models. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI'02)*, 2002.
15. A. Maedche, B. Motic, N. Silva, and R. Volz. Mafra - an ontology mapping framework in the semantic web. In *Proceedings of the ECAI Workshop on Knowledge Transformation*, Lyon, France, July 2002.
16. E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, December 2001.
17. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, Inc., second edition edition, 2003.
18. L. Xu and D.W. Embley. A composite approach to automating direct and indirect schema mappings. *Information Systems*, available online April 2005.

Visual Ontology Alignment for Semantic Web Applications

Jennifer Sampson and Monika Lanzemberger*

Norwegian University of Science and Technology, Vienna University of Technology
sampsonj@idi.ntnu.no, lanzemberger@ifs.tuwien.ac.at

Abstract. Ontologies play an important role for the semantic web because they aim at capturing domain knowledge in a generic way and provide a consensual understanding of a domain. Due to the number of ontologies available the need for mapping or bringing them into alignment has prompted a surge of tools and algorithms to be developed for this purpose. We describe some of these ontology alignment tools and discuss issues and shortcomings in current state of the art. From this analysis we propose the use of visualization techniques to facilitate understanding of the ontology alignment results. Finally we briefly describe AlViz, our visual ontology alignment tool.

1 Introduction

Ontology alignment is the process where for each entity in one ontology we try to find a corresponding entity in the second ontology with the same or the closest meaning. We align ontologies to share common understanding of the structure of information among people or software agents. While it may be possible to introduce standards for ontology languages, it is impractical to promote the use of a global ontology or knowledge base within a community of interest. It is impossible for human experts in scientific domains to reach full and complete agreement, so in these cases it becomes more desirable that a system can help resolve inconsistencies. In our research we will address the following research questions related to ontology alignment: (1) Which alignment tools exist and how do they perform the necessary tasks? (2) What are the main issues and shortcomings of current ontology alignment tools? (3) How can visualization support the ontology engineers' understanding and interpretation of alignment results?

In the next section we will briefly describe some of the ontology management tools.

2 Ontology Alignment State of the Art

We compared a number of the ontology management tools [1,2,3,4,5,6,7,8] to help establish requirements for our new tool for supporting ontology alignment. The results are shown in Table 1.

* Part of this work was completed while M. Lanzemberger was an ERCIM research fellow at IDI, Norwegian University of Science and Technology (Trondheim, Norway) and CITI, Centre de Recherche Public Henri Tudor (Luxembourg).

Table 1. State of the art ontology management tools

Alignment Tool	Prompt Suite	Chimæra	ONION	GLUE	FCA	FOAM	IF-MAP	Momis
Developed by	Stanford Medical Informatics	KSL Stanford University	Mitra Wiederhold	Uni of Washington	Stumme & Maedche	Uni of Karlsruhe	Uni of South Hampton	Uni Modena and Reggio Emilia
Purpose	Ontology Merging & Mapping	Ontology Merging	Information Integration	Database Schema	Ontology Mapping	Ontology Alignment	Ontology Merging Mapping theory	Information Integration
Input	Protégé Knowledge Model OKBC	OKBC compliant	ONION ontology format	Relational	Frame logic, Text documents	OWL Lite	Reference ontology	ODL
Concepts	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Properties	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Instances	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Structure	Yes	Yes	Yes	Yes	No	Yes	Yes	No
User Input	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Lexicon or auxiliary information	No	No	WordNet	Training Matches	Extraction of instances from text	Machine learning	No	DL Reasoner Wordnet Artemis
Cardinality	1:n	1:1	1:1	1:1	1:1	1:1/1:n	1:1/1:n	n:1
Visualization	PromptViz	No	No	No	Node-link visualization	No	No	Classes and attributes diagram
Support	Treemap layout					No	No	
Output	Merged ontology	Merged ontology	Articulation rules	Mappings for two ontologies	XML merged ontology	Mapped entity pairs,	Merged ontology in RDF	Merged ontology
Evaluation	Precision recall f-measure Experimental	Empirical evaluation	Precision recall for corpus based word relations	Empirical evaluation using real world examples	Precision recall Experimental	Precision recall f-measure Experimental	Experimental	Application of Artemis in a research project

Prompt [1] is an ontology merging tool that takes two ontologies and creates a single merged ontology. Prompt is an interactive tool which allows the user to give some information about relatedness of concepts and uses this information to determine the next operation and to identify potential conflicts. The tool relies on a linguistic matcher to give initial suggestions of potential mappings which are then refined and updated in later stages. Prompt leaves the user in control of making the decisions, it makes suggestions that a user can choose to follow or not. An extended version of Prompt, AnchorPrompt [9] considers similarity between class hierarchies, and the use of similarity scores in determining the potential match between classes. The main idea behind the AnchorPrompt algorithm is that if two pairs of terms from source ontologies are similar and there are paths connecting the terms, then the elements in those paths are often similar as well [9].

Chimæra [2] is another interactive ontology merging tool, similar to Prompt, however, since it uses only a class hierarchy in its analysis it does not locate all the mappings that Prompt establishes. When using Chimæra a user can bring together ontologies developed in different formalisms, during which s/he can request an analysis or guidance from the tool. Chimæra can, for example, point out to the user a class in the merged ontology that has two slots obtained from the different source ontologies or has two subclasses that came from the different ontologies. Foam is an ontology mapping algorithm that combines different similarity measures to find one-to-one mapping candidates between two ontologies. In [10] the authors describe how a range of similarity measures are used in Foam [6] for finding alignments between entities ranging in increasing semantic complexity. Once the individual measures are calculated for each entity (e.g., URI, RDF/S primitives or domain specific features) they are input to the similarity aggregation using a sigmoid function. Foam produces as output a set of actual mappings and a set of questionable mappings that require user verification.

GLUE [4] is a system that uses machine-learning techniques to semi automatically find mappings between heterogeneous ontologies. The multiple learners within the GLUE system use concept instances and the taxonomic structure of ontologies. GLUE uses a probabilistic model to combine the results from the different learners. The result of the matching is a set of similarity measures, specifying whether concepts in one ontology O_1 are similar to concepts in the other ontology O_2 . Onion (ONtology compositiON) [3] is another architecture which supports a scalable framework for ontology integration. In [16] the authors claim that the main innovation in ONION is that it uses articulations of ontologies to interoperate among ontologies. An articulation generation function in ONION is a matching or alignment algorithm between ontologies.

Through an evaluation of the literature and analysis of current tools for ontology management we have identified several research issues. First, there is a general lack of consensus in the literature on terminology for ontology management. This process has been referred to as: ontology mapping, ontology alignment, ontology integration and ontology merging. However, there are subtle differences between the different terms. Alignment may often include a transformation of two ontologies removing the unnecessary information and integrating missing information. In contrast to mapping, ontology alignment may result in changes to one or more of the ontologies. Ontology merging and ontology integration are closely related. Ontology integration is where we build a new ontology reusing other available ontologies which can be assembled, extended and specialized. Ontology merging is where two different ontologies within the same domain are merged into a single unifying ontology, the input ontologies usually have overlapping parts.

In addition [11] commented that understanding semantic relations between entities of different ontologies is a cognitively difficult task. In order to align domain ontologies an ontology developer or end user is required to define the mapping between the ontologies either manually or by using a semi automated

tool. Ontology mapping is largely a human-mediated process, although there are numerous tools, which can help with identifying differences between ontologies. The reason being that most of the kinds of conflicts or mismatches discovered by such tools require a human to recognize that different symbols have been used to represent the same concept, or that the same symbols may sometimes represent different concepts. Moreover, the degree of automation of the alignment algorithm is important especially for web service composition and agent communication, however ontology alignment will never be completely automatic, some degree of human validation is required. There is also a lack of empirical validation of ontology alignment tools using real world ontologies. Existing tests for ontology alignment tools focus on light weight ontologies, often with no properties or instances. Experiments on a broader scale need to be initiated over wider domains to evaluate the scalability of the existing approaches. This also corresponds to the lack of gold standard ontologies to be used as reference ontologies.

Furthermore, from an analysis of current ontology tools, we find that the presentation of ontology alignment results is limited. Text based listings of correspondences do not adequately highlight the intricate relationships between entities. Consequently it is difficult to understand the semantic relationship between entities and the semantic meaning of the mapping. And it is difficult to estimate the impact of mapping decisions made during the alignment process. Many ontology alignment tools have not been evaluated with respect to run time complexity. As a result many research prototype alignment tools suffer from problems with scalability and algorithm complexity. Although [12] recently demonstrated that a combination of similarity functions applied to a sigmoid function reaches high quality mapping levels very quickly.

Our survey showed so far only a few alignment tools implemented interactive functionality to support the required user interaction suitably. In particular, we identified two relevant tools among the numerous efforts in ontology management: OLA [13] and to a certain degree Prompt-Viz [14] are related to our work. In general, current ontology visualization relies on simple types such as two-dimensional trees or graphs where nodes represent concepts and the edges in-between show one specific relationship among these concepts. Further properties are usually displayed as textual components within the visualization. Whereas OLA uses such two-dimensional graphs enriched with additional cues, Prompt-Viz adopts a tree map [15] which helps the user in gaining overview. Our literature study indicated a broad interpretation of ontology visualization differing among the various tools resulting in an unclear discrimination between text and visualization. To avoid confusion we start with a definition of information visualization suitable for the context of our research. Information visualization uses visual metaphors to ease the interpretation and understanding of multi-dimensional data in order to provide the user with relevant information. Graphical primitives such as point, line, area or volume are utilized to encode information by position in space, size, shape, orientation, color, texture, and

other visual cues, connections & enclosures, temporal changes, and viewpoint transformations [16].

3 Alignment Relation Categories

We have identified categories of relations that can exist between entities in two ontologies O_1 and O_2 . The purpose was to provide categories suggesting alignment type and thereby providing an ontology engineer or group of domain experts information about the candidate alignments. Our categories for classifying alignment types are shown at Table 2. The basic notion is that we are providing an explanation for ‘why’ there is an alignment between entities and using information visualization techniques as the transportation mechanism for this explanation. Moreover, we recognize that there may be several ways in which a pair of entities x, y from two ontologies O_1, O_2 may be related, e.g., syntactically equal and similar.

Table 2. Categories of relations between entities in ontologies

OWL Ontology Construct	Comparison Relationship	Description
Concept	Equal	The concept identifiers are the same.
	Syntactically equal	Concept names are the same.
	Similar	Concepts can be similar in a number of different ways. e.g., Super and subclasses may be the same, or concept properties may be the same.
	Broader than	One concept may be broader than another.
	Narrower than	One concept may be narrower than another.
Data Properties	Different	Two classes have no common characteristics.
	Equal	Data property identifiers are the same.
	Syntactically Equal	Data property names are the same.
	Similar	Data properties can be similar in a number of ways. e.g., Data super or sub properties are the same, or data members are the same.
Object Properties	Different	Two data properties have no common characteristics.
	Equal	The object property identifiers are the same.
	Syntactically Equal	Object property names are the same.
	Similar	Object properties may be similar in a number of ways. e.g., Object super and sub properties are the same, or object properties members are the same.
Instance	Different	Two object properties have no common characteristics.
	Equal	Instance identifiers are the same.
	Syntactically Equal	Instance names are the same.
	Similar	Instances may be similar in a number of ways. e.g., Instance properties are the same, or related by the same parent concept.
	Different	Two instances have no common characteristics.

For the purposes of implementation we found that Foam [6] algorithm implemented a large number of rules and within the rule set we were able to make correspondences between our categories and a combination of the rules in the Foam algorithm. Within each of these categories we specify how the rules determine whether a concept is equal to another concept, syntactically equal, similar or different to another concept (likewise for properties and instances). For example, one indicator that establishes whether two concepts are equal is the fact that they both share the same instances, another indicator is whether or not they have the same URI. We can say that it is probable that two concepts are the same when the labels are the same, in this case we say they are syntactically equal. Furthermore, entities can be defined as being broader than or narrower than other entities in the hierarchy. We define that concept x of ontology 1 is broader than concept y of ontology 2, where $((x, r \in O_1)(y, w \in O_2))$ and

$$\text{Broader than}(x, y) := (r \sqsubseteq x) \wedge (y \sqsubseteq w) \wedge (r = w)$$

Similarly, if the superclass of concept x is similar to the subclass of concept y we deem concept x to be narrower than entity y . We define that concept x of ontology 1 is narrower than concept y of ontology 2, where $((x, z \in O_1)(y, t \in O_2))$ and

$$\text{Narrower than}(x, y) := (x \sqsubseteq z) \wedge (t \sqsubseteq y) \wedge (z = t)$$

Our tool AlViz provides a visualization of these different categories of associations between entities in related ontologies, next we briefly describe AlViz.

4 AlViz – A Visualization Tool for Ontology Alignment

AlViz, is implemented as multiple-view plug-in for Protégé [17] based on a available solution [18], which transforms the original small world algorithm from 3 to 2 dimensions. The tool consists of two types of views coupled by the linking and brushing technique. AlViz applies J-Trees as one out of two types of views. Such trees consist of a root node, expanded or collapsed branch nodes and leaf nodes displaying the hierarchical structure by indentation. They support the access and manipulation of instances and single items within classes quite effectively and are well established within the Protégé community. We integrate another visualization type: small world graphs [19]. Therefore, as a second view, graphs help the user to examine the structure of the ontology intuitively. This method uses clusters to group the nodes of a graph according to the selected level of detail. The nodes represent the entities (concepts or instances) connected to each other according to the selected relations, also called mutual properties, such as **IsA**, **IsPart**, **IsMember**, **locatedIn**, **IsDistinct**. So, each source ontology is visualized as a clustered graph where the edges represent the selected mutual property (or a cumulation of properties is possible as well). Our tool provides zooming in and out capabilities which allows a user to explore the ontology on different levels of detail. In fig. 1 and fig. 2 we show the same source ontology visualized at different levels of detail. Clustering emphasizes the structure of the

ontology (refer fig. 2). Although the small world graphs like all spring-embedded algorithms bear the problem of high computational complexity - usually $O(N^3)$, clustering the graph improves program's interactivity. The tool is fast enough to perform at interactive speeds because on average there are only $O(\text{Log}(N))$ clusters visible. Our current solution manages up to about 1000 entities per ontology.

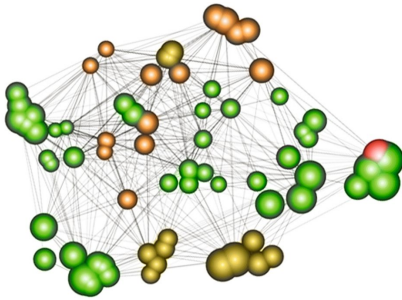


Fig. 1. Small world graph for a Tourism ontology

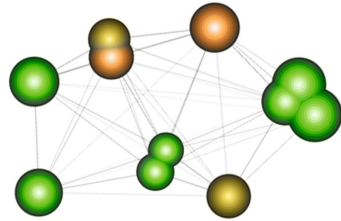


Fig. 2. Highly clustered small world graph of the same Tourism ontology

4.1 Visual Ontology Alignment of ISO 15926

A project was undertaken to transform the ISO 15926 Core Data Model into the web ontology language (OWL). In order to achieve this, two alternative models were created for representing various data model entity types as OWL classes and properties. The ISO 15926-2 is a data model with three key concepts: individuals, classes and relationships. The classes have few, if any, attributes, and all characteristics of an individual or a class are assigned via relationships. Relationships are also used to establish relations like inheritance, composition, inclusion, and classification [20]. The first ontology contained 246 classes, 18 datatype properties and 102 object properties. The second ontology contained 156 classes, 17 datatype properties and 115 object properties. We align these ontologies to establish relations between the two different versions. To improve the understanding and interpretation of the alignments results we unravel the combined similarities into the similarity types for each entity. The visualization of the alignment of the two ontologies using AlViz is shown at fig. 3. The visualization of the ISO ontologies show us that ontologies with only IsA relations may result in quite simple graphs in terms of the number of edges. The clustered graphs indicate that the ISO ontologies are quite similar overall but that ISO 1 contains more knowledge than ISO 2 represented by an additional subgraph. Such additional subgraphs are perceived easily and typically they are colored green (similar) or yellow (different) because neither equal nor syntactically equal associates can be found in the second ontology. The J-Trees on the left hand side show the details. Also, since the Foam algorithm compares the `rdf:label` of entities and not the entity name, there may be situations where the relation syntactically equal is

not rated. This occurs because often the entity `rdf:label` is missing or different to the entity name. In this case the j-trees can help indicate where some labels should be adapted.

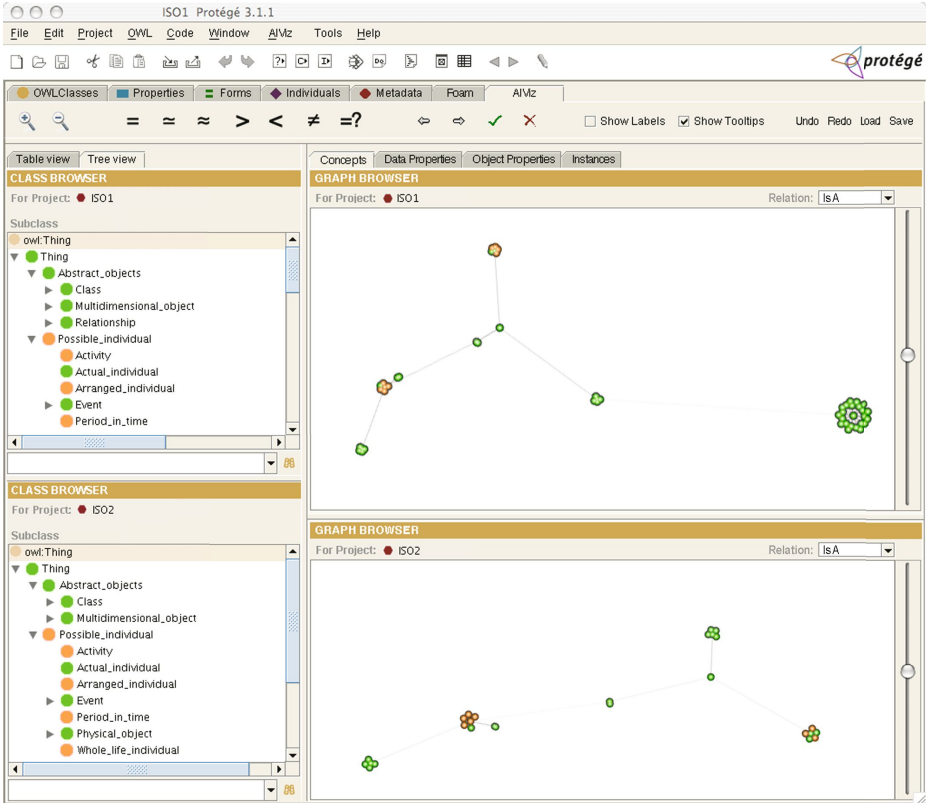


Fig. 3. AIViz: the four views of the tool visualize two ontologies named ISO1 and ISO2. The nodes of the graphs and dots next to the list entries represent the similarity of the ontologies by color. The size of the nodes results from the number of clustered concepts. The graphs show the `IsA` relationship among the concepts. Green indicates similar concepts available in both ontologies, whereas orange nodes represent syntactically-equal concepts. The sliders to the right adjust the level of clustering.

5 Conclusion

We completed a state of the art survey on ontology alignment tools, in an effort to understand current algorithms and shortcomings of the existing approaches. Our findings suggest that one of the main problems with current approaches is associated with the volume and complexity of the results produced by these algorithms. We have determined a number of research issues from the state of the art on visualization of ontologies: first, many tools exist for visualizing ontologies,

however there are very few visualization tools for assisting with ontology alignment. Second, existing tools for visualizing ontologies lack support for visualizing certain ontology characteristics or are designed for a specific range of queries. Third, time complexity of the layout algorithm needs to be low to maintain good interaction for the users. Fourth, the layout needs to make efficient use of the screen 'real estate' so large numbers of concepts can be displayed simultaneously. Fifth, in order to support an overview and detail approach appropriately, multiple views are needed for ontology alignment. To facilitate user understanding of the meaning of the ontology alignment we propose the use of visualization techniques to graphically display data from ontology mappings. Our tool, Alviz, is to help the user determine the following: 1) Location: Where do most of the mappings between ontologies occur? 2) Impact: Do the mapping choices directly or indirectly affect parts of the ontology the user is concerned about? 3) Type: What kinds of alignments occur between the ontologies? 4) Extent: How different is the aligned ontology to the source ontologies? By exploring such questions in a multiple-view visualization the user may be able to understand and enhance the alignment results. While we recognize that much work remains to be done, initial results using our tool are promising.

Acknowledgement. This work is partially supported by the Norwegian Research Foundation in the Framework of Information and Communication Technology (IKT-2010) program - the ADIS project.

References

1. Noy, N.F. and Musen, M.A.: 'The PROMPT Suite: Interactive Tools for Ontology Merging and Mapping. *International Journal of Human-Computer Studies*, 59(6):983–1024. (2003).
2. McGuinness, D. Fikes, R, Rice, J. and Wilder, S. An Environment for Merging and Testing Large Ontologies, 7th International Conference on Principles of Knowledge Representation and Reasoning, (2000).
3. Mitra, P. Kersten, M. and Wiederhold, G. Graph Oriented Model for Articulation of Ontology Independencies, 7th International Conference Extending Databases Technology, (2000).
4. Doan, A. Madhavan, J. Domingos, P and Halevy, A.: Ontology Matching: A machine learning approach, In *Handbook on Ontologies*, Staab, S. and Studer, R. (eds), (2004).
5. Stumme, G. and Maedche, A.: FCA-Merge: Bottom up merging of ontologies, 7th International Conference on Artificial Intelligence, (2001).
6. Foam.: Framework for Ontology Alignment and Mapping <http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/> (2005).
7. Kalfoglou, Y. Schorlemmer, M.: Information Flow Based Ontology Mappings, 1st International Conference on Ontologies, Databases and Application of Semantics, (2002).
8. Semantic Integration of Semistructured and Structured Data Sources. *SIGMOD Record*, 28(1):54–59. (1999).

9. Noy, N.F. and Musen, M.A.: Anchor-Prompt: Using Non Local Context for Semantic Matching, Workshop on Ontologies and Information Sharing at IJCAI-2001, (2001).
10. Ehrig, M., Haase, P., Stojanovic, N, Hefke, M.: Similarity for Ontologies - A Comprehensive Framework 13th European Conference on Information Systems. (2005).
11. Noy, N.: Ontology Management with the Prompt Plugin, 7th International Protégé Conference, (2004).
12. Ehrig, M. and Staab, S.: Efficiency of Ontology Mapping Approaches, International Workshop on Semantic Intelligent Middleware for the Web and the Grid, (2004).
13. Euzénat, J., Loup, D., Touzani, M. and Valtchev, P.: Ontology Alignment with OLA. 3rd EON Workshop, 3rd International Semantic Web Conference. (2004).
14. Perrin, D.: PROMPT-Viz: Ontology Version Comparison Visualizations with Treemaps. M.Sc. Thesis, University of Victoria. (2004).
15. Shneiderman, B.: Tree Visualization with Tree-maps: A 2-D Space-filling Approach. ACM Transactions on Graphics, 11(1), (1992).
16. Card, S.K., Mackinlay, J.D. and Shneiderman, B. (Eds.): Readings in information visualization. Morgan Kauffman, (1999).
17. Gennari, J. H. Musen, M. A. Ferguson, R. W. Grosso, W. E. Crubezy, M. Eriksen, H. Noy, N. F. and Tu S. W.: The Evolution of Protégé: An Environment for Knowledge Based Systems Development, International Journal of Human Computer Studies, 58(1), 89–123, (2003).
18. Ingram, S.F.: An Interactive Small World Graph Visualization. University of British Columbia, Technical Report, (2005).
19. van Ham, F. and van Wijk, J.J.: Interactive Visualization of small world graphs. In Proc. of IEEE Symposium on Information Visualization. (2004).
20. Christiansen, T. Jensen, M. Leal, D. Price, D and Valen-Sendstad.: Representing the ISO 15926-2 Core Data Model in OWL, Version 1.3. Available at: <http://www.olf.no/io/kunnskapsind/?28132> (2005).

Automatic Creation of Web Services from Extraction Ontologies

Cui Tao, Yihong Ding, and Deryle Lonsdale

Brigham Young University, Provo, Utah 84602, U.S.A.
{ctao, ding}@cs.byu.edu, lonz@byu.edu

Abstract. The Semantic Web promises to provide timely, targeted access to user-specified information online. Though standardized services exist for performing this work, specifying these services is too complex for most people. Annotating these services is also problematic. A similar situation exists for traditional information extraction, where ontologies are increasingly used to specify information used by various extraction methods. The approach we introduce in this paper involves converting such ontologies into executable Java code. These APIs act individually or compositionally as services for Semantic Web extraction.

1 Introduction

One goal of the Semantic Web is to enable personalized, automatic, targeted access to information that can be useful to a user. This might include finding out the availability and price of a book, or reserving and ticketing travel itineraries. Currently tools to provide such services are in their infancy, but much human intervention is needed to hard-code and hand-specify their functionality.

In order to increase the use of services with the Semantic Web, we need a more automatic way of making them machine-interpretable, and we need to annotate them with a more systematic description of their respective semantic domains, conceptual coverage, and executable functions. This goal is perhaps optimally realized when a Web service is annotated by an ontology. An ontology enables domain experts to declare standardized, sharable machine-processable knowledge. In a traditional Web setting we have found ontologies useful for information extraction applications. In this paper we show how ontologies can serve two useful purposes in the creation of Semantic Web services.

First, we map an extraction ontology to a set of Java APIs, through which we can automatically create atomic information extraction Web services. Each of these atomic Web services instantiates a lexical ontology concept; domains with extensive vocabularies could spawn a considerable number of derived services. Since each atomic service directly derives from formal definitions in an ontology, we can use this information to sidestep the traditional requirement for hand-annotation of the service, a difficult process. Second, given these Java-instantiated services and their corresponding ontological properties, we can automatically compose complex Web services to respond to users' requests.

Our goal is thus to develop a method of automatically creating Web services based on extraction ontologies. In this paper we begin by introducing extraction ontologies. In the next section we sketch the process of compiling these ontologies into executable Java classes. Next we discuss web services and their creation via composition of these classes. Finally, we mention future work and applications.

2 Extraction Ontologies

An extraction ontology is a conceptual-model instance that serves as a wrapper for a narrow domain of interest such as car advertisements. We use OSM [3] as the semantic data model for an extraction ontology; we also augment OSM to allow regular expressions as descriptors for constants and context keywords. The conceptual-model instance includes objects, relationships, constraints over these objects and relationships, descriptions of strings for lexical objects, and keywords denoting the presence of objects and relationships among objects. When we apply an extraction ontology to a source document, the ontology identifies instantiated objects and relationships and associates them with named object sets and relationship sets in the ontology. This wrapping of recognized strings makes them machine-interpretable in terms of the schema inherent in the conceptual-model instance.

In essence, an extraction ontology is semantically equivalent to a Semantic Web ontology written in OWL¹. We are not using OWL directly because our OSM extraction ontology contains formal specifications for data extraction patterns beyond standard OWL. To be compatible with the Semantic Web standards, we have developed an OWL-OSM converter that transforms the two ontological representations; however, this ontology conversion research is beyond the scope of this paper.

An extraction ontology consists of two components: (1) an object/relationship-model instance that describes sets of objects, sets of relationships among objects, and constraints over object and relationship sets, and (2) for each object set, a data frame that defines the relevant extraction patterns. Figure 1 shows part of our car-ads extraction ontology, including object/relationship model declarations (Lines 1-8) and sample data frames (Lines 9-18).

An object set in an extraction ontology represents a set of objects which may either be lexical or non-lexical. Data frames with declarations for constants that can potentially populate the object set represent lexical object sets, and data frames without constant declarations represent non-lexical object sets. *Car*, for example, is a non-lexical object set. *Year* (Line 9) and *Mileage* (Line 14) are lexical object sets whose character representations have a maximum length of 4 characters and 8 characters respectively. We describe the constant lexical objects and the keywords for an object set by regular expressions using Perl-like syntax. We denote a relationship set by a name that includes its object-set names (e.g. *Car has Year* in Line 2 and *PhoneNr is for Car* in Line 8). The *min:max* pairs, such as *1:**, *0:**, or *0:1*, in the relationship-set name are participation

¹ Web Ontology Language (OWL), <http://www.w3.org/2004/OWL/>

```

1. Car [-> object];
2. Car [0:1] has Year [1:*];
3. Car [0:1] has Make [1:*];
4. Car [0:1] has Model|Trim [1:*];
5. Car [0:1] has Mileage [1:*];
6. Car [0:*] has Feature [1:*];
7. Car [0:1] has Price [1:*];
8. PhoneNr [1:*] is for Car [0:1];
9. Year matches [4]
10.     constant {extract "\d{2}";
11.               context "\b'[4-9]\d\b";
12.               substitute "^" -> "19"; },
13.     ...
14. Mileage matches [8]
15.     ...
16.     keyword "\bmiles\b", "\bmi\.", "\bmi\b",
17.             "\bmileage\b", "\bodometer\b";
18. ...

```

Fig. 1. Car-Ads Extraction Ontology (Partial)

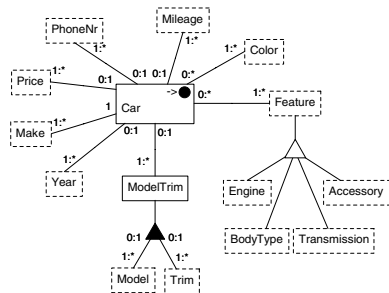


Fig. 2. The Graphical View of the Car Ontology

constraints. *Min* designates the minimum number of times an object in the object set can participate in the relationship set and *max* designates the maximum number of times an object can participate, with * designating an unspecified maximum number of times.

Figure 2 shows the equivalent graphical view of the ontology in Figure 1. A dashed box represents a lexical object set and a solid box represent a non-lexical object set. Lines between object sets represent the relationship sets among them and the digit pairs on both ends of the lines represent participation constraints. A black triangle represents an aggregation which constitutes a part/subpart relationship. In Figure 2, for the *Model|Trim* aggregation, we implicitly have the binary relationship sets *Model* is part of *Model|Trim* and *Trim* is part of *Model|Trim*. OSM uses a clear triangle to denote a generalization/specialization and connects a generalization at an apex of the triangle and to a specialization at the opposite base. In Figure 2, *Feature* is the generalization of *Engine*, *BodyType*, *Accessory*, and *Transmission*. In this research, we convert the knowledge represented in OSM to the Java programming language. The next section discusses this translation process.

3 From Extraction Ontologies to Java

We have developed an ontology compiler that translates from an ontology language (in this paper, we use OSM-L, which is a language for the OSM model) to Java. This compiler takes an extraction ontology as input, and outputs Java APIs automatically. It also helps ontology writers to find syntax errors which are usually hard to find by visual inspection. Other research such as [7] and [2] have also tried to automatically translate an ontology such as OWL or RDF to Java APIs. These approaches, however, only focus on mapping—not compiling—certain desirable properties from the ontologies to corresponding executable Java code. In addition, they do not provide translating ontologies to Web services.

In the rest of this section we discuss how we use Java classes and methods to describe the knowledge represented in an ontology. In particular, we present our use of Java to describe the five major components of an extraction ontology: object sets, data frames, relationship sets, aggregation, and generalization/specialization. The first two map to atomic Web services (Section 4.1), and the rest to complex Web services (Section 4.2).

Object Set. An object set describes information about a concept in a source ontology. In our system, an interface is generated automatically for each concept in the source ontology. We choose to use interfaces because Java only supports multiple inheritance through interfaces. An interface, however, can only provide static variables and abstract methods. We also design the compiler to generate an implemented class for each interface. For the car ontology in Figure 2, for example, there are 15 concepts. The ontology compiler generates 15 interfaces as well as 15 implemented classes. All of these interfaces use the same template as Figure 3 (a) shows². Each interface declares five static data fields. The *name* variable stores the concept's name. The *type* variable specifies whether the concept is a primary concept. The *length* variable corresponds to the matching length in the ontology. The *dataFrame* vector stores all the extraction rules defined by the ontology and the *relationSet* stores all the relationships between this concept and any other concepts. There are also three static methods: *recognize*, *listRelation*, and *checkSuperClass*. The first is to recognize this concept from a source document depending on the extraction rules defined in the data frame, which we will discuss in Data Frame section. The second method is to store all the relationships that this concept participates in. All such relationships are stored in a vector called *relationSet*, which we will discuss in the Relationship Set section. The third one is to check all the generalization concepts of a concept, which we will discuss in the Generalization/Specialization section.

Data Frame. There are two kinds of object sets: lexical objects and non-lexical ones. A non-lexical object set describes an abstract concept that does not have any value to extract, just like an interface cannot be instantiated. For a non-lexical object, its *dataFrame* is *null*, and its *recognize* method only has an empty body in the implemented class. For lexical objects, on the other hand, we want to save the information from their data frames and implement the *recognize* methods.

A data frame contains a set of extraction rules. Each extraction rule is an instance of the *BaseDFRule* class. Figure 3(b) shows the UML diagram of the

² The interface/class for each concept has its own name (the concept name for the interface name and the concept name concatenating an “impl” for the class name). We use *Concept* here to illustrate the template. We decided not to generate an overall concept class with each individual concept as an instance of this overall class. Because we want to provide an individual Web service for each concept, it is more convenient to make each concept an interface, which allows packing each concept separately. The basic Web service can use one concept interface and the implemented class without touching any other interface/class.

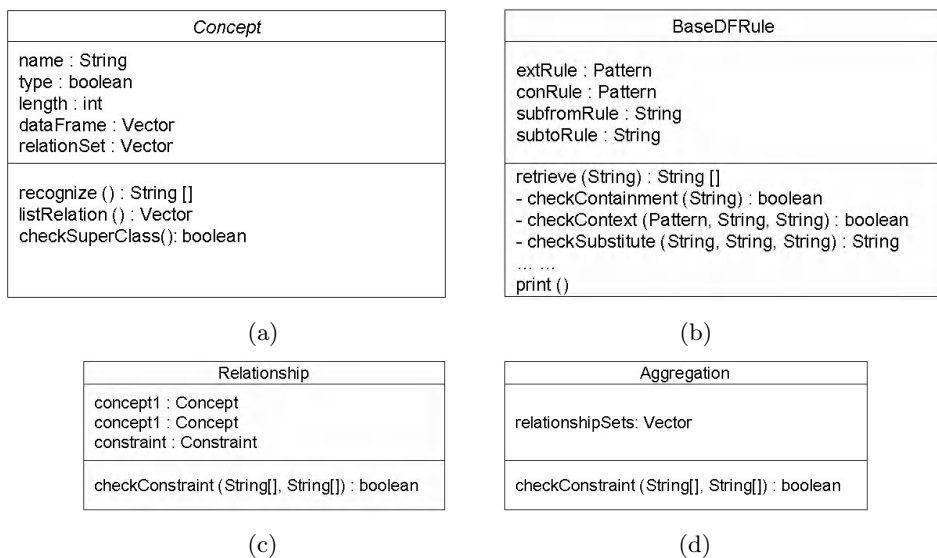


Fig. 3. The UML Diagrams for the Generated Java Classes

BaseDFRule class. We use four data fields to store the four parts of an extraction rule: extraction pattern, context pattern, substituteFrom pattern, and substituteTo pattern. This class also contains methods that extract information according to the extraction rule. The method *retrieve* takes a string as input, finds all the substrings that match the extraction rule, and then returns the extracted results in an array. There is a set of private methods that help this process. How to extract the information via an extraction rule is beyond the scope of this paper; please refer to [3] for detailed information. In each concept implementation class, the vector *dataFrame* stores all the extraction rule instances contained in the data frame of this concept. The *recognize* method processes one extraction rule a time by calling its *retrieve* method, and recognizes all the strings that match the concept.

Relationship Set. The ontology compiler generates a *Relationship* class to represent relationship sets in a source ontology. Figure 3(c) shows the UML diagram of the *Relationship* class. Each relationship set among two concepts is an instance of the *Relationship* class. The *Relationship* class has three data fields: the two concepts involved and the participation constraint among these two concepts. One method, *checkConstraint*, is implemented to check the participation constraints. This method takes the extraction results of the two concepts, and checks whether the constraints hold. For non-lexical object sets, we assume that the participation constraints hold automatically. Consider the relationship set *Car*[0:1] has *Year*[1:*] as an example. Since *Car* is a non-lexical object set, the code does not need to check the constraint [1:*]. The *Year* concept, on the other hand, is a lexical object set, so we need to check if the constraint [0:1] holds. If the system retrieves more than one *Year* value from a source record, then the method will return false and the system will return a failure condition.

A concept might be involved in many relationship sets. The *listRelation* method in its corresponding concept implementation class stores all the Relationship instances that this concept directly involved in the *relationSet* vector.

Aggregation. An aggregation has one or more is-subpart-of relationship sets. We use an Aggregation class to represent aggregation relationship sets. Each aggregation in an ontology is an instance of the *Aggregation* class. Figure 3(d) shows the UML diagram of the *Aggregation* class. The *relationshipSets* vector stores all the binary relationship sets (as in instances of the *Relationship* class). The *relationshipSets* Vector for the *Model|Trim* aggregation, for example, stores two *Relationship* instances: *Model [1:∗] is-subpart-of Model|Trim [0:1]* and *Trim [1:∗] is-subpart-of Model|Trim [0:1]*. The method *checkContainment* checks whether the aggregation condition holds by checking whether each extracted value of the sub-concept is a substring of the super-concept. If not, the system will throw an error to the user.

Generalization/Specialization. A generalization/specialization specifies an *is-a* relationship, identical to inheritance in Java. A specialization concept should inherit all the relationship sets its generalization concepts have. The interface of each specialization concept extends all its generalization concepts. Since Java supports multiple inheritance over interfaces, one child interface can extend more than one parent interface. In our car ontology, for example, the interfaces Engine, BodyType, Accessory, and Transmission extend the interface Feature. In an implementation class, the *listRelation* method also calls the parent class's *listRelation*, so that the parent's relation sets can be added to the child *relationSet* vector. The *checkSuperClass* is implemented to find all the super classes (generalization concepts) of a class (concept). The parent class list can be obtained through the *getInterface* method in the *Class* class in the Java standard class library.

4 Web Service Creation

Ontologies enable machine communication, and the Semantic Web is a typical environment that requires machine communication. Web services, on the other hand, are typical ways of performing machine communication. In the environment of the Semantic Web, Web services describe conventions for mapping knowledge easily and conveniently into and out of Web application programs. Hence we deem it desirable not only to generate Java APIs based on declarative domain knowledge in extraction ontologies, but also to directly generate machine-interpretable Web services using these generated Java APIs.

4.1 Atomic Web Services for Data Recognition

A Web service is a remote service that can not only satisfy a request from a client but can also be accessed via a standard specification interface. There is no need for a service requester to know the details about internal service


```

<definitions name='edu.byu.deg.examples.PriceService'
...
  xmlns:cps='http://www.deg.byu.edu/wsd1/edu.byu.deg.examples.PriceService/'
... >
  <message name='PriceRecognizer0SoapIn'>
    <part name='inputDoc' type='xsd:string' />
  </message>
  <message name='PriceRecognizer0SoapOut'>
    <part name='inputDoc' type='xsd:string' />
  </message>
  <portType name='edu.byu.deg.examples.PriceService'>
    <operation name='PriceRecognizer' parameterOrder='inputDoc'>
      <input name='PriceRecognizer0SoapIn' message='cps:PriceRecognizer0SoapIn' />
      <output name='PriceRecognizer0SoapOut' message='cps:PriceRecognizer0SoapOut' />
    </operation>
  </portType>
...
</definitions>

```

Fig. 4. Part of PriceService.wsd1

implementation. Instead, a service requester only needs to know the input and output of an operation, and the location provided by the service provider, both of which are described in the Web service description language (WSDL)³. The central problem of creating Web services is, therefore, to create a WSDL file that specifies a desired Web service.

In general, we can create an atomic data recognition Web service for each concept in an ontology. As described above, there is a *recognize* method in the generated Java interface/class of each concept. We therefore build a Web service based on this method for each concept. These generated services are *atomic* because each of them recognizes data instances with respect to one and only one ontology concept. In essence, none of them should be further decomposable. Moreover, users can compose these atomic Web services to be more complex Web services.

Figure 4 shows an example of an atomic Web service. This service is automatically generated from a Java program that is built through the ontology compilation process presented in the last section. The WSDL file is created for the Price recognition Java class. The service describes a remotely executable method named “PriceRecognizer” that takes an argument named “inputDoc.” Both the return datatype and the datatype of the input argument are “xsd:string”. With this information, users can directly invoke the service to retrieve price values within the “inputDoc”. When invoking the service, users do not need to know either the implementation details or even the specific programming language used for the implementation.

Several available tools can accomplish such transformations by directly generating WSDL files from Java classes. For instance, our example in Figure 4 is generated by the *java2wsdl* command-line tool contained in the GLUE platform⁴. GLUE also provides facilities that allow users to publish generated Web services that may be useful to other people.

³ Web Services Description Language (WSDL), <http://www.w3.org/TR/wsd1/>

⁴ The GLUE platform, <http://www.themindelectric.com>

4.2 Complex Web Services Through Composition

Complex services are composed with multiple simple atomic services. Web services represent loosely coupled interactions which are well suited to integrate disparate software domains and bridge incompatible technologies. It is therefore favorable to compose atomic Web services to accomplish complex operations, which is known as the process of Web service composition.

An essential requirement of automated service composition is that machine agents can interpret both the functionalities and the meanings of the operands in a basic Web service. Through a WSDL file, machines can execute a service correctly. The same WSDL descriptions, however, do not provide any explanation of the intent of a service or what the meanings of inputs and outputs are. Hence machines do not know what a Web service targets. This problem of lack of semantic explanation for Web service functionality has already been identified as a major problem for Web service research [1].

To solve this problem, we usually require Web services to be annotated before machines can automatically compose them. We use ontologies to denote the inputs, outputs, and parameters of a service. This solution is known as Web service annotation (such as [1], [5], [6], [8], and etc.). This type of service annotation operations, however, requires additional processing after services are generated. Not surprisingly, service annotation is not trivial.

Our method of automatically generating atomic Web services from extraction ontologies provides a resolution to this service interpretation problem. Because our generated services are based on ontologies from the onset, each of its generated features has its original formal definition in the starting extraction ontology. For this reason, the after-generation Web service annotation is no longer needed.

The simplest composition of atomic data recognition services involves retrieving a binary relation between two concepts. In an extraction ontology, we do not define relationship recognizer methods. Hence our relationship identification is based on a necessary but incomplete checking of a discovered relation. Our method determines a discovered binary relationship to be a defined relationship in an ontology when the following three conditions are satisfied simultaneously: (1) domain checking: the application domain matches a defined ontology; (2) concept-pair checking: the two concepts in the discovered binary relationship matches two concepts in the defined ontology; and (3) participation-constraint checking: the participation constraint of the discovered relation matches the participation constraint of the target relationship in the defined ontology. Although in theory this relationship-checking method is not complete enough to determine a relationship set, we find that it works very well in practice, especially when the scope of an application domain is narrow [3]. Because our research focuses on service composition involving the generated atomic ones, this narrow-domain assumption always holds.

We defer the domain-checking aspect to future discussions. In essence, we can view the domain checking problem as a standard ontology matching problem. Our previous experience in solving the ontology matching problem [4] allows us to simply assume prior knowledge about the ontology the user seeks.

With these discussions, we can reduce our generation problem for complex services to automatic composition of a service that captures two concepts and their participation constraints based on a known ontology. For example, assume we want to produce a complex service “car has price” as the ontology in Figure 2 shows. According to our relation-checking method, the dynamically created car-price complex service is composed of three basic services: the car recognition service, the price recognition service, and a check constraint service as Figure 3(c) shows.

More complex would be a recognition service for the relation “price for make.” In the original ontology as Figure 2 shows, there are no explicitly declared links between the two concepts *Price* and *Make*. So the service composition method needs to perform a relation discovery process to retrieve an implicit relation between the two concepts that can be derived by declarative specifications of ontology relationships. In our example, we know both “car has price” and “car has make.” Through a standard ontology inference process, we can derive a relation of “price for make” as well as its participation constraints. After the implicit relation is generated, our system can process it through the same complex service composition procedure as before.

As we discussed earlier, two typical relationships in extraction ontologies are aggregation and generalization/specialization. For an aggregation complex service request, in addition to the normal service composition procedure, we add a checkConstraint service as Figure 3(d) shows. For generalization/specialization, we add a checkSuperClass service as Figure 3(a) shows.

Although these examples are simple, we can perform the same composition process recursively. That is, each constructed service can be an unit, and we can perform a binary composition of any two constructed units. Therefore, this recursive service composition process can eventually produce very complex services based on users’ requests. Even better, since in each iteration the new composed service has itself mapped to machine-interpretable formal semantics, the new composed service will be inherently machine-interpretable no matter how complicated it is. There is no need for additional service annotation processes in these automatically generated Web services.

5 Conclusion and Future Work

In this paper we have sketched a two-step process for creating Semantic Web services. The first, which is fully implemented, involves compiling extraction ontologies to Java code that represents atomic executable Web services. The second stage involves composing these Web services together recursively with information derivable from the original ontologies. This additionally provides an automatic means for annotating these services in a standardized and formal manner.

Several directions remain to be pursued. First, we have yet to fully explore the second-stage composition process, particularly with respect to exploitation of the full range of ontological relationships possible and the resulting complexity

and consistency of the derived services. The relative benefits and challenges to this approach versus less automated approaches is also unclear at this point.

Second, we have yet to develop a rigorous testing methodology for assessing how well the system can assure appropriate coverage of services, accuracy of results given user queries, and quantifiable reduction in annotation efforts over more traditional efforts. This would involve comparing the results with those obtained via other proposed ontology-based service frameworks.

Third, we intend to explore development of a comprehensive Semantic Web services environment that allows users to specify a query of interest, match it with pre-existing extraction ontologies of appropriate domain and coverage and thereby select which pre-existing services are the most appropriate. In the absence of pre-existing services, the tool would allow the user to select the ontologies that are the most relevant, supervise (if desired) their mapping to atomic services, and direct (again, if desired) their composition into more complex services which can then be executed to satisfy the user's request.

References

1. M. L. Brodie. Illuminating the dark side of Web services. In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB'03)*, pages 1046–1049, Berlin, Germany, September 2003.
2. A. Eberhart. Automatic generation of Java/SQL based inference engines from RDF schema and RuleML. In *Proceedings of the First International Semantic Web Conference on The Semantic Web (ISWC'02)*, pages 102–116, London, UK, June 2002.
3. D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith. Conceptual-model-based data extraction from multiple-record Web pages. *Data & Knowledge Engineering*, 31(3):227–251, November 1999.
4. D.W. Embley, L. Xu, and Y. Ding. Automatic direct and indirect schema mapping: Experiences and lessons learned. *SIGMOD Record*, 33(4):14 – 19, December 2004.
5. D. Fensel and C. Bussler. The Web service modeling framework WSMF. *Electronic Commerce: Research and Applications*, 1:113–137, 2002.
6. Andreas Heß, E. Johnston, and N. Kushmerick. ASSAM: A tool for semi-automatically annotating Semantic Web services. In *Proceedings of the 3rd International Semantic Web Conference (ISWC'04)*, Hiroshima, Japan, November 2004.
7. A. Kalyanpur, D. Pastor, S. Battle, and J. Padget. Automatic mapping of OWL ontologies into Java. In *Proceedings of Software Engineering and Knowledge Engineering (SEKE'04)*, Banff, Canada, June 2004.
8. A. Patila, S. Oundhakar, and K. Verna. METEOR-S Web service annotation framework. In *Proceedings International WWW Conference*, pages 553–562, New York, NY, May 2004.

An Architecture for Emergent Semantics

Sven Herschel, Ralf Heese, and Jens Bleiholder

Humboldt-Universität zu Berlin

Unter den Linden 6, 10099 Berlin, Germany

{herschel, rheese, bleiho}@informatik.hu-berlin.de

Abstract. Emergent Semantics is a new paradigm for inferring semantic meaning from implicit feedback by a sufficiently large number of users of an object retrieval system. In this paper, we introduce a universal architecture for emergent semantics using a central repository within a multi-user environment, based on solid linguistic theories.

Based on this architecture, we have implemented an information retrieval system supporting keyword queries on standard information retrieval corpora. Contrary to existing query refinement strategies, feedback on the retrieval results is incorporated directly into the actual document representations improving future retrievals.

An evaluation yields higher precision values at the standard recall levels and thus demonstrates the effectiveness of the emergent semantics approach for typical information retrieval problems.

1 Introduction

The elementary challenge in all retrieval tasks is to find an object representation that can later effectively and efficiently be matched against a user query in order to find and rank the objects according to the user's needs.

Researchers in information retrieval (IR), to select a prominent example, have been very successful in finding document representations for later retrieval. Albeit being today's state of the art, the vector space model [1] used in conjunction with Latent Semantic Indexing [2], constitute only a syntactical approach in finding a so-called semantic representation.

Emergent semantics aims to emerge object representations by aggregating many user's opinions about the object content, therefore providing object representations that a majority of actual users of a system agree upon. We believe that finding such a representation considerably improves precision, since it was created by the users themselves. A basic example illustrates the idea behind emergent semantics: In a park near our campus, the landscape architects decided on not paving walkways initially. Instead, they covered the entire area with lawn. After a year they came back and knew exactly where to pave walkways, since the walkers had obviously decided which pathways they will use by actually walking them: the paths were all torn and muddy. We claim that this approach can be transferred into the area of computer science and, termed emergent semantics, represents a major advancement in the way object representations are created and maintained.

Our contribution with this paper is a formal approach to the emergent semantics paradigm, an architecture for utilizing its full potential and an implementation within the area of IR that demonstrates the possibilities associated with this paradigm. Our preliminary results show higher precision values at the standard recall levels, thus demonstrating the effectiveness of the emergent semantics approach for typical IR problems.

Structure of this paper. This paper is structured as follows: in Section 2, “Groundwork”, we introduce the linguistic background of syntax, semantics, and pragmatics. We then adopt these cognitions for computer science by introducing a universal architecture for emergent semantics in “Model House” (Section 3). In Section 4, “Construction”, we introduce our implementation of this architecture for a classic IR scenario in order to be able to present the promising results of our evaluation in Section 5. Related work is outlined in “Neighborhood” (Section 6) and “Roof and windows” (Section 7) concludes the paper with an outlook on open research issues and future work.

2 Groundwork

In this section, we introduce the linguistic background of syntax, semantics, and pragmatics in order to motivate our universal architecture for emergent semantics. This is essential for understanding the process leading from the user need, expressed by a query, to the retrieval and ranking performed by the system.

Semiotics is the study of signs, their meaning, and their interpretation by humans. The three subfields of semiotics are, in accordance with Morris [3]:

Syntax: the study of signs and their interrelation

Semantics: the study of signs and their relation to the objects they represent

Pragmatics: the study of signs and their relation to the user interpreting them

According to this theory, every sign is assigned a meaning within the context of a person’s understanding. So the letters *t*, *r*, *e*, and *e* form the word *tree* which, in English, refers to the biological wooden structure. Different people, however, may have something different in mind when being confronted with the word *tree*. Concepts may reach from a beautiful day in the park to the latest forest fire in Portugal. Similarly, the section headlines of this paper were deliberately chosen to potentially carry different meanings within this paper and outside its scope.

Algorithms for generating object representations from objects are usually syntactical with the implied hope of inferring something similar to semantics by applying clever extraction algorithms. An example for such an advanced algorithm is Latent Semantic Indexing (LSI) in the context of textual IR [2] which claims to extract document representations that are – by human judgment – considered good representations for the respective documents.

The semiotic triangle (see Figure 1) puts the three semiotic concepts (syntax, semantics, and pragmatics) into relation. The directed edges between the concepts stand for semiotic transitions that can potentially be explored by computer science algorithms.

Latent semantic indexing tries to walk the line from syntax to semantics: they analyze the document content and create a term-document matrix identifying the most important terms for each document and the most discriminating terms within a document collection. These algorithms therefore extract the meaning of a document by representing a document with terms found in the document collection. The obvious drawback of this purely syntactical approach is that only terms already found in the document collection itself can be used within document representations.

Emergent semantics in turn walks the line from pragmatics to semantics. It aggregates different users' opinions about the meaning of an artifact and creates an artifact representation that the majority of users agree upon.

Pragmatics includes the specific user background, i.e., culture, education, and social background, as well as very time-dependent influences like the user's mood, for example. It is our assumption that this diverse background will lead to a lot of noise, if applied unconditionally to the document representations. It is therefore necessary to somehow eliminate this noise. This elimination of unwanted noise can essentially be accomplished in two ways: (1) by applying user specific filters ("user profiles") and possibly aggregating these into opinion networks ("collaborative filtering") or (2) by assuming that the majority of users knows best and incorporating users' opinions directly into the document representation – through the noise semantics will emerge.

We believe that an architecture for emergent semantics needs to provide facilities for both approaches (see Section 3), however, we already achieved promising results with the later approach alone (see Section 4).

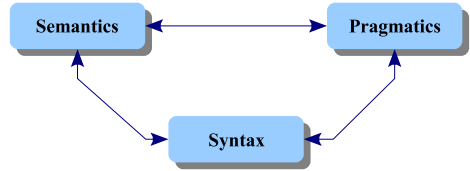


Fig. 1. Semiotic triangle

3 Model House

In this section, the basic building blocks of an emergent semantics (EmSem) architecture are introduced and their functionality is explained. The architecture is best explained describing the query processing workflow. Please refer to Figure 2 for an overview of both the architecture and the fundamental query processing steps. For now we assume a simple environment consisting of a central repository storing all data and a large number of clients querying the server.

3.1 Ingredients

In the context of emergent semantics, we understand object retrieval as a four step process. At first, a user develops an idea of her information need and formulates a *query* to the best of her knowledge (arrow 1). This query implicitly includes her individual context, e.g., her academic or social background.

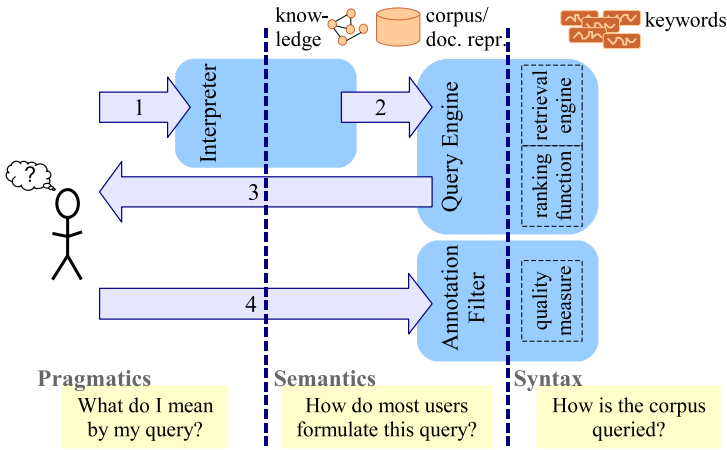


Fig. 2. An architecture for emergent semantics

The query is then analyzed by a component we term the *interpreter*. The interpreter is responsible for reformulating the query in such a way that the user's pragmatic background is resolved and explicitly stated as part of the query. This is accomplished in two ways: (a) utilizing query expansion or query reduction strategies in order to include a specified user context and thus reduce the possibilities of (mis)interpreting the query, (b) calibrating the query to be in accordance with the retrieval system, i.e., replacing terms with terms from a controlled vocabulary. An example for (a) is detailed in [4].

The result of this interpretation step is what we term a *canonical query*, i.e., a query that does not contain any pragmatic context any more (arrow 2). The interpreter thus bridges pragmatics and semantics on the query processing side.

The canonical query is then fed to the retrieval system. Three ingredients are used by the query engine to retrieve and rank suitable objects: *keywords* (a syntactical representation of stored objects), *corpus knowledge* (knowledge that is derived from the entirety of the object collection) and *external knowledge* (knowledge that is independent from the object collection). The query engine therefore bridges semantics and syntax within our architecture.

The result of the query engine is a list of ranked results (arrow 3) which is returned to the user. These results include an object surrogate which the user evaluates to determine which documents fulfill her information need.

By actually retrieving the relevant documents the user feeds her *original query*, including all its implicit pragmatic context, back to the system (arrow 4).

If this last step is performed a sufficiently large number of times by a sufficiently large number of users, new document semantics will be created. This is the reason for us calling it *emergent semantics*: the pragmatic view of many users of an information system is gradually converted into document semantics.

In complex scenarios, an annotation filter with a specified quality measure, could reject the addition of new keywords, i.e., if it is not in accordance with regulations or to suit a specific retrieval model.

3.2 Distinctions

Three aspects of the emergent semantics paradigm should be emphasized, since they differentiate the approach from current state of the art:

Entirely new keywords. EmSem allows entirely new terms to be introduced into the system. Even advanced approaches like LSI or query expansion can only work with terms already contained in the collection. In addition, we tackle the synonym problem, since a sufficient number of users will annotate the document with relevant synonyms.

Changing document representations. Keeping things simple, EmSem directly alters document representations. No new layers in query processing, no user specific state to be held, instant gratification to all users.

Living document representations. The entire process is based on the assumptions that most users “know best”. If users change their mind about the supposed meaning of an object, the meaning of the respective object will change over time (e.g., historic events that are reevaluated after some time, cars that become oldtimers, or changes in the use of language).

4 Construction

In this section, we instantiate the previously motivated architecture within the area of information retrieval. IR aims at satisfying a user information need usually expressed in natural language [5]. To accomplish this task and to effectively rank documents according to a user’s needs, an IR system models the relationship between a query and the relevant documents to this query. Approaches include the Boolean model, the vector space model [6], or the probabilistic model [7]. In the following, we introduce basic terminology and give a global view on the problem of IR within our architecture.

4.1 Terminology

We consider a collection of objects, e.g., (text) documents or images, forming the *corpus* $\mathcal{C}$ on which retrieval is performed. A common way to summarize content and meaning of these objects is to represent them using keywords (interchangeably called terms in this paper). Therefore, each document contained in the corpus is represented by a finite set of terms taken from a *universe of terms* $\mathcal{T}$. This annotation is called the document representation.

Definition 1 (Document Representation). *The document representation of a document is a set of terms: $r(d) = \{t_1, \dots, t_n\}, d \in \mathcal{C}, t_i \in \mathcal{T}$.*

In full-text IR, the prevalent form of IR today, these keywords correspond to the words of a (text) document, usually preprocessed and filtered, i.e., by a stopword filter or a stemmer, eliminating the most frequent words of a language and converting the remaining words into their canonical form.

The user expresses her information need by issuing a query to the system in natural language. As a query itself is interpreted as a document, we define it in the same way as the document representation:

Definition 2 (Query). *A user query is a set of terms: $q = \{t_1, \dots, t_n\}$, $t_i \in \mathcal{T}$.*

As a result of query evaluation, the system returns a ranked list of document surrogates to the user. Based on this list, the user selects the documents which fulfill her information need. We define the answer to a query as follows:

Definition 3 (Answer). *Let q be a query. The answer of q is defined as $D_q = \{d_1, \dots, d_n\}$, $d_i \in \mathcal{C}$. We denote with $D_r \subseteq D_q$ the set of documents classified as relevant by the user.*

Please note, that set D_r is specific for each user and each query of the system. Particularly, this implies that a specific D_r does not necessarily contain *all* relevant documents but only the ones *classified* as relevant by the user.

4.2 Running the System

Before effective IR can be performed, the entire collection must be indexed. This phase is called the bootstrapping phase and usually needs to be performed only once for each collection. As a result of bootstrapping, the components *keywords* and *corpus knowledge* of our architecture (see Figure 2) are initialized: while *keywords* represent the syntactical document representations, stemmed and reduced by stopwords, the *corpus knowledge* in our case is the term-document matrix of our collection filled with TF/IDF weights.

The *retrieval engine* is based on an inverted index of all terms $\mathcal{T}$ and the *ranking function* is based on the vector space model: both documents and queries are represented as term vectors carrying term weights from the term-document matrix above. The similarity between a query and each document is calculated as the angle between these vectors: the smaller the angle, the more relevant is the document for the respective query.

After the query has been processed, the user selects the relevant documents D_r by actually retrieving them. We assume that the result list contains enough information that a user can decide on the relevance of a document. This actual retrieval of the document leads to the document annotation of the relevant documents being completed with the query: $\forall d \in D_r : r(d) = r(d) \cup r(q)$. The intuition behind this is that if a document is found by the terms within the query and this document is additionally marked as relevant then all terms of the query are also related to the content of the document.

Since the TF/IDF-matrix depends on the document representation, some parts of the matrix have to be recalculated. The following list enumerates the possible changes to the document representation and recalculated elements of the TF/IDF-matrix. Let be $t \in r(q)$ and be $d \in D_r$:

1. Existing term ($t \in r(d)$): The weight of the term t increases for the document d , all other values for other documents remain unchanged.

2. New term/document combination ($t \notin r(d) \wedge t \in \mathcal{T}_C$): Weights for all documents are calculated, because the document frequency changes for t .
3. New term to corpus ($t \notin r(d) \wedge t \notin \mathcal{T}_C$): Since the document frequency of t is known in advance, it is sufficient to calculate the weight of the term t with regard to d .

In the context of the architecture presented in the previous section, adding all query terms to the document representation is only one way of a quality measure for the annotation filter, where all terms are considered to be of some and equal good quality. In the following, we outline some more sophisticated strategies to modify the document representation: A variation of the method described above is to ignore terms contained in the query having a high document frequency. These terms are not discriminative and thus, do not improve the document representation. As a side effect of omitting such terms, a smaller region of the TF/IDF-matrix has to be recalculated. Although they have yet not been validated we present two further strategies: (a) inclusion of additional external knowledge and (b) measuring the quality of a document representation with regard to its semantic precision. The first approach utilizes external knowledge to select the terms of a query being added to the document representation. For example, the decision of adding a query term may be based on a domain ontology, e.g., a term is only added if it is contained in the ontology. Considering the second approach, we will develop a quality function which measures the quality of the document representation and only add a new term if it increases the quality of the document representation.

5 Assessment

To evaluate our approach we chose the standard “Communications of the ACM” information retrieval collection (CACM). We chose this collection because it features an overlap between query terms as well as between corresponding result sets (see “Evaluation remarks” below).

5.1 Evaluation Setup

We used the document title and, if available, the document abstract, for indexing and retrieval. They were tokenized and indexed by the Apache Lucene inverted index [8] using the Vector Space Model with TF/IDF weighting. Then, the following steps were repeated for each query in the corpus:

Query. From the CACM corpus, a query is chosen and presented to the system as a disjunctive query (“or” semantics).

Retrieval and ranking. Documents are retrieved from the index and ranked using the vector space model with TF/IDF weighting of terms.

Feedback. According to the given gold standard, precision and recall measures are calculated. The query is then tokenized and attached to all relevant documents as content.

5.2 Evaluation Results

As performance indicators for our emergent semantics approach, we determined precision-recall-measures along the eleven standard recall levels. See [5] for a discussion of individual IR performance indicators.

The results in Figure 3 demonstrate a major improvement in retrieval performance after feeding back all query terms unconditionally. These results mean that retrieval performance increases with each query if multiple users pose the same query to the system. In addition, we experimented with a *quality measure* (see Figure 2), which only allows feeding back terms with a document frequency smaller than 300 (10% of the documents). This prevents feedback of frequent terms. However, we were surprised to see that the results were identical to the results before; we expected these frequent terms to introduce a lot of noise and therefore reduce precision. We believe that this lack of noise results from the very distinctive query terms within the CACM corpus: since the queries mostly contain infrequent terms, the amount of noise introduced did not do much harm.

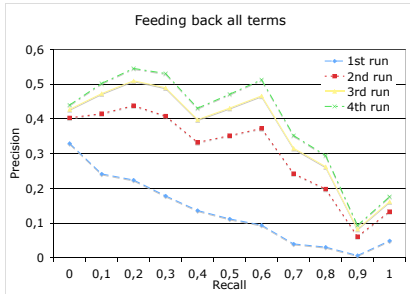


Fig. 3. Augmenting document representation with all query terms.

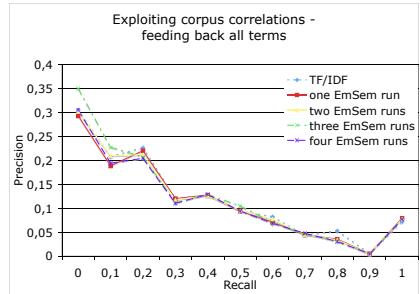


Fig. 4. Augmenting corpus part 1 with all query terms. Querying corpus part 2.

These results might have been anticipated: Few people would deny that feeding back the exact queries into the system will lead to improved retrieval performance for the same queries. Therefore, we split the corpus into halves and processed queries of the first half as described above. After different numbers of these runs (which we call “EmSem runs” in the figure), the second half was queried without feeding back these queries into the system. We were therefore able to measure the impact of query processing from the first half of the corpus onto the second (see Figure 4). While these results were not as stunning as the results in Figure 3, we were content to see that emergent semantics made an impact even in this scenario where neither the overlap between queries nor the overlap between result sets is big enough for a real-world scenario.

5.3 Evaluation Remarks

We chose the CACM collection after evaluation of the (also standard) Medline collection. It turned out, however, that there is no overlap in retrieval results

between the queries defined in the Medline collection and therefore the emergent semantics approach will not work. Ideally, we look for a collection of highly correlated queries with a highly overlapping result set. In such a scenario, emergent semantics unfolds its full potential.

It should be noted that we do compare our results to today's best precision/recall values. It is our ambition to introduce the new paradigm of actually modifying document representations instead of using (possibly user-specific) query refinement strategies. Therefore, the baseline of our comparison is standard document retrieval using the vector space model and TF/IDF weighting. We expect emergent semantics techniques to have a similar impact on more sophisticated IR algorithms.

6 Neighborhood

Emergent semantics claims to integrate many users' opinions on objects in order to find better document representations. While the user context plays an important role both in collaborative filtering approaches [9,10] as well as in contextual service adaptation [4], they both introduce a separate layer into the object retrieval process. Emergent semantics as a paradigm leaves the underlying retrieval system untouched and achieves its goal through direct modification of document representations. This is also contrary to the standard approach in using relevance feedback where the query is expanded [5].

Annotation or tagging systems [11] bear some similarity to the emergent semantics paradigm, however, they usually require the user to explicitly determine suitable tags or annotations for the object. During system usage, object descriptions are not altered. Emergent semantics takes advantage of the user query (implicit information) to accomplish its goal. Similarly, Grosky et al. emerge the semantics of multimedia objects (i.e., web pages) by including objects along a user's browsing path into the context of the respective object [12].

Emergent semantics through gossiping [13] aims at determining schema mappings between independent information provider nodes through measuring the information quality along feedback cycles.

Emergent semantics relies on an underlying IR infrastructure, whether this is based on the vector space model with TF/IDF weights [1] or with LSI weights [2]. It advances these approaches to the integration of users' opinions into the system, thus allowing for more representative terms or even additional terms that have not yet existed within the collection. In addition, recalculation of LSI weights in the event of changes in the document collection is quite expensive.

7 Roof and Windows

In this paper, we presented emergent semantics, a new paradigm for integrating many users' opinions directly into an object retrieval system. On the linguistic side, this paradigm represents the shift from many user's individual pragmatic

views to a unifying semantic representation of the object. We introduced an architecture capturing all aspects of emergent semantics and detailed the expected functionality of its components. An implementation of the emergent semantics approach within the area of IR, including promising results within a standardized evaluation on the CACM corpus, demonstrates the feasibility of our approach.

Future work includes exploration of other applications of emergent semantics, especially its potential ability to reduce the size of object representations, while still allowing for good retrieval results. We will also evaluate the applicability of the emergent semantics paradigm within a distributed environment.

Acknowledgment. This research was supported by the German Research Society (DFG grants no. NA 432 and GRK 316), and by the German Ministry of Research (InterVal Berlin Research Center for the Internet Economy).

References

1. Salton, G., McGill, M.: Introduction to Modern Information Retrieval. McGraw-Hill Book Company (1984)
2. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science* **41**(6) (1990) 391–407
3. Morris, C.W.: Foundations of the Theory of Signs. Chicago University Press, Chicago (1938)
4. Jacob, C., Radusch, I., Steglich, S.: Enhancing legacy services through context-enriched sensor data. In: Proceedings of the International Conference on Internet Computing. (2006)
5. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. ACM Press (1999)
6. Salton, G., Lesk, M.E.: Computer evaluation of indexing and text processing. *Journal of the ACM* **15**(1) (1968) 8–36
7. Robertson, S.E., Jones, K.S.: Relevance weighting of search terms. *Journal of the American Society for Information Science* **27**(3) (1976) 129–146
8. The Apache Software Foundation: <http://lucene.apache.org> (2006)
9. Shardanand, U., Maes, P.: Social information filtering: algorithms for automating “word of mouth”. In: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, ACM Press. (1995) 210–217
10. Oard, D.W.: The state of the art in text filtering. *User Modeling and User-Adapted Interaction* **7**(3) (1997) 141–178
11. Heflin, J., Hendler, J., Luke, S.: SHOE: A knowledge representation language for internet applications. Technical Report CS-TR-4078 (UMIACS TR-99-71), University of Maryland (1999)
12. Grosky, W.I., Sreenath, D.V., Fotouhi, F.: Emergent semantics and the multimedia semantic web. *SIGMOD Rec.* **31**(4) (2002) 54–58
13. Aberer, K., Cudré-Mauroux, P., Hauswirth, M.: The chatty web: emergent semantics through gossiping. In: Proceedings of the Twelfth International Conference on World Wide Web, New York, NY, USA, ACM Press (2003) 197–206

Semantic Web Techniques for Personalization of eGovernment Services

Fabio Grandi¹, Federica Mandreoli², Riccardo Martoglia², Enrico Ronchetti²,
Maria Rita Scalas¹, and Paolo Tiberio²

¹ Alma Mater Studiorum – Università di Bologna, Italy
Dipartimento di Elettronica, Informatica e Sistemistica,
{fgrandi, mrscalas}@deis.unibo.it

² Università di Modena e Reggio Emilia, Italy
Dipartimento di Ingegneria dell'Informazione, Modena,
{fmandreoli, rmartoglia, eronchetti, ptiberio}@unimo.it

Abstract. In this paper, we present the results of an ongoing research involving the design and implementation of systems supporting personalized access to multi-version resources in an eGovernment scenario. Personalization is supported by means of Semantic Web techniques and is based on an ontology-based profiling of users (citizens). Resources we consider are collections of norm documents in XML format but can also be generic Web pages and portals or eGovernment services. We introduce a reference infrastructure, describe the organization and present performance figures of a prototype system we have developed.

1 Introduction

The Semantic Web (SW), once it has been proposed as the next generation of the existing Web [4], has solicited a crop of scientific interest both from academic and industrial research and has gained strong momentum for the last five years. However, after years of intensive research and impressive scientific results, SW is still in search for killer applications and real-world use cases which could demonstrate, beyond any reasonable doubt, its added value as enabling technology.

On the other hand, the last years have also witnessed a very strong, world-wide institutional effort towards the implementation of **eGovernment** (eGov) support services, which constitute an enormous challenge for the deployment of semantics and the exploitation of domain knowledge in the design, construction and operation of Web information systems. As a matter of fact, whereas SW technologies can be an ideal platform to envisage a knowledge-based, user-centric, distributed and networked eGov, the eGov domain can provide an ideal testbed for existing SW research and for the development of software applications with “ontologies under the hood”. In this context, the first call is for *interoperability*: manifold semantic differences have to be settled in order to provide seamless services to citizens, as the eGov domain provides for differences in the interpretation of laws, regulations, life events, administrative processes, service workflows, best-practices, to be taken into account within and across regions, nations and

continents (not to talk about the usage of many different languages). The second call is for *personalization*: the achievement of a high level of integration and involvement of the citizens in the eGov and eGovernance activities, the necessity to fairly deal with different categories of citizens (including disadvantaged ones, with a potential risk of increasing digital divide), the requirements to support flexible, user-friendly, precise, targeted and non-baffling services, all claim for the personalization of the services offered and information supplied.

Whereas most of the recent and ongoing research on the convergence between SW and eGov is on the interoperability side (see e.g. [2]), we move on the personalization side, which we consider a legitimate corollary of the solution of the interoperability problems. If interoperability, including semantic integration of systems, processes and of the exchanged information, is the basis for the realization of complex networked eGov services, semantics-aware personalization of the Public Administration (PA) activities that concern the citizens and of the online offered services is aimed at improving and optimizing the involvement of citizens in the eGovernance process. In particular, we consider ontology-based user-profiling and personalized access to online resources (internally available in multi-version format), which may range from guided browsing of PA informative Web sites and portals to selective querying collections of norm documents, and to enactment of customized Web services [15] implementing administrative processes. Notice that, although all these kinds of resources are already available in existing eGov Web information systems, personalization is either completely absent or at most “predefined” in the Web site structure/contents or service definition/workflow (for example, hardwired in eGov portals by human experts according to the *life events* metaphor [9]). Effective, automatic, flexible, on-demand, “intelligent” and, last but not least, efficient personalization facilities are lacking.

In this paper, we present the results of an ongoing research started in 2003 (see [11]). In the first part of this research we focused on the design and implementation of Web information systems for personalized access to norm repositories. Building upon previous work on temporal management of multi-version norm documents [6], we developed a platform for semantics-aware personalized access to the repository. Personalization is based on the maintenance of an ontology which classifies the citizens according to the limited applicability of norm provisions. Semantic information is then used to map the citizen’s identity onto the applicable norms in the repository thanks to an intelligent and efficient retrieval system. The ongoing research concerns the application of our ontology-based personalization techniques to the choice and execution of eGov Web services. The research activity will be described in Section 2 whereas conclusions can be found in Section 3.

2 Personalized Access to eGovernment Resources

In the framework of eGov, a large number of online resources including PA portals, informative Websites, usable administrative services are progressively

being made available to citizens. In particular, collections of norm texts and legal information presented to citizens (e.g. stored in large repositories in XML format [10]) are being made available and becoming popular on the internet owing to big investments and efforts made by governments and administrations. Such portals or websites are usually equipped with a keyword-based search engine or contain indexes and predefined navigation paths for user guidance (e.g. following the life events approach).

The main objective of such activity has been the development of techniques allowing an effective and efficient personalized access to multi-version norm repositories. First of all, the fast dynamics involved in normative systems implies the coexistence of multiple *temporal versions* of the norm texts stored in a repository, since laws are continually subject to amendments and modifications (e.g. it is crucial to reconstruct the consolidated version of a norm as produced by the application of all the modifications it underwent so far). Moreover, another kind of versioning plays an important role, because some norms or some of their parts have or acquire a limited applicability. For example, a given norm defining tax treatment may contain some articles which are only applicable to particular classes of citizens: one article is applicable to unemployed persons, one article to self-employed persons, one article to public servants only and so on. Hence, a citizen accessing the repository may be interested in finding a *personalized version* of the norm, that is a version only containing articles which are applicable to his/her personal case. Notice that personalization avoids in some cases the user to have to go through a huge amount of irrelevant text to find out the relevant one and, thus, may help to make the search feasible. For instance, the annual budget law of a state, maybe composed of several hundreds of articles, may contain one article whose provisions have some consequences on the way research funds must be managed by universities (maybe without ever explicitly mentioning “university” in the text). One university professor may be interested in accessing the repository to retrieve the personalized version of the budget law, which will only contain the pertinent article, without having to go through the whole norm text, which would result in a very time-consuming and daunting activity.

Introducing the civic ontology. In general, in order to enhance the participation of the citizens to an eGovernance procedure of interest through the provision of personalization facilities, automatic and accurate positioning of them within the reference legal framework is needed. To this purpose, we propose to employ SW techniques and introduce an ontology, called *civic ontology*, which corresponds to a classification of citizens based on the distinctions introduced by subsequent norms which imply some limitation (total or partial) in their applicability. In the following, we refer to such norms as *founding acts*. Hence, in order to define a mapping between ontology classes and relevant norm parts, applicability annotations are added to the XML encoding of norms. More precisely, a semantic versioning dimension is introduced in the multi-version data model used for the representation and storage of the XML resources.

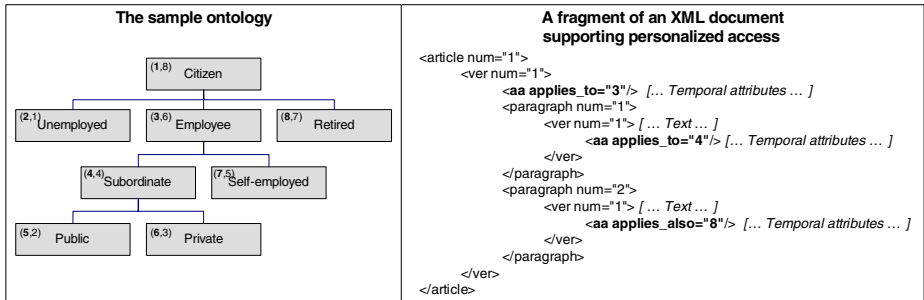


Fig. 1. An example of civic ontology, where each class has a name and is associated to a (pre,post) pair, and a fragment of a XML norm containing applicability annotations

For instance, the left part of Fig. 1 depicts a simple civic ontology built from a small corpus of norms ruling the status of citizens with respect to their work position. The right part shows a fragment of a multi-version XML norm text supporting personalized access with respect to this ontology, where the “aa” tag (applicability annotation) contains references to reference classes in the ontology.

At the current stage of the research, semantic information is mapped onto a *tree-like* civic ontology, that is based on a taxonomy of citizens induced by IS-A relationships. The tree-like civic ontology is sufficient to satisfy basic application requirements as to applicability constraints and personalization services, although more advanced application requirements may need a more sophisticated ontology definition. The adoption of tree-like ontologies allows us to exploit the pre-order and post-order properties of trees in order to enumerate the nodes and quickly check ancestor-descendant relationships between the classes. These codes are represented in the upper left part of the ontology classes in the Figure, in the form: (pre-order,post-order). For example, the class “Employee” has pre-order “3”, which is also its identifier, whereas its post order is “6”. The pre- and post-order information is then used to process queries in a very efficient way.

The article in the XML fragment on the right part of Fig. 1 is composed of two paragraphs and contains applicability annotations (tag *aa*). Notice that applicability is inherited by descendant nodes unless locally redefined. Hence, by means of redefinitions we can also introduce, for each part of a document, complex applicability properties including extensions or restrictions with respect to ancestors. For instance, the whole article in the Figure is applicable to civic class “3” (tag *applies_to*) and by default to all its descendants. However, its first paragraph is applicable to class “4”, which is a restriction, whereas the second one is applicable to class “8” (tag *applies_also*), which is an extension. The representation of extensions and restrictions gives rise to high expressiveness and flexibility in an eGovernment scenario, where personalization requirements have to be met.

The reference infrastructure. In order to use the semantic versioning mechanism for personalization, we define the citizen’s *digital identity* as the total

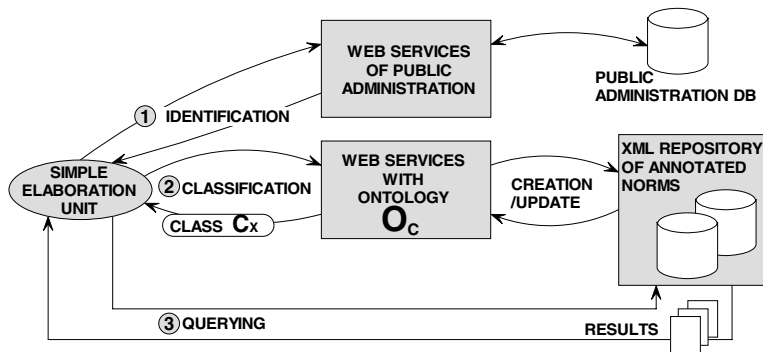


Fig. 2. The Complete Personalization Infrastructure

amount of information concerning him/her –necessary for the sake of classification with respect to the ontology– which is available online [14]. Such information must be retrievable in an automatic, secure and reliable way from the PA databases through suitable Web services (*identification services*). For instance, in order to see whether a citizen is married, a simple query concerning his/her marital status can be issued to registry databases. In this way, the classification of the citizen accessing the repository makes it possible to produce the most appropriate version of all and only norms which are applicable to his/her case.

Hence, the resulting complete infrastructure which is needed to perform all the required tasks is composed of various components that exchange information and cooperate to produce the final results as shown in Fig. 2. Firstly, in order to obtain a personalized access, a secure authentication is required for a citizen accessing the infrastructure. This is performed through a simple elaboration unit, also acting as user interface, which processes the citizen's requests and manages the results. Then, we can identify the following phases:

- the **identification phase** (step 1 in Fig. 2) consists of calls to identification services to reconstruct the digital identity of the authenticated user on-the-fly. In this phase the system collects pieces of information from all the involved PA web services and composes the identity of the citizen.
- the citizen **classification phase** (step 2 in Fig. 2) in which the classification service uses the collected digital identity to classify the citizen with respect to the civic ontology (O_C in Fig. 2), by means of an embedded reasoning service. In Fig. 2, the most specific class C_X has been assigned to the citizen;
- finally, in the **querying phase** (step 3 in Fig. 2) the citizen's query is executed on the multi-version XML repository, by accessing and reconstructing the appropriate version of all and only norms which are applicable to the class C_X .

In order to supply the desired services, the digital identity is modelled and represented within the system in a form such that it can be translated into the same language used for the ontology. In this way, during the classification

```

FOR    $a IN norm
WHERE  textConstr ($a//paragraph//text(), 'health AND care')
AND    tempConstr ('vTime OVERLAPS PERIOD('2002-01-01','2004-12-31')')
AND    applConstr ('class_7')
RETURN $a

```

Fig. 3. An XQuery-equivalent executable query

procedure, the matching between the civic ontology classes and the citizen's digital identity can be reduced to a standard reasoning task (see [3,8]).

Furthermore, the civic ontology used in steps 2 and 3 requires to be created and constantly maintained: each time a new founding act is enforced, the execution of a **creation/update phase** is needed. Notice that this process is a delicate task which needs advice by human experts and “official validation” of the outcomes and, thus, it can only partially be automated.

The resources of interests (e.g. norm documents) are stored in the XML repositories in a compact format according to a multi-version data model supporting temporal and semantic versioning (details can be found in [7]). Notice that temporal and limited applicability aspects may also interplay in the production and management of versions. However, since temporal and semantic versioning are treated in an orthogonal way in our model, also complex situations can be easily captured.

The queries that can be supported can contain four types of completely orthogonal constraints (temporal, structural, textual and applicability) allowing us to specify very specific searches in the XML norm repository. Let us focus first on the applicability constraint. Consider again the ontology and norm fragment in Fig. 1 and let John Smith be a “self-employed” citizen (i.e. belonging to class “7”) retrieving the norm: hence, the sample article will be selected as pertinent, but only the second paragraph will be actually presented as applicable. Furthermore, the applicability constraint can be combined with the other three ones in order to fully support a multi-dimensional selection. For instance, John Smith could be interested in all the norms ...

- which contain paragraphs (*structural constraint*) ...
- dealing with health care (*textual constraint*), ...
- which were in force between 2002 and 2004 (*temporal constraint*), ...
- which are applicable to his personal case (*applicability constraint*).

Such a query can be issued to our system using the standard XQuery FLWR syntax [16] as in Fig. 3, where `textConstr`, `tempConstr`, and `applConstr` are suitable functions allowing the specification of the textual, temporal and applicability constraints, respectively (the structural constraint is implicit in the XPath expressions used in the XQuery statement). Notice that the temporal constraints can involve several time dimensions (see [6]), allowing high flexibility in satisfying the information needs of users in the eGovernment scenario. In particular, it is possible to extract consolidated current versions from the

multi-version repository, or to access past versions of particular norm texts, all consistently reconstructed by the system on the basis of the user's specifications and personalized on the basis of his/her identity.

Implementation and performance evaluation. In order to test the efficacy of the proposed approach, we built a prototype system supporting our data model. The system is based on a Multi-version XML Query Processor designed on purpose, which is able to manage the XML data repository and to support all the temporal, structural, textual and applicability query features in a single component. In addition to the introduction of the semantic versioning dimension, the system represents a complete redesign and extension of a previous system supporting temporal versioning described in [6], which we had built on top of a commercial DBMS with XML storage and query support. Details of the migration and a comparison between the two systems can be found in [7,12].

The prototype is implemented in Java JDK 1.5 and exploits ad-hoc data structures (relying on embedded "light" DBMS libraries) and algorithms which allow users to store and reconstruct on-the-fly the XML norm versions satisfying the four types of constraints. Such a component stores the XML norms in a partitioned way, which is used, during query answering, in order to efficiently perform structural-join algorithms [1] specifically adapted and tuned for the temporal/semantic multi-version context. Textual constraints are handled by means of an inverted index. Owing to the properties of the adopted pre- and post-order encoding of the civic ontology classes, the system is able to very efficiently deal with applicability constraints during query processing by means of simple comparisons involving such encodings and semantic annotations.

The experiments have been effected on a P4 2.5Ghz Windows XP workstation, equipped with 512MB RAM and a RAID0 cluster of two 80GB EIDE disks with NT file system (NTFS). We performed the tests on three XML document sets of increasing size: collection C1 (5,000 XML normative text documents), C2 (10,000 documents) and C3 (20,000 documents). We will describe in detail only the results obtained on the collection C1, then we will briefly describe the scalability performance shown on the other two collections. The total size of the collections is 120MB, 240MB, and 480MB, respectively.

Experiments were conducted by submitting queries of five different types (Q1-Q5). Table 1 presents the features of the test queries and the query execution time for each of them. All the queries require structural support (St constraint); types Q1 and Q2 also involve textual search by keywords (Tx constraint), with different selectivities; type Q3 contains temporal conditions (Tm constraint) on three time dimensions: transaction, valid and publication time; types Q4 and Q5 mix the previous ones since they involve both keywords and temporal conditions. For each query type, we also present a personalized access variant involving an additional applicability constraint (Ap constraint), denoted as Qx-A in the first column of Table 1. "XML-Native" denotes the system described in this paper, whereas "DOM-based" represents our previous prototype only supporting temporal versioning.

Table 1. Features of the test queries and query execution time (time in milliseconds, collection C1)

Query	Constraints					Selectivity	Performance (msec)	
	Tm	St	Tx	Ap			DOM-based	XML-Native
<i>Q1</i>	-	✓	✓	-		0.6%	2891	1046
<i>Q2</i>	-	✓	✓	-		4.02%	43240	2970
<i>Q3</i>	✓	✓	-	-		2.9%	47638	6523
<i>Q4</i>	✓	✓	✓	-		0.68%	2151	1015
<i>Q5</i>	✓	✓	✓	-		1.46%	3130	2550
<i>Q1-A</i>	-	✓	✓	✓		0.23%	n/a	1095
<i>Q2-A</i>	-	✓	✓	✓		1.65%	n/a	3004
<i>Q3-A</i>	✓	✓	-	✓		1.3%	n/a	6760
<i>Q4-A</i>	✓	✓	✓	✓		0.31%	n/a	1020
<i>Q5-A</i>	✓	✓	✓	✓		0.77%	n/a	2602

Let us first focus on queries without personalized access. Our approach shows a good efficiency in every context, providing a short response time (including query analysis, retrieval of the qualifying norm parts and reconstruction of the result) of approximately one or two seconds for most of the queries. Notice that the selectivity of the query predicates does not impair performances (as it happened to the “DOM-based” approach), even when large amounts of documents containing some (typically small) relevant portions have to be retrieved, as it happens for queries Q2 and Q3. Our new system is able to deliver a fast and reliable performance in all cases, since it practically avoids the retrieval of useless document parts. Furthermore, for the same reasons, the main memory requirements of the new system are very small, less than 5% w.r.t. “DOM-based” approach.

The time needed to answer the personalized access versions of the Q1–Q5 queries is approximately 0.5-1% more than for the original versions. Moreover, since the applicability annotations of each part of an XML document are stored as simple integers, the size of the tuples with applicability annotations is practically unchanged (only a 3-4% storage space overhead is required with respect to documents without semantic versioning), even with quite complex annotations involving several applicability extensions and restrictions.

Finally, we ran the same queries of the previous tests on the larger collections and saw that the computing time always grows sub-linearly with the number of documents. For instance, query Q1 executed on the 10,000 documents of collection C2 (which is as double as C1) took 1,366 msec (i.e. the system was only 30% slower); similarly, on the 20,000 documents of collection C3, the average response time was 1,741 msec (i.e. the system was less than 30% slower than with C2). Also with the other queries the measured trend was the same, thus showing the good scalability of the system in every type of query context.

Current extensions. In our current research work, we are extending our ontology-based personalization approach to the definition and management of multi-version eGov Web services. For instance, ontology-based personalization

has been used in the field of eLearning services [5,13] and we are experimenting the adoption of similar techniques for the eGov application domain. In particular, we are applying our semantic versioning techniques also to solve this problem.

In particular, the application scenario and the reference infrastructure remain the same as in Fig. 2, along with the ontology management module and the identification and classification services. However, the XML repository and the query engine are being extended to also deal with the data required for the definition of multi-version Web services (including the specific data possibly required by a Workflow Management System to enact and support the execution of a workflow instance underlying the personalized eGov service delivered). Once the citizen has been classified with respect to the ontology during the identification phase, the semantic information is then used during the query phase to extract the data needed to build the personalized version of the requested Web service.

For instance, the eGov service for the “change of address” procedure may be different for public servants with respect to other categories of workers (e.g. public servants may have to declare that their new residence is within a fixed distance from their workplace, if required by law; hence a specific subservice for this task has to be included in the selected Web service). Hence, if the user has been classified as public servant, the query engine must retrieve a personalized version of the Web service for the “change of address” process for the citizen’s case.

However, techniques very similar to the ones adopted for semantic annotation of XML documents and for efficient querying of the multi-version repository can be used also in such a case. Preliminary experiments are under way.

3 Conclusions

In this paper, we presented the results of a still ongoing research activity we are carrying out in the context of a national research project in order to support efficient and personalized access to multi-version resources in an eGovernment scenario. We defined a data model supporting ontology-based personalized access to XML documents, built a prototype system implementing the data model and evaluated its performance through some exploratory experiments. The results we obtained are very encouraging as to query response time, storage requirements and system scalability figures. Current efforts are focused on the extension of our approach to the support of ontology-based personalization of eGov services, as outlined at the end of the previous section.

In the future, we will strengthen the proposed approach, in particular by considering more advanced application requirements leading to a more sophisticated (e.g. graph-based) ontology definition, and by completing the required technological infrastructure with the specification and implementation of the remaining auxiliary services, including advanced reasoning services for management of the ontology. Further work will also include the assessment of our developed systems in a concrete working environment, with real users and in the presence of a large repository of real legal documents. In particular, a civic ontology based on a corpus of real norms (concerning infancy schools) is currently under development.

References

1. S. Al-Khalifa, H.V. Jagadish, J. M. Patel, Y. Wu, N. Koudas, and D. Srivastava. Structural joins: A primitive for efficient XML query pattern matching. In *Proc. of 18th ICDE Conf.*, San Jose, CA, 2002.
2. A. Abecker, A. Sheth, G. Mentzas, and L. Stojanovic (Eds.). *Semantic Web Meets eGovernment – Papers from the AAAI Spring Symposium*. AAAI Press, Menlo Park, CA, 2006.
3. F. Baader, I. Horrocks, and U. Sattler. Description Logics for the Semantic web. *Künstliche Intelligenz*, 16(4):57–59, 2002.
4. T. Berners-Lee, J. Hendler, O. Lassila, ‘ The Semantic Web, *Scientific American* 284(5):34-43, 2001.
5. R. Denaux, D. Dimitrova, and L. Aroyo. Interactive Ontology-based user modeling for personalized learning content management. In *Proc. of AH’04 Semantic Web for E-Learning Workshop.*, Eindhoven, The Netherlands, 2004.
6. F. Grandi, F. Mandreoli, and P. Tiberio. Temporal modelling and management of normative documents in XML format. *Data & Knowledge Engineering*, 54(3):327–354, 2005.
7. F. Grandi, F. Mandreoli, R. Martoglia, E. Ronchetti, M.R. Scalas, and P. Tiberio. Personalized access to multi-version norm texts in an e-government scenario. In *Proc. of 4th EGOV Conf.*, LNCS No. 3591, Copenhagen, Denmark, 2005.
8. I. Horrocks, and P.F. Patel-Schneider. Reducing OWL entailment to Description Logic satisfiability. In *Proc. of ISWC 2003*, Sanibel Island, FL, 2003.
9. The Italian e-Government portal. <http://www.italia.gov.it/>.
10. The “norme in rete” (norms on the net) home page. <http://www.normeinrete.it>.
11. The “Semantic web techniques for the management of digital identity and the access to norms” PRIN Project Home Page. <http://www.cirsfd.unibo.it/eGov03/>.
12. F. Mandreoli, R. Martoglia, F. Grandi, and M.R. Scalas. Efficient management of multi-version XML documents for e-Government applications. In *Proc. of 1st WEBIST Conf.*, Miami, FL, 2005.
13. L. Razmerita, and G. Gouarderes. Ontology-based user modeling for personalization of grid learning services. In *Proc. ITS’04 GRID Learning Services Workshop*, Maceio, Brazil, 2004.
14. S. Rodotà. Introduction to the “One world, one privacy” session. In *Proc. of 23rd Data Protection Commissioners Conf.*, Paris, France, http://www.paris-conference-2001.org/eng/contribution/rodota_contrib.pdf, 2001.
15. Web services activity. W3C Consortium, <http://www.w3.org/2000/xp/Group/>.
16. XML Query language. W3C Consortium, <http://www.w3.org/XML/Query>.

Query Graph Model for SPARQL

Ralf Heese

Humboldt-Universität zu Berlin
Databases and Information Systems
`rheese@informatik.hu-berlin.de`

Abstract. Several query languages for RDF have been proposed before the World Wide Web Consortium started to standardize SPARQL. Due to the declarative nature of the proposed query languages, a query engine is responsible to choose an efficient evaluation strategy. Although all of the RDF repositories provide query capabilities, some of them disregard declarativeness during query evaluation.

In this paper, we propose a query graph model (QGM) for SPARQL supporting all phases of query processing. On top of the QGM we defined transformations rules to simplify the query specification as a preliminary step of query execution plan generation. Furthermore, the query graph model can easily be extended to represent new concepts.

1 Introduction

Since the introduction of RDF researchers have been investigating approaches to manage RDF data. Thus, several RDF repositories have been developed often exploiting existing database technologies, e.g., relational databases [1,2,3] or Berkley-DB [4]. The development of RDF repositories came along with several proposals for a query languages. Combining concepts of these languages the World Wide Web Consortium currently standardizes a query language for RDF, namely SPARQL Protocol And RDF Query Language (SPARQL), which is being adopted by recent implementations of RDF repositories. Due to its declarative nature, a query engine has to choose an efficient way to evaluate a query. The query engines of current RDF repositories usually translate a SPARQL query into queries of the underlying DBMS. However, parts of the query processing are covered by the repository implementation. As the following example¹ demonstrates, the achieved results are not satisfactory.

Question: ... Below is a very simple Jena/ARQ program. It is a toy standalone program that builds a database of 10,000 trees describing families (dad, mom, kids...), and then does various queries.

The queries are very simple (e.g. families where dad=="Peter"), but the program runs **very* slowly*. ...

¹ Shortened discussion. See <http://groups.yahoo.com/group/jena-dev/message/21436> for the full text. The question was posted on Mar 8, 2006.

Answer: ... Put the more specific part of the query first; it makes a significant difference. ...

Reply: ... My time went from 33000ms $\rightarrow$ 150ms. ...

The query engine of the used RDF repository, Jena, evaluates the triple pattern of the query in sequence of their occurrence. As a consequence, the user has to choose the right order of triple pattern to minimize the query execution time – contradictory to the nature of a declarative query language.

In this paper, we make a first step into optimization of SPARQL queries. During compilation of a query, the query processor has to gather and store information which are relevant for query evaluation. Therefore, we adopted the well-known query graph model (QGM) developed for the Starburst database management system [5] to represent SPARQL queries. Transformation rules provide the means for reformulating and simplifying the query specification.

The paper is structured as follows. In Section 2 we put the proposed query graph model into the perspective of query processing in database management systems. Afterwards, Section 3 introduces the adapted query graph model and describes how a SPARQL query is represented by it. Furthermore, we present a basic set of transformation rules laying the foundations for the generation of query execution plans. Finally, we discuss related work in Section 4 and provide a conclusion of this paper in Section 5.

2 Query Processing

Query processing in relational databases has been researched for decades. As a result, many models and algorithms have been developed and evaluated, which aim at decreasing the execution time of relational queries. Due to the achievements of relational database research, most of the current approaches to manage RDF data rely on relational technology, e.g., RDFSuite [1], Sesame [2], and Jena2 [3]. We suggest to adapt the query graph model (QGM) described by Pirahesh et al. in [5] as foundation of the query processing in RDF repositories. In this section we put the query graph model into the perspective of query processing in databases before we explain it in more detail.

In the context of databases, query processing is usually divided into four phases as depicted in Figure 1: query parsing, query rewriting, query execution plan (QEP) generation, and QEP execution. In Figure 1 the small arrows depict the information flow; large arrows depict the control flow.

In the first phase the query processor parses the query and constructs an internal representation that describes the query. We adapted the query graph model (QGM) described in [5] for the

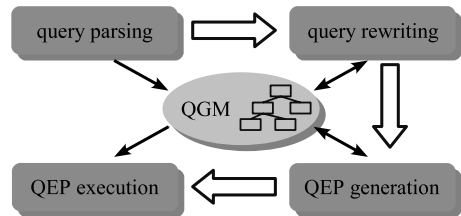


Fig. 1. Phases of the query processing

the

internal representation of a SPARQL query. During compilation the query graph model represents a key data structure for the query processing comprising all relevant information about the query.

In the following query rewriting phase, the generated QGM is transformed into a semantically equivalent one to achieve a better execution strategy when processed by the plan optimizer. The transformation rules mainly aim at reducing the complexity of the query specification by merging graph patterns, e.g., avoiding join operations, and eliminating redundant or contradicting restrictions. Basic transformation rules are presented in Section 3.3.

In the third phase the plan optimizer generates a set of query execution plans (QEPs) choosing between different data access strategies and operator implementations. Furthermore, it selects the indexes to be invoked (see Section 3.4). Using a cost function, the plan optimizer calculates the costs for each QEP and picks the most beneficial plan for execution. The cost function may consider criteria such as involved indexes, characteristics of the physical storage, and data distribution. The information relevant for the current query are also maintained in the QGM. Finally, the chosen QEP is forwarded to the query execution engine which interprets the QEP and generates the result of the query.

3 Query Graph Model

In [5] the authors developed the *query graph model* (QGM) which defines a conceptually more manageable representation of an SQL query. The QGM representation of a query consists of boxes containing subqueries and edges connecting these boxes. Each of the boxes is optimized separately. If possible the transformation rules of the QGM merge and eliminate boxes, so that the optimizer can consider larger parts of the query during optimization. The model is furthermore extensible, e.g., to represent extensions to the query language and to keep additional information about the query and its data sources. In this section we describe the adaption of the query graph model to represent SPARQL queries. We begin with the description of the basic elements of the model and then explain the translation of SPARQL into the model.

3.1 Basics

The fundamental elements of the query graph model are boxes and edges (see Figure 2). A box represents an operation to be executed during query evaluation, e.g., matching graph patterns or verifying value constraints. An edge symbolizes the data flow between two operations (boxes) indicating that an operation consumes the output of another. Edges are directed and point to the data consumer.

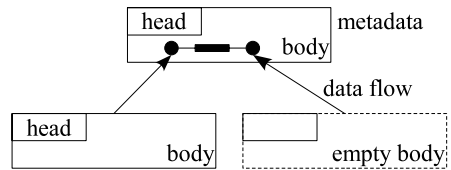


Fig. 2. Basic elements of query graph model

A box consists of three parts: head, body and metadata. The head lists the variables that are bound during the evaluation of the operator and are needed for subsequent operations. The body of a box describes the operation performed to bind the variables or to generate the output. Additionally, a box may carry meta-data providing additional information about the operation, e.g., **TABLE** indicating a select query. Derived from the current SPARQL specification we distinguish between the following categories of operations:

Data access. Operations of this category provide access to the data graphs specified in the **FROM** and **FROM NAMED** clauses.

Restrictions. This category comprises operations that apply graph pattern and value constraints to their input.

Fusion. Operations combining several input sources to a single result set, for instance join and union, belong to this category.

Solution constructor/modifiers. This category contains operations which generate the result of a query or modify the answer set, e.g., **ORDER BY**.

3.2 Representing SPARQL Queries

In the following, we focus on the translation of a SPARQL query into its corresponding query graph model. We illustrate the translation process with Example 1 that returns a list of names with their corresponding mail address and, optionally, their homepage, if (a) the mail address is an IRI, (b) there exists an IRI pointing to an image, and (c) both graphs contain the same mail address.

Example 1.

```

1  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
2  SELECT ?name ?mbox ?hpage
3  FROM      <http://example.org/foaf/aliceFoaf>
4  FROM NAMED <http://example.org/foaf/bobFoaf>
5  WHERE {
6    { ?x foaf:name ?name ;
7      foaf:mbox ?mbox .
8      FILTER ( isIRI(?mbox) ) .
9      OPTIONAL { ?x foaf:homepage ?hpage }
10   }
11   { ?x foaf:image ?img .
12     FILTER ( isIRI(?img) ) .
13   }
14  GRAPH <http://example.org/foaf/bobFoaf>
15    { ?x foaf:mbox ?mbox . }
16 }
```

A SPARQL query consists basically of three parts: (a) the result specification (line 2), (b) the dataset definition part (lines 3–4), and (c) the restriction definition part (lines 5–16). For further explanation of the syntax we refer to [6].

Corresponding to the query structure, the generation of a QGM runs through the following stages:

1. Create a box for each data source.
2. Generate a box for graph patterns and value constraints and combine them by adding operations of the fusion category.
3. Depending on the type of the query construct a box which produces the result set.

Before we go into details, we present the corresponding query graph model of Example 1 in Figure 3.

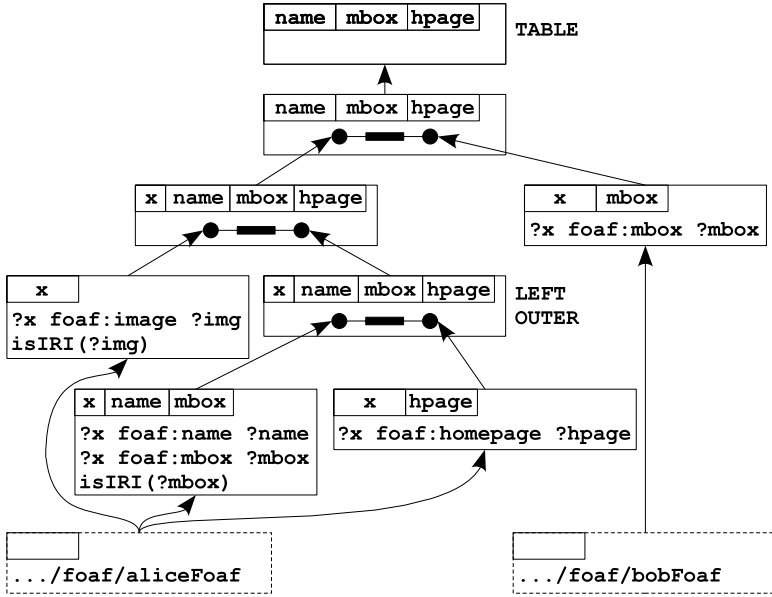


Fig. 3. Query graph model of Example 1

First, the access to a data graph is represented by a box with a dashed outline and the respective IRI as its body. This is independent of how an RDF graph is accessed during query execution – in context of the default graph or as a named graph. Furthermore, this box may carry additional information about the data source which are important for the QEP generation, e.g., latency of a data source in a distributed environment. If the default graph is constructed from several RDF graphs, an additional box is inserted into the QGM representing the merge of these graphs. Another special case occurs if the query contains a **NAMED** clause with a variable instead of the IRI of a named graph. In this case, an operation box is added which iterates over all given named graphs and binds the variable to the corresponding IRI. The head of this box holds the name of the variable.

The **WHERE** clause forms the main part of a query and possibly contains the following structures: (optional) graph patterns, union, and value constraints. To translate the **WHERE** clause we first create a box for each restriction (a graph

pattern or a value constraint). Hereby, the body of the box holds the restriction and the head holds the variables occurring in the restriction. To limit the number of generated boxes, graph patterns and value constraints on the same level are combined to a single box, e.g., lines 6–8 of Example 1 become a single box in Figure 3.

Next, data flow information is added to the QGM. Except for the union operator, which combines the results of two restrictions, the input of an operation originates either from the default graph or a named graph. Accordingly, data flows between the restrictions and their corresponding data sources are generated. Afterwards, the outputs of boxes being on the same level are combined by inserting join (a box with two connected circles) or union operations. Regarding joins we distinguish three cases:

- (a) If the box represents an **OPTIONAL** clause, we use an outer join ($\bowtie$, $\bowtie$) to indicate that the variables of this box need not to be bound, i.e., lines 6–8 and line 9 of Example 1 imply a left outer join;
- (b) if a value constraint is involved on the same level, we generate an appropriate θ -join ($\bowtie_\theta$), where θ stands for the value constraint. For example, Figure 4 shows a part of a query which retrieves all resources having birthday after `ex:alice`; and
- (c) if none of the first two cases apply we generate a natural join ($\bowtie$).

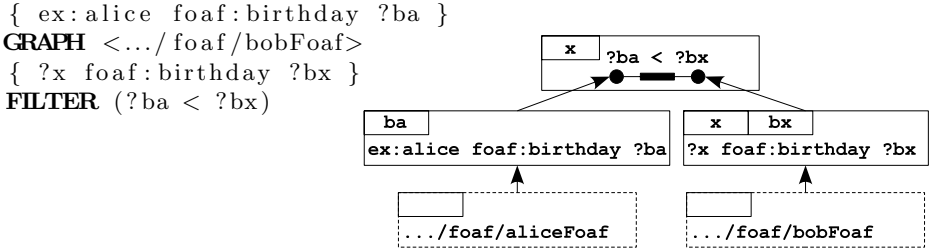


Fig. 4. Extract of a query (θ -join) and the corresponding part of the QGM

Finally, we describe the realization of solution sequence modifiers and result construction operations. The list of solution modifiers specified in SPARQL include **DISTINCT**, **LIMIT**, **OFFSET**, and **ORDER BY**. While the first three modifiers are translated into annotations of the box constructing the result, we decided to assign a separate box to the **ORDER BY** clause. This is because the **ORDER BY** clause may contain user-defined functions to determine the order and may have impact on the overall optimization of the query, e.g., an index could already provide the desired order. The ordering criteria are kept in the body of the box.

The upmost box of a QGM is responsible to construct the result of the whole query. In [6] there are several forms of output defined: a list with variable bindings, a graph, or a Boolean. Depending on the form of output the box contains different information that are summarized in Table 1.

Table 1. Information hold by a solution constructor

Keyword	Head	Body	Annotation
SELECT	projected variables	—	TABLE
CONSTRUCT	—	construction template	GRAPH
DESCRIBE	—	—	GRAPH
ASK	—	—	BOOLEAN

3.3 Transformation Rules

Since SPARQL is a declarative language there often exist several phrasings of a query. However, these should perform equivalently. Furthermore, the syntax of SPARQL allows constructing complex queries, e.g., by nesting graph patterns. Such queries often occur due to automatic generation of queries. Query rewriting using transformation rules is one way of meeting the challenge. Currently, we investigate rules mainly aiming at reducing the number of operations in the query graph model. Therefore, the list below is not yet complete.

Associativity of joins. Given three operations o_1, o_2 , and o_3 then the join order can be changed as follows:

- $(o_1 \bowtie o_2) \bowtie o_3 = o_1 \bowtie (o_2 \bowtie o_3)$
- $(o_1 \ltimes o_2) \bowtie o_3 = o_1 \ltimes (o_2 \bowtie o_3)$ and $(o_1 \bowtie o_2) \bowtie o_3 = o_2 \bowtie (o_1 \bowtie o_3)$

Eliminating (θ) -joins. If two restrictions operate on the same graph and participate in a (θ) -join then the three boxes can be merged into a single box. In case of a θ -join the former join condition becomes part of the new box. If the lists of bound variables differ the union of both list forms the set of variables provided by the new operation. Written as formula, let be r_1, r_2 two restrictions, be θ a value constraint, and be G a graph, then: $r_1(G) \bowtie_{\theta} r_2(G) = (r_1 \wedge r_2 \wedge \theta)(G)$

Eliminating redundant graph pattern. By appropriate application of the first two rules, operations are eliminated that contain the same restrictions (graph pattern or value constraints) and operate on the same graph.

Eliminating unsatisfiable boxes. If a box contains contradicting graph patterns or value constraints then the box and all connected parts can be removed from the QGM, for example a variable $?x$ can never satisfy the value constraint $\text{isURI}(?x) \ \&\& \ \text{isLiteral}(?x)$. This rule requires knowledge about the schema of the data source and inference. As a consequence it may become possible to immediately return the empty set or to delete a complete subgraph of a QGM, e.g., an unsatisfiable optional part of a query.

Eliminating variables. If a value constraint tests two variables for equality, e.g., $\text{FILTER}(?x = ?y)$, then all occurrences of the first variable can be substituted by the second and vice versa. Furthermore, variables that are not used in subsequent operations are removed from the head of a box.

Figure 5 depicts the transformed QGM of Figure 3 after applying the first three rules.

3.4 Aspects of Query Plan Generation

While the previous section proposed rewriting rules for the query graph model, this section provides an example of index selection, to demonstrate the

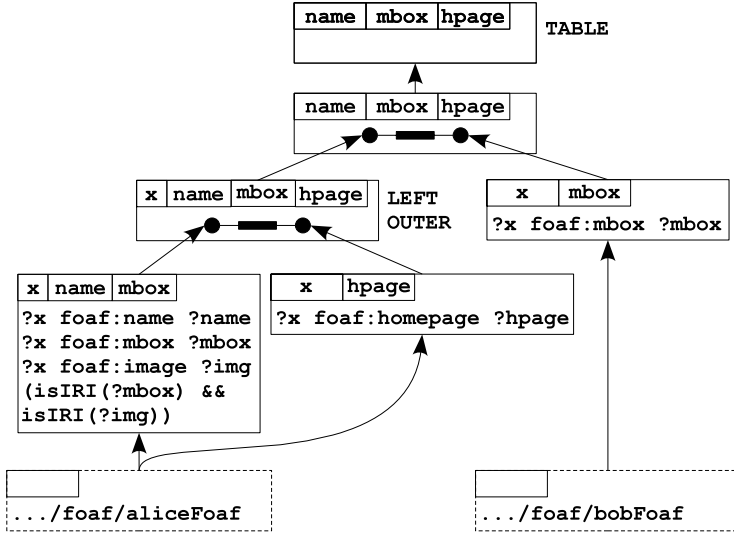


Fig. 5. Resulting query graph model after rewriting

advantages of the query graph model during the generation of QEPs. Besides query rewriting index selection is an important step during query optimization. In the context of databases indexes provide a structure to access database entries efficiently that satisfy a certain constraint. In most cases it is sufficient to create an index on a single attribute. Using relational database management systems (RDBMS) as storage for RDF data, we think that specialized index structures are necessary which are aware of the RDF data model. Otherwise, the RDBMS has, by some means or other, to cope with many join operations.

Making a first step towards index structures for RDF repositories, we assume that the user can create indexes on graph patterns, i.e., an index *idx* on all resources of *db* having the properties *foaf:name* and *foaf:mbox*:

```
CREATE INDEX idx ON db (?x foaf:name ?y ;
                           foaf:mbox ?z .)
```

Definition 1 (Graph Pattern Index). A graph pattern index *I* is a data structure defined on a graph pattern *P* providing an efficient access to solutions for *P*.

In our approach, the plan optimizer processes each box containing a graph pattern separately and determines the indexes to be invoked during the evaluation of this part. Thus, the plan optimizer has to solve the following problem:

Problem 1 (Index selection). Given a graph pattern *P* and a set of graph pattern indexes $I = \{I_1, \dots, I_n\}$, find a subset of graph pattern indexes $J \subseteq I$ such that *J* covers *P* to a maximum extent while minimizing the evaluation cost.

By increasing the number of triple pattern that can be considered at the same time, the optimizer can take more indexes into account and, therefore, choose

from a larger number of alternatives. In this regard, the query graph model becomes a key data structure, because the transformation rules lead to larger graph patterns.

In our future work, we will additionally investigate to what extent schema information can be exploited to rewrite graph patterns such that an index can be invoked. Besides index selection the plan optimizer has to make further decisions, which will be in the focus of our future research: exploit graph pattern entailment and determine the order of graph access.

4 Related Work

Although many approaches have been investigated on how to store and to query RDF data, less attention has been paid to query optimization as a whole. However, some phases of query processing have already been considered. Regarding query rewriting Cyganiak [7] and Frasnica et al. [8] proposed algebras for RDF derived from the relational algebra which allow to construct semantically equivalent queries. Furthermore, Serfiotis et al. developed algorithms for the containment and the minimization of RDF/S query patterns in [9] which only consider hierarchies of concepts and properties. However, all these approaches consider only a small part of the query processing, while the proposed query graph model supports all phases of query processing.

To improve the query execution time, several ways of storing RDF data were developed and evaluated, e.g., Jena [3], Sesame [2], Redland [4] and a path-based relational RDF database [10]. But as the introductory example demonstrated, we have reasons to believe that the developer of current RDF repositories have to re-engineer the query engines to declaratively deal with queries.

Christophides et al. focused on indexing RDF data, e.g., in [11] they developed labeling schemes to access subsumption hierarchies efficiently. Again, developing index algorithms is only a small part of query processing; it is also important to decide on the effectiveness of invoking an index.

5 Conclusion and Future Work

Over the last years several approaches on storing and querying RDF data have been developed and evaluated. Although some research has been undertaken to provide means for query rewriting, we think that query optimization as a whole has not been considered so far. In this paper, we proposed a query graph model (QGM) based on the work of Pirahesh et. al. [5] which supports all phases of the query processing. For instance, transformation rules enable the query processor to simplify the query specification. The transformation rules also support the selection of indexes to be invoked during query execution. Furthermore, the query graph model can easily be extended to represent new concepts. This is important since the current SPARQL specification defines only basic query structures, e.g., widely used structures such as subqueries, views, and group by are not supported at the moment, but will be added in near future.

Future work will include a more formal definition of the transformation rules and their implementation on the basis of the Jena Semantic Web Framework. Another research direction considers the index selection problem as described in Section 3.4. In this regard, we will also evaluate the usefulness of transformation rules based on schema information, e.g., `owl:inverseProperty`. As a long-term goal we will investigate extensions to the current SPARQL specification, e.g., subqueries and views, and develop appropriate transformation rules.

References

1. Alexaki, S., Christophides, V., Karvounarakis, G., Plexousakis, D., Tolle, K.: The RDFSuite: Managing Voluminous RDF Description Bases. In Decker, S., Fensel, D., Sheth, A.P., Staab, S., eds.: *Proceedings of the Second International Workshop on the Semantic Web*. (2001) 1–13
2. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A generic architecture for storing and querying RDF and RDF schema. In Horrocks, I., Hendler, J.A., eds.: *Proceedings of the First International Semantic Web Conference*. Volume 2342 of *Lecture Notes in Computer Science*, Springer (2002)
3. Wilkinson, K., Sayers, C., Kuno, H., Reynolds, D.: Efficient RDF Storage and Retrieval in Jena2. In Cruz, I.F., Kashyap, V., Decker, S., Eckstein, R., eds.: *Proceedings of the First International Workshop on Semantic Web and Databases*. (2003)
4. Beckett, D.: The Design and Implementation of the Redland RDF Application Framework. In: *Proceedings of the Tenth International Conference on World Wide Web*, New York, NY, USA, ACM Press (2001)
5. Pirahesh, H., Hellerstein, J.M., Hasan, W.: Extensible/rule based query rewrite optimization in Starburst. *SIGMOD Records* **21**(2) (1992) 39–48
6. Prud'hommeaux, E., Seaborne, A.: *SPARQL Query Language for RDF* (2006) W3C Working Draft.
7. Cyganiak, R.: A relational algebra for sparql. Technical Report HPL-2005-170, HP Laboratories Bristol (2005)
8. Frasnica, F., Houben, G.J., Vdovjak, R., Barna, P.: RAL: an Algebra for Querying RDF. In Feldman, S.I., Uretsky, M., Najork, M., Wills, C.E., eds.: *Proceedings of the 13th International conference on World Wide Web*, New York, NY, USA, ACM Press (2004)
9. Serfiotis, G., Koffina, I., Christophides, V., Tannen, V.: Containment and Minimization of RDF/S Query Patterns. In: *International Semantic Web Conference*. *Lecture Notes in Computer Science*, Springer (2005) 607–623
10. Matono, A., Amagasa, T., Yoshikawa, M., Uemura, S.: A path-based relational rdf database. In: *CRPIT '39: Proceedings of the sixteenth Australasian conference on Database technologies*, Darlinghurst, Australia, Australia, Australian Computer Society, Inc. (2005) 95–103
11. Christophides, V., Karvounarakis, G., Scholl, D.P.M., Tourtounis, S.: Optimizing Taxonomic Semantic Web Queries Using Labeling Schemes. *Web Semantics: Science, Services and Agents on the World Wide Web* **1**(2) (2004) 207–228

Author Index

- Albert, Manoli 63
 Arévalo Rosado, Luis Jesús 257
- Baumann, Peter 75
 Bauzer Medeiros, Claudia 110
 Belussi, Alberto 100
 Benjamins, V. Richard 269
 Bleiholder, Jens 425
 Bussler, Christoph 301
- Cachero, Cristina 329
 Calero, Coral 329
 Cantador, Iván 271
 Cappiello, Cinzia 339
 Cardoso de Paiva, Anselmo 150
 Castells, Pablo 271
 Cavassana Costa, David 150
 Chang, Jae-Woo 130
 Cherfi, Samira Si-Saïd 323, 325
 Chiang, Roger 171
 Claramunt, Christophe 73
 Condori-Fernández, Nelly 352
- de Souza Baptista, Cláudio 150
 de Vries, Denise 209
 Ding, Yihong 395, 415
 Dirk, Draheim 1, 43
 Domínguez, Eladio 237
 Dong, Chao 382
 Duval, Erik 372
- Eckstein, Silke 227
 Eder, Johann 183, 217
 Elmasri, Ramez 393
 Embley, David W. 395
- Fernández González, Juan María 257
 Fernández-Medina, Eduardo 32
 Ficiaro, Paolo 339
 Foxvog, Doug 301
 Freitag, Burkhard 53
- Genero, Marcela 329
 George, Betsy 85
- Grandi, Fabio 207, 435
 Guppenberger, Michael 53
- Hakkarainen, Sari 281
 Han, Hyoil 393
 Heese, Ralf 425, 445
 Hella, Lillian 281
 Hepp, Martin 269
 Herschel, Sven 425
 Hofreiter, Birgit 19
 Huemer, Christian 19
- Jing, Ning 120
- Karlapalem, Kamalakar 3
 Kasumi Sasaoka, Liliana 110
 Kim, Sang-Mi 130
 Kim, Yong-Ki 130
 Kim, Young-Chang 130
 Knolmayer, Gerhard F. 362
 Kopecký, Jacek 312
 Korherr, Birgit 7
 Krishna, P. Radha 3
 Kupfer, Andreas 227
 Kurz, Stefan 53
- Lanzenberger, Monika 405
 Lara, Rubén 271
 Lee, Hyunja 291
 Lee, Sang-goo 291
 Lee, Sang-Won 247
 Lee, Suekyung 291
 Lehmann, Marek 183
 Liao, Wei 120
 Liegl, Philipp 19
 List, Beate 7
 Lloret, Jorge 237
 Lonsdale, Deryle 415
 Lutteroth, Christof 43
 Lytras, Miltiadis 269
- Mandreoli, Federica 435
 Martoglia, Riccardo 435
 Mathiak, Brigitte 227
 Meireles Teixeira, Mario 150

- Meliá, Santiago 329
 Mišić, Vojislav B. 171, 203
 Muñoz, Javier 63

 Negri, Mauro 100
 Neumann, Karl 227

 Ochoa, Xavier 372

 Paolino, Luca 160
 Pastor, Óscar 63, 173, 193, 352
 Pelagatti, Giuseppe 100
 Pelechano, Vicente 63, 173, 193
 Pernici, Barbara 339
 Piattini, Mario 32
 Poels, Geert 323, 325
 Polo Márquez, Antonio 257

 Roddick, John F. 209
 Rodrigues da Silva, Elvis 150
 Rodríguez, Alfonso 32
 Ronchetti, Enrico 435
 Röthlin, Michael 362
 Rubio, Ángel L. 237
 Ruiz, Marta 193
 Ryu, Keun Ho 140

 Sampaio, Pedro R. Falcone 382
 Sampaio, Sandra de F. Mendes 382
 Sampson, Jennifer 405

 Scalas, Maria Rita 435
 Schuster, Rainer 19
 Sebillio, Monica 160
 Shekhar, Shashi 85
 Shim, Junho 291
 Song, Il-Yeol 5
 Strasunskas, Darijus 281

 Tahamtan, Amirreza 183
 Tang, Guifen 120
 Tao, Cui 415
 Tiberio, Paolo 435
 Torres, Victoria 173
 Tortora, Genny 160
 Trujillo, Juan 5
 Tung, Hoang Do Thanh 140
 Tuxen, Stine 281

 Vangenot, Christelle 73
 Vitiello, Giuliana 160

 Weber, Gerald 1, 43
 Wiggisser, Karl 217

 Xu, Li 395

 Zapata, María A. 237
 Zapletal, Marco 19
 Zhong, Zhinong 120