Lecture Notes in Computer Science 2503
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Tokyo

Stefano Spaccapietra Salvatore T. March
Yahiko Kambayashi (Eds.)

Conceptual Modeling ER 2002

21st International Conference on Conceptual Modeling
Tampere, Finland, October 7-11, 2002
Proceedings

Springer

Volume Editors

Stefano Spaccapietra
Ecole Polytechnique Federale Lausanne (EPFL)
Database Laboratory, School of Computer and Communication Sciences
1015 Lausanne, Switzerland
E-mail: stefano.spaccapietra@ep .ch

Salvatore T. March
Vanderbilt University
Owen Graduate School of Management
Nashville, TN 37203, USA
E-mail: sal.march@owen.vanderbilt.edu

Yahiko Kambayashi
Kyoto University
Graduate School of Informatics, Department of Social Informatics
Yoshida Sakyo Kyoto 606-8501, Japan
E-mail: yahiko@db.soc.i.kyoto-u.ac.jp

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Conceptual modeling : proceedings / ER 2002, 21st International Conference on Conceptual Modeling Tampere, Finland, October 7 - 11, 2002. Stefano Spaccapietra ... (ed.). - Berlin ; Heidelberg ; New York ; Hong Kong ; London ; Milan ; Paris ; Tokyo : Springer, 2002
(Lecture notes in computer science ; Vol. 2503)
ISBN 3-540-44277-4

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

ISSN 0302-9743
ISBN 3-540-44277-4 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by DA-TeX Gerd Blumenstein
Printed on acid-free paper SPIN 10870821 06/3142 5 4 3 2 1 0

Foreword

For more than 20 years, the series of Conceptual Modeling – ER conferences has provided a forum for research communities and practitioners to present and exchange research results and practical experiences in the fields of database design and conceptual modeling. Throughout the years, the scope of these conferences has extended from database design and specific topics of that area to more universal or refined conceptual modeling, organizing originally weak or ill-structured information or knowledge in more cultured ways by applying various kinds of principles, abstract models, and theories, for different purposes. At the same time, many technically oriented approaches have been developed which aim to facilitate the implementation of rather advanced conceptual models.

Conceptual modeling is based on the process of conceptualization, and it is the core of system structuring as well as justification for information systems development. It supports and facilitates the understanding, explanation, prediction, and reasoning on information and knowledge, and their manipulation in the systems, in addition to understanding and designing the functions of the systems.

The conceptualization process aims at constructing concepts relevant for the knowledge and information system in question. Concepts in the human mind and concept descriptions in computerized information systems are quite different things by nature, but both should be taken into account in conceptual modeling. Usually concept descriptions are properly observed, but concepts in the human mind and their properties are often neglected quite carelessly.

Conceptual models are created using these concepts. Conceptual modeling means creating conceptual models that describe the abstract system of the Universe of Discourse (UoD) and its information content, in which we are interested. Conceptual models are needed in designing and defining the knowledge content of the UoD. They consist of concepts and rules of the UoD, for example, the concepts and rules of an enterprise.

In recent years a great deal of work has been done in searching for basic systems of concepts for constructing various types of conceptual models, called ontologies, which can be applied to develop advanced, high-quality conceptual-level information systems. It seems that research of various ontology types and construction methods of conceptual-level information systems by using these various ontology types will engage researchers, teachers and philosophers for many years to come.

There are many approaches and aspects to be taken into account in conceptual modeling for information systems and databases. At the 21st International Conference on Conceptual Modeling – ER 2002, three keynote speakers explored special questions of conceptual modeling, and the authors of 30 conference papers presented their latest results in the field. In addition, the conference program consisted of five tutorials, four workshops, and one panel discussion. There were also poster and industrial presentation sessions.

The technical program of the conference was selected by the program committee consisting of three co-chairs and 68 members. The Program Committee Co-chairs, who also prepared the final program, were Stefano Spaccapietra, Salvatore T. March and Yahiko Kambayashi. The Tutorial Chairs, Veda C. Storey and Vijayan Sugumaran, gathered together five interesting tutorials. The Panel Chair Heinrich C. Mayr prepared the panel on "Do we need an ontology of ontologies?". I warmly thank them all for their excellent work on the conference. I would also like to thank the area liaisons, Tok Wang Ling, Klaus-Dieter Schewe, David W. Embley, and Alberto H.F. Laender, for supportive co-operation and publicity.

The Workshop Co-chairs, Antoni Olivé, Masatoshi Yoshikawa, and Eric S.K. Yu, selected four interesting workshops: "Evolution and Change in Data Management", "Conceptual Modeling Approaches to Mobile Information Systems Development", "Conceptual Modeling Quality", and "Conceptual Modeling Approaches for e-Business: A Web Service Perspective". I appreciate their accurate work very much. The papers of these workshops will be published in a separate LNCS volume.

I would also like to express my appreciation to other members of the organizing committee, Jyrki Nummenmaa (Local Arrangements, Demonstrations and Posters), Olavi Maanavilja (Industrial), Benkt Wangler (Publicity), Erkki Mäkinen (Registration), Martti Juhola (Treasurer), Jarkko Leponiemi and Toni Pakkanen (Webmasters), and Tapio Niemi and Kati Viikki (Social Activities).

In addition, I would like to express my appreciation to the University of Tampere and its Rector, Dr. Jorma Sipilä, for their important support and co-operation. I would like to extend my sincere thanks to the members of the Steering Committee, especially Tok Wang Ling (Chair), Bernhard Thalheim, and Peter P. Chen (Emeritus), who initiated these conferences and showed that there is much interesting work to be done in this field.

Last, but not least, I gratefully thank all the sponsors for their help and support, and the participants, who make a conference successful. I hope that this conference will be fruitful and valuable in the advancement of the research and practice of conceptual modeling.

October 2002 Hannu Kangassalo

Preface

Conceptual modeling is fundamental to the information systems discipline, including new e-world activities. It has become a major mechanism for understanding and representing organizations and the information systems that support them. ER 2002 encompasses the entire spectrum of conceptual modeling. It addresses research and practice in areas such as theories of concepts and ontologies underlying conceptual modeling, methods and tools for developing and communicating conceptual models, and techniques for transforming conceptual models into effective information system implementations, including advanced applications such as e-commerce, knowledge management, learning environments, telecommunications, and enterprise management systems.

Conceptual models instantiate various levels of abstraction. They must facilitate understanding and foster communication between technology experts and those who would benefit from the application of those technologies. They must enable users of these technologies to understand their current applications and visualize new applications. To do so our understanding and knowledge about information and how to describe, represent, and intelligently utilize it must be further developed. ER conferences are devoted to exposing and promoting advances in such development. They invite researchers and practitioners from both computer sciences and management information systems. Reports of new ideas and approaches, useful experiences and informative experiments are all welcome. Regular and industrial papers are solicited along with proposals for workshops, tutorials, panel sessions and posters. In particular, ER 2002 has emphasized conceptual modeling issues related to enterprise-wide information systems, and information systems to support virtual organizations.

Nearly 130 papers were submitted to the conference. Each paper was reviewed by program committee members and reviewers selected by them. After much electronic discussion and debate the Program Committee selected 30 of them for inclusion in the conference. These are organized into 10 sessions and address both theory and practice. Included are: two sessions dealing with Ontology, two dealing with methods, and one each dealing with applications, XML, quality, Web environments, meta-models, and integration. Research paper presentations are complemented in the conference program with invited keynote talks by three outstanding contributors, one panel on the use of ontology in conceptual modeling, and three tutorials. Abstracts of these additional presentations are included in this volume. The conference is also complemented with a series of workshops, whose proceedings are published as a separate volume.

Thanks are due to many people who worked to make the program a success. These include: the Workshop Co-Chairs, Antoni Olivé, Universitat Politècnica de Catalunya, Spain, Masatoshi Yoshikawa, NAIST, Japan, and Eric S.K. Yu, University of Toronto, Canada; the Tutorial Chairs, Veda C. Storey, Georgia State University, USA and Vijayan Sugumaran, Oakland University, USA; the

Panel Chair, Heinrich C. Mayr, University of Klagenfurt, Austria; the Industrial Chair, Olavi Maanavilja, M-real Corporation, Finland; and the Demonstration and Poster Chair, Jyrki Nummenmaa, University of Tampere, Finland. Since the program committee chairs were located on three different continents and none is located in Finland, coordination and communication were major issues. Jarkko Leponiemi of Tampere Polytechnic, Finland, did an outstanding job of managing the conference Website and the paper review system and coordinating communication with authors. Finally, we would like to thank all authors of submitted papers, who played the key role in materializing our dreams of an excellent conference.

October 2002

Stefano Spaccapietra
Salvatore T. March
Yahiko Kambayashi

ER 2002 Conference Organization

Conference Chair

Hannu Kangassalo, University of Tampere, Finland

Program Co-Chairs

Stefano Spaccapietra, EPFL Lausanne, Switzerland
Salvatore T. March, Vanderbilt University, USA
Yahiko Kambayashi, University of Kyoto, Japan

Workshop Co-Chairs

Antoni Olivé, Universitat Politècnica de Catalunya, Spain
Masatoshi Yoshikawa, NAIST, Japan
Eric S.K. Yu, University of Toronto, Canada

Tutorial Chairs

Veda C. Storey, Georgia State University, USA
Vijayan Sugumaran, Oakland University, USA

Panel Chair

Heinrich C. Mayr, University of Klagenfurt, Austria

Industrial Chair

Olavi Maanavilja, M-real Corporation, Finland

Demonstration and Poster Chair

Jyrki Nummenmaa, University of Tampere, Finland

Publicity Chair

Benkt Wangler, University of Skövde, Sweden

Local Organization Committee

Local Arrangements: Jyrki Nummenmaa, University of Tampere, Finland
Registration Chair: Erkki Mäkinen, University of Tampere, Finland
Webmasters: Jarkko Leponiemi, Tampere Polytechnic, Finland,
and Toni Pakkanen, University of Tampere, Finland
Treasurer: Martti Juhola, University of Tampere, Finland
Social Activities Chairs: Tapio Niemi, CERN, Switzerland,
and Kati Viikki University of Tampere, Finland

Steering Committee Representatives

Tok Wang Ling, Steering Committee Chair, National University of Singapore
Bernhard Thalheim, Brandenburg University of Technology, Germany
Peter P. Chen, Steering Committee (Emeritus), USA

Area Liaisons

Asia: Tok Wang Ling, National University of Singapore, Singapore
Australia and Pacific Area: Klaus-Dieter Schewe, Massey University,
New Zealand
North America: David W. Embley, Brigham Young University, USA
South America: Alberto H. F. Laender, Federal Univ. of Minas Gerais, Brazil

Workshops

ECDM 2002

2nd International Workshop on Evolution and Change in Data Management

Chairs: Fabio Grandi (University of Bologna, Italy)
John Roddick (Flinders University, South Australia)

MobIMod 2002

ER/IFIP WG8.1 Workshop on Conceptual Modeling Approaches to Mobile Information Systems Development

Chairs: John Krogstie (SINTEF and Norwegian Institute of Science and Technology, Norway)
Keng Siau (University of Nebraska-Lincoln, USA)
Kalle Lyytinen (Case Western Reserve University, USA)

IWCMQ 2002

International Workshop on Conceptual Modeling Quality

Chair: Mario Piattini (University of Castilla-La Mancha, Spain)

eCOMO 2002

Joint Workshop on Conceptual Modeling Approaches for e-Business: A Web Service Perspective

Chairs: Heinrich C. Mayr (University of Klagenfurt, Austria)
Willem-Jan van den Heuvel (Tilburg University, The Netherlands)

The proceedings for these workshops are published in a separate LNCS volume.

Program Committee

Gove Allen, Tulane University, USA
Paolo Atzeni, University of Rome, Italy
Hiroshi Arisawa, Yokohama National University, Japan
Dinesh Batra, Florida International University, USA
Joachim Biskup, University of Dortmund, Germany
Mokrane Bouzeghoub, University of Versailles, France
Marco A. Casanova, Catholic University of Rio de Janeiro, Brazil
Tiziana Catarci, University of Rome, La Sapienza, Italy
Stefano Ceri, Milan Polytechnic, Italy
Roger H.L. Chiang, University of Cincinnati, USA
Wesley Chu, University of California at Los Angeles, USA
Deb Dey, University of Washington, USA
David W. Embley, Brigham Young University, USA
Andreas Geppert, University of Zurich, Switzerland
Paulo Goes, University of Connecticut, USA
Nicola Guarino, National Research Council LADSEB-CNR, Italy
Terry Halpin, Microsoft Corporation, USA
Jean-Luc Hainaut, University of Namur, Belgium
Matthias Jarke, Technical University of Aachen, Germany
Paul Johannesson, Stockholm University, Sweden
Alberto Laender, Federal University of Minas Gerais, Brazil
Stephen W. Liddle, Brigham Young University, USA
Tok Wang Ling, National Univ. of Singapore, Singapore
Akifumi Makinouchi, Kyushu University, Japan
Heinrich C. Mayr, University of Klagenfurt, Austria
Robert Meersman, Free University of Brussels, Belgium
Elisabeth Metais, University of Versailles, France

Takao Miura, Hosei University, Japan
Mukesh Mohania, IBM. India Research Lab, India
Renate Motschnig-Pitrik, University of Vienna, Austria
John Mylopoulos, University of Toronto, Canada
Jyrki Nummenmaa, University of Tampere, Finland
Daniel O'Leary, University of Southern California
Antoni Olivi, University of Catalunya, Spain
Maria Orlowska, University of Queensland, Australia
Josi Palazzo de Oliveira, Federal University of Rio Grande do Sul, Brazil
Christine Parent, University of Lausanne, Switzerland
Guenther Pernul, University of Essen, Germany
Alain Pirotte, Catholic University of Louvain, Belgium
Sandeep Purao, Georgia State University, USA
Sudha Ram, University of Arizona, USA
John Roddick, Flinders University of South Australia, Australia
Elke A. Rundensteiner, Worcester Polytechnic Institute, USA
Sumit Sarkar, University of Texas at Dallas, USA
Klaus-Dieter Schewe, Massey University, New Zealand
Michael Schrefl, University of Linz, Austria
Wang Shan, Renmin University of China, China
Keng Siau, University of Nebraska-Lincoln, USA
Arne Solvberg, Norwegian University of Science and Technology, Norway
Il-Yeol Song, Drexel University, USA
Veda Storey, Georgia State University, USA
Vijayan Sugumaran, Oakland University, USA
Mohan Tanniru, Oakland University, USA
Zahir Tari, RMIT University, Australia
Toby Teorey, University of Michigan, USA
Bernhard Thalheim, Brandenburgian Technical University, Germany
Olga De Troyer, Vrije Universiteit Brussel, Belgium
Ramesh Venkataraman, Indiana University, USA
Yair Wand, University of British Columbia, Canada
Kyu-Young Whang, Korea Advanced Institute of Science and Technology, Korea
Roel Wieringa, University of Twente, The Netherlands
Carlo Zaniolo, University of California at Los Angeles, USA
Yanchun Zhang, University of Tasmania, Australia
Esteban Zimányi, Université Libre de Bruxelles (ULB), Belgium

External Reviewers

Zareh Aghbari, Kyushu University, Japan
Birger Andersson, Stockholm University, Sweden
Anteneh Ayanso, University of Connecticut, USA
Maria Bergholtz, Stockholm University, Sweden

Organized By

University of Tampere, Finland

Sponsored By

Association for Computing Machinery
The ER Institute
eTampere

In Cooperation with

Finnish Information Processing Association
City of Tampere

Corporate Sponsors

Nokia
TietoEnator

Table of Contents

Conceptual Modelling and Ontology: Possibilities and Pitfalls

Ron Weber[1]

Faculty of Business, Economics and Law
The University of Queensland
Australia 4072
weber@bel.uq.edu.au

The task of conceptual modelling is to build a high-quality representation of selected phenomena in some domain. The conceptual models that result facilitate the design, implementation, operation, and maintenance of information systems. Because conceptual modelling activities usually occur during the initial stages of the systems development process, modelling errors and omissions that are not detected early often have costly repercussions. Thus, research that seeks to improve conceptual-modelling practice is important.

Information systems professionals generate conceptual models (*scripts*) using (a) conceptual modelling *grammars* (e.g., the entity-relationship modelling grammar), and (b) conceptual modelling *methods*. Their work occurs within a particular organizational *context*. Research on conceptual modelling grammars, methods, scripts, and contexts offers many rich opportunities with potentially important theoretical and practical outcomes.

In this presentation, however, I will focus on some research possibilities that exist in relation to conceptual modelling grammars. In particular, I will argue that theories of ontology–theories about the structure and dynamics of the real world–can be used to provide the basis for evaluating existing grammars to determine their ability to generate models (scripts) that are complete and clear. I will describe briefly some studies that I have undertaken with colleagues to test predictions we have made based on our ontological evaluations of grammars. Some of our predictions have been counterintuitive and contrary to current conceptual modelling practice–for instance, that optional properties should not be used in conceptual models and that attributes should not be attached to relationships. Our empirical results, however, have supported out predictions. In essence, our theories have been "stronger" than out intuition.

I will also argue that theories of ontology should be used to provide the theoretical foundation for designing new conceptual modelling grammars. If we continue to develop new grammars without adequate theoretical foundations, we are doomed to repeat the errors of the past. We will have learned little from history. Nonetheless, much work on conceptual modelling grammars still proceeds almost devoid of any substantive theoretical basis. In due course, I believe the folly of this approach will become clear.

[1] Director of Research

Finally, I will discuss some pitfalls associated with using theories of ontology to evaluate conceptual modelling grammars and testing propositions based on these evaluations. In particular, I will examine how one chooses a theory of ontology in the first place to evaluate conceptual modelling grammars, whether relativistic notions undermine the value of ontological evaluations of grammars, and why testing propositions based on the evaluations often proves to be problematical.

An Ontology for m-Business Models

Yves Pigneur

University of Lausanne, Ecole des HEC, CH-1015 Lausanne
Yves.Pigneur@unil.ch

Abstract. The m-business landscape never stops to change and the impacts on the mobile market are constant as players reposition themselves on the market according to the new opportunities and threats brought by rapid technological developments. This paper proposes a conceptual tool to better understand this player arena. Its objective is to provide the researchers with an ontology for analyzing and assessing the business models adopted by these players.

1 Introduction

Nowadays rapid developments in wireless networks and mobile information systems, are observed and adopted, as illustrated by recent research projects and reports [5]. New business models are constantly emerging and can become a major stake in the e-business game. Understanding them and helping to design them are important issues. Conceptual modeling and formalized framework can help. We propose an ontology for defining business models in the m-business arena.

The next section sketches some research issues in the m-business arena. Then, section 3 tackles three perspectives of m-business: applications, open issues, and key players. Finally, section 4 suggests an ontology for assessing the business models.

2 Research in m-Business

There are several ways to assess the mobile technology, and to identify research issues in m-business. Well-known authors of the IS community recently published research directions and agendas. Among them, Lyytinen and Yoo [4] suggest a framework, which identifies research issues in nomadic computing environment at the individual, the team, the organizational, and the inter-organizational levels.

Lehner and Watson [2] concentrate on a stakeholder perspective, an application perspective, and a market player's institutional perspective. In the latter, they propose relevant research problems such as the business models, the useful alliances and the driving forces for cooperation, the interaction between market players, among others.

Camponovo and Pigneur [1] focus on the mobile market players landscape, and aim at sketching a conceptual tool for analyzing and visualizing the key players, their business models, their interactions and their dependencies.

3 Applications, Issues and Actors in m-Business

The demand for m-business *applications* is emerging. They concern information sharing or access, communication or messaging, and transaction or activity coordination. They deal with entertainment and gaming, healthcare, retailing, ticketing, in-car electronic, payment and cash, sale force management and CRM, team coordination and schedule synchronization. Mobile systems provide users and companies with solutions without regard to time and space.

The emerging m-business systems raise a set of commercial and research *issues* [7]. These issues can be mainly categorized into services issues (integration *Vs.* autonomy), infrastructure issues (operator-driven *Vs.* self-organized), and device issues (mono- *Vs.* multi-purpose). Some others concern the regulatory context (bandwidth and privacy).

The wireless market is highly fragmented and has witnessed a large number of market *actors*. The primary participants are access device manufacturers, content providers and aggregators, mobile network operators. Many other players show up in the landscape such as regulation authorities, standardization groups, consumer groups, airports and other “venues” (specially if one considers *WLAN* technology). Since no single player can provide its customers with an end-to-end solution on its own, fostering viable alliances and actors networks is a key challenge. Partnership management is becoming a core competence of the m-business players. Therefore it is not enough to examine the actor’s role. The relationships and interactions among the actors have to be assessed too.

For assessing the role of the different m-business key players, it is recommended to briefly but clearly describe their business models.

4 Ontology for m-Business Models

Our e-business model ontology outlines what value a company offers to which customer segments. It describes the architecture of the firm and its network of partners for creating, marketing and delivering value and relationship capital, in order to generate profitable and sustainable revenue streams.

We design this ontology based on an extensive literature review on business model [6] and on enterprise ontology. By merging the conceptually rich business model approach with the more rigorous ontological approach [8] and by applying it to e-business [3], we achieve an appropriate foundation for tools that would allow the understanding, sharing and communication, change, measuring and simulation of e-business models.

Our e-Business Model Ontology is the conceptualization and formalization into elements, relationships, vocabulary and semantics of the essential objects in the e-business model domain. The ontology is structured into several levels of decomposition with increasing depth and complexity. The first level of decomposition of our ontology contains the four main pillars of a business model, which are the products and services a firm offers, the relationship it maintains with its customers, the infrastructure necessary in order to provide this and finally, the financials, which are the expression of business success or failure (see figure 1).

The *product innovation* element covers all aspects related to the offering of the firm. This comprises not only its products and services but also the manner in which it differentiates itself from its competitors. The element product innovation is composed of the *value propositions* the firm offers to specific *target customer segments*, and the *capabilities* a firm has to provide in order to deliver this offering.

The *customer relationship* element describes the way a firm goes to market and gets in touch with its customers. This is composed of the *feel & serve* element, which defines the distribution channels, the *information strategy* for the collection and application of customer information for supporting customization and personalization, and the *trust & loyalty* element, which is essential in an increasingly "virtual" business world.

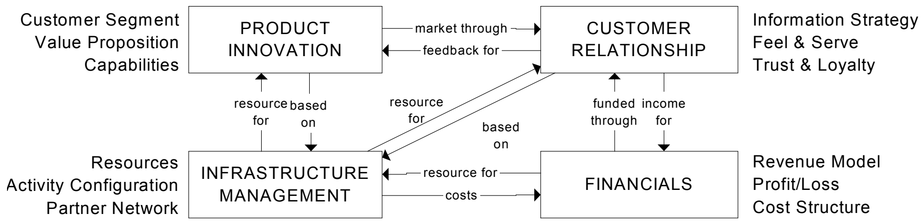


Fig. 1. The main components of the Business Model Ontology

The *infrastructure management* describes the value configuration that is necessary in order to deliver the value proposition and to maintain a customer relationship. It is composed of the *activity configuration* (value chain), the in-house *resources* and assets and the firm's partner network to fulfill these activities.

The *financials* are the culmination of an e-business model. The best products and services and the finest customer relationship are only valuable to a firm if it guarantees long-term financial success. The financial aspects element is composed of the company's *revenue model* and its *cost structure*, which determines the *profitability* of a company.

5 Conclusion

The future of m-business is very uncertain. Therefore it is recommended to adopt long-range strategic planning, scenario-based forecasting [9], and simulation approaches. Such approaches should be better supported and improved by conceptual modeling, ontology and other frameworks for defining and assessing business models.

6 References

1. Camponovo, G., Pigneur, Y.: Analyzing the Actor Game in m-Business. Proc. First International Conference on Mobile Business, Athens (2002)
2. Lehner, F., Watson, R: From e-Commerce to m-Commerce: Research Directions. Working paper. University of Regensburg: Chair of Business Informatics (2001)
3. Linder, J. C, Cantrell, S.: Changing Business Models: Surveying the Landscape. Working Paper, Institute for Strategic Change, Accenture (2001)
4. Lyytinen, K., Yoo, Y.: The Next Wave of Nomadic Computing: a Research Agenda for Information Systems Research. Sprouts Working Papers on Information Environments, Systems and Organizations, 1 (2001)
5. Müller-Veerse, F., et al. : UMTS Report – An Investment Perspective, Durlacher (2001)
6. Osterwalder, A., Pigneur, Y.: An e-Business Model Ontology for Modeling e-Business. Proc. 15th Bled Electronic Commerce Conference (2002)
7. Tarasewich, P., Nickerson, R., Warkentin, M.: Issues in Mobile E-Commerce, Communication of the Association for Information Systems, 8, January (2002) 41-64
8. Ushold, M., King, M.: Towards a Methodology for Building Ontologies, Proc. Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, August (1995)
9. Van der Heijden, K.: Scenarios: The Art of Strategic Conversation, John Wiley & Sons (1996)

Modeling Dynamics of Business Processes: Key for Building Next Generation of Business Information Systems

Ilia Bider[1] and Paul Johannesson[2]

[1]IbisSoft, Box 19567, SE 104 32 Stockholm Sweden
ilia@ibissoft.se
[2] Department of Computer and Systems Sciences
Stockholm University and Royal Institute of Technology
Forum 100, SE 164 40 Kista, Sweden
pajo@dsv.su.se

Abstract. During the past ten years, requirements on functionality of business information systems have been slowly changing. This shift consists of moving from traditional command based applications to the applications of workflow and groupware type. Such applications are aimed at "controlling" business processes. Designing an appropriate "control" system presumes that the nature of the process that we want to control is fully understood and modeled. The objective of the tutorial is "to understand the problems and solutions of designing a new generation of business information systems". It is intended for Software Engineers involved in business applications development, Business Analysts, and Researchers interested in business process modeling.

Today, at least half of the industrial software development is connected to business application development. During the past ten years, requirements on functionality of business applications have been slowly changing. This shift consists of moving from the command-based applications to the applications of workflow and groupware type. In other words, the shift can be described as moving from the traditional, "*human-assisting*" systems, to a new generation of "*human-assisted"* systems.

A *human-assisting* system helps a human being only to perform certain activities, e.g. to write a letter, to print an invoice, to complete a transaction, etc. The relations between these activities, and the aim of the whole process are beyond the understanding of the system, but are a prerogative of the human participant. In a *human-assisted* system, the roles are reversed, the system has a complete picture of the process and is involved in all activities. Only when the system cannot perform some activity on its own, it will ask the human participant for assistance.

The difference between the old and new generations is essential, and it can be traced in all aspects of system development, as shown in Table 1.

Table 1. Aspects of system development

Aspect	**Old generation**	**New generation**
Modeling	Data Modeling	Process Modeling
Data Base	Static and passive	Dynamic and active
User Interface	Functional (multilevel menus)	Navigational
Organizational aspect	Follows existing management schemes	Suggests new management schemes

A new generation of applications is aimed at "controlling" business processes. Designing an appropriate "control" system presumes that the nature of the process that we want to control is fully understood. That is why in Table 1, the traditional data modeling is substituted by process modeling.

According to the most general definition, a business process is a set of partially ordered activities aimed at reaching a well-defined goal, for example:

- Reaching an agreement in business negotiations.
- Discharging a patient from the hospital in a (relatively) healthy state.
- Closing a sale.

Each process engages a number of participants that can be roughly divided into two categories: passive participants, and active participants. Passive participants are the participants that are consumed, produced or changed during the execution of activities, for example, a document being written, a car being assembled, a patient being treated in the hospital, an organization being reorganized. Active participants, or agents, are those participants that perform actions aimed at the passive participants.

A business process is a complex phenomenon, and there are different methods of representing the development of the process in time. The following views on the process development are the most common:

1. Input/output flow. The focus is on passive participants that are being consumed, produced, or changed by the activities.
2. Workflow. The focus is on order of activities in time.
3. Agent-related workflow. The focus is on order in which the agents get and perform their part of work.
4. State flow. The focus is on changes produced in the part of the world that embraces the given process.

The view to take depends on many factors, one of them being the mission of the information system under development as shown in Table 2. Other can be defined as follows:

- degree of physicalness and mobility of passive participants,
- level of specialization and mobility of active participants
- degree of precision of operational goals,
- autonomy and friendliness of process environment,
- nature of activities,
- degree of orderliness of process flow,
- level of process maturity in the organization.

Building a model of a real business process is a challenging task because:

- Business processes are not always clearly visible as they may go through the whole, often functionally structured organization.
- Written information about business process is often non-existing or not reliable. The only practical way to obtain reliable information for creating a model of a real business process is by interviewing the people engaged in the process.

In many business domains, the experts are not technicians, but may be professionals of any kind, doctors, nurses, teachers, lawyers, clerks, etc. For these professional, presentation of the model in some formal language, or complex diagrammatic notation would be inappropriate. To have some means to present the experts with a process model in an understandable for them form is essential for the success of the modeling work.

Table 2. How to choose process view

System mission	**Process view**
Integrate existing systems	Input/output flow
Facilitate coordination / communication	Agent-related view
Introduce strict order in production-like processes	Workflow
Navigate each process to its goal	State flow

The means of representing a model for domain experts depend on the chosen view on business process dynamic. For the input/output, agent-related and workflow views, some form of a diagrammatic presentation is normally used. For the state flow view both diagrammatic, and non-diagrammatic presentations are possible. A non-diagrammatic presentation shows examples of the trajectories of the process in state space.

Ontology-Driven Conceptual Modelling

Nicola Guarino and Luc Schneider

National Research Council, Institute for Cognitive Sciences and Technologies
ISTC-CNR, c/o ISIB-CNR (formerly LADSEB-CNR)
Corso Stati Uniti 4, I-35127 Padova, Italy
Nicola.Guarino@ladseb.pd.cnr.it
schneider@ladseb.pd.cnr.it

In the last decade, ontologies have become a hot issue in the conceptual modeling community, which is chiefly due to the emergence of new application areas such as Electronic Commerce and the Semantic Web. The implementation of large-scale information systems, as well as their semantic interoperability, seem to be only tractable on the basis of ontologies as abtract domain models.

Despite the urgency of clarifying the basics of ontology engineering and modelling, practice in this area are still in their infancy, and inexperienced modelers with little ontological training are prone to make recurrent formal and conceptual mistakes that could be easily avoided.

This tutorial is intended for practitioners who are interested in designing ontologies to support knowledge engineering and management, database modeling, software engineering, as well as business process modeling and enterprise integration. Participants will walk away with a "tool bag" of solution strategies for common modeling problems, that are the result of a re-visitation of basic conceptual modeling primitives in the light of formal-ontological principles.

A prerequisite of the tutorial is a certain familiarity with first order logic; basic notions of modal logic are also welcome, but not mandatory. Unlike other ontology tutorials, we will only marginally address questions like "what is an ontology","what can you do with an ontology", or "why should I use an ontology", but assume participants have answered these questions for themselves and are already interested in building ontologies.

The tutorial will cover background material on conceptual modeling, knowledge structuring, and ontology in general. First we will present the basic conceptual tools for ontological analysis, including the theory of parts and wholes (mereology), the theory of unity and plurality, the theory of essence and identity, as well as the theory of dependence. Then we will show, how one can, on the basis of these tools, specify a minimal set of metaproperties characterising the ontological nature of concepts and relations used in conceptual models. These metaproperties can be used to develop a formal classification of concepts supporting basic conceptual modeling choices. The final part of the tutorial will be entirely devoted to guidelines for ontology design, especially for cleaning-up taxonomies.

Advanced OO Modelling: Metamodels and Notations for the New Millennium

Brian Henderson-Sellers

Faculty of Information Technology, University of Technology
Sydney, P.O. Box 123 Broadway 2007 NSW Australia
brian@it.uts.edu.au

Building an object-oriented model requires knowledge of process and techniques; whereas representing the model requires the use of a notation underpinned by a rigorous definition. Today this usually starts with a metamodel. Together, the metamodel and the notation are known as a "modelling language".

UML is the OMG's standard modelling language, accepted as Version 1.1 in September 1997, the current version being 1.4 [1]. To test out new ideas for possible inclusion in the evolving UML, a second modelling language, OML, has been developed in full cognizance of the OMG process. Its Version 1.0 was released in March 1997 [2] and Version 1.1 [3,4] is in accordance with the OMG standard. OML clarifies some of the ambiguities in the UML and offers further sophisticated and crucial extensions. As a UML variant [5], the OML is useful to test out some of the possible UML modifications that might go into UML2.0.

Advanced use of the Unified Modeling Language (UML) Version 1.4 is described both directly and in terms of possible future extensions (in Version 2.0) derivable from ideas tested in OML. In particular, concepts such as roles, stereotypes and responsibilities are discussed together with relationship modelling, with especial emphasis on the representation for the whole-part relationship. The tutorial provides sufficient information on a range of advanced modelling issues for the finer points of the architectural components of the UML to be used successfully on commercial projects.

References

1. OMG, 2001, OMG Unified Modeling Language Specification, Version 1.4, September 2001, OMG document ad/01-09-68 through 80
2. Firesmith, D., Henderson-Sellers, B. and Graham, I., 1997, *OPEN Modeling Language (OML) Reference Manual*, SIGS Books, NY
3. Firesmith, D.G. and Henderson-Sellers, B., 1998, Upgrading OML to Version 1.1: Part 1. Referential relationships, JOOP/ROAD, 11(3), 48-57
4. Henderson-Sellers, B. and Firesmith, D., 1998, Upgrading OML to Version 1.1: Part 2. Additional concepts and notations, JOOP/ROAD, 11(5), 61-67
5. Henderson-Sellers, B., Atkinson, C. and Firesmith, D.G., 1999, Viewing the OML as a variant of the UML, *«UML»'99 – The Unified Modeling Language. Beyond the Standard* (eds. R. France and B. Rumpe), Lecture Notes in Computer Science 1723, Springer-Verlag, Berlin, 49-66

Ontology-Driven Conceptual Modelling: Advanced Concepts

Nicola Guarino and Luc Schneider

National Research Council, Institute for Cognitive Sciences and Technologies
ISTC-CNR, c/o ISIB-CNR (formerly LADSEB-CNR)
Corso Stati Uniti, 4, I-35127 Padova, Italy
Nicola.Guarino@ladseb.pd.cnr.it
schneider@ladseb.pd.cnr.it

There is a growing awareness amongst conceptual modellers that their tools (languages and methodologies) are in need of theoretical elucidation and foundation. Conceptual modeling frameworks, e.g. ER or UML, presuppose ontological notions such as entity, object, class, attribute, association, and aggregation, which are still used in a more or less intuitive und unreflected way. Furthermore, the design and maintenance of large scale information systems presupposes well-understood and reusable top-level ontologies.

The tutorial is intended for researchers interested in ontological foundations of conceptual modelling frameworks, as well as in the role for well-founded ontologies for information systems.

In the first part of the tutorial, we will re-visit the basic conceptual modeling primives in the light of formal ontology. In this context, we will briefly discuss also related approaches based on the ontology of Mario Bunge. In the second part, we will outline our concept of a library of foundational ontologies, and present some elements from a first reference module, the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE). Finally, we will discuss the general perspectives of a unified ontology-driven conceptual modeling methodology.

Workflow Management in Electronic Commerce

Paul Grefen

Computer Science Department, University of Twente
P.O. Box 217, 7500 AE Enschede, Netherlands
www.cs.utwente.nl/~grefen
grefen@cs.utwente.nl

1 Introduction

In electronic commerce scenarios, effectiveness and efficiency of business process execution are of paramount importance for business success. Even more than in traditional commerce scenarios, they determine the chances of survival of organizations in fast moving, highly competitive electronic markets. To obtain the required levels of effectiveness and efficiency, well-structured automated business process support is required. For typical business-to-consumer (B2C) electronic commerce, process support is usually of an intra-organizational nature. For business-to-business (B2B) electronic commerce, however, process support across organizational boundaries is often required as the basis for virtual enterprises.

This tutorial addresses the application of workflow management (WFM) for process support in both these cases. The tutorial is organized into three parts. In the first part, we pay attention to 'classical' workflow management in the context of a single organization. In the second part, we extend this to workflow management across the boundaries of organizations. In the third part, we further extend this model by making service processes – implemented as workflows – the objects traded in e-commerce scenarios. We outline each part in the sections below.

2 Intra-organizational Workflow Management

The basics of intra-organizational workflow management are treated in short in the first part of this tutorial and placed in the context of electronic commerce. We start with discussing the overall goal and main functionalities of workflow management. Then, we pay attention to both conceptual modeling and high-level architectural aspects.

The main workflow management concepts are discussed and placed in a general workflow management model. This model consists of process, organization and information submodels. The right way of coupling these submodels is essential for flexible workflow management.

The conceptual workflow model is a basis for workflow management systems that manipulate instances of these models. The overall architecture of WFM systems is treated by means of reference architectures. We show how WFM systems fit in the context of information infrastructures.

3 Cross-Organizational Workflow Management

Cross-organizational workflow management (XO-WFM) is required in B2B e-commerce scenarios to support business processes spanning two (or more) organizations. XO-WFM requires specific extensions of basic workflow support as discussed in Part 1 of this tutorial, both conceptually and architecturally.

These extensions are treated in this part of the tutorial. We discuss which additional requirements have to be met by XO-WFM with respect to intra-organizational workflow management. This is the basis for XO-WFM concepts and models on the one hand and XO-WFM architectures on the other hand. We show how XML can be used to achieve interoperability in workflow specification between organizations. We discuss how chosen organization forms for e-commerce influence the choice of abstract architectures for workflow management.

XO-WFM is treated in the context of a number of B2B e-commerce application areas. One specific area – in which workflow management is the central ingredient – is further elaborated in the next part of this tutorial.

4 Workflow Management in Dynamic Service Outsourcing

An important class of B2B electronic commerce is dynamic service outsourcing, in which services are the objects of commerce. In this paradigm, consumer organizations buy the execution of services by provider organizations. Fine-grained monitoring and control of outsourced services is important in scenarios where complex, mission-critical processes are outsourced. Dynamic service outsourcing requires models and mechanisms for service trading between organizations and service enactment across organization boundaries – hence a tight coupling of e-commerce and workflow management must be supported.

The third part of the tutorial goes into the role of workflow management for service enactment in this context. We explain the concept of service outsourcing and illustrate the paradigm by means of an abstracted architecture. We show that the concept of electronic contract is required to formalize the business relationship between partners engaging in outsourcing. Workflow specifications of the services to be outsourced are part of these contracts. We discuss a high-level conceptual model for electronic contracts and show how this can be represented in an XML-based contract specification language. We end the tutorial by briefly presenting two business cases in which the service outsourcing paradigm is applied.

Do We Need an Ontology of Ontologies?

Heinrich C. Mayr

IWAS, Institute of Business Informatics and Information Systems
University of Klagenfurt
A-9020 Klagenfurt, AUSTRIA
mayr@ifit.uni-klu.ac.at

Ontologies are "in." For a number of years, ontology has been one of the areas under broad research attention, especially in the domain of information systems. Actually, there is nearly no IS-related conference that does not provide sessions or contributions related to ontology – the ER'2002 Conference is another proof of that.

There is a relatively clear understanding from the root domain, namely philosophy, what ontology is all about, namely to deal with theories about the nature of things in general (as opposed to theories of things in particular). However, this notion seems more and more to be used as a pure buzzword by various authors and in various fields. Today "ontology" comes with a bulk of rather different meanings: Some use it for standardized parameter definitions of technical interfaces, others for models or meta-models (e.g. "global schema"), others for explicit natural language specifications of domain specific vocabularies, others as "a core body of knowledge that can be incorporated into any knowledge base", others in it's initial philosophical meaning.

Ontologies are also related to different levels of generality (from "universal" via "domain specific" to "group specific" and "personal"), and they are discussed at several levels of formality. They are said to useful for developing methods and tools in the context of requirements elicitation and engineering for information systems, natural language processing and, especially, semantic web services by contributing to the quality of interoperating systems, and by reducing costs.

The panel session is intended to bring some light and structure into that muddle. Experts with different point of views will discuss their understanding of ontology and relate to topics like taxonomy of ontologies, inter-ontology relationships, relationships to modelling and metamodeling, quality of ontologies, grounding models/metamodels in ontologies, value and use of ontologies, limitations of ontologies, and others, thus trying to find an answer to the panel's title.

Development of a Conceptual Data Model for Digital Spatio-Temporal Geographical Information, with Application to Several Themes and GIS

Nico Van de Weghe

Department of Geography, Ghent University
Krijgslaan 281 Gebouw S8 (B.2.73)9000 Gent
nico.vandeweghe@rug.ac.be

A Geographical Information System (GIS) is a system capable of inputting, analyzing and visualizing geographical data. Despite several theoretical studies concerning spatio-temporal data models (e.g. snapshot model, space-time composite model, triad model), the current commercial systems do not support the temporal aspect, i.e. maintenance of history information and storage and querying and visualization of geographical data with a temporal dimension. As the link between the geographical data and the non-geographical attributes is important in a GIS, the (spatial) data model is mostly based on a hybrid approximation of a two- or three-dimensional space and an n-dimensional semantic table. A spatio-temporal data model, however, is far more complex than a space + time model due to the complexity of space and time.

The way people conceptualize space and time is an important consideration for the design of a temporal GIS, because a better match with people's thinking is expected to lead to easier-to-use information systems. The ontology of space-time, essential in order to develop a common conceptual framework, deals with the nature of space and time and their unique interactions. The spatio-temporal data model therefore has to deal with the various complexities of time: topology of the time-axis, multi-dimensionality of time, dating of time, multi-scalability of time, vagueness of time, complexity of changes, etc. The approach also needs to become broadly interdisciplinary. Existing theories in a range of fields, from physics and geography to cognitive psychology and philosophy need to be examined within the context of digital database representation and use.

Existing models with applications from e.g. criminology, geology and archaeology, as well as the new spatio-temporal conceptual model dealing with the spatio-temporal complexity will be implemented in ArcInfo 8.0.2. This software is used because of its extension of the geo-relational data model into an object-oriented model that allows users to add behavior, properties, rules, and relationships to their data. The diagram notation for ArcObjects model is based on UML. ArcObjects technology is based on the Component Object Model (COM) protocol. Customization is performed using the built-in Visual Basic for Applications (VBA) scripting capabilities. Using C++, one can also define custom features (objects with a geometric shape) with specialized behavior. The implementation will give feedback to guide further optimization.

Using Semantic Rules Database to Dynamically Set up the ICSpace Virtual Building

Ozifrankly Silva, Tatiana Tavares, Márcia Lucena, and Guido Souza Filho

DIMAp - Federal University of Rio Grande do Norte, 59072-970 Natal, Brazil
{ozzy,tati,guido}@natalnet.br
marciaj@dimap.ufrn.br
http://www.natalnet.br

Abstract. The ICSpace is based on a virtual building specified in VRML. In this virtual building, any kind of artistic manifestation, such as pictures, videos, poems, music is exhibited in thematic rooms. One of the main features of the ICSpace is that its VRML interface dynamically adapts itself to the quantity and the type of the works being exhibited. The focus of this work is to describe ICSpace implementation, based on rules database that supports reconfiguration of the virtual building.

1 Introduction

The ICSpace implementation is based on a multimedia database and a database management system that handles, manages medias and supports the virtual building reconfiguration. This reconfiguration gives an dynamic property to ICSpace. To support this dynamism, the VRML code of the ICSpace building elements are also stored in the database application.

2 The Proposal

The ICSpace is an opened space in the internet for art works exhibition and appreciation. For the artists, ICSpace provides the virtual space to exhibit their works and mechanisms to include and handle them. The visitors can appreciate the available works in virtual rooms, to get additional information regarding the works and communicate with the authors. The visitors may record commentaries, feel the presence of other visitors in the same room and interact with them. In the current stage of this project, different technologies were approached to reach the goals of each implementation phase, as VRML, HTML, Oracle 8.0.4 and PL/SQL to implement the triggers which perform the semantic rules. We also used JSP, servlets and JDBC to establish the communication between the Oracle database. One version of this system can be found in (http://www.natalnet.br/~icspace).

On the Transformation of Object Oriented Conceptual Models to Logical Theories: From EROOS to ID-Logic

Pieter Bekaert[*] and Bert Van Nuffelen[**]

Department Of Computer Science
KULeuven, Leuven, Belgium
{Pieter.Bekaert,Bert.VanNuffelen}@cs.kuleuven.ac.be

In [1], the authors present a semi-automatic transformation from object-oriented conceptual models to logical theories. By associating a logical theory with a conceptual model, the best of both worlds is combined. On one hand, the object-oriented software development paradigm is recognized to be well-suited to build maintainable and communicable conceptual models. On the other hand, the logical programming paradigm offers semantically founded concepts to represent knowledge and the powerful logical inference systems make it possible to prototype solutions to computational tasks.

More specific, the work is done in the context of the EROOS method[1] and ID-Logic. Both share the goal to represent a problem domain in a natural and declarative way. However they put different accents: Namely the EROOS method for building object oriented conceptual models stresses the importance of guiding the domain expert in the specification of his knowledge in an object oriented environment. Whereas ID-Logic stems from the research for a solid epistomological foundation for knowledge representation in a logical context.

Important is the strong link of ID-Logic with computational logic. Complex problems such as database querying, scheduling and planning specified in ID-Logic are nowadays within the reach of general purpose logical solvers. One of such solvers is the Asystem, an abductive solver. This solver handles ID-Logic theories by transforming them into a corresponding abductive logic program, the input of the Asystem. This connection enables the potential of generating prototype systems from an EROOS conceptual model through the defined transformation in [1]. The poster presentation will demonstrate all mentioned techniques separately together with the connection defined in [1].

[*] Research Assistant of the Fund for Scientific Research - Flanders (Belgium) (F.W.O.-Vlaanderen)

[**] Research supported by Research Fund K.U.Leuven

[1] All references to the mentioned methodologies and systems can be found in [1].

References

[1] Pieter Bekaert, Bert Van Nuffelen, Maurice Bruynooghe, David Gilis, and Marc Denecker. *On the Transformation of Object-Oriented Conceptual Models to Logical Theories.* In The 21st International Conference on Conceptual Modeling (ER2002). Springer-Verlag, 2002. Accepted.

Component Construction of Database Schemes

Bernhard Thalheim

Computer Science Institute
Brandenburg University of Technology at Cottbus
PostBox 101344, D-03013 Cottbus
thalheim@informatik.tu-cottbus.de

Abstract. Principles of database modeling have been intensively investigated in the late 70ies or early 80ies. The principles have been based on constructs such as subtypes, supertypes, restructuring through normalization, types construction by constructors, generic models and associations with pre-specified semantical meaning such as relationship types. Whenever a schema is becoming too large schema developers get lost in the web of types. The classical approach is a repair approach, i.e., whenever a schema becomes too large then use techniques for surveying. This paper aims in developing general principles for pragmatistic development of large database schemata: many-dimensionality, star and snowflake subschemata, bridges, nesting, lifespan, logs, meta-characterizations, variants and occurrences, quality, temporality and abstraction layers. Therefore, this approach allows to treat the 'lost on the schema' problem in parallel to development.

1 Introduction

Some Observations.

Beyond small examples used in monographs or textbooks, real-life database schemata tend to be large, unsurveyable, incomprehensible and partially inconsistent due to application, the database development life cycle and due to the number of team members involved at different time intervals. Thus, consistent management of the database schema might become a nightmare and may lead to legacy problems. The size of schemata may be very large[1,2].

There is a considerable effort for handling large schemata. In [Moo01] most of the methods proposed so far[3] have been generalized to *map abstractions*. Classical approaches are trying to get a treatment after the schema size has caused

[1] The SAP R/3 database schema uses meanwhile more than 21.000 relation types, 40.000 attributes, 40.000 views and more than 400.000 functions. The schema has a large number of redundant types which redundancy is only partially maintained.

[2] We observed a high similarity within the solutions while comparing, surveying and systematizing [Tha00b] the database schemata brought to our knowledge.

[3] Methods known in literature due to extensive research in the 80ies and early 90ies are structured entity charts, subject-oriented schemata, clustered entity models, structured data modeling, clustered ER models, leveled ER models, and financial services data models.

mismatches, redundancy, misinterpretation, etc. ('late cure medicine'). Abstraction of sub-schemata has been tackled by compression methods, e.g., [ACW93, RaS97, TWB89]. [Moo01] stated further that diagrams quickly become unreadable once the number of entity and relationship types exceeds about *twenty*.

Large database schemata can be drastically simplified if techniques of modular modeling such as *design by units* [Tha00a] are used. Modular modeling is an abstraction technique based on principles of hiding and encapsulation. Design by units allows to consider parts of the schema in a separate fashion. The parts are connected via types which function similar to bridges. Our approach allows to treat the schema complexity in a *preventive manner* instead of the classical approach based on *repair and cure*.

Constructing Database Schemes Based On Component Ware.

Systematic schema development can be based on guidelines for development of database applications such as [Iso91] or the codesign framework [Tha00a].

A better approach to scheme development is the use of building blocks and of composition methods for schema composition:

Development of building blocks: There are parts in the database schema which cannot be partitioned into smaller parts without loosing their meaning. Typical such parts are kernel types together with their specialization types. Kernel types may be represented by *star* or *snowflake sub-schemata*.

Development of composition methods: Composition of sub-schemata to larger schemata may be based on generalizations of operations used within the ER model itself. Typical composition methods are based on *bridge types*, *nesting of types*, *lifespan variation types*, *log/history types*, *meta-characterization types*, *occurrence types*, *temporality types*, and *abstraction association*.

Rules for application of composition methods: Constraints for application of composition methods allow to keep track on restrictions, special application conditions and on the context of the types to be composed.

Our approach is based on principles of component construction [Bro97]. The composition theory of stream processing functions [Bro97] is combined with the interface scripting [NiM95] on the basis of mixins [AnZ98]. In our understanding a database component is *database scheme that has an import and an export interface for connecting it to other components by standardized interface techniques*. A (HERM) scheme [Tha00a] consists of a structural description (schema) and a behavioral description.

Survey on the Paper.

The next section discusses inherent many-dimensionality within large database schemata and derives component separation. Based on this observation we derive building blocks for large schemata based on star and snowflake sub-schemata. Next we discuss the composition methods together with their rules for application. Due to space limitations we concentrate the paper on structural aspects (schemata). The treatment of the behavioral aspects is similar.

2 Component Construction Based on Dimensionality

Discovery of Internal Structuring.

Let us consider the schema[4] displayed in Figure 3 in the Appendix[5]. Information on products may be structured into information on the *kernel properties* of products such as the kinds of products ranging from physical products called item, the associations among products themselves and the characteristics of products, the *associations* to other ER types within the application such as to organizations acting in various roles, the *meta-characterization information* such as the application conditions and the categorization, and the *log and utilization information*. Thus, the product schema itself has four different dimensions.

Database schemata are structured by a number of dimensions:

Specialization dimension: Types may be specialized based on roles objects play or on categories into which objects are separated. Specialization dimensions usually lead to hierarchies such as subtypes, role hierarchies, categorization hierarchies, and to version dimensions (development, representational, or measure versions).

Association dimension: Things in reality do not exist in separation. Therefore, we are interested too in representing their associations through bridging related types and in adding meta-characterization on data quality.

Usage, meta-characterization or log dimension: Data may be integrated into complex objects at runtime, may store log information such as the history of database evolution, the association to business steps and rules, and the actual usage of the information stored in the database at a certain time. Further, objects may described with meta-properties such as category, source and quality information.

Data quality, lifespan and history dimension: Since data may vary over time and we may have used at different moments of time different facts on the same thing we model the data history. Typical information added for characterizing data collects quality information, e.g. source data, data on the business process, data on source restrictions, data quality parameters etc. The history and time dimensions distinguish between transaction time, user-defined time, validity time, and availability time.

[4] The schema is not complete. Since we are interested in discussing the product schema we do not represent in detail modeling of *Party* and *PartyAddress*. The complete schema is discussed in [Tha00b].

[5] We use the extended entity-relationship model (HERM) [Tha00a] for the representation. It generalizes the classical entity-relationship model by adding constructs for richer structures such as complex nested attributes, relationship types of higher-order, cluster types that allow disjoint union of types (displayed by $\bigoplus$), by an algebra of operations, by rich sets of integrity constraints, by transactions, by workflows, interaction stories, and by views. An IsA-association and the subtype can be compacted to a unary relationship type which can be extended by specific identification. Techniques for translation of a HERM specification to relational and object-relational specification are discussed in detail in [Tha00a].
For surveys etc. see: http://www.informatik.tu-cottbus.de/∼thalheim/slides.htm

One dimension considered to be harmful is the explicit maintenance of schema parts that contain analysis/design types together with implementation types, storage types and representation types, thus, creating a redundant schema.

Intext.

According to [Wis01], a set of properties that corresponds to a specific application context is called *intext*. The internal structuring of a kernel type such as *Product* is called *intext*. Intext is based on the three major kinds of abstraction [Tha00a]: Component abstraction used for construction of types by constructors $\subset, \times$ and $\mathcal{P}$, localization abstraction for representing repeating, shared or local patterns of components or functions that are *factored out*, and implementation abstraction or modularization that allows [Tha00a] to selectively retain information about structures.

Context.

The associations to other related units or sub-schemata is called the *context* of the type. Objects and things they represent do not exist as stand-alone concepts. They are related to other objects and they have a context in applications. There are several (orthogonal) dimensions for context description, e.g.:

schema and units associated are connected through bridges,
source and acquisition information is an orthogonal dimension of context,
time dimension may occur in various facets of time,
versions are used to show the life cycle of the objects, and
security information is another orthogonal dimension.

Components in General.

We generalize the theory of components [Bro97] to database components. Components may be considered as input-output machines that are extended by the states Σ of the database with corresponding input view $I^{\mathcal{V}}$ and output view $O^{\mathcal{V}}$. Input and output of components is based on channels C. The structuring of channels is described by the function $type : C \rightarrow \mathcal{V}$ for the view schemata $\mathcal{V}$. The views are used for collaboration with the environment via data exchange. In general, the input and output sets may be considered as abstract words from M^* or as words on the database structuring.
A database component $C = (I^{\mathcal{V}}, O^{\mathcal{V}}, \Sigma, \Delta)$ is specified by

syntactic interface providing names (structures, functions) with parameters and database structure for Σ and $I^{\mathcal{V}}, O^{\mathcal{V}}$,

behavior relating the $I^{\mathcal{V}}, O^{\mathcal{V}}$ (view) channels
$\Delta : (\Sigma \times (I^{\mathcal{V}} \rightarrow M^*)) \rightarrow \mathcal{P}(\Sigma \times (O^{\mathcal{V}} \rightarrow M^*))$
and an initial state σ_0 .

Component operations such as merge, fork, transmission are definable via application of superposition operations (identification of channels, permutation of channels, renaming of channels, introduction of fictitious channels, and composition (parallel composition with feedback)).
The composition $C_1 \otimes C_2 = (I^{\mathcal{V}}, O^{\mathcal{V}}, \Sigma, \Delta)$ of two components $C_1 = (I_1^{\mathcal{V}}, O_1^{\mathcal{V}}, \Sigma_1, \Delta_1)$ and $C_2 = (I_2^{\mathcal{V}}, O_2^{\mathcal{V}}, \Sigma_2, \Delta_2)$ is defined via assignment

of syntactic interfaces (with unification of output channels to input channels) $I^{\mathcal{V}} = (I_1^{\mathcal{V}} \cup I_2^{\mathcal{V}}) \setminus (O_1^{\mathcal{V}} \cup O_2^{\mathcal{V}})$, $O^{\mathcal{V}} = (O_1^{\mathcal{V}} \cup O_2^{\mathcal{V}}) \setminus (I_1^{\mathcal{V}} \cup I_2^{\mathcal{V}})$, and internal channels $Z^{\mathcal{V}} = (I_1^{\mathcal{V}} \cap O_2^{\mathcal{V}}) \cup (I_2^{\mathcal{V}} \cap O_1^{\mathcal{V}})$.
The composition of the behavior function $\Delta = \Delta_1 \otimes \Delta_2$ defined by

$$\Delta((\sigma_1, \sigma_2), x) \;=\; \{\, ((\sigma_1', \sigma_2'), y|_{O^{\mathcal{V}}}) \mid y|_{I^{\mathcal{V}}} = x|_{I^{\mathcal{V}}} \wedge \\ (\sigma_1', y|_{O_1^{\mathcal{V}}}) \in \Delta_1(\sigma_1, y|_{I_1^{\mathcal{V}}}) \wedge (\sigma_2', y|_{O_2^{\mathcal{V}}}) \in \Delta_2(\sigma_2, y|_{I_2^{\mathcal{V}}}) \,\}$$

for $\Sigma = \Sigma_1 \times \Sigma_2$. Composition is displayed in Figure 1.

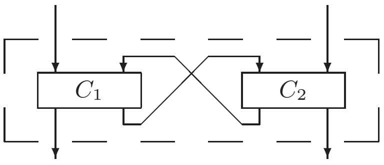


Fig. 1. The Composition of Database Components

3 Star and Snowflake Sub-schemata as Building Blocks

Practioneers observed that star or snowflake schemata are often used in practical database operating. The OLAP community restricted database operating to star and snowflake schemata as the *main building block* for schemata. Star and snowflake schemata can be supported by database views in traditional way [LST99]. Star structuring and snowflake structuring is becoming popular in the XML community. XML documents are nothing else than HERM views on the HERM database. This point of view enables in DBMS-supported, consistent maintenance of XML-based web sites.

Star Schemata.

A star schema for a database type C_0 is defined by

- the (full) (HERM) schema $\mathcal{S} = (C_0, C_1, ..., C_n)$ covering all types on which C_0 has been defined,
- the subset of *strong types* $C_1, ..., C_k$ forming a set of keys $K_1, ..., K_s$ for C_0, i.e., $\cup_{i=1}^{s} K_i = \{C_1, ..., C_k\}$ and $K_i \rightarrow C_0$, $C_0 \rightarrow K_i$ for $1 \leq i \leq s$ and $card(C_0, C_i) = (1, n)$ $(1 \leq i \leq k)$.
- the extension types $C_{k+1}, ..., C_m$ satisfying the (general) cardinality constraint $card(C_0, C_j) = (0, 1)$ $((k+1) \leq i \leq n)$.

The extension types may form their own $(0, 1)$ specialization tree (hierarchical inclusion dependency set). The cardinality constraints for extension types are partial functional dependencies.

There are various variants for representation of a star schemata:

- Representation based on an entity type with attributes $C_1, ..., C_k$ and $C_{k+1}, ..., C_l$ and specializations forming a specialization tree $C_{l+1}, ..., C_n$.

- Representation based on a relationship type C_0 with components $C_1, ..., C_k$, with attributes $C_{k+1}, ..., C_l$ and specializations forming a specialization tree $C_{l+1}, ..., C_n$. In this case, C_0 is a *pivot element* [BiP00] in the schema.
- Representation by be based on a hybrid form combining the two above.

Let us consider the example in Figure 3. The inherent star schema is represented in Figure 2. It is based on the entity-representation option.

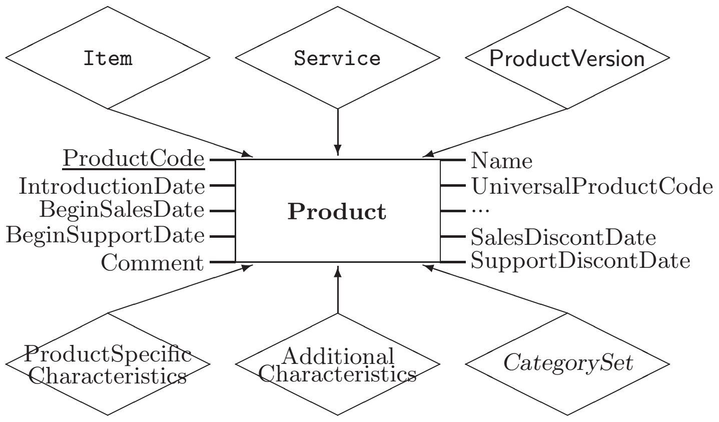


Fig. 2. Star Schema *Product_Intext_Data* for the *Product* Application in Figure 3

Thus, a **star schema** is usually characterized by a kernel entity type used for storing basic data, by a number of dimensions that are usually based on subtypes of the entity type such as `Service` and `Item`, and on subtypes which are used for additional properties such as *AdditionalCharacteristics* and *ProductSpecificCharacteristics*. These additional properties are clustered according to their occurrence for the things under consideration. Furthermore, products are classified by a set of categories. Finally, products may have their life and usage cycle, e.g., versions. Therefore, we observe that the star schema is in our case a schema with four dimensions: subtypes, additional characterization, life cycle and categorization.

Varying Star Sub-Schemata for Variable Integration.

Star schemata may occur in various variants within the same conceptual schema. Therefore, we need variants of the same schema for integration into the schema. We distinguish the following variants:

Integration and representation variants: For representation and for integration we can define views on the star type schema with the restriction of invariance of identifiability through one of its keys. Views define ‘context’ conditions for usage of elements of the star schema.

Versions: Objects defined on the star schema may be a replaced later by objects that display the actual use, e.g., *Products* are obtained and stored in the *Inventory*.

Variants replacing the entire type another through renaming or substitution of elements.

History variants: Temporality can be explicitly recorded by adding a history dimension, i.e., for recording of instantiation, run, usage at present or in the past, and archiving.

Lifespan variants of objects and their properties may be explicitly stored. The lifespan of products in the acquisition process can be based on the *Product-Quote-Request-Response-Requisition-Order* cycle.

Snowflake Sub-Schemata.

Star schemata may be extended to snowflake schemata. Database theory folklore uses star structures on the basis of α-acyclic hypergraphs. Snowflake structuring of objects can be caused by the internal structure of functional dependencies. If for instance, the dependency graph for functional dependencies forms a tree then we may decompose the type into a snowflake using the functional dependency $X \rightarrow Y$ for binary relationship types R on X, Y with $card(R, X) = (1, 1)$ and $card(R, Y) = (1, n)$.

This observation is not surprising. The only case when decomposition algorithms are applicable and generate the same results as synthesis algorithms is the case that the dependency graph for functional dependencies is hierarchical.

Let us now generalize this observation. A snowflake schema is a

- star schema $\mathcal{S}$ on C_0 extended or changed by
  - variations $\mathcal{S}^*$ of star schema (with renaming)
  - with strong 1-n-composition by association (glue) types $A_{\mathcal{S}}^{\mathcal{S}'}$ associating the star schema with another star schema $\mathcal{S}'$ either with full composition restricted by the cardinality constraint $card(A_S^{S'}, S) = (1, 1)$ or with weak, referencing composition restricted by $card(A_S^{S'}, S) = (0, 1)$,
- and which structure is potentially C_0-acyclic.

A schema $\mathcal{S}$ with a 'central' type C_0 is called *potentially* C_0*-acyclic* if all paths p, p' from the central type to any other type C_k are

- either entirely different on the database, i.e., the exclusion dependency $p[C_0, C_k] \,||\, p'[C_0, C_k]$ is valid in the schema
- or completely identical, i.e. the pairing inclusion constraints $p[C_0, C_k] \subseteq p'[C_0, C_k]$ and $p[C_0, C_k] \supseteq p'[C_0, C_k]$ are valid.

The exclusion constraints allow to form a tree by renaming the non-identical types. In this case, the paths carry different meanings. The pairing inclusion constraints allow to cut the last association in the second path thus obtaining an equivalent schema or to introduce a mirror type C_k' for the second path. In this case, the paths carry identical meaning.

4 Composition of Sub-schemata

We consider now a schema defined as a component. A database component $C = (I^{\mathcal{V}}, O^{\mathcal{V}}, \Sigma, \Delta)$ is called *unwrapped* if the schema $\mathcal{S}$ of Σ is identical to $I^{\mathcal{V}}$ and $O^{\mathcal{V}}$ and called wrapped in the other case.

A database component is called *symmetric* if $I^{\mathcal{V}} = O^{\mathcal{V}}$ and the views are a sub-schema of $\mathcal{S}$. Symmetric components can be represented by framed schemata. *We introduce composition of schemata based on symmetric components.* The extension to database components is similar.

A framed (component) schema is a symmetric database component

$\mathcal{S}_F$ = *F(interface types; schema types)*

i.e., $F(\alpha_1, ..., \alpha_n; R_1, ..., R_m)$.

The interface types can be considered as parameters to be used for composition. A framed schema without parameters is finalized.

4.1 General Composition Constructors

The general composition theory may be based on the theory of graph grammars [EEK99] which has been already used for the CASE tool RADD [Tha00a].

Composition is based on constructors used for definition of HERM types:

Composition through association is based on the (Cartesian) product constructor (or tuple constructor). It is used for direct association of database components. The Cartesian product can be restricted and can be combined with projection, i.e., with views on the combined schema. Thus, we may use the join constructor

$$\bowtie_{InputView_1 := OutputView_2}$$

or in detail $\bowtie_{(I^{\mathcal{V}}_{1,2} := O^{\mathcal{V}}_{2,1}) \wedge (I^{\mathcal{V}}_{2,1} := O^{\mathcal{V}}_{1,2})}$

for the composition $C_1 \otimes C_2 = (I^{\mathcal{V}}, O^{\mathcal{V}}, \Sigma, \Delta)$ with the association of the view $I^{\mathcal{V}}_{1,2}$ of the first component to the view $O^{\mathcal{V}}_{2,1}$ of the second component and of $I^{\mathcal{V}}_{2,1}$ to $O^{\mathcal{V}}_{1,2}$, correspondingly, instead of the Cartesian product.

The views to be joined must be *type-compatible*, i.e., the assignment operation is supported by the type inference system that is derived from the schema specification of the components.

We distinguish the following variants:

Joining types: Types combined by an injective function associating each type of the assignment view with one and only one type of the assigned view.

Associating through linking and referencing: Association is based on a variety of link properties such as *potential integration*, *existence constraints*, *referential integrity* with enforcement or with assurance, and *citations* without integrity. Furthermore, functionality of destination component may be inherited.

General schema join: Association may be based on cooperating views [Tha00a]. The data exchange is explicitly specified by views on the views which are type-compatible and through which functionality is partially exported.

Folding and unfolding schemata: Schemata which are similar can be integrated into one schema by adding a contraction type, by nesting of components or by generalizing or parameterizing the presentation:

Bulk construction: Given a *CentralType C* and associated types which are associated by a set of relationship types $\{A_1, ..., A_n\}$ by the occurrence frame F. The occurrence frame can be $\forall$ (if the inclusion constraints $A_i[C] \subseteq A_j[C]$ are valid for all $1 \leq i, j \leq n$) or a set of inclusion constraints. Now we combine the types $\{A_1, ..., A_n\}$ into the type *BulkType* with the additional component *ContractionAssistant* and the attributes *Identif* (used for identification of objects in the type *ContractionAssisitant* if necessary), *ContractionDomain* with *dom(ContractionDomain)* = $\{A_1, ..., A_n\}$ and *dom(ContractionBinder)* = F.

Component nesting is based on parameterizations of the schema used for the addition of components. Identification of components is inherited to the type using the component. The default inheritance is based on the primary identification. We may however use secondary identification due to specifics of the application.

Formally, given a sub-schema `R` with a component `S` and a sub-schema `T`, one of its identification `Identif` and some additional components `OtherComponents`. Component nesting is defined by

`Component(R(S),T(Identif,OtherComponents))` .

Component nesting generalizes entity model clustering, entity clustering, entity and relationship clustering, entity tree clustering and design-by-units method [Tha00a].

Presentation nesting: Each class can be displayed in various different formats depending on the usage, profiles of the user, and technical environment. We use *views* for generating the variety of different formatting dimensions. Typical such formatting dimensions may be hierarchical schemata used for website documents. Necessity of nesting has been also observed while adding measures applicable to types in various fashions. We may partially keep this information in a separate form and then add the corresponding bulk types to the schema.

Collection construction allows to integrate schemata into one schema and to consider the schemata as elements of the generalized schema. Specific class constructions are based on the following constructors:

Set constructor: Schemata are composed to a set of schemata. We must maintain identifiability of the elements and may add additional properties. Thus, we use the formal representation

`Set(Element(Identif,OtherComponents),AdditionalAttr)`

with the parameters `Element, Identif, OtherComponents` and `AdditionalAttr` constrained by the restriction that the type `Element` uses `Identif`.

Set-of framed schema are supported by set operators such as select-n-by-criteria, select-arbitrarily-n, disable optional components.

Bag constructor: Bags are collections of elements which may not be unique in the set and occur several times. The order of the elements is unimportant.

We use the formal representation
`Bag(Element(Identif,OtherComponents),OccurNo,`
`AdditionalAttr)`
with the parameters `Element, Identif, OtherComponents, OccurNo` where `OccurNo` is used for representing the number of occurrences of the corresponding element.

Tree constructor: Schemata may be related to each other by a tree and represented by
`Tree(Element(Identif,OtherComponents),AdditionalAttr)` .

List constructor: Lists are based on sequential association of schemata and represented by
`List(Element(Identif,OtherComponents),AdditionalAttr)` .
The list constructor can be generalized to the *multi-list constructor*.

4.2 Main Composition Operations

We observe six main composition operators which can be used for constructing larger schemata from smaller ones:

Composition through bridge or hinge types $\bowtie$ allows to associate two or more subschemata into one schema. Bridges enable in separating units in schemata from other units. Hinge types are simple gluing types. Hinge types may be entity, relationship or cluster types.
A typical hinge type used in Figure 3 is the relationship type *ProducerOf*. This type associates organizations and people to *Product*.

Composition through contraction or nesting of types ν is used to abstract from specific properties and to combine several types into one type using an additional *folder* type.
For instance, the associations among *Product* in Figure 3 can be presented by the types *ConsistsOf*, *ProductSubsitute* and *ProductObsolence*. We can use the entity type *AssociationKind* with the attributes *Identif* and *Description* and the domain
dom(Description) = { *ConsistsOf*, *ProductSubsitute*, *ProductObsolence*} .
The three association types can be equivalently represented by the type
Associated = (IsAssociated : Product, With : Product, AssociationKind, attr(Associated))
with a set of attributes additionally characterizing the association.

Composition through basing types on other types allows to represent the development of real-life things and to store information gained and added throughout the lifespan of the things.

Composition through adding orthogonal dimensions such as time, quality information, and meta-characteristics.

Composition through adding records on utilization of objects and materializing actions performed within the database.

Composition through adding facilities for versions and occurrences allows to keep track on the development of real-life things and their corresponding objects.

The composition rules do not use renaming. This list may be extended by other ER schema production rules. The last four composition operators can be expressed through the first two composition operators.

The second composition rule forms a critical pair together with each the other composition rules. We may, however, prove that conflicts may be resolved:

Theorem 1 *If the set of framed schemes is finite then the six composition operators are confluent, satisfy the Church-Rosser property and are acyclic.*

The proof of this theorem is based on the property that HERM schemata are hierarchical schemata and the set of critical pairs is finite.

Let us now consider the construction of the *Product_Schema* displayed in Figure 3. This schema is constructed using the following sub-schemata:

- the sub-schema *Product_Intext_Data* in Figure 2 consisting of the types *Product*, *Item*, *Service*, *ProductCharacteristics*, and the sub-types of the last two types and the framed variant
  Product_Schema $(\lambda, \mathit{Product})$;
- the framed *Categorization* schema
  Categorization_Schema $(\alpha, \mathit{Code};$
  $\mathit{ProductCategoryClass}(\alpha, \mathit{ProductCategory}(\mathit{Code})))$,
- the framed *Availability* schema
  Availability_Schema $(\alpha; \mathit{ProductObsolesence}(\alpha))$,
- the framed *Product_Association* schema
  Product_Associations_Schema $(\alpha, \beta, \delta;$
  $\mathit{Associates}(\mathit{what} : \alpha, \mathit{toWhat} : \beta, \mathit{kind} : \delta))$,
- the framed *Inventory* schema
  Inventory_Schema $(\gamma, \epsilon; \mathit{Inventory}(\mathit{what} : \gamma, \mathit{store} : \epsilon))$
  with the labels *what* and *store* for the components of the binary type *InventoryItem*,
- the framed *Container* schema
  Container_Schema $(\zeta, \eta; \mathit{Container}(\mathit{kind} : \zeta, \mathit{partyAddress} : \eta))$
- the framed *Producer* schema
  Producer_Schema $(\alpha; \mathit{ProducerOf}(\alpha, \mathit{Organization}))$
- and the *Pricing_Schema* schema which is itself a complex framed schema
  Princing_Schema $(\alpha; \mathit{ProductPriceComponent}(\alpha))$.

The *Product_Schema* in Figure 3 is obtained by

$$\begin{aligned}
\mathit{Product\_Schema} \bowtie_{\alpha:=CategorySet}\ & \mathit{Categorization\_Schema} \\
\bowtie_{\alpha:=Product}\ & \mathit{Availability\_Schema} \\
\bowtie_{\alpha:=Product,\beta:=Product,\delta=ProductSubstitute \bigoplus ConsistsOf}\ & \\
& \mathit{Product\_Association\_Schema} \\
\bowtie_{\zeta:=ContainerType,\eta:=Address}\ & \mathit{Container\_Schema} \\
\bowtie_{\gamma:=Item,\epsilon:=Container \bigoplus Address}\ & \mathit{Inventory\_Schema} \\
\bowtie_{\alpha:=Product}\ & \mathit{Producer\_Schema} \\
\bowtie_{\alpha:=AdditionalCharacteristics}\ & \mathit{Pricing\_Schema}\ .
\end{aligned}$$

4.3 Deriving Other Operations for Component Construction

The approach presented so far is a very general one. We can specialize the rules for component construction in order to handle meta-schema development:

Lifespan recording: Things represented by objects evolve over lifespan. For instance, before ordering a product, the company employees consult various catalogs and product information provided either by the supplier or the departments in their company. Based on the quote, various requests for offers are send to the suppliers. According to the response, employees decide which offer is the most appropriate. Finally, an order is generated. Thus, the order has evolved over lifespan. The order itself is a complex object. It is based on a requisition and involves a party twice for issuing and billing. It is restricted by business rules.

Shuffling meta-characteristics into schemata: Databases often are associated to agreements and adjustments such as policies, laws, regulations. Temporality is another meta-characterization: Transaction time, user-defined time, validity time and availability time are often orthogonal meta-properties. Storage alternatives such as class-wise, strongly identifier-based storage or object-wise storage lead to different sources. Often we need in applications also an explicit representation of the data quality. Data quality is essential whenever we need to distinguish versions of data based on their quality and reliability. This information is necessary especially in the case of data obtained from various sources: Source dimension, intrinsic data quality parameters, accessibility data quality, contextual data quality, and representational data quality.

Recording usage, logs and history: The database structure depicted in Figure 3 demonstrate that objects may integrate their usage scheme. The type *Product* is a complex type. We use products in different occasions in a variety of views, e.g. views for pricing, views for characterization with variants, and views for budgeting of work assets. Thus, we find that the scheme of utilization should be reflected by the database schema.
Log information is used for recording the actions of people in business processes, e.g., the involvement of people during the ordering process. The evolution of data can be recorded in separate manner.

Managing variants and occurrences: Objects may have general characterizations through base types or *potential types* and may have specific actual characterizations for the class of actual objects specified by *actual types*. As discussed above, types may be based on other types, e.g., *Request* is based on *Quote*. The based-on occurrence is often the basis for treatment of complex documents. We distinguish between raw documents, operational documents, blueprints, submissions, and archived documents.

5 Concluding

The approach presented so far is a generalization of view integration approaches which have been intensively discussed in the last two decades and view cooperation approaches developed in [Tha00a]. The approach enables in simpler construction of schemata. Those schemata are not minimal in size. However, they are simpler to maintain, better to survey and thus allow simpler operating of the database system. The approach is not intending to cover all development methods. We are interested in developing a safe and simple pragmatistic approach to modeling of database and information systems and thus gain advantages such as:

Smaller sub-schemata: The entire schema may be cut down to sub-schemata which associations are maintained by specific bridge or hinge types.
Simpler combination of sub-schemata: The schemata can be combined based on either view integration or view cooperation [Tha00a] approaches.
Abstraction layering within the schema: The combination of sub-schemata may be considered at various abstraction layers and therefore in different level of detail.
Simpler integrity maintenance: The enforcement may be concentrated on a certain part of the database.
Control of dimensionality through component construction: Due to the separation between the components the schema can be surveyed, handled simpler and used for generation of views.
Modularity: Internal structuring enhances modularity by encapsulating volatile implementation details behind stable interfaces.
Reusability: The abstract specification provided by sub-schemata enhances reusability by defining generic components that can be re-applied to create new applications.
Extensibility: Internal structuring of schemata enhances extensibility by providing explicit hook methods that allow applications to extend the specification.

The size of the schemata developed with our approach is not the same as the size of schemata developed through monolithic approaches. The schemata developed by our approach are, however, easier to survey and to use. Therefore, the detection and correction of errors, wrong integrity constraints, and wrong types is simpler. The development of queries is easier as well.

The approach is not yet complete without considering specification of functionality and interactivity. We concentrate our research efforts to the development of such generic methods which support functionality and interactivity. Distribution concepts can be added similarly to the other concepts for composing schemata. The general theory of integrity constraints is the most difficult open research question.

References

[ACW93] J. Akoka and I. Comyn-Wattiau, A framework for automatic clustering of semantic models. Proc. ER'93, LNCS 823, Springer, 1994, 438-450. 21

[AnZ98] D. Ancona and E. Zucca, A theory of mixin modules: Basic and derived operators. Mathematical Structures in Computer Science, 1998, 8 (4), 401-446. 21

[BiP00] J. Biskup and T. Polle, Decomposition of database classes under path functional dependencies and onto contraints. Proc. FoIKS'2000, LNCS 1762, Springer, 2000, 31-49. 25

[Bro97] M. Broy, Compositional refinement of interactive systems. Journal of the ACM, 44, 6, 1997, 850-891. 21, 23

[EEK99] H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg (eds.), Handbook of graph grammars and computing by graph transformations. Vol. 2: Applications, languages and tools. World Scientific, Singapore, 1999. 27

[Iso91] ISOTEC, Methoden des Fachkonzepts. Plönzke Informatik GmbH, Wiesbaden 1991. 21

[LST99] J. Lewerenz, K.-D. Schewe, and B. Thalheim, Modeling data warehouses and OLAP applications by means of dialogue objects. Proc. ER'99, Springer, Berlin, LNCS 1728, 1999, 354-368. 24

[Moo01] D. L. Moody, Dealing with complexity: A practical method for representing large entity-relationship models. PhD., Dept. of Information Systems, University of Melbourne, 2001. 20, 21

[NiM95] O. Nierstrasz and T. D. Meijler, Research directions in software composition. ACM Computing Surveys, 27, 2, 1995, 262-264. 21

[RaS97] O. Rauh and E. Stickel, Konzeptuelle Datenmodellierung. Teubner, Stuttgart, 1997. 21

[SIG97] L. Silverston, W. H. Inmon, and K. Graziano, The data model resource book. Jon Wiley & Sons, New York, 1997.

[Tha00a] B. Thalheim, Entity-relationship modeling – Foundations of database technology. Springer, Berlin, 2000.
See also http://www.informatik.tu-cottbus.de/∼thalheim/HERM.htm 21, 22, 23, 27, 28, 32

[Tha00b] B. Thalheim, The person, organization, product, production, ordering, delivery, invoice, accounting, budgeting and human resources pattern in database design. Preprint I-07-2000, Computer Science Institute, Brandenburg University of Technology at Cottbus, 2000. 20, 22, 33

[TWB89] T. J. Teorey, G. Wei, D. L. Bolten, and J. A. Koenig, ER model clustering as an aid for user communication and documentation in database design. CACM, 32, 8, 1989, 975-987. 21

[Wis01] P. Wisse, Metapattern - Context and time in information models. Addison-Wesley, Boston, 2001. 23

Acknowledgement

We are thankful to the reviewers requesting to sharpen the profile of the paper and to concentrate on component construction theory. We acknowledge the efforts of members of DBIS groups and of our cooperation partners engaged in the development and application of the database design workbenchs $(DB)^2$ and RADD, employed in modeling or developing dozens of large database applications or participating in more than 30 projects for development of large information-intensive websites. The experience gained in a large number of database development projects has been summarized and generalized in [Tha00b] and led to the development of component frameworks.

Appendix: The Diagramm of the Product Schema

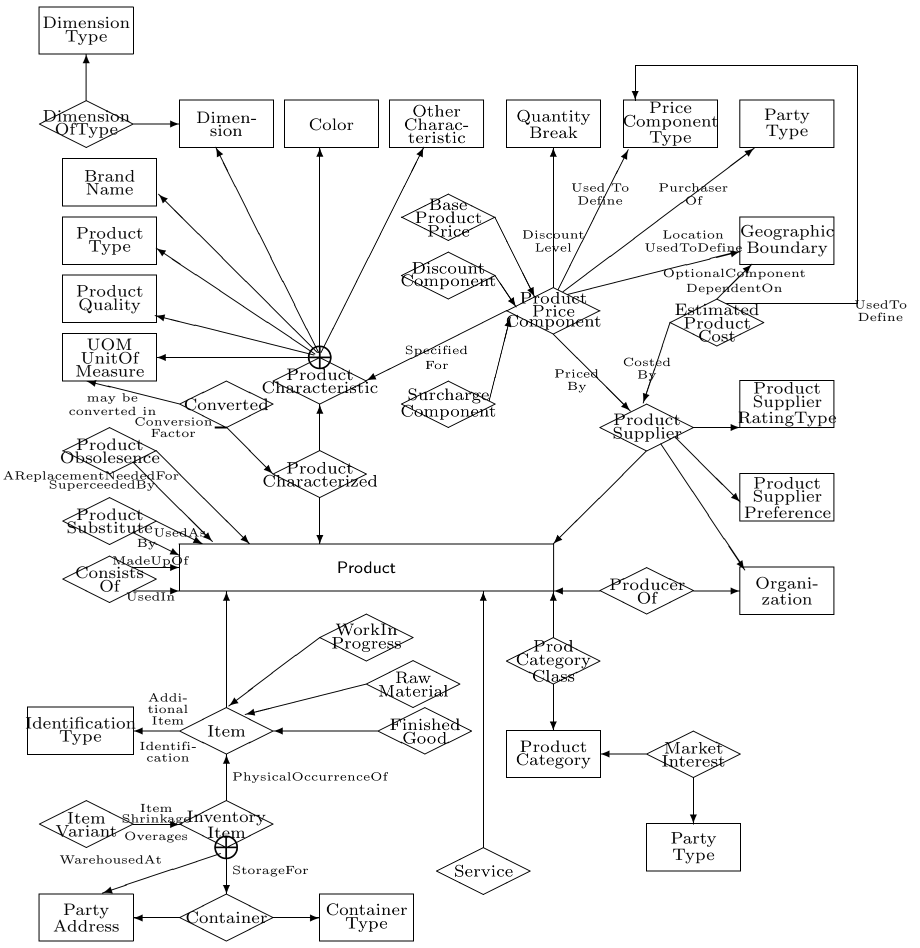


Fig. 3. HERM Schema for Information on Products (without attributes)

Multirelational Semantics for Extended Entity-Relationship Schemata With Applications

Sebastiano Vigna

Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, Via Comelico 39/41, I-20135 Milano MI, Italy. vigna@dsi.unimi.it

Abstract. This paper describes a multirelation semantics for a fragment of the Extended Entity-Relationship schemata formalism based on the bicategorical definition of multirelations. The approach we follow is elementary—we try to introduce as few notions as possible. We claim that bicategorical algebra handles gracefully multirelations and their operations, and that multirelations are essential in a number of applications; moreover, the bicategorical composition of multirelations turns out to correspond to natural joins. From the formal semantics we derive an algorithm that can establish statically the possibility of building parallel ownership paths of weak entities. The ideas described in this paper have been implemented in a free tool, ERW, which lets users edit sets and multirelations instantiating an EER schema *via* a sophisticated web interface.

1 Introduction

Entity-Relationship (ER) schemata are a popular conceptual model. They were originally introduced by Chen [Che76], and later extended in several ways; the more common extensions are usually termed *Extended ER* (EER) [Tha00]. The basic idea is that using sets and relations we can model objects of the real world and their inter-relationships. An EER schema is used to design conceptually a database: then, a *reification* process produces a logical database schema (e.g., in SQL) [EN94].

The starting point of this paper was the creation of a set of specifications and tools implementing the following idea: that conceptual design is sufficient to completely define an application (at least in a default, standardized form).

Since our first requirement was platform independence, we decided that the user interface should be web-based. Only a few years ago, poor standardization of browser document object models, and lack of flexibility, would have made this goal impractical, but this is no longer true.

The result is ERW, a system that, essentially, lets you edit instances of EER schemata. We stress the fact that the user edits such instances, rather than databases, because in our view the underlying relational database is just a support for a clearly defined multirelational semantics for schema instances.

More in detail, one defines an EER schema using ERL, an XML-based language (currently only a fragment of the EER formalism is supported, the most notable omissions being n-ary relations and multiple-value attributes). Then, a Java$^{\mathrm{TM}}$ tool named ERtool reifies the schema into a set of SQL tables using a standard algorithm (ERtool is also able to generate SGML documentation about the reification process). Moreover, it produces a set of definition files that are used by a run-time environment written in PHP, a powerful and very popular server-side scripting language. The run-time environment creates forms that let the user interact with the schema instance, with natural operations such as "associate this entity to this entity", and so on. ERW is free software downloadable from `http://erw.dsi.unimi.it/`.[1]

During the development of ERW, we decided to found the manipulation of entities and relationships on a clear, completely formal semantics. This was made necessary by the complexity of the database management code, which has to handle cardinality constraints, subtypes and ownership. In applications, moreover, we found that *multirelations* are very useful. In a multirelation, two

[1] Note that WebML [CFB00] has a certain intersection with ERW. There are two main differences between ERW and WebML. The first is in focus: ERW does not generate web sites, but just sophisticated user interfaces accessible with a browser. On the other hand, the database support of ERW is much wider than that of WebML.

entities can be related "more than once". For instance, if a library has an entity type for persons and one for books, the "loan" relationship type (with, say, start and end date as attributes) will have to be instantiated by a multirelation with attributes, as a customer can borrow a book several times.

Since defining multirelations in terms of tuples leads to the same problems that plague improperly defined multisets [Mon87, Bli89], we decided to try a different approach, by using a natural generalization of the categorical definition of relation. The result is a very natural semantics, also because the object and arrows in categories parallel exactly the notions of entity type and relationship type in EER schemata.

Another obstacle met during the development of ERW was the "weak status of weak entity types" [BS99]. ERW supports single-owner weak entities only, but detecting double ownership is a nontrivial problem if subtyping gets in the way, as ownership can be inherited (if subtyping is not allowed, detecting double ownership is trivial under the single-owner assumption).

This paper presents the solutions we have found for the problems above. First of all, it extends the semantics of EER schemata by providing a clear, mathematically sound description of multirelations and their operations. Note that this does not require a change in the EER syntax: multirelations are simply an extended *cardinality constraint*, that complements the well-known `(0:1)`, `(1:N)` and so on. The semantics is given in the same spirit of the original paper by Chen [Che76]: entities and relationships are *not* tuples; rather, they have a well-defined semantics on their own; then, attributes are defined as functions mapping entities and relationships into a suitable domain.

Finally, we present a static double-ownership detection algorithm that works for multiple inheritance and multiple owners, and prove it sound and complete using the formal semantics above. The algorithm is currently implemented in ERtool. Lack of space prevent us from giving complete proofs, which are present in the full version of this paper, downloadable from the author's home page.

2 The Categorical View of Binary Relations

A binary relation R from set X to set Y is a subset of the cartesian product $X \times Y$, that is, a set of pairs of the form $\langle x, y\rangle$. This is the well-known set-theoretical notion of relation. *Composition* of relations is easily obtained with the "common middle" operation: if R goes from X to Y and S goes from Y to Z then RS is defined as the set of pairs $\langle x, z\rangle \in X \times Z$ for which there is a $y \in Y$ such that $x \mathrel{R} y$ and $y \mathrel{S} z$.

Category theorists see relations in a different way [SCK84]: a relation is a set R (*not* a set of pairs), endowed with two functions, the *left (first) projection* and the *right (second) projection*, often called *left leg* and *right leg* because of the typical way they are drawn:

$$\begin{tikzcd}
 & R \arrow[dl, "\pi_0^R"'] \arrow[dr, "\pi_1^R"] & \\
X & & Y
\end{tikzcd}$$

Moreover, we require that that two legs are *jointly monic*: by this we simply mean that there are no elements $r, s \in R$ such that $\pi_0^R(r) = \pi_0^R(s)$ and $\pi_1^R(r) = \pi_1^R(s)$.

A reader acquainted with EER schema reification will certainly notice "this looks like an SQL table reifying a relationship type", and indeed the interesting point about the categorical definition of relation is that it is very close to the relational algebra used to formalize relational databases.

Clearly, a categorical relation can be mapped to a relation, just by taking the set

$$\bigl\{\langle \pi_0^R(r), \pi_1^R(r)\rangle \bigm| r \in R\bigr\}.$$

Conversely, any set-theoretic relation can be made into a category-theoretic relation taking the two natural projections as legs.

2.1 Straight To Multirelations

We have not yet seen the real contribution of the categorical viewpoint. As we stated in the introduction, our goal is to give a firm mathematical foundation to a multirelational semantics of schemata.

How should we define multirelations? If we start from the set-theoretical definition, we encounter a number of difficulties. Should we assign a natural number of each pair, denoting its multiplicity, or use some other trick?

In the categorical view, instead, multirelations are *more natural* than relations. The definition of a multirelation is simply obtained by *dropping the joint monicity condition*, that is, via a true generalization process. We state this formally:

Definition 1. *A (binary) multirelation from set X to set Y is a set M endowed with two function, the* left leg M_0 *and the* right leg M_1*:*

$$\begin{tikzcd} & M \arrow[dl, "M_0"'] \arrow[dr, "M_1"] & \\ X & & Y \end{tikzcd}$$

The reader should notice that now it can happen that two elements $r, s \in M$ satisfy $M_0(r) = M_0(s)$ and $M_1(r) = M_1(s)$. In this case, the elements $M_0(r)$ and $M_1(r)$ are related more than once.

A category theorist calls a set with two such functions a *span*; indeed, Definition 1 is simply the categorical definition of a span. The interesting point is that spans form themselves a *bicategory* (a structure slightly weaker than a category), and have a well-defined composition [SCK84]. Since we have now a clear definition of a multirelations, but no hint on their composition, we will look into the standard composition of spans.

2.2 Composition

Let us introduce the last categorical concept we need, the *pullback*. A pullback of two functions with the same codomain, say $f : X \to Z$ and $g : Y \to Z$, is defined as the universal commutative square of functions

$$\begin{tikzcd} P \arrow[r, "\alpha_0"] \arrow[d, "\alpha_1"'] & X \arrow[d, "f"] \\ Y \arrow[r, "g"'] & Z \end{tikzcd}$$

By *commutative*, we mean that paths starting and ending at the same points of the diagram give the same functional composition. In the case above, it means that doing α_0 followed by f is the same as α_1 followed by g, that is, that the equation

$$f \circ \alpha_0 = g \circ \alpha_1 \tag{1}$$

is satisfied. By *universal*, we mean that this diagram is the best possible, in the following sense: given any other commutative diagram with the same shape, say

$$\begin{tikzcd}
Q \arrow[r, "\beta_0"] \arrow[d, "\beta_1"'] & X \arrow[d, "f"] \\
Y \arrow[r, "g"'] & Z
\end{tikzcd} \tag{2}$$

there *exactly one* function $Q \to P$ such that the following diagram commutes:

$$\begin{tikzcd}
Q \arrow[dr] \arrow[drr, "\beta_0"] \arrow[ddr, "\beta_1"'] & & \\
& P \arrow[r, "\alpha_0"] \arrow[d, "\alpha_1"] & X \arrow[d, "f"] \\
& Y \arrow[r, "g"'] & Z
\end{tikzcd} \tag{3}$$

In this case, P, together with the *projections* α_0 and α_1, is called the pullback of f and g. Note that commutativity of (3) is a fairly complicated condition to write as a set of equations, since there are many paths of arrows between two nodes.

All this may seem very abstract, but it's closer to databases than it could seem. Let us try to build a set P satisfying the requirements above: first of all we notice that to each element p of P we can assign an element of X $(\alpha_0(p))$ and an element of Y $(\alpha_1(p))$. Thus, we can assign to each element of P a pair $\langle x, y\rangle$. Of course, not all pairs are possible: indeed, since (1) must be satisfied, it must always happen that $f(x) = g(y)$. Moreover, it is not possible that two elements p and p' of P get the same pair $\langle x, y\rangle$: indeed, consider as set Q a singleton $\{q\}$ and define $\beta_0(q) = x$, $\beta_1(q) = y$. This would define a commutative diagram as in (2), and thus it should happen that there is a *unique* function from Q to P making (2) commutative: but this is not true! There are at least *two* such functions, the one mapping q to p and the one mapping q to p'. Thus no two elements of P can be mapped to the same pair of elements from $X \times Y$.

All in all, we can build a pullback P using the subset of $X \times Y$ that satisfy $f(x) = g(y)$, and using as projections the standard projections. Note that from a categorical viewpoint, this subset of $X \times Y$ is indeed *one* of the many possible ways of building a pullback of f and g. We insist on this point because it will be fundamental in what follows.

It is now time to show why ever we need pullbacks. How do category theorists define the composition of multirelations (spans)? It's easy: consider two multirelations

$$\begin{tikzcd}
 & M \arrow[dl, "M_0"'] \arrow[dr, "M_1"] & & N \arrow[dl, "N_0"'] \arrow[dr, "N_1"] & \\
X & & Y & & Z
\end{tikzcd}$$

Now, note that the "v"-shaped pair of maps looks really like a candidate for building a pullback. This brings us to

$$\begin{tikzcd}
 & & P \arrow[dl, "\alpha_0"'] \arrow[dr, "\alpha_1"] & & \\
 & M \arrow[dl, "M_0"'] \arrow[dr, "M_1"] & & N \arrow[dl, "N_0"'] \arrow[dr, "N_1"] & \\
X & & Y & & Z
\end{tikzcd}$$

By composing the α_i's with the external legs, we finally get our composition:

$$\begin{tikzcd}
 & P \arrow[dl, "M_0 \circ \alpha_0"'] \arrow[dr, "N_1 \circ \alpha_1"] & \\
X & & Z
\end{tikzcd}$$

Which are the elements of P? Using the construction above, we would say that they are the pairs $\langle r, s\rangle \in M \times N$ such that $M_1(r) = N_0(s)$; the legs of the span map these elements to $M_0(r)$ and $N_1(s)$, respectively. But remember, this is just one of the possible construction for the composition of M and N.

This does not tell us much. We can get more intuition about this operation if we take M and N to be relations, represented in the standard set-theoretical way, and their projections. Using again the construction above, we obtain that the elements of P are $\langle\langle x, y\rangle, \langle y', z\rangle\rangle$ such that $\alpha_1(\langle x, y\rangle) = \alpha_0(\langle y', z\rangle)$, that is, $y = y'$; all in all, P would contain the tuples of the form $\langle\langle x, y\rangle, \langle y, z\rangle\rangle$ (with $\langle x, y\rangle \in M$ and $\langle y, z\rangle \in N$).

This really looks like relational algebra: indeed, if M and N were relations we would have just defined the *equijoin* operator of relational algebra. Note that usually the *natural join*, in which duplicated columns are omitted, is taken as a distinct operation, but in our case this is not necessary: the set of triples $\langle x, y, z\rangle$ such that $\langle x, y\rangle \in M$ and $\langle y, z\rangle \in N$ *can as well be taken as the pullback of* M_1 *and* N_0, as the pullback is defined by a property of its projections, not as a specific set; in categorical-theoretic terms, the two definitions are *naturally*[2] *equivalent.* The projections simply take out of the triples the corresponding elements of X and Z, so that we can compose again.

The example of equijoin and natural join points out the important fact that *composition of multirelations is not unique, but defined up to isomorphism.* We shall not pursue the point here, but from the definition of pullback one can show that all possible pullbacks are naturally isomorphic. Equijoins and natural joins are just two possible pullbacks for relations. A consequence is that multirelations form a *bicategory*, as opposed to a *category*: composition is not unique, but defined up to a unique isomorphism.

All in all, spans are a nicely structured and clearly defined model for multirelations. Moreover, their natural composition reduces to the natural join in the case of relations, and thus the relational algebra is easily extended to an algebra of multirelations. We will not pursue here the details of this extension, but we believe that the connection between the bicategorical algebra of relations and relational algebra is very strong. By means of pullbacks, the usual properties of

[2] The original purpose of category theory was exactly to give a precise definition of the term "natural" in mathematics.

joins can be proved also for multirelations, and an EER calculus that defines precisely the semantics of queries can be derived.

3 Schemata and Instances

For the rest of the paper, we shall not be concerned with attributes. Our interest is in giving a clear semantics to EER schema instances using sets and multirelations, and attributes would uselessly clutter our presentation. It is not difficult to extend the bicategorical semantics given in the previous section adding an attribute map for every set involved (including the set defining a multirelation, which gives us also attributes for relationship types). At that point, one can easily discuss keys, attribute constraints and so on. Our interest now is mainly in a formal semantics that is sufficient to express cardinality constraints on multirelation and to validate the ownership structure of a schema.[3]

In our simplified view, an EER schema (of binary relations) $\mathscr{S}$ is given by a set $\mathscr{E}$ of entity types, a set $\mathscr{R}$ of relationship types, a source function $s : \mathscr{R} \to \mathscr{E}$ and a target function $t : \mathscr{R} \to \mathscr{E}$ (note that in order to give a formal semantics to an EER schema without roles you need to take directed relationship types). Moreover, each relationship type has a source and a target *cardinality constraint*, which is a symbol out of `(0:1)`, `(1:1)`, `(0:N)`, `(1:N)`, `(0:M)`, `(1:M)`. The ordered pair of cardinality constraint of a relationship type is usually written as `(-:-)`$\to$`(-:-)`.

Finally, a relationship type may be marked optionally as either `ISA` or `WEAK`. In the first case, its constraint must be `(1:1)`$\to$`(0:1)`; in the second case, it must be `(1:1)`$\to$`(-:N)`.

[3] The same approach is used in the original paper by Chen [Che76], where entity sets and relationship sets are first used to give semantics to the relational structure of a schema, and attributes are defined afterwards by means of suitable mappings from those sets.

Whenever there is an `ISA` relationship type from E to F, E is said to be a *direct subtype* of F, and F a *direct supertype* of E. A *subtype* of E is either E or a direct subtype of a subtype of E (analogously for supertypes).

With respect to typical EER schema definitions, we introduce two new cardinality types: `(0:M)` and `(1:M)`. The informal meaning of these types is that a relationship type with such specifiers can be instantiated by a multirelation; when `(0:N)` and `(1:N)` are used, instead, the relationship type must be instantiated by a relation. In this way we do not disturb the standard syntax of EER schemata to introduce our new semantics.

There is an important syntactic rule that must be respected: if one constraint of a relationship type is given by a `(-:M)` specifier, than the same must happen for the other one. Indeed, while all other constraints combine freely, if you allow a multirelation on the source you must do the same for the target, and viceversa.

3.1 Instances

We are finally in a position to give the definition of the multirelation-based schema semantics.

Definition 2. *An instance σ for a schema $\mathscr{S}$ is given by a map σ assigning to each entity type E in $\mathscr{E}$ a set $\sigma(E)$ and to each relationship type R in $\mathscr{R}$ a span $\sigma(R)$ satisfying the following properties:*

1. *The left leg of $\sigma(R)$ must end in $\sigma(s(E))$, and the right leg in $\sigma(t(E))$; in other words, if R is a relationship type from entity type E to entity type F, then $\sigma(R)$ must be a span*

$$\begin{tikzcd} & \sigma(R) \arrow[dl] \arrow[dr] & \\ \sigma(E) & & \sigma(F) \end{tikzcd}$$

that is, a span with a left leg ending in $\sigma(E)$ and a right leg ending in $\sigma(F)$[4].

[4] The categorically-minded reader will have noticed that this condition represent simply the definition of a *functor* (from a graph to the bicategory of spans of sets).

2. *A cardinality constraint of the form* `(1:-)` *requires that the corresponding leg be a surjective function.*
3. *A cardinality constraint of the form* `(-:1)` *requires that the corresponding leg be an injective function.*
4. *A cardinality constraint of the form* `(-:N)` *on either leg requires that the span be jointly monic (i.e., it must represent a relation).*
5. *Whenever a relationship type is marked* `ISA`*, its bijective leg is the identity, and its injective leg is an inclusion.*
6. *Whenever entity types E and F have a common supertype and $x \in \sigma(E) \cap \sigma(F)$, there is a common subtype G of E and F such that $x \in \sigma(G)$*[5].

Note how cardinality constraints map very naturally on function properties if we look at multirelations as spans. The reader should have no difficulty in showing that the above definitions correspond exactly to the usual constraints when we restrict to relations. For instance, if a relationship type R from entity types E to entity type F has a constraint `(0:1)` on E, then elements of $\sigma(E)$ should have at most one relationship with elements of $\sigma(F)$ (that is, we assume the *participation interpretation* of cardinality constraints). But this is exactly what condition (3) says: if the leg ending on $\sigma(E)$ is injective, it cannot happen that an $x \in \sigma(E)$ is related with two elements of $\sigma(F)$: this would require the existence of two elements $r, s \in \sigma(R)$ mapped to x, which is impossible if the leg ending in $\sigma(E)$ is injective.

The definition above treats subtyping as *set inclusion*: all elements assigned to a certain type *are also* assigned to supertypes. This is essentially the original semantics of subtyping given in [Che76], and also ERW treats subtypes this way. Indeed, the multirelation instantiating an `ISA` relationship type must have the

Indeed, from the definition above one could immediately derive a mathematical definition of *isomorphism* and of *transformation* of instances, using natural transformations [Mac71].

[5] The last two conditions are an elementary phrasing of the fact that the map from the binary-meet completion of the type order to the chosen universe of sets induced by σ must be *stable* [Ber78].

following form:

$$\begin{tikzcd} & X \arrow[dl, "1_X"'] \arrow[dr, "\iota"] & \\ X & & Y \end{tikzcd}$$

where $\iota : X \hookrightarrow Y$ is the inclusion of X as a subset of Y. This actually forces to interpret subtypes as subsets (of course, attributes of an element of $\sigma(X)$ viewed as an element $\sigma(X)$ or as an element of $\sigma(Y)$ will be different).

Condition (6) is essential to give a precise definition of what we mean by an *entity.* It forces, for each entity type E and each $x \in \sigma(E)$, the existence of a *unique* subtype F of E such that $x \in \sigma(F)$ but $x \notin \sigma(G)$ for any proper subtype G of F; such an F is called the *type of* $x \in \sigma(E)$. An *entity* is now a pair E, $x \in \sigma(E)$ such that E is the type of x. Note that we did not restrain entity sets to be disjoint, so an x of type E and an x of type F are actually *distinct entities.*

The definition we gave of cardinality constraints is also valid for multirelations. Since it is very natural, and fits naturally the classical case, we believe it is a step in the right direction. There is indeed another possibility for condition (3): we could allow more than one relationship, but always with the same element. However, this would give rise to a rather clumsy definition, and we see no motivation for such a change.

4 Checking Subtyping and Ownership

ERW supports single-inheritance subtyping and single-owner weak entities. This means than every entity type can be defined as a subtype of another entity type (using an `ISA` relationship type), and can have an owner entity type (defined using a `WEAK` relationship type; in this case the relationship type is instantiated to an *identifying function*).

Since we have a completely clear formal semantics, we shall give a polynomial algorithm that guarantees that it never happens for an entity to own another entity by means of two ownership paths[6]. To be as general as possible, the algorithm will allow multiple inheritance and multiple owners, even if these features are not currently supported by ERW.

4.1 The Algorithm

We start with two definitions from graph theory [Ber85]:

Definition 3. *A path is* simple *if it does not contain the same node twice, unless it is the first and the last node of the path. Two paths of a graph are said to be* parallel *if they start on the same node and end on the same node.*

Definition 4. *A* path *in an instance of a schema $\mathscr{S}$ as a sequence of entities and relationships x_0, r_0, x_1, r_1, ..., x_{n-1}, r_{n-1}, x_n and types E_0, R_0, E_1, R_1, ..., E_{n-1}, R_{n-1}, E_n of $\mathscr{S}$ such that*

1. *x_i is of type E_i for $0 \leq i \leq n$ and $r_i \in \sigma(R_i)$ for $0 \leq i < n$;*
2. *E_i is a subtype of the source of R_i for $0 \leq i < n$;*
3. *E_{i+1} is a subtype of the target of R_i for $0 \leq i < n$;*
4. *the left leg of $\sigma(R_i)$ maps r_i to x_i, and the right leg maps r_i to x_{i+1} for $0 \leq i < n$, that is, each relationship in the sequence relates the entities immediately before and immediately after.*

If all relationship types R_i are marked `WEAK` for $0 \leq i < n$, we say that the path is an ownership path. *A* cycle *is a path satisfying $x_0 = x_n$ and $E_0 = E_n$.*

We believe this formal definition captures exactly the notion of ownership path; we now define a static condition on a schema that is equivalent to the possibility of building two distinct ownership paths between entities. By *distinct* we mean that the list of entities or the list of relationship types of the path are different.

Given an EER schema $\mathscr{S}$, consider the graph having as nodes entity types. Now, for each relationship type R marked `WEAK` going from E to F, for each

[6] Since the empty path is a legal ownership path, by which an entity owns itself, this definition also forbids cycles.

subtype E' of E and each subtype F' of F add an arc from E' to F' coloured by R. Denote this graph with $W(\mathscr{S})$.

Definition 5. *We say that $\mathscr{S}$ has* double ownership *if in the graph $W(\mathscr{S})$ there are two parallel paths α and β such that the list of colours on the two paths is different.*

Theorem 1. *If an EER schema $\mathscr{S}$ has not double ownership, no instance of $\mathscr{S}$ contains two parallel ownership paths.*

Of course, the previous theorem is just half of what we need: it just guarantees that double ownership is strong enough to avoid parallel ownership paths, but it could be too strong. However, it turns out this is not the case:

Theorem 2. *If an EER schema $\mathscr{S}$ has double ownership, there is a subschema $\mathscr{S}'$ of $\mathscr{S}$ and an instance of $\mathscr{S}'$ containing two parallel ownership paths.*

Note that the above theorem does not state the there is an instance *of $\mathscr{S}$* that contains parallel ownership paths. The only reason for this limitation is that, in principle, complex interaction between cardinality constraints could make it impossible putting in an instance the entities which are necessary to build the paths. To all practical purposes, however, this distinction is irrelevant.

4.2 Running Time

In principle, checking Definition 5 requires exploring all possible paths and check their colouring, which would require superexponential time. However, we can do much better:

Theorem 3. *Double ownership for a schema with n entity types, m `WEAK` relationship types and largest type hierarchy of size t can be checked in time $O\bigl(n^3 + tmn + t^2m^2\bigr)$.*

Of course, t is bounded by n and m is bounded by n^2. However, in real schemata they are much smaller, as the underlying graph is usually sparse.

References

[Ber78] G. Berry. Stable models of typed λ-calculi. In G. Ausiello and C. Böhm, editors, *Proceedings of the 5th Colloquium on Automata, Languages and Programming*, volume 62 of *Lecture Notes in Computer Science*, pages 72–89, Udine, Italy, 1978. Springer–Verlag.

[Ber85] Claude Berge. *Graphs.* North–Holland, Amsterdam, 1985.

[Bli89] Wayne D. Blizard. Multiset theory. *Notre Dame J. Formal Logic*, 30(1):36–66, 1989.

[BS99] Mira Balaban and Peretz Shoval. Resolving the "weak status" of weak entity types in entity-relationship schemas. In *Proc. of 18th Int'l Conference on Conceptual Modeling ER'99*, number 1728 in Lecture Notes in Computer Science, pages 369–383, 1999.

[CFB00] Stefano Ceri, Piero Fraternali, and Aldo Bongio. Web Modeling Language (WebML): a modeling language for designing web sites. In *Proc. Ninth World Wide Web Conference*, 2000.

[Che76] Peter Pin-Shan Chen. The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[EN94] Ramez Elmasri and Shami Navathe. *Fundamentals of Database Systems.* Benjamins/Cummings, 1994.

[Mac71] Saunders Mac Lane. *Categories for the Working Mathematician.* Springer–Verlag, 1971.

[Mon87] G. P. Monro. The concept of multiset. *Z. Math. Logik Grundlag. Math.*, 33(2):171–178, 1987.

[SCK84] Ross Howard Street, Aurelio Carboni, and Stefano Kasangian. Bicategories of spans and relations. *J. Pure Appl. Algebra*, 33:259–267, 1984.

[Tha00] B. Thalheim. *Entity-Relationship Modeling.* Springer–Verlag, 2000.

A Meta-model for e-Contract Template Variable Dependencies Facilitating e-Negotiation

S.C. Cheung[1], Patrick C.K. Hung[2], and Dickson K.W. Chiu[3]

[1]Department of Computer Science, Hong Kong
University of Science and Technology, HK
[2] Department of Management Sciences, University of Waterloo, Ontario, Canada
[3]Department of Computer Science and Engineering
Chinese University of Hong Kong, HK
scc@cs.ust.hk
ckphung@engmail.uwaterloo.ca
kwchiu@acm.org

Abstract. Contracts are important for attaining business process interoperability and enforcing their proper enactment. An e-Contract is the computerized facilitation or automation of a contract in a cross-organizational business process. In this paper, motivated by a lease template example, we propose a meta-model for e-Contract templates and template variables. Furthermore, we discover different types of relationships, such as partial order and indivisibility, among template variables. In addition, negotiators can logroll among template variables, focusing on tradeoffs among these instead of arguing about single variables. Based on these relationships, we present an algorithm to determine a negotiation plan of an e-Contract. Furthermore, we propose a model for e-Negotiation of contracts based on contract templates and workflow management technologies. As a result, the process of contract e-Negotiation can be streamlined.

1 Introduction

A contract is a binding agreement between two or more parties, defining the set of obligations and rewards in a business process. Contracts are important for attaining business process interoperability and enforcing their proper enactment, because it reduces uncertainty associated with the interactions between organizations. In USA [27], the federal government spends about USD$200 billion annually buying goods and services from over 300,000 vendors. A typical supermarket chain requires negotiating annually contracts of over 50,000 product items. It is obvious that negotiating contracts play an important role in business processes.

Negotiation is a decision process in which two or more parties make individual decisions and interact with each other for mutual gain [32]. Proposals are sent to the other parties, and a new proposal may be generated after receiving a counter proposal.

The process continues until an agreement or a deadlock is reached, or even one or more parties quit. Each party needs to determine reactions of the other parties and obtain their responses, and each party also needs to estimate the outcomes that the other parties would like to achieve. Whereas each party has its own utility function [29], they tend to be ignorant of the others' values and strategies, especially in a non-cooperative environment. As a result, negotiations may involve high transaction costs and do not always reach the best solution. It is obvious that negotiation of contracts is a critical business activity. Negotiation of contracts involves two or more parties multilaterally bargaining for mutual gain in order to achieve a mutual beneficial agreement, but each has competing interests concerning the specific terms of that agreement. The Internet has recently become a global common platform where organizations and individuals communicate among each other to carry out various commercial activities and to provide value-added services. However, as many business activities become automated as electronic transactions, negotiation of contracts between human can be a bottleneck. A major problem of this is its slowness, which is further complicated by issues of culture, ego and pride [32].

An *e-Contract* is the computerized facilitation or automation of a contract in a cross-organizational business process. As current trends in e-commerce may accelerate the widespread use of e-Contracts in the business world [25], the ability of an e-commerce system to quickly create mutually beneficial e-Contracts will become a critical success factor for organizations. An automated e-Contract negotiation system should conduct negotiation to create value by interacting with different parties to create mutually acceptable deals. This is particularly applicable to standard business interactions that could be taken place over the Internet, such as real-estate transactions, purchase and sale of goods, etc.

New contracts for these business interactions are typically defined based on standard *contract templates*. Specific business interactions not covered by the clauses in standard contract templates can be provided as *contract variations* or *contract escalations*. A contract template is the reference document based on which a new contract is negotiated. A contract template consists of a number of *contract clauses*, each addressing a specific concern in the business interaction. Each contract clause contains a set of *template variables* whose values are to be negotiated. The following gives an example of a contract clause in a lease contract template. The brackets identify template variables in the clause.

Tenant is required to pay landlord [ ] months' deposit, amount to [ ], which is refundable without interests upon contract termination on the condition that ...

Referring to the example above, contracts management involves the evaluation of several variables and of several values per variable. In a result, a large combination of values has to be evaluated by decision makers. The evaluation of a large combination of variables is time consuming and could be difficult for decision makers from a cognitive perspective. Most of the related work in contract negotiations takes the consideration of template variables in isolation. Decision makers are assumed to consider these variables one at a time, in a stepwise fashion, instead of integrating multiple template variables into a single package so that potential tradeoffs can be recognized [10]. We do believe that contracts with similar attributes can be evaluated similarly.

The remainder of this paper is organized as follows: Section 2 describes a motivating example. Section 3 presents a meta-model for e-Contract templates and template variables. Section 4 discusses the e-Contract template variable dependencies. Section 5 presents a model for e-Negotiation of contracts. Next, Section 6 discusses related works. Lastly, Section 7 discusses the conclusion and future work.

2 Motivating Example

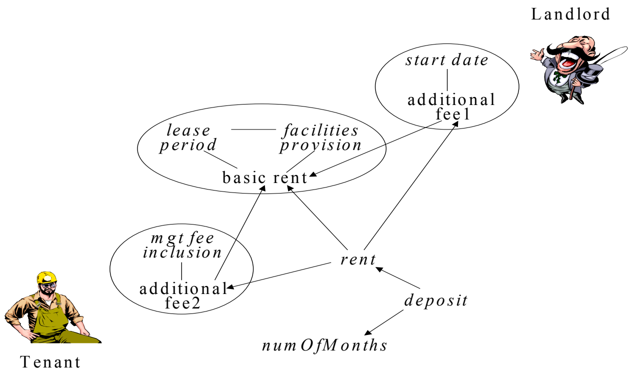


Fig. 1. A Selection of Template Variables in a Lease Contract

Fig. 1 presents a scenario of negotiation over a selection of template variables in a lease contract template. Although the motivating example involves only two parties, the framework we discussed in this paper is applicable to multiple parties. The template variables, which appear in a contract template, are printed in italic. The non-italic ones are referred to as *auxiliary variables*, which are introduced to facilitate negotiation activities, which is an iterative and incremental process. Without the introduction of these auxiliary variables, the landlord and tenant would need to negotiate at the same time the alternatives of all five template variables: start date, lease period, facilities provision, management (mgt) fee inclusion and rent. The complexity of negotiation activities would therefore increase geometrically with the number of alternatives attained by each variable. The concept of auxiliary variables will be re-addressed in more details in Section 4.

Planning is the most important part of negotiation. There is usually one major issue (e.g., price), and several minor issues (e.g., start date) in any negotiation. In negotiation, there are three types of planning [22]: (1) Strategic planning is used to define long-range goals and to position oneself in order to achieve long-rang goals, (2) Tactical planning is the process of developing short-range tactics and plans to achieve long-range goals, and (3) Administrative planning is the process by which both manpower and information are marshaled to make the negotiation proceed smoothly. Therefore, we have to establish a negotiation plan that specifies a procedure for discussing those template and auxiliary variables, which are important, and the order in which they will be discussed.

A negotiation plan describes the order of variables in a contract template that is being negotiated during the contract negotiation process. Most of the human contract negotiation plan focuses on negotiable variables only. Non-negotiable variables refer to those issues or factors that cannot be negotiated. When the tenant would like to rent an apartment and several apartments that are available from several different landlords, the locations of the apartments are not negotiable because they are major decision factors when the tenant makes choices to negotiate with a specific landlord. Further, the tenant should have a budget (i.e., reservation price) for the rent in his mind so that the tenant may be more interested to negotiate in other variables such as the lease period, facilities provision, management fee inclusion, start date and deposit (i.e., number of months). Therefore, the tenant should have a negotiation plan based on his preference order. On the other side, the landlord would also have its own negotiation plan.

Further, we do believe that an automated e-Contract negotiation system should also depict an entire e-Contract template so that negotiators can logroll among template variables, focusing on tradeoffs among these instead of arguing about single variables. *Logrolling* is the exchange of loss in some variables for gain in others resulting in mutual gain [30]. There can be complex interacting tradeoffs between variables in an e-Contract. In this example of a lease contract template, the template variables representing the lease period, the facilities provision and the rent could involve complex tradeoffs in the negotiation of a lease contract. We therefore identify two general requirements for such an automated e-Contract negotiation system. First, it should be able to model the relations among template variables in an e-Contract template. These variables may involve complex tradeoffs and logrolling in e-Contract negotiation. Second, it should provide analysis of these relations and work out a plan to facilitate subsequent negotiations.

In the next section, motivated by a lease contract template example, we propose a meta-model for e-Contracts and e-Contract templates. Furthermore, we discover different types of relations, such as partial order and indivisibility, among template variables. Based on these relations, we present an algorithm to determine a negotiation plan that facilitates the collaborative negotiation process. This research has two salient features, as it is aimed at modeling: (1) e-Negotiation of contracts, and (2) e-Contract template variable relations.

3 A Meta-Model for e-Contract Templates

This section presents our meta-model for an e-Contract template in Unified Modeling Language (UML [26]), which is a modeling language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. Fig. 2 presents a meta-model of an e-Contract template in UML. An e-Contract, which involves at least two parties, sets the variables of an e-Contract template to agreed values. A template consists of a number of contract clauses, each of which concerns some parties to be bound by the e-Contract. Typical contract clauses include *obligation*, *permission* and *prohibition* [15][23]. For example, a tenant is obliged to pay rent to the landlord monthly and a tenant is not allowed to sublease the leased property. A complex contract clause may consist of several simpler clauses. In an e-Contract template, a contract clause may contain a number of template variables, such as the rent and lease period in a lease contract. These variables are to be refined in an e-Contract through negotiations. There are two possible relations among template variables. A template variable *precedes* another variable if the former is to be refined before the latter. As mentioned in the previous section, logrolling is a common scenario in contract negotiation. A template variable *indivisibly relates* to another variable if both of them are to be logrolled altogether. These two relations will be explained in more details in Section 4.

Fig. 3 gives an example lease e-Contract template instantiated from the meta-model in Fig. 2. This template provides a basis on which the negotiations as described in Fig. 1 may take place. The lease e-Contract consists of five contract clauses; each in turn contains one or two template variables. Directed and undirected thick lines represent

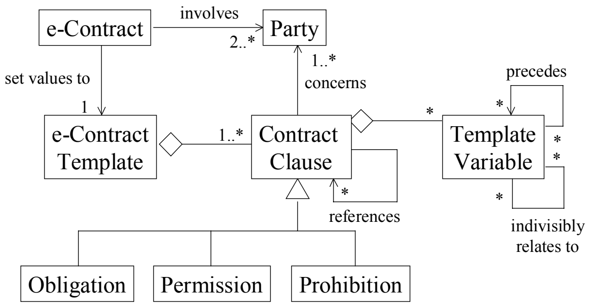


Fig. 2. Meta-Model of an e-Contract Template

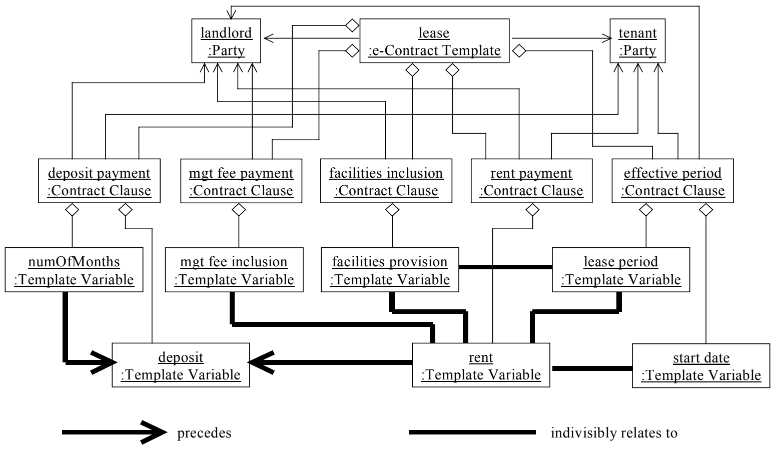

landlord :Party
lease :e-Contract Template
tenant :Party
deposit payment :Contract Clause
mgt fee payment :Contract Clause
facilities inclusion :Contract Clause
rent payment :Contract Clause
effective period :Contract Clause
numOfMonths :Template Variable
mgt fee inclusion :Template Variable
facilities provision :Template Variable
lease period :Template Variable
deposit :Template Variable
rent :Template Variable
start date :Template Variable
precedes
indivisibly relates to

Fig. 3. A Lease e-Contract Template as an Instance of the Meta-model in Fig. 2

the *precedes* and the *indivisibly relates to* relationship in the meta-model, respectively. For example, facilities provision, a lease period and rent are indivisibly related and thus they will be in the same logroll during e-Contract negotiations.

4 Template Variable Relations and e-Negotiation Plan

To facilitate the derivation of negotiation plans, let us examine the conceptual structure of template variables. Given two template variables, they could fall into one of the three following categories.

1. The variables need to be negotiated in a bundle at the same time, such as facilities provision, a lease period and rent.
2. The variables have a partial order, where one should be negotiated before another, such as rent and deposit.
3. The variables are to be individually negotiated, such as the start date and the management fee inclusion.

Most contract templates may contain some principal template variables, which relate to many other variables, such as the rent in a lease contract or the price in a sales contract. Since negotiation is an incremental and iterative process, these principal variables are often decomposed into parts such that each part can be separately negotiated in different rounds. This is to allow a smaller set of variables to be negotiated each round. This leads to a more effective negotiation process [10]. These parts are referred to as *auxiliary variables*. They do not appear at the contract template but are introduced to facilitate contract negotiation. In the motivating example, basic rent, additional fee1 and additional fee2 are auxiliary variables introduced to represent the various factors that add up to the rent. Figure 4 presents an UML model of auxiliary and template variables, where variables generically refer to both auxiliary and template variables. In this paper, we assume the responsibilities of parties to identify auxiliary variables.

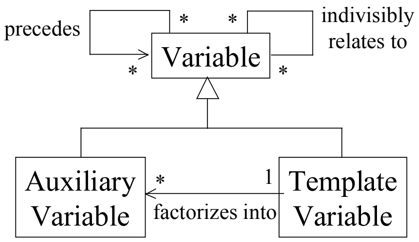


Fig. 4. An UML Model of Auxiliary and Template variables

4.1 Indivisible Components of Variables

Some variables need to be negotiated together at the same time. It occurs when the tradeoffs of these variables cannot be individually or sequentially considered during negotiation. In the motivating example, the variables of facilities provision, lease period and basic rent fall into this category. These variables mutually depend on one another. Let *indivisible component IC* refer to a set of variables so that every variable in an *IC* mutually depends on each other in the set. For example, the set {facilities provision, lease period, basic rent} is an indivisible component. However, total rent and basic rent should not belong to the same indivisible component because they are not to be negotiated at the same time. A singleton indivisible component contains a variable, which is not to be negotiated with other variables at the same time. Note that each variable belongs to one and only one indivisible component. The following is a list of indivisible components in the motivating example.

- {facilities provision, lease period, basic rent}
- {start date, additional fee1}
- {management fee inclusion, additional fee2}

4.2 Partial Ordering of Variables

To model the intention to negotiate some variables before others, we introduce the precede relation. A variable x precedes another variable y, written as $x \prec\prec y$, when the negotiation of x is to precede that of y. Precede relation between variables can be found in the following situations.

- A template variable is factored into several auxiliary variables. Therefore, these auxiliary variables should be negotiated before the template variable, e.g., basic rent $\prec\prec$ rent.
- A template variable depends on another template variable, which requires the latter to be negotiated before the former, e.g., rent $\prec\prec$ deposit.
- Two template variables can be mutually derived from one another but one is more predominant, and the more predominant one is used in the negotiation. For example, rent is more predominant than stamp duty, we have rent $\prec\prec$ stamp duty.

The precede relation is transitive but not reflexive and not commutative. For those situations where the parties have their own precede relations, an overall precede relation is derived from the union of individual relations. If derived precede relation violates the commutative property, measures can be taken to resolve inconsistency. For example, if $x \prec\prec y$ and $y \prec\prec x$ then both relations are deleted from the precede relation, and both x and y become elements of the same indivisible component. This is to facilitate the derivation of an acyclic negotiation plan to be presented in the next subsection.

Two variables can be individually negotiated if they do not belong to the same indivisible component and they are not related by the precede relation. Let $IC(v)$ denotes the indivisible component that contains a variable *v*. It follows that two variables that are related by the precede relation should not belong to the same indivisible component, i.e.,

$$\text{for any variables x and y, x} \prec\!\prec \text{y} \Rightarrow \text{IC(x)} \neq \text{IC(y)}$$

4.3 Derivation of Negotiation Plans

From the precede relation, an *immediate precede relation*, denoted as $\prec$, can be mechanically constructed by removing implied precede relation from $\prec\!\prec$ using the following steps.

1. Identify a set of variables that are not preceded by other variables in $\prec\!\prec$.
2. Assign level 0 to those variables. Let *level*(*x*) be the level assigned to a variable *x*.
3. Use minimum number of levels to assign the level of other variables such that for any variable pair x and y, level(x) < level(y) iff x $\prec\!\prec$ y.
4. Initialize $\prec$ to an empty set. For each $x \prec\!\prec y$, insert $x \prec y$ if $level(x) + 1 = level(y)$.

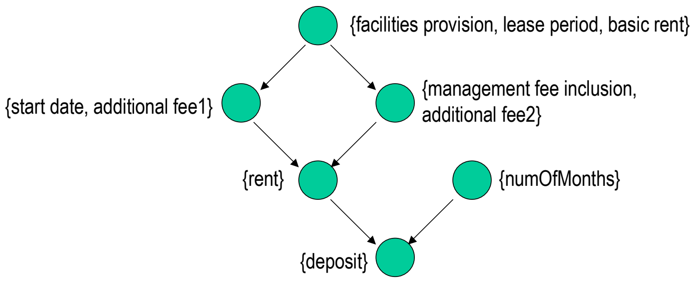

{facilities provision, lease period, basic rent}
{start date, additional fee1}
{management fee inclusion, additional fee2}
{rent}
{numOfMonths}
{deposit}

Fig. 5. Negotiation Plan of the Motivating example

Let $G = (V, E)$ be a directed acyclic graph, where V = the set of all indivisible component, and $E = \{ (u, v) \mid \exists\, x \in u, y \in v \text{ such that } x \prec y \}$. The directed graph presents a negotiation plan so that the negotiation of the variables in a vertex in *G* can only take place after the negotiation of its parents.

Fig. 5 presents the negotiation plan of the motivating example. According to the plan, the variables of facilities provision, lease period and basic rent is to be first negotiated. Alternatively, negotiation may start at the discussion of the number of months, i.e., numOfMonths, required for deposit. However, most users will probably consider the first set of variables more important and therefore prefer to negotiate them first.

5 A Model for e-Negotiation of Contracts

Whereas much research has focused on supporting negotiation activities with various information technologies, the proposed approach in this paper concentrates on the e-Negotiation for a new e-Contract based on an e-Contract template. In this section, we present an e-Negotiation concept model and a process model to support it.

5.1 e-Negotiation Conceptual Model

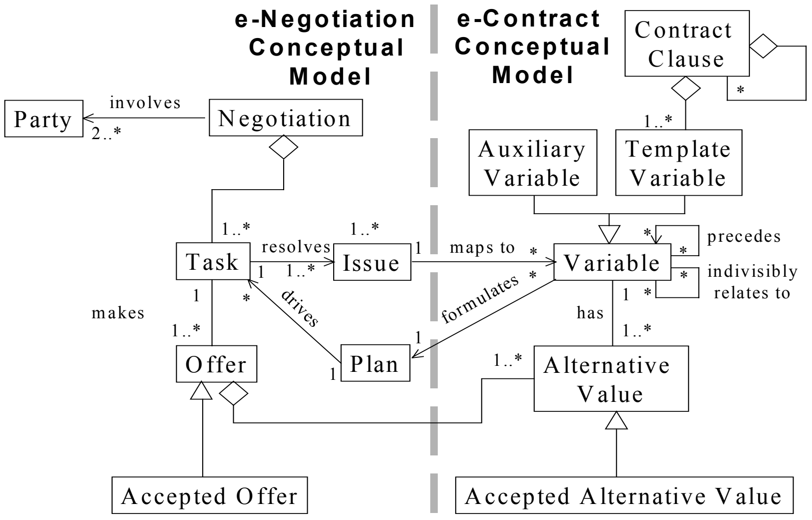


Fig. 6. Conceptual Model of e-Negotiation and e-Contract

Fig. 6 presents the e-Negotiation conceptual model and its relationships to the e-Contract conceptual model. The right hand side is an extraction of the e-Contract conceptual model while the left hand side represents a conceptual model of an e-Negotiation. An e-Negotiation is made of up tasks, each of which aims at resolving an issue or a collection of co-related issues. For instance, issues in the lease example are facilities provision, inclusion of management fee, the effective dates of the lease and the deposit to be paid by the tenant. Each of these issues maps to a set of variables and their relationships. For instance, the issue of facilities provision is mapped to the indivisibility relations among three variables basic rent, facilities provision and lease period. An e-Negotiation plan can be formulated based on the relationships across variables. The plan presents a strategy to drive and organize various tasks in an e-Negotiation. Multiple offers and counter offers are made in a task until a consensus has been reached.

A task in an e-Negotiation represents some work that needs to be executed by a set of parties that can be a negotiator, or even a program such as Negotiation Support Systems (NSS) to resolve a specific issue. Further, there may be a set of criteria to be enforced in the set of tasks. There are three types of criteria [1], which can be classified into three classes: static, dynamic and hybrid. Static criteria can be checked by simply considering the e-Negotiation of contracts specification. For example, the landlord can set criteria to seek potential tenants who don't have pets. Dynamic criteria are those whose conformance cannot be checked without executing the e-Negotiation of contracts. For example, the landlord can set a time limit (e.g., a month) for seeking the potential tenants. Hybrid criteria can be checked during the specification of an e-Negotiation of contracts or the execution of an e-Negotiation of contracts or the execution. If they are found to be inconsistent by performing preliminary consistency verification, they will certainly not be satisfied by the execution.

5.2 e-Negotiation Process Model

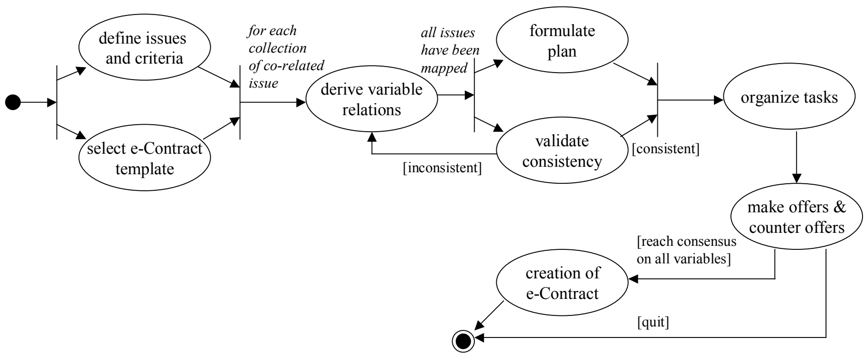

define issues and criteria
select e-Contract template
for each collection of co-related issue
derive variable relations
all issues have been mapped
formulate plan
validate consistency
[inconsistent]
[consistent]
organize tasks
make offers & counter offers
[reach consensus on all variables]
creation of e-Contract
[quit]

Fig. 7. Interactions of e-Negotiation Process in UML Activity Diagram

Fig. 7 depicts our e-Negotiation process in the notation of UML activity diagram. This negotiation process is driven by our meta-model for e-contract templates as described in previous sections. Both parties have to participate each constituting activity of the process. When an e-Negotiation process is started, it triggers two parallel activities. One activity is to identify issues and define the criteria to be adopted in the e-Negotiation process. For instance, the landlord may set criteria on the minimum rent such as $18,000 and on the minimum lease period such as one year. The other activity is to select an appropriate e-Contract template in the domain of the e-business under negotiation.

Then, the set of issues identified is mapped to the template variables and their inter-relationships in the e-contract template. Auxiliary variables are introduced to structure the e-Negotiation process if necessary. The specified relationships are validated against the required properties. For instance, no two variables in the precede relation are related commutatively. From the inter-relationships of the variables, a negotiation plan is derived based on the algorithms discussed in Section 4. In real-life, the formulation of a negotiation plan may involve several iterations before attaining the consensus of all parties. This reflects the inter-relationships among the variables may not often be captured precisely in one-shot. e-Negotiation tasks can be organized based on an agreed plan. Offers and counter offers are made in each task in order to attain a set of agreed values for the variables involved. Finally, the e-Negotiation process either leads to a success creation of an e-Contract if all tasks succeed; otherwise leads to nothing.

Note that in the task "formulate plan", we construct a detailed workflow to realize the activity "make offers and counter offers." Fig. 8 depicts the actual constructed detail activity for our lease negotiation example. The flow reflects the organization of e-Negotiation tasks derived from the plan as shown in Fig. 5.

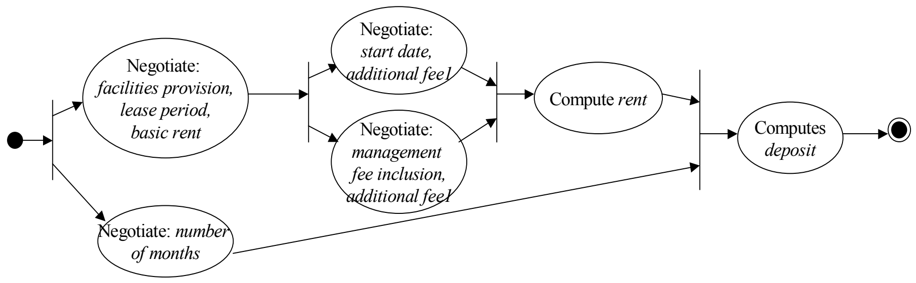

Negotiate: *facilities provision, lease period, basic rent*
Negotiate: *start date, additional fee*
Negotiate: *management fee inclusion, additional fee*
Negotiate: *number of months*
Compute *rent*
Computes *deposit*

Fig 8. The Task *Make Offers and Counter Offers* for Lease Negotiation in UML

5.3 Facilitating the e-Negotiating Process with E-ADOME WFMS

We have extended a flexible, web-enabled workflow management system (WFMS), ADOME-WFMS [5][6], into E-ADOME [4] to provide support for specifying, executing and monitoring composite E-services. In particular, we strengthen the external interface layer to interact with different types of agents over the Internet more effectively. The most recently update is the employment of Web service [7] support to replace a traditional web-server.

The E-ADOME *Agent Internet Interface Layer* allows the ADOME-WFMS to interact with agents through the Internet. It supports effective management of agents; cooperative exception handling, user-driven computer supported resolution of unexpected exceptions, and workflow evolution with the following components. *Internet Message Sender* sends alerts to users and agents via ICQ or E-mail. This module also sends out requests to other software agents using a compatible API. *Internet Event Interceptor* receives responses or alerts from software agents through a compatible API and translates them to ADOME events (which include exceptions). Note that the agents may be internal or external to the organization and may be another WFMS. Furthermore, an *Access Security Layer* is also added to handle external communications.

In this section, we have presented a novel model of e-Negotiation of contracts based on e-contract templates, and a practical e-Negotiation process based on template variable dependencies. Furthermore, facilitated by E-ADOME WFMS, a prototype e-Negotiation system can be built rapidly, supporting the cross-organizational process via contemporary Internet technologies. With such a solid and convenient platform, we are prototyping an e-Negotiation system based on the techniques developed in this paper. In particular, we are using the lease contract template as our primary test application.

6 Related Work

Modeling of e-contracts can be dated back to the Contract Net Protocol [28]. However, they only concentrated on low-level transaction aspects. Gisler [11] presented a framework for legal e-contracts, but not a mechanism for modeling e-contracts. Grosof [14] introduced a declarative approach to business rules in e-commerce contracts by combining Courteous Logic Program and XML. Marjanovic and Milosevic [23] gave a contract model based on contract clauses include obligation, permission and prohibition. Recently, Karlaplem [20] proposed a meta-model for e-contracts with E-R diagrams and generation of workflows to support e-contracts, but do not consider the notion of workflow views or commitments in e-contracts. We have also done some work [4] on e-Contract enactment but not on contract negotiation.

Crossflow [13] models virtual enterprises based on a service provider-consumer paradigm, in which organizations (service consumers) can delegate tasks in their workflows to other organizations (service providers). High-level support is obtained by abstracting services and offering advanced cooperation support. Virtual organizations are dynamically formed by contract-based matchmaking between service providers and consumers. Though Crossflow includes detailed work for contracts, it does not provide such a sophisticated mechanism as workflow views for information and control exchange between workflows of different organizations. Contract enforcement is also not as straightforward as that provided by E-ADOME workflow views equipped ECA-rules mechanisms based on cross-organizational events.

Computer applications were first employed for negotiation support in the 1960s. In the 1980s, computer-based Negotiation Support Systems (NSS) emerged, and they were typically used for training and research in a laboratory environment but rarely used in practice [8]. In general, NSS have the following basic features [19]: (1) a formalism to describe the negotiation activity in terms of choices and outcomes, (2) a way to generally characterize the associated outcome probabilities, and (3) a methodology for processing the model to evaluate the expected value of choice alternatives. NSS normally assist negotiators to assess situations, generate and evaluate options, and implement decisions.

However, most of NSS do not consider the generation of contracts as an outcome of negotiation process, as illustrated in the following examples. NEGOTIATOR [3] seeks to guide negotiators to move their individual goals and judgments to enhance the chance of achieving a common solution. It supports problem adaptation through information sharing, concession making, and problem restructuring or re-framing. However, NEGOTIATOR only helps the negotiators make decisions without any support to other entities involved in negotiation. Further, NEGOTIATOR does not consider contracts as an outcome of negotiation process. INSPIRE (InterNeg Support Program for Intercultural Research) [21] is a web-based prototype for supporting inter-cultural as well as intra-cultural negotiations. It can conduct negotiation anonymously, evaluate the goodness of an offer, and review the history of a negotiation. INSPIRE supports the communication among negotiators by exchanging messages, but it does not directly deal with the interactions among different entities. Further, it does not consider how to handle the outcome of negotiations.

In addition, Griffel [12] presents an application of contract negotiation by mobile agents. A contract is represented as an object that can be accessed by the negotiation partners. Each negotiation partner has the opportunity to change or insert clauses. However, it does not describe any formal logical model to support their approach. It only provides a conceptual view of their approach. Further, a logical formalism [31] is proposed to represent the content of business contracts based on the Formal Language for Business Communication (FLBC). This formalism is proposed to negotiate and process contracts based on the event semantics. However, it does not provide any conceptual and logical model to model the negotiation of contracts. Yu and Mylopoulus [33] considers dependencies of business goals but not down to practical details of variable dependencies of contract negotiation.

7 Conclusion and Future Work

In this paper, motivated by a lease contract template example, we have proposed a pragmatic approach to e-Negotiation based on e-Contract templates. This research has developed a meta-model for e-Contracts and e-Contract templates. Furthermore, we have discovered different types of relations, such as partial order and indivisibility, among template variables. Based on these relations, we have presented an algorithm to determine a negotiation plan that facilitates collaborative negotiation processes. From the negotiation plan, we developed a practical model of e-Negotiation of contracts based on workflow technology. We are currently facilitating this e-Negotiating process with E-ADOME WFMS. As such, the process of contract e-Negotiation can be streamlined.

This work can be expanded in several directions. We are investigating the one-to-many (more than two parties at one time) negotiation of contract. We are also interested in the role of negotiation mediators (especially real-estate brokers and agents), and how our current solution can help them, because this is a large business in Hong Kong. For applications, we are also interested in sales contracts of real estates, supply chain and investment. In addition, we are investigating the ranking of different types of issues and criteria [2] for logrolling issues.

References

1. Bertino, E., Ferrari E., Atluri, V.: The Specification and Enforcement of Authorization Constraints in Workflow Management Systems. ACM Transactions on Information and System Security, 2 (1) 209-240 (1999)
2. Brams, S., Kilgour, M.: Fallback Bargaining. Group Decision and Negotiation 10 287-316 (2001)

3. Bui, T., Shakun, M.: Negotiation Processes, Evolutionary Systems Design, and Negotiator. Group Decision and Negotiation 5 339-353 (1996)
4. Chiu, D.K.W., Karlapalem, K., Li, Q., Kafeza, E.: Workflow View Based E-Contracts in a Cross-Organizational E-service Environment. Distributed and Parallel Database (2002 to appear)
5. Chiu, D.K.W., Li, Q., Karlapalem, K.: A Meta Modeling Approach for Workflow Management System Supporting Exception Handling. Information Systems 24 (2) 159-184 (1999)
6. Chiu, D.K.W., Li, Q., Karlapalem, K.: Web Interface-Driven Cooperative Exception Handling in ADOME Workflow Management System. Information Systems 26 (2) 93-120 (2001)
7. Chopra, V., et. al.: Professional XML Web Services. Wrox Press (2001)
8. Delaney, M., Foroughi, A., Perkins, W.: An empirical study of the efficacy of a computerized negotiation support system (NSS). Decision Support Systems 20 (3) 185-197 (1997)
9. Erikson, H.-E, Penker., M.: Business Modeling with UML: Business Patterns at Work. John Wiley. New York (2000)
10. Foroughi, A., Jelassi, M.: NSS Solutions to Major Negotiation Stumbling Blocks. In: Proceedings of the 23th Hawaii International Conference on System Sciences, vol. 4 2-11 (1990)
11. Gisler, M., Stanoevska-Slabeva, K., Greunz, M.: Legal Aspects of Electronic Contracts. In: CAiSE*00 Workshop of Infrastructures for Dynamic Business-to-Business Service Outsourcing (IDSO'00), Stockholm (2000)
12. Griffel, F., Tu, M., Munke, M., Merz, M., Lamersdorf W., da Silva., M.: Electronic contract negotiation as an application niche for mobile agents. In: Proceedings of First International Enterprise Distributed Object Computing Workshop 354 –365 (1997)
13. Grefen, P., Aberer, K., Hoffner, Y., Ludwig., H.: CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises. International Journal of Computer Systems Science & Engineering, 15 (5) 277-290 (2000)
14. Grosof, B.: A declarative approach to business rules in Contracts: Courteous Logic Programs in XML. In: Proceedings of the 1st ACM Conference on Electronic Commerce (EC99), Denver, Colorado, USA, Nov. 3-5 (1999)
15. Herring, C., Milosevic, Z.: Implementing B2B contracts using BizTalk. In: Proceedings of the 34th Hawaii International Conference on System Sciences 4078 – 4087 (2001)
16. Hung, P. C. K.: Modeling e-Negotiation Activities. Master Thesis, Department of Management Sciences, University of Waterloo, Canada (2002)
17. Hung, P. C. K., Mao, Ji-Ye: Modeling e-Negotiation Activities with Petri Nets. In: Proceedings of the 35th Hawaii International Conference on System Sciences (CD-ROM) (2002)
18. Hwang, C., Yoon, K.: Multiple Attribute Decision Making: Methods and Applications. Springer-Verlag Berlin Heidelberg, New York (1981)
19. InterNeg: For and about Negotiations. http://interneg.carleton.ca (2000)

20. Karlaplem, K., Dani, A., Krishna., P.: A Frame Work for Modeling Electronic Contracts. In: Proceedings 20th International Conference on Conceptual Modeling (ER2001) 193-207 (2001)
21. Kersten, G., Noronha, S.: WWW-based negotiation support: design, implementation, and use. Decision Support Systems 25 (2) 135-154 (1999)
22. Lewicki, R., Litterer, J.: Negotiation. Irwin (1985)
23. Marjanovic, O., Milosevic, Z.: Towards formal modeling of e-Contracts. In: Proceedings of 5th IEEE International Enterprise Distributed Object Computing Conference 59 –68 (2001)
24. Meister, D.: A Prescriptive Analysis Negotiator Support System. Ph.D. Thesis, Department of Management Sciences, University of Waterloo (1993)
25. Nabil, A., Yesha, Y.: Electronic Commerce: Current Research Issues and Applications. Springer-Verlag Berlin Heidelberg (1996)
26. OMG, Object Management Group: Foreword UML specification 1.4. September (2001)
27. Robinson, W.: Electronic brokering for assisted contracting of software applets. In: Proceedings of the 13th Hawaii International Conference on System Sciences, 4 449 –458 (1997)
28. Smith, R.: The contract net protocol: High Level Communication and Control in a Distributed Problem Solver. IEEE Transactions on Computers 29(12) 1104-1113 (1980)
29. Stigum, B. P. and Wenstop, F.: Foundations of Utility and Risk Theory with Applications, D. Reidel Publishing Company (1983)
30. Tajima, M., Fraser, N.: Modelling Multi-Issues Negotiation. Technical Paper, Department of Management Sciences, University of Waterloo (2000)
31. Tan, Y.-H., Thoen, W.: Using Event Semantics for Modeling Contracts. In: Proceedings of the 35th Hawaii International Conference on System Sciences (CD-ROM) (2002)
32. Thompson, L.: The Mind and Heart of the Negotiator. Prentice-Hall (1998)
33. Yu, E., Mylopoulus, J.: Using Goals, Rules, and Methods to Support Reasoning in Business Process Reengineering. International Journal of Intelligent Systems in Accounting, Finance and Management, John Wiley & Sons, 5 (1) 1-13 (1996)

On the General Ontological Foundations of Conceptual Modeling

Giancarlo Guizzardi[1], Heinrich Herre[2,3], and Gerd Wagner[4]

[1] Centre for Telematics and Information Technology, University of Twente, Enschede, The Netherlands
guizzard@cs.utwente.nl
[2] Institut für Informatik, University of Leipzig, Germany
herre@informatik.uni-leipzig.de,
[3] Institute for Formal Ontology and Medical Information Science (IFOMIS), University of Leipzig, Germany
[4] Faculty of Technology Management, Eindhoven University of Technology, The Netherlands
G.Wagner@tm.tue.nl,
http://tmitwww.tm.tue.nl/staff/gwagner

Abstract. As pointed out in the pioneering work of [WSW99,EW01], an upper level ontology allows to evaluate the ontological correctness of a conceptual model and to develop guidelines how the constructs of a conceptual modeling language should be used. In this paper we adopt the General Ontological Language (GOL), proposed in [DHHS01], for this purpose. We discuss a number of issues that arise when applying the concepts of GOL to UML class diagrams as a conceptual modeling language. We also compare our ontological analysis of some parts of the UML with the one proposed in [EW01].

1 Introduction

Conceptual modeling is concerned with identifying, analyzing and describing the essential concepts and constraints of a domain with the help of a (diagrammatic) modeling language that is based on a small set of basic meta-concepts (forming a *metamodel*). Ontological modeling, on the other hand, is concerned with capturing the relevant entities of a domain in an ontology of that domain using an ontology specification language that is based on a small set of basic, domain-independent ontological categories (forming an *upper-level ontology*). While conceptual modeling languages (such as *Entity-Relationship diagrams* or *UML class diagrams*)are evaluated on the basis of their successful use in (the early phases of) information systems development, ontology specification languages and their underlying upper level ontologies have to be rooted in principled philosophical theories about what kinds of things exist and what are their basic relationships with each other.

We adopt the position that conceptual modeling languages should be founded on an upper-level ontology referring to reality in a philosophically justified way.

Frequently, the goal of conceptual modelers is not to capture the real structure of some domain but merely to capture some conceptualization of it. It is, however, well-known that not all conceptualizations of a domain are equally suitable. The choice of an adequate upper-level ontology is crucial for achieving an adequate conceptualization. We adopt the General Ontological Language (GOL), proposed in [DHHS01], for this purpose. The upper level ontology of GOL is under development at the Institute for Formal Ontology and Medical Information Science at the University of Leipzig, Germany (http://www.ifomis.uni-leipzig.de). The project is a collaboration between philosophers, linguists and other cognitive scientists and computer and information scientists.

In section 2, we present an introduction to those parts of the GOL ontology that form the basis of conceptual modeling, summarizing [DHHS01] and adapting it for this purpose. In section 3, we briefly discuss a number of UML's class modeling concepts: object class, datatype, powertype, abstract class, and association from an ontological point of view. In section 4, we discuss the Bunge-Wand-Weber (BWW) Ontology, an important line of research that is closely related to ours. Finally, in section 5, we briefly mention some other important work.

2 Basic Elements of the Upper Level Ontology of GOL

The basic elements of the upper level ontology of GOL[5] can be visually described by means of the UML class diagram shown in Figure 1.

2.1 Urelements and Sets

One of the basic distinctions of GOL is the distinction between *urelements* and *sets*. We assume the existence of both urelements and sets in the world and presuppose that both the impure sets and the pure sets constructed over the urelements belong to the world. This implies, in particular, that the world is closed under all set-theoretical constructions.

Urelements are entities which are not sets. They form an ultimative layer of entities without any set-theoretical structure in their build-up. Neither the membership relation nor the subset relation can unfold the internal structure of urelements. In GOL, urelements are classified into two main categories: *individuals* and *universals*. There is no urelement being both an individual and a universal. This is expressed in GOL by the following axioms:

$$(U1)\ \forall x(Ur(x) \leftrightarrow Ind(x) \vee Univ(x))$$
$$(U2)\ \neg\exists x(Ind(x) \wedge Univ(x))$$

2.2 Individuals

Individuals may be *substances*, *moments*, *processes*, *chronoids*, or *topoids*. The pre-cognitive part of the GOL ontology, that is being developed at the Institute for Formal Ontology and Medical Information Science at the University of

[5] Also called *General Formal Ontology (GFO)* in [DHH02].

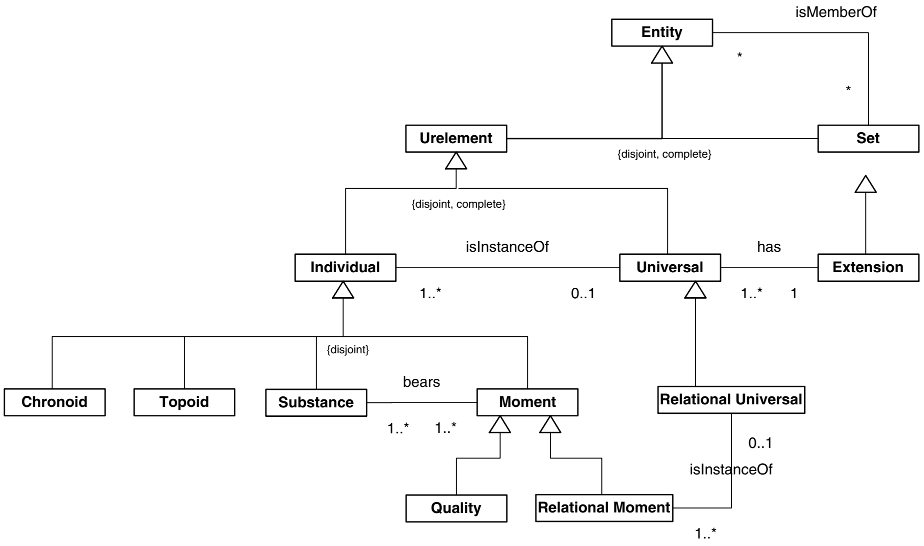


Fig. 1. A UML class diagram describing the basic concepts of GOL.

Leipzig, is called *Basic Formal Ontology (BFO)*. In BFO, a distinction is made between 3D-individuals and 4D-individuals. 3D-individuals are wholly present at every time at which they exist; they have no temporal parts. 4D-individuals are extended in space as well as in time: they have temporal parts. Examples of 3D-individuals are substances, qualities, forms, roles, and functions. 4D-individuals include processes, aggregates of processes, and temporal boundaries of processes, which are also called *Ingarden's events* [Ing64].

Substances and Moments A substance is that which can exist by itself; this implies that a substance is *existentially independent* from other individuals. Existential independence was introduced by E. Husserl: An individual A is existentially independent from an individual B if and only if it is logically possible for A to exist even if B does not exists.

Typical examples of substances are: an individual person, a house, the moon, a car. Every substance is founded on matter. Substances come into existence because the matter is formed in various ways which give rise to pieces separated off in more or less stable ways from their surroundings and possessing qualities of different sorts.

A *moment* is an individual which can only exist in other individuals (in the way in which, for example, an electrical charge can exist only in some conductor). Typical examples of moments are: a color, a connection, a purchase order. Moments have in common that they are all dependent on substances. Some moments are one-place *qualities*, for example color or temperature. But there are also *relational moments* – for example flight connections or purchase orders – which depend on several substances.

The inherence relation i glues moments to the substances which are their bearers. For example it glues your hair color to your hair, or the charge in a specific conductor to the conductor itself. Substances must bear moments, and moments must inhere in substances. This is axiomatically expressed as follows:

$$(SM1)\ Subst(x) \rightarrow \exists y(Mom(y) \wedge i(y,x))$$
$$(SM2)\ \forall x(Mom(x) \rightarrow \exists y(Subst(y) \wedge i(x,y))$$

GOL adopts the *non-migration principle*: it is not possible that a quality q inheres in two different substances a and b:

$$i(q,a) \wedge i(q,b) \Rightarrow a = b$$

This implies that the inherence relation i is functional.

Chronoids and Topoids Chronoids and topoids are instances of the universals *Time* and *Space*, respectively. Chronoids can be understood as temporal durations, and topoids as spatial regions having a certain mereotopological structure.

Every substance x has a certain maximal temporal extent, a chronoid which we denote by *lifetime(x)*. The substance x exists during *lifetime(x)*. Also, every moment m inhering in x has a lifetime, which is such that *lifetime(m)* $\leq$ *lifetime*(x). Moreover, if n is a relational moment connecting substances $x_1, \ldots, x_k$, then *lifetime(n)* $\leq$ *lifetime*(x_i), $i \leq k$.

2.3 Universals

A universal is an entity that can be instantiated by a number of different individuals which are similar in some respect. Following Aristotle, we assume that the universals exist in the individuals (*in re*) but not independently from them. Therefore any universal, in order to exist, must possess instances, implying that the poetic concept *Unicorn* does not correspond to a universal.

For every universal U there is a set $Ext(U)$, called its *extension*, containing all instances of U as elements. It is, however, not the case that every set is the extension of a universal (there is no such axiom in GOL).

There are two kinds of universals that are of particular interest: ***quality universals***, such as *Color* and *Weight*, and ***relational universals***, such as *Flight-Connection* ('...is connected with ...') or *Purchase* ('...purchases ...from ...').

Every universal has an *intension* which, in GOL, is captured by means of an axiomatic specification, i.e., a set of axioms that may involve a number of other universals representing its essential features. A particular form of such a specification of a universal U, called *elementary specification*, involves a number of universals $U_1, \ldots, U_n$ and corresponding functional relations $R_1, \ldots, R_n$ which attach instances from the U_i to instances of U, expressed by the following axiom:

$$\forall a(a :: U \rightarrow \exists e_1 \ldots \exists e_n \bigwedge_{i \leq n} (e_i :: U_i \wedge R_i(a, e_i)))$$

The universals $U_1, \ldots, U_n$ used in an elementary specification are called *features.* A special case of an elementary specification is a *quality specification* where $U_1, \ldots, U_n$ are quality universals, the attachment relations R_k correspond to the inherence relation i, and the instances of U are substances.

Humans, as cognitive subjects, grasp universals by means of concepts that are in their head and cannot capture the universals completely, but only as approximate views. We emphasize that universals whose instances are pre-cognitive individuals belong to the real world and are themselves independent from cognition. Concepts, on the other hand, are cognitive entities that refer to universals.

Meta-Universals of Finite Order Ordinary universals are universals of first order and the instances of universals of (n+1)-th order are univerals of n-th order. Instantiation relations of n-th order are denoted by $::_n$, and the relation $::_1$ is also notated as $::$. Since no universal is a set, it follows that all universals (of whatever order) are urelements.

2.4 Relations and Relational Universals

Relations are entities which glue together other entities. Without relations the world would fall into many isolated pieces. Every relation has a number of *relata* or *arguments* which are connected or related by it. The number of a relation's arguments is called its arity. Relations can be classified according to the types of their relata. There are relations between sets, between individuals, and between universals, but there are also *cross-categorical* relations for example between urelements and sets or between sets and universals.

We divide relations into two broad categories, called *material* and *formal*, respectively. The relata of a material relation are mediated by individuals which are called *relators.* Relators are individuals with the power of connecting entities; a *FlightConnection*, for example, is a relator that connects airports.

A formal relation is a relation which holds between two or more entities directly – without any further intervening individual. Examples of formal relations are: 5 *isGreaterThan* 3, this day *isPartOf* this month, N *isSubsetOf* Q.

Holding Relation and Facts One important formal relation is called the *holding relation.* If r is a relator connecting the entities $a_1, \ldots, a_n$, $n \geq 1$, then we say that $r, a_1, \ldots, a_n$ (in this order) stand to each other in the holding relation, symbolically $h(r, a_1, \ldots, a_n)$.

The fact that h holds directly suffices to block the obvious regress which would arise if a new material relation were needed to tie h to $r, a_1, \ldots, a_n$, and so on. Holding holds directly.

If r connects (holds of) the entities $a_1, \ldots, a_n$, then this yields a new individual which is denoted by $\langle r : a_1, \ldots, a_n \rangle$. Individuals of this latter sort are called *material facts.*

An example of a binary relator is the flight connection c_3427 between Berlin and Paris. c_3427, Berlin, Paris stand in the holding relation, symbolically expressed by $h(c_3427, \mathrm{Berlin}, \mathrm{Paris})$, or by the fact $\langle c_3427 : \mathrm{Berlin}, \mathrm{Paris} \rangle$. An example of a unary relator is an individual quality q which inheres in a substance s. We can express this by $i(q, s)$, or by $h(q, s)$, or by the fact $\langle q : s \rangle$.

A material fact $\langle r : a_1, \ldots, a_n \rangle$ has a duration, which depends on the lifetime of the relator r. We write $\langle r : a_1, \ldots, a_n; t \rangle$ if t is a chronoid which is a part of the lifetime of r, i.e. this fact exists at least during the chronoid t.

Relators of Finite Order Relators can be classified with respect to their order. A relator is said to be of first order if it connects substances exclusively. Examples of first-order relators are those relational moments – for example flight connections or purchases – whose arguments are substances. A relator is of $(n+1)$th order if the highest order of relators it conncects is equal to n.

For example, if John makes a reservation for a rental car, then there is an individual reservation r relating John and the car rental company. Clearly, r is a first order relator. There is another relator, say *assignment*, connecting a specific car to the reservation r. Then, this assignment relator is of second order.

Relator Universals A *relator universal* is a universal whose instances are relators. For every relator universal R there exists a set of facts, denoted by *facts(R)*, which is defined by the instances of R and their corresponding arguments. We assume the axiom that for every relator universal R there exists a *factual universal* $F = F(R)$ whose extension equals the set *facts(R)*. Take, for example, the relator universal *Conn* whose instances are individual flight connections. Then we may form a factual universal $F(Conn)$ having the meaning *'An airport X is connected to an airport Y'* whose instances are all facts of the form $\langle c : a, b \rangle$, where c is an individual connection and a, b are individual airports.

Formal Relations A formal relation is a relation which holds between two or more entities directly – without any further intervening individual. A formal relation may be either an extensional relation (i.e. a set) or it may be given by a relation universal (having an intension and an extension). If R is a formal relation and $[a, b] : R$ then $\langle R : a, b \rangle$ is called a formal fact.

Extensional Relations Extensional relations are sets (or set-theoretical classes) of lists. Obviously, every extensional relation is formal. We assume the axiom that for every relation universal R there is a set-theoretical class $Ext(R)$ being the extension of R. An extensional relation can be the extension of many different relation universals.

2.5 Basic Ontological Relations

We can distinguish a number of basic ontological relations which form an important part of the upper level ontology of GOL. The first and most familiar one is set-theoretic membership, denoted by $\in$. Further basic relations include:

- the proper and reflexive part-of relations, denoted by $<$ and $\leq$
- the contextual part-of relation, denoted by $<_U$, where the universal U denotes the context
- the holding relation h
- the inherence relation, denoted by i
- the instantiation relation, denoted by $::$

We discuss some of these basic ontological relations in more detail.

Part-Whole Relation There are many different part-whole relations between individuals. They can be classified by means of the axioms they satisfy. All part-whole relations are asymmetric and transitive. In addition to formal part-whole relations, there are also material part-whole relations.

Part-whole relations may be either proper (denoted by $<$) or reflexive (denoted by $\leq$). We use the following definitions:

overlap $ov(x, y) =_{df} \exists z(z \leq x \wedge z \leq y)$
reflexive part-whole $x \leq y =_{df} x = y \vee x < y$

A proper part-whole relation $<$ is a strict partial order, that is, it satisfies the following axioms:

irreflexivity $\neg x < x$
asymmetry $x < y \rightarrow \neg y < x$
transitivity $x < y \wedge y < z \rightarrow x < z$

In addition, it may satisfy some of the following axioms:

weak supplementation $x < y \rightarrow \exists z(z \leq y \wedge \neg ov(z, x))$
supplementation $\neg x \leq y \rightarrow \exists z(z \leq x \wedge \neg ov(z, y))$
exclusivity $(z < x \wedge z < y) \rightarrow (x \leq y \vee y \leq x)$

Contextual Part-Whole Relation The contextual part-whole relation $x <_U y$ has the meaning: "U is a universal and x is a part of y in the context of U". Briefly, if x is a U-part of y in this sense, then x and y are parts of instances of U and $x \leq y$. But more is involved, since again the notions of *granularity* and *point of view* are an issue. We propose the following axiom: for every universal U there are universals $U_1, \ldots, U_n$ such that $x <_U y$ implies that x, y are instances of one of the U_i and every instance of one of the U_i is part of an instance of U.

Consider the following example, taken from the domain of biology. Let T be the biological universal whose instances are those organisms called trees. Then

$x <_T y$ describes the part-whole relation based on the granularity of the context of whole trees. A biologist is interested in describing the structure of trees only in terms of parts of a certain minimal size. She is not interested in atoms or molecules. There is a finite number of universals $\{U_1, \ldots, U_n\}$ by which the biologically relevant parts of trees are demarcated. All such parts of trees are either instances of some U_i, $1 \leq i \leq n$, or they can be decomposed into a finite number of parts, each of which satisfies this condition. Examples of universals U_i within the granularity of the tree context would be *branch of a tree*, *leaf of a tree*, *trunk of a tree*, *root of a tree*, and so on.

We have the following axioms:

$$(CPW1)\ \forall xyU(x <_U y \rightarrow Univ(U) \wedge x < y)$$
$$(CPW2)\ \forall xyzU(x <_U y \wedge y <_U z \rightarrow < x <_U z)$$

3 Ontological Foundations of the UML

In the sequel, we refer to the *OMG UML Specification 1.4*, when we cite text in italics using page references in the form of [p.2-31].

For simplicity, we simply say *conceptual model* when we mean a conceptual model of a domain in the form of a *UML class model.* Whenever the context is clear, we omit the name space prefix and simply say 'universal', 'class', etc., instead of 'GOL:universal', 'UML:class', etc.

3.1 Classes and Objects

In the UML, "an object represents a particular instance of a class. It has identity and attribute values." While in the UML *objects* are instances of *classes*, *individuals* are instances of *universals* in GOL. The relationship between UML:classes and GOL:universals may be described by the following observation.

Observation 1 *A class may represent a universal by representing its quality specification (that captures its* intension*) in the form of a list of attributes, such that each quality universal is represented by a distinct attribute (a function that assigns a value from a value domain to an instance of the class). Since a function is a set, this representation of a quality universal by an attribute is a reduction (or approximation). While, according to the* non-migration priciple*, it is not possible that the same quality inheres in two different substances, it is quite common that two different instances of a class have the same attribute value*

A *"Class describes a set of Objects sharing a collection of Features, including Operations, Attributes and Methods, that are common to the set of Objects."* [p.2-26] *"The model is concerned with describing the intension of the class, that is, the rules that define it. The run-time execution provides its extension, that is, its instances."* [p.3-35] Attributes come with associated data types. Since in conceptual modeling, the behavior of objects is normally not taken into consideration, we exclude the 'operations' and 'methods' of an object from our discussion.

We may observe a direct correspondence between universals and classes of a certain kind, as stated in the following principles.

Principle 1 (Class) *In a conceptual model, any first-order universal U of the domain may be represented as a concrete class C_U. Conversely, for all concrete classes (of a conceptual model of the domain) whose instances are basic objects or links (representing individuals), there must be a corresponding first-order universal in the domain.*

Principle 2 (Attribute) *Any feature of a universal U that is captured by a quality universal in a quality specification for U may be represented by an attribute of the corresponding class C_U in a conceptual model of the domain.*

In a conceptual model, any individual of the domain that is an instance of a universal may be represented as an object (or link) of the class representing the universal. Notice that classes are not sets: while the latter are defined only in terms of their *extension*, the former are defined both in terms of their extension and in terms of their *intension*. In general, two classes C_1 and C_2 with identical extensions, $Ext(C_1) = Ext(C_2)$, even if they have the same set of attributes, are not equal, $C_1\ /\ \not{C}_2$.

3.2 DataType

"A data type is a type whose values have no identity; that is, they are pure values. Data types include primitive built-in types (such as integer and string) as well as definable enumeration types (such as the predefined enumeration type Boolean whose literals are false and true)." [p.2-33]

Data types are sets. This means that two data types denoting the same set of possible values are equal, by the extensionality axiom of set theory. For instance, a C++ compiler may admit the data types *ULONG* and *DWORD*. Since both types denote the set of positive 32-bit-expressible integers, they are identical: *ULONG* = *DWORD*.

3.3 Powertype

"A Powertype is a user-defined metaelement whose instances are classes in the model." A powertype is a special class, designated with the stereotype 'powertype'. It represents a higher-order universal of order n whose instances are universals of order $n-1$. Unfortunately, the UML does not provide higher-order 'isInstanceOf' relationships.

3.4 Abstract Class

"Abstract constructs are not instantiable and are commonly used to reify key constructs, share structure, and organize the UML metamodel. Concrete metamodel constructs are instantiable and reflect the modeling constructs used by

object modelers (cf. metamodelers). Abstract constructs defined in the Core include ModelElement, GeneralizableElement, and Classifier. Concrete constructs specified in the Core include Class, Attribute, Operation, and Association." [p.2-12]

What is the status of abstract classes in an ontologically well-founded conceptual model? It seems that an abstract class does not denote a universal but rather a conceptual construction in the form of a hierarchy whose bottom elements denote universals.

3.5 Association

In the UML, the ER concept of a *relationship type* is called *association. "An association defines a semantic relationship between classifiers. The instances of an association are a set of tuples relating instances of the classifiers. Each tuple value may appear at most once."* [p. 2-19] *"An instance of an Association is a Link, which is a tuple of Instances drawn from the corresponding Classifiers."* [p. 2-20]

The *OMG UML Specification* is somehow ambiguous in defining *associations*. An association is primarily considered to be a 'connection', but, in certain cases (whenever it has 'class-like properties'), an association may be a class: *"An association class is an association that is also a class. It not only connects a set of classifiers but also defines a set of features that belong to the relationship itself and not any of the classifiers."* [p.2-21]

An association A between the classes $C_1, \ldots, C_n$ of a conceptual model can be understood in GOL as a relation R between the corresponding universals $U_1, \ldots, U_n$ induced by the relational universal whose extension consists of all relational moments corresponding to the links of A. Let $\phi(a_1, \ldots, a_n)$ denote a condition on the individuals $a_1, \ldots, a_n$. Then

$$[a_1, \ldots, a_n] : R_A(U_1, \ldots U_n) \longleftrightarrow \bigwedge_{i \le n} a_j :: U_i \ \wedge \ \phi(a_1, \ldots, a_n)$$

An association is called *material* if there is a relator universal F such that the condition ϕ is obtained from F as follows:

$$\phi(a_1, \ldots, a_n) \longleftrightarrow \exists k(k :: F \ \wedge \ h(k, a_1, \ldots, a_n))$$

An example of a ternary material association is *purchFrom* corresponding to a relator universal *Purchase* whose instances are individual purchases. These individual purchases connect three individuals: a person, say John, an individual good, e.g. the book *Speech Acts* by Searle, and a shop, say Amazon. Thus,

$$[\text{John}, \text{SpeechActsBySearle}, \text{Amazon}] : R_{\text{purchFrom}}(\text{Person}, \text{Good}, \text{Shop}),$$

since *John::Person*, *SpeechActsBySearle::Good* and *Amazon::Shop*, and there is a specific purchase event *p::Purchase* such that

$$h(p, \text{ John, SpeechActsBySearle, Amazon}).$$

We obtain the following definition for the triple $[a_1, a_2, a_3]$ being a link of the association *purchFrom* between `Person`, `Good` and `Shop`:

$$\begin{aligned}&[a_1, a_2, a_3] : R_{\text{purchFrom}}(\text{Person}, \text{Good}, \text{Shop})\\ &\longleftrightarrow a_1 :: \text{Person} \wedge a_2 :: \text{Good} \wedge a_3 :: \text{Shop}\\ &\quad \wedge \exists p(p :: \text{Purchase} \ \wedge \ h(p, a_1, a_2, a_3))\end{aligned}$$

4 The Bunge-Wand-Weber (BWW) Ontology

The approach found in the literature that is closest to the one presented here is the approach by Evermann and Wand [EW01] and Wand, Storey and Weber [WSW99]. In these two articles, the authors report their results in mapping common constructs of conceptual modeling to an upper level ontology. Their approach is based on the BWW ontology, a framework created by Wand and Weber on the basis of the original metaphysical theory developed by Mario Bunge in [Bun77] and [Bun79].

In this section we will make a comparison between GOL and BWW in terms of their theories and of their corresponding mapping approaches.

4.1 Things and Substances

The concepts of substance (in GOL) and of thing in BWW are both based on the Aristotelian idea of substance, i.e.,

1. an essence which makes a thing what it is;
2. that which remains the same through changes;
3. that which can exist by itself, i.e., which does not need a 'subject' in order to exist.

In BWW, a thing is defined as a substantial individual with all its substantial properties: "a thing is what is the totality of its substantial properties" [Bun77]. As a consequence, in BWW, there are no bare individuals, i.e., things without properties: a thing has one or more substantial properties, even if we, as cognitive subjects, are not or cannot be aware of them. Humans get in contact with the properties of things exclusively via the thing's attributes, i.e. via a chosen representational view of its properties.

A thing in BWW can be directly mapped to the concept of substance in GOL. The axiom (SM1) dictates that a GOL:substance has a non-empty appearance, i.e. every substance bears at least one moment. As it will be explained in the next section, qualities and relational moments in GOL are equivalent to the concepts of intrinsic and mutual properties in BWW, respectively. Not only the two theories confirm each other regarding this issue but the mapping directive presented in this paper also conforms to the BWW-to-UML mapping presented in [EW01].

4.2 Properties and Moments

In BWW, a thing has necessarily at least one property. Likewise, a property exists only in connection with things. A property whose existence depends only on a single thing is called an *intrinsic property* (e.g. the height of a person). A property that depends on two or more things is called a *mutual property* (e.g. being a student is a mutual property between a person and an educational institution). In BWW, only things possess properties. As a consequence, a property cannot have properties. This dictum leads to the following modeling principle: "Associations should not be modeled as classes", (Rule 7) in [EW01]. Contrary to this principle, GOL allows associations, representing relational universals, to have attributes and to participate in second or higher-order associations. Thus, while the BWW approach prohibits to use association classes in conceptual modeling, they are allowed in GOL.

There is an important distinction between the properties possessed by a thing and the representations of these properties by means of attributes. Attributes are characteristics assigned to things according to human perception or conceptualization. Attributes are state functions that provide a mapping from a thing to 'co-domains' whose members can be substantial (attributes representing mutual properties) or conceptual (attributes representing intrinsic properties). In other words attribute functions are used to represent both, intrinsic and mutual properties.

The strong distinction between properties and their representations helps clarifying a common misinterpretation of what a mutual property means. When we say, for example, that "John and Mary are married to each other", we acknowledge the existence of "being married to each other" as a mutual property of John and Mary. At first glance, one could find it counterintuitive to classify this as a mutual property since "being married to Mary" would be the property of John while "being married to John" would be the property of Mary. However, one should notice that we can only get in contact with properties of things via their attributes. Thus, "being married to Mary" is actually an attribute function that represents this mutual property from John's point of view, and not a property.

Again the concepts of BWW-property and GOL-moment can directly be related. Like a property, a moment can only exist if it inheres in an individual substance (axiom SM2). Moments are bound to their associated substances via the inherence relation. This relation can only link moments to substances, i.e. like a property, a moment cannot have moments.

Substances come into existence because matter is formed in specific ways possessing qualities of different sorts. Based on this, we can say that substances have qualities independently of our perception or consideration of these qualities. However, we can only reason about these qualities through their representations as moments. Thus, we can conclude that qualities and relational moments in GOL are, actually, the equivalent of BWW intrinsic and mutual properties, respectively.

In [EW01], it is also proposed that attribute functions representing BWW intrinsic and mutual properties are represented as attributes and associations in UML, respectively.

So, some BWW concepts have a direct counterpart in GOL:

BWW concept	*GOL concept*
substantial thing	substance
intrinsic property	quality
mutual property	relational moment

4.3 Natural Kinds and Universals

In BWW, the definition of a class is based on the notion of the scope of a property. A scope S of a property p is a function assigning to each property that exists in a domain a set of things from that domain, i.e., $S(p)$ is the set of things in the domain that possess property p. A *class* is then defined as the scope of a property.

If we have a non-empty set P of properties, the intersection of the scopes of all members of P is called a *kind*. Finally, a kind whose properties satisfy certain 'laws' (in the sense of integrity constraints) is called a *natural kind*.

In [EW01], a set of attributes (defined as *state functions*) that describe things with common properties is called a *functional schema*. A UML-class cannot be mapped to any of the BWW concepts class, kind or natural kind, because the latter are defined extensionally (as special sets), while the former is defined intensionally and has an extension at run-time. Evermann and Wand therefore propose that "a UML-class is equivalent to a functional schema" of a natural kind.

While in the BWW ontology, the part-whole relation may only hold between substances ('a composite thing consists of other things'), it holds more generally between individuals in GOL. For example, processes can have temporal parts, such as the lifecycle of a product may consist of a design, a production and a maintenance phase.

5 Other Related Work

For a comparison between GOL and other upper-level ontologies, such as the IEEE Standard Upper Ontology, the reader is referred to [DHHS01].

The ONTOCLEAN methodology proposed in [GW02] proposes a number of tests for identifying ill-defined generalization relationships. It is based on a formal ontological framework derived from philosophical ontology the authors have previously presented in a number of papers. The goals of ONTOCLEAN are similar to those of GOL: the process of building ontologies to be used for information systems engineering must become a rigorous engineering discipline based on scientific principles.

6 Conclusions

This paper is only a first step in the attempt to use the General Ontological Language (GOL) and its underlying upper level ontology to evaluate the ontological correctness of UML as a conceptual modeling language, and to develop guidelines that assign well-defined ontological semantics to UML constructs. One important issue for future work is the correct treatment of UML's aggregation concept on the basis of a suitable formalization of the part-whole relation.

Acknowledgements We are grateful to Barbara Heller for fruitful discussions and for providing valuable input to the issues of this article.

References

[Bun77] M. Bunge. *Treatise on Basic Philosophy. Vol. 3. Ontology I. The Furniture of the World.* D. Reidel Publishing, New York, 1977.

[Bun79] M. Bunge. *Treatise on Basic Philosophy. Vol. 4. Ontology II. A World of Systems.* D. Reidel Publishing, New York, 1979.

[DHH02] W. Degen, B. Heller, and H. Herre. GOL: A framework for building and representing ontologies in ontological spring: A reader in formal and applied ontology. Technical Report IFOMIS Reports No. 1, Institute for Formal Ontology and Medical Information Science (IFOMIS), Universitat Leipzig, Germany, 2002.

[DHHS01] W. Degen, B. Heller, H. Herre, and B. Smith. GOL: Towards an axiomatized upper level ontology. In Barry Smith and Nicola Guarino, editors, *Proceedings of FOIS'01*, Ogunquit, Maine, USA, October 2001. ACM Press.

[EW01] J. Evermann and Y. Wand. Towards ontologically based semantics for UML constructs. In H.S. Kunii, S. Jajodia, and A. Solvberg, editors, *Proceedings of ER 2001*, pages 354–367. Springer-Verlag, 2001.

[GW02] N. Guarino and C. Welty. Evaluating ontological decisions with ONTOCLEAN. *Communications of the ACM*, 45(2):61–65, February 2002.

[Ing64] R. Ingarden. *Der Streit um die Existenz der Welt*, volume 1. Max Niemeyer Verlag, Ebingen, 1964.

[WSW99] Y. Wand, V.C. Storey, and R. Weber. An ontological analysis of the relationship construct in conceptual modeling. *ACM Transactions on Database Systems*, 24(4):494–528, December 1999.

Finding and Characterizing Changes in Ontologies

Michel Klein[1], Atanas Kiryakov[2], Damyan Ognyanov[2], and Dieter Fensel[1]

[1] Vrije Universiteit Amsterdam
{michel.klein|dieter}@cs.vu.nl
[2] OntoText Lab. Sofia
{naso|damyan}@sirma.bg

Abstract. Recently, the interest in the use of ontologies — which can be seen as formal representations of conceptual models — has increased because of the excitement about the vision of a “Semantic Web”. When ontologies are used on the web, the distributed and dynamic nature of it requires advanced support for change management. This paper discusses the working of OntoView, a web-based change management system for ontologies. OntoView provides a transparent interface to different versions of ontologies, by maintaining not only the transformations between them, but also the conceptual relation between concepts in different versions. It uses several rules to find changes in ontologies and it visualizes them — and some of their possible consequences — in the file representations. The user is able to specify the conceptual implication of the differences, which allows the interoperability of data that is described by the ontologies. This paper briefly describes the system and presents the mechanism that we used to find and classify changes in RDFS / DAML ontologies. It also shows how users can specify the conceptual implication of changes to help interoperability.

1 Ontologies as Conceptual Models on the Web

Ontologies are specifications of a formal and common understanding of a domain, which try to reduce the gap between the way in which humans and machines handle information. They are developed for knowledge sharing and reuse (see [8]). In the last few years, there has been a lot of interest in ontologies. The current excitement about the vision of a Semantic Web [4] form an additional stimulant for the interest in ontologies. In this vision, ontologies have a role in defining and relating concepts that are used to describe data on the web. The Semantic Web is the idea of having data on the Web defined and linked in a way that it can be used for more effective discovery, automation, integration, and reuse across various applications[3].

The distributed and dynamic character of the web emphasizes a specific issue of ontology research: the evolution and versioning of ontologies. Ontologies are often developed by several persons and continue to evolve over time. Moreover, domain changes, adaptations to different tasks, or changes in the conceptualization might cause modifications of the ontology. This will likely cause incompatibilities in the applications and ontologies that refer to them, and will give wrong interpretations to data or make data inaccessible [9].

[3] `http://www.w3.org/2001/sw/Activity`

To handle ontology changes, a change management system is needed that keeps track of changes and versions of ontologies. Moreover, it is necessary to maintain the links between the versions and variants that specify the relations and updates between the versions. These links can be used to re-interpret data and knowledge under different versions. The ontologies and their relations together form a *web* of ontologies. The specification of these links is thus very important.

In this paper, we present a web-based system that supports the user in specifying the conceptual relation between version of concepts. The system, called OntoView, also maintains those links, together with the transformations between them. It use them to provide a transparent interface to different versions of ontologies, both at a specification level as at a conceptual level. It can also export the differences between versions as separate "mapping ontologies", which can be used as adapters for the re-interpretation of data and other ontologies.

The system could be used in a scenario where ontologies are used to describe data on the internet. Users can copy and adapt ontologies of others, e.g. to fulfil their specific needs. After a specific user changed the ontology off-line, he can use the system to compare the new version of the ontology with the old one and characterize the conceptual implications of changes. The system then exports the conceptual relations between the concepts in the different versions together with meta-data. This exported mapping ontology can be used to translate or to reinterpret instance data or other ontologies automatically.

Most of the ideas underlying the system are not depending on a specific ontology language. However, the implementation of specific parts of the system will be dependent on the used ontology language, for example the mechanism to detect changes. Throughout this article, we will use DAML+OIL[4] [6,7] and RDF Schema (RDFS) [5] as ontology languages. These two languages are widely considered as basis for future ontology languages for the Web.

The rest of the paper is organized as follows. In the next section, we discuss some issues about update relations between ontologies. In section 3, we give a brief overview of the versioning system and describe its the main functions. Section 4 describes the main feature of the system: comparing ontologies. In that section, we explain the mechanism we used to find changes in RDF-based ontologies and present some of the rules that we used to encode change types. Finally, we conclude the paper in section 5.

2 The Update Relation between Ontologies

There are two important things to discuss when considering an update relation. First, this is the difference between update relations and conceptual relations inside an ontology.

Ontologies usually consist of a set of class (or concept) definitions, property definitions and axioms about them. The classes, properties and axioms are related to each other and together form a model of a part of the world. We call these relations conceptual relations inside the domain of interest. A change constitutes a new version of the

[4] Available from `http://www.daml.org/language/`

ontology and defines an orthogonal update relation between the original version of the ontology and the new version. This update relation between two ontologies also entails update relations between concepts and properties in the old version and those in the new version. These implied update relations are depicted in Figure 1.

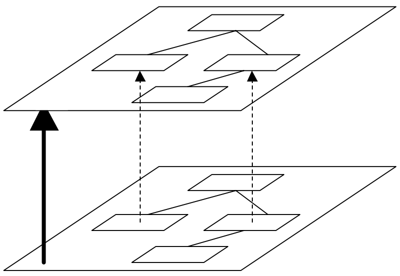


Fig. 1. An update relation (thick arrow) and the implied conceptual relations (dashed arrows) between classes in two version of an ontology.

The update relations between two versions of a concept, e.g. between class $A_{1.0}$ and class $A_{2.0}$, are more than pure conceptual relation in the domain of interest. The update relation also describes meta-information about the change of the concept. We can distinguish the following properties of an update relation:

- **transformation** or **actual change**: a specification of what has actually changed in an ontological definition, specified by a set of change operations (cf. [1]), e.g., change of a restriction on a property, addition of a class, removal of a property, etc.;
- **conceptual relation**: the logical relation between constructs in the two versions of the ontology, e.g., specified by equivalence relations, subsumption relations, or logical rules;
- descriptive meta-data like **date**, **author**, and **intention** of the update: this describes the when, who and why of the change;
- **valid context**: a description of the context in which the update is valid. In its simplest form, this might consist of the date when the change is valid in the real world, conform to *valid date* in temporal databases [12] (in this terminology, the "date" in the descriptive meta-data is called *transaction date*). More extensive descriptions of the context, in various degrees of formality, are also possible.

A well-designed ontology change specification mechanism should take all these characteristics into account.

Another issue to discuss about ontology updates is the possible discrepancy between changes in the specification and changes the conceptualization. We have seen that a ontology is a *specification* of a *conceptualization*. The actual specification of concepts and properties is thus a *specific representation* of the conceptualization: the same concepts could also have been specified differently. Hence, a change in the specification does not necessarily coincide with a change in the conceptualization [9], and changes in the specification of an ontology are not per definition ontological changes.

For example, there are changes in the definition of a concept which are not meant to change the concept itself: attaching a slot "fuel-type" to a class "Car". Both class-definitions still refer to the same ontological concept, but in the second version it is described more extensively. Theoretically, the other way around is also possible: a concept could change without a change in its specification. However, this usually means that the concept is badly modelled.

It is important to distinguish changes in ontologies that affect the conceptualization from changes that don't. In [13] the following terms are used:

- **conceptual change**: a change in the way a domain is interpreted (conceptualized), which results in different ontological concepts or different relations between those concepts;
- **explication change**: a change in the way the conceptualization is specified, without changing the conceptualization itself.

A specific modification of an ontology cannot automatically be classified as belonging to one of these categories, because it is basically a decision of the modeler. However, heuristics can be applied to suggest the effects of changes. We will discuss that later on.

3 General Description of OntoView

OntoView is a web-based system under development that provides support for the versioning of online ontologies, which might help to solve some of the problems of evolving ontologies on the web. Its main function is to help the a user to manage changes in ontologies and keep ontology versions as much interoperable as possible. It does that by comparing versions of ontologies and highlighting the differences. It then allows the users to specify the conceptual relation between the different versions of concepts. This function is described more extensively in the next section.

The system can also function as a storage system for version of ontologies, providing a transparent interface to arbitrary versions of ontologies. To achieve this, the system maintains an internal specification of the relation between the different variants of ontologies, with the aspects that were defined in section 2: it keeps track of the **meta-data**, the **conceptual relations** between constructs in the ontologies and the **transformations** between them.

OntoView is inspired by the Concurrent Versioning System CVS, which is used in software development to allow collaborative development of source code. The first implementation is also based on CVS and its web-interface CVSWeb[5]. However, during the ongoing development of the system, we are gradually shifting to a complete new implementation that will be build on a solid storage system for ontologies, e.g., Sesame[6].

Besides the ontology comparison feature, the system has the following functions:

[5] Available from `http://stud.fh-heilbronn.de/~zeller/cgi/cvsweb.cgi/`

[6] A demo is available at `http://sesame.aidministrator.nl`

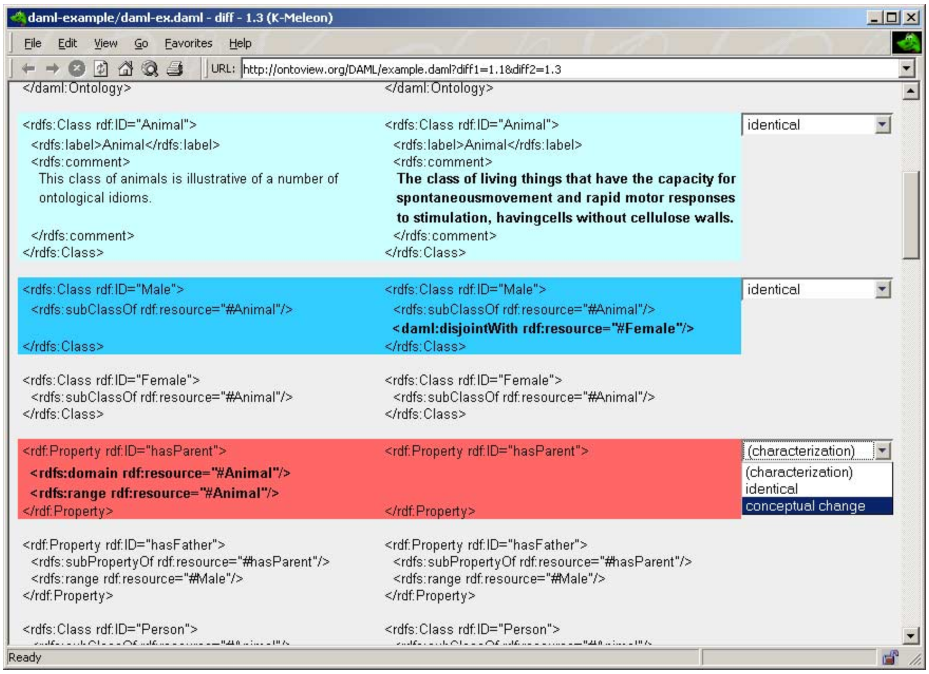
- **Reading changes and ontologies.** OntoView will accept changes and ontologies via several methods. Currently, ontologies can be read in as a whole, either by providing a URL or by uploading them to the system. The user has to specify whether the provided ontology is new or that it should be considered as an update to an already known ontology. In the future, OntoView will also accept changes by reading in transformations, mapping ontologies, and updates to individual definitions. These update methods provides the system with different information than the method described above. For that reason, this also requires an adaptation of the process in which the user gives additional information.
- **Identification.** Identification of versions of ontologies is very important. Ontologies describe a consensual view on a part of the world and function as reference for that specific conceptualization. Therefore, they should have a unique and stable identification. A human, agent or system that conforms to a specific ontology, should be able to refer to it unambiguously.
- **Analyzing effects of changes.** Changes in ontologies do not only affect the data and applications that use them, but they can also have unintended, unexpected and unforeseeable consequences in the ontology itself [11].

  OntoView provides some basic support for the analysis of these effects. First, on request it can also highlight the places in the ontology where conceptually changed concepts or properties are used. For example, if a property "hasChild" is changed, it will highlight the definition of the class "Mother", which uses the property "hasChild". In the future, this function should also exploit the transitivity of properties to show the propagation of possible changes through the ontology.

  Further, we expect to extend the system with a reasoner to automatically verify the changes and the specified conceptual relations between versions. For example, we could couple the system with FaCT [3] and exploit the Description Logic semantics of DAML+OIL to check the consistency of the ontology and look for unexpected implied relations.
- **Exporting changes.** The main advantage of storing the conceptual relations between versions of concepts and properties is the ability to use these relations for the re-interpretation of data and other ontologies that use the changed ontology. To facilitate this, OntoView can export differences between ontologies as separate mapping ontologies, which can be used as adapters for data sources or other ontologies. They only provide a partial mapping, because not all changes can be specified conceptually, e.g. complicated changes like splits of concepts, or deletions.

4 Comparing Ontologies

One of the central features of OntoView is the ability to compare ontologies at a structural level. The comparison function is inspired by UNIX `diff`, but the implementation is quite different. Standard `diff` compares file version at line-level, highlighting the lines that textually differ in two versions. OntoView, in contrast, compares version of ontologies at a *structural* level, showing which definitions of ontological con-


daml-example/daml-ex.daml - diff - 1.3 (K-Meleon)
File Edit View Go Favorites Help
Ready

URL: http://ontoview.org/DAML/example.daml?diff1=1.1&diff2=1.3

```
</daml:Ontology>                                   </daml:Ontology>

<rdfs:Class rdf:ID="Animal">                       <rdfs:Class rdf:ID="Animal">                        identical
  <rdfs:label>Animal</rdfs:label>                    <rdfs:label>Animal</rdfs:label>
  <rdfs:comment>                                     <rdfs:comment>
    This class of animals is illustrative of a number of    The class of living things that have the capacity for
    ontological idioms.                                spontaneousmovement and rapid motor responses
                                                       to stimulation, havingcells without cellulose walls.
  </rdfs:comment>                                    </rdfs:comment>
</rdfs:Class>                                      </rdfs:Class>

<rdfs:Class rdf:ID="Male">                         <rdfs:Class rdf:ID="Male">                          identical
  <rdfs:subClassOf rdf:resource="#Animal"/>          <rdfs:subClassOf rdf:resource="#Animal"/>
                                                     <daml:disjointWith rdf:resource="#Female"/>
</rdfs:Class>                                      </rdfs:Class>

<rdfs:Class rdf:ID="Female">                       <rdfs:Class rdf:ID="Female">
  <rdfs:subClassOf rdf:resource="#Animal"/>          <rdfs:subClassOf rdf:resource="#Animal"/>
</rdfs:Class>                                      </rdfs:Class>

<rdf:Property rdf:ID="hasParent">                  <rdf:Property rdf:ID="hasParent">                   (characterization)
  <rdfs:domain rdf:resource="#Animal"/>                                                                  (characterization)
  <rdfs:range rdf:resource="#Animal"/>                                                                   identical
</rdf:Property>                                    </rdf:Property>                                     conceptual change

<rdf:Property rdf:ID="hasFather">                  <rdf:Property rdf:ID="hasFather">
  <rdfs:subPropertyOf rdf:resource="#hasParent"/>    <rdfs:subPropertyOf rdf:resource="#hasParent"/>
  <rdfs:range rdf:resource="#Male"/>                 <rdfs:range rdf:resource="#Male"/>
</rdf:Property>                                    </rdf:Property>

<rdfs:Class rdf:ID="Person">                       <rdfs:Class rdf:ID="Person">
```

Fig. 2. Comparing two ontologies

cepts or properties are changed. An example of such a comparison of two versions of a DAML+OIL ontology is depicted in Figure 2.[7]

4.1 Types of Change

The comparison function distinguishes between the following types of change:

- Non-logical change, e.g. in a natural language description. In DAML+OIL, this are changes in the rdfs:label of an concept or property, or in a comment inside a definition. An example is the first highlighted change in Figure 2 (class "Animal').
- Logical definition change. This is a change in the logical definition of a concept. Examples of such changes are alterations of subClassOf, domain, or range statements. Additions or deletions of local property restrictions in a class are also logical changes. The second and third change in the figure is (class "Male" and property "hasParent") are examples of such changes. Note that there are also logical changes that do not affect the semantics. However, we
- Identifier change. This is the case when a concept or property is given a new identifier, i.e. a renaming.
- Addition of definitions.

[7] This example is based on fictive changes to the DAML example ontology, available from `http://www.daml.org/2001/03/daml+oil-ex.daml`.

– Deletion of definitions.

Each type of change is highlighted in a different color, and the actually changed lines are printed in boldface.

Most of these changes can be detected completely automatically, except for the identifier change, because this change is not distinguishable from a subsequent deletion and addition of a simple definition. In this case, the system uses the location of the definition in the file as a heuristic to determine whether it is an identifier change or not.

It is a deliberate choice not to show all changes, but only the ones which we think that are of interest to the ontology modeler. This choice is explained in the next paragraphs, together with the mechanism that we use to detect and classify changes. Experimental validation should show whether this list of change types is sufficient.

4.2 Detecting Changes

There are two main problems with the detection of changes in ontologies. The first problem is the abstraction level at which changes should be detected. Abstraction is necessary to distinguish between changes in the representation that affect the meaning, and those that don't influence the meaning. It is often possible to represent the same ontological definition in different ways. For example, in RDF Schema, there are several ways to define a class:

```
<rdfs:Class rdf:ID="ExampleClass"/>
```

or:

```
<rdf:Description rdf:ID="ExampleClass">
  <rdf:type  rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
```

Both are valid ways to define a class and do not change the ontology. Thus, detecting changes in the *representation* alone is not sufficient.

However abstracting too far is also a problem: considering the *logical meaning* only is not enough. In [2] is shown that different sets of ontological definitions can yield the same set of logical axioms. Although the logical meaning is not changed in such cases, the ontology definitely is. Finding the right level of abstraction is thus important.

Second, even when we found the correct level of abstraction for change detection, the conceptual implication of such a change is not yet clear. Because of the difference between conceptual changes and explication changes (as described in section 2), it is not possible to derive the conceptual consequence of a change completely on basis of the visible change only (i.e., the changes in the definitions of concepts and properties). Heuristics can be used to suggest conceptual consequences, but the intention of the engineer determines the actual conceptual relation between versions of concepts.

In the next two sections, we explain the algorithm that we used to compare ontologies at the correct abstraction level, and how users can specify the conceptual implication of changes.

4.3 Rules for Changes

The algorithm uses the fact that the RDF data model [10] underlies a number of popular ontology languages, including RDF Schema and DAML+OIL. The RDF data model basically consists of triples of the form `<subject, predicate, object>`, which can be linked by using the object of one triple as the subject of another. There are several syntaxes available for RDF statement, but they all boil down to the same data model. An set of related RDF statements can be represented as a graph with nodes and edges. For example, consider the following DAML+OIL definition of a class "Person".

```
<daml:Class rdf:ID="Person">
  <rdfs:subClassOf rdf:resource="#Animal"/>
  <rdfs:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#hasParent"/>
      <daml:toClass rdf:resource="#Person"/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

When interpreted as a DAML+OIL definition, it states that a "Person" is a kind of "Animal" and that the instances of its "hasParent" relation should be of type "Person". However, for our algorithm, we are first of all interested in the RDF interpretation of it. That is, we only look at the triples that are specified, ignoring the DAML+OIL meaning of the statements. Interpreted as RDF, the above definition results in the following set of triples:

subject	**predicate**	**object**
Person	rdf:type	daml:Class
Person	rdfs:subClassOf	Animal
Person	rdfs:subClassOf	*anon-resource*
anon-resource	rdf:type	daml:Restriction
anon-resource	daml:onProperty	hasParent
anon-resource	daml:toClass	Person

This triple set is depicted as a graph in Figure 3. In this figure, the nodes are resources that function as subject or object of statements, whereas the arrows represent properties.

The algorithm that we developed to detect changes is the following. We first split the document at the first level of the XML document. This groups the statements by their intended "definition". The definitions are then parsed into RDF triples, which results in a set of small graphs. Each of these graphs represent a specific definition of a concept or a property, and each graph can be identified with the identifier of the concept or the property that it represents.

Then, we locate for each graph in the new version the corresponding graph in the previous version of the ontology. Those sets of graphs are then checked according to a number of rules. Those rules specify the "required" changes in the triples set (i.e., the graph) for a specific type of change, as described in section 4.1.

Rules have the following format:

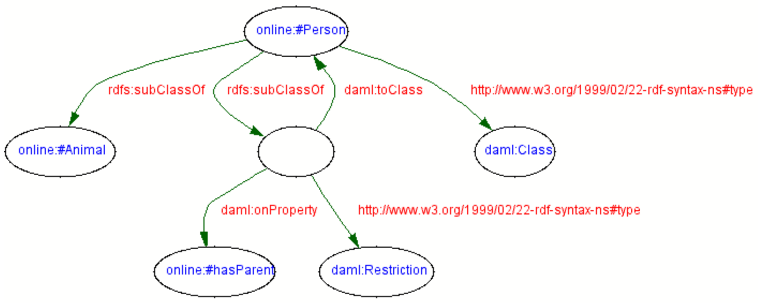


Fig. 3. An RDF graph of a DAML class definition.

```
IF exist:old
     <A, Y, Z>*
   exist:new
     <X, Y, Z>*
   not-exist:new
     <X, Y, Z>*
THEN change-type A
```

They specify a set of triples that should exists in one specific version, and a set that should not exists in another version to signal a specific type of change. With this rule mechanism, we were able to specify almost all types of change, except the identifier change. Here we also used some heuristics, based on the location of the definition in the file. We list two example rules below.

A change in the value of a local property:

```
IF exist:old
     <X, rdfs:subClassOf, Y1>
     <Y1, rdf:type, daml:#Restriction>
     <Y1, daml:onProperty, Y2>
     <Y1, daml:toClass, Z>
   exist:new
     <X, rdfs:subClassOf, Y1>
     <Y1, rdf:type, daml:#Restriction>
     <Y1, daml:onProperty, Y2>
   not-exist:new
     <Y1, daml:toClass, Z>
THEN logicalChange.localPropertyValue X
```

A change in the property type:

```
IF exist:old
```

```
      <X, rdf:type, rdf:#Property>
      <X, rdf:type, daml:#UniqueProperty>
    exist:new
      <X, rdf:type, rdf:#Property>
    not-exist:new
      <X, rdf:type, daml:#UniqueProperty>
THEN logicalChange.propertytype X
```

The rules are specialized for a specific RDF-based ontology language (in this case DAML+OIL), because they encode the interpretation of the semantics of the language for which they are intended. For another language, other rules would have been necessary to specify other differences in interpretation. The semantics of the language are thus encoded in the rules. For example, the last example not looks at changes in values of predicates (as the first does), but at a change in the type of property. This is a change that is related to the specific semantics of DAML+OIL.

In the prototype system, we were able to specify all changes types that we wanted to detect via this rule format.

4.4 Specifying the Conceptual Implication of Changes

The comparison function also allows the user to *characterize* the conceptual implication of the changes. For the first three types of changes that were listed in section 4.1, the user is given the option to label them either as "identical" (i.e., the change is an explication change), or as "conceptual change", using the drop-down list next to the definition (Figure 2). In the latter case, the user can specify the conceptual relation between the two version of the concept. For example, the change in the definition of "hasParent" could by characterized with the relation `hasParent`$_{1.1}$ `subPropertyOf` `hasParent`$_{1.3}$.

5 Conclusion

Conceptual models will play an important role in the envisaged "Semantic Web". Versioning support for such ontologies is essential when they are used in such a distributed and dynamic context. In this paper we have analyzed the versioning relation and have described a system that provides support for change management of online ontologies.

In the system that we described, all the dimensions of a versioning relation are specified separately: the descriptive **meta-data**, the **conceptual relations** between constructs in the ontologies, and the **transformations** between the ontologies themselves. This allows both complete transformations of ontology representations and partial data reinterpretations. The conceptual relations can be exported and used to adapt data sources and ontologies.

We described how the systems support users in comparing ontologies, and what the problems and challenges are. We presented a algorithm to perform a comparison for RDFS-based ontologies. This algorithm doesn't operate on the representation of the ontology, but on the data model that is underlying the representation. By grouping the RDF-triples per definition, we still retained the necessary representational knowledge.

We also explained how users can specify the conceptual implication of changes to help interoperability. This honors the fact that it is not possible to derive all conceptual implications of changes automatically. An important advantage of this approach is that there is no write-access needed to the original ontologies. The exported conceptual relations between versions of concepts live on their own as separate mapping ontologies.

The described system is not yet finished and should be developed further. We believe that it will significantly simplify the change management of ontologies and thus help the interoperability of different domain models on the web.

References

1. J. Banerjee, W. Kim, H.-J. Kim, and H. F. Korth. Semantics and Implementation of Schema Evolution in Object-Oriented Databases. *SIGMOD Record (Proc. Conf. on Management of Data)*, 16(3):311–322, May 1987.
2. S. Bechhofer, C. Goble, and I. Horrocks. DAML+OIL is not enough. In *Proceedings of the International Semantic Web Working Symposium (SWWS)*, Stanford University, California, USA, July 30 – Aug. 1, 2001.
3. S. Bechhofer, I. Horrocks, P. F. Patel-Schneider, and S. Tessaris. A proposal for a description logic interface. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Proceedings of the International Workshop on Description Logics (DL'99)*, pages 33–36, Linköping, Sweden, July 30 – Aug. 1 1999.
4. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.
5. D. Brickley and R. V. Guha. Resource Description Framework (RDF) Schema Specification 1.0. Candidate recommendation, World Wide Web Consortium, Mar. 2000.
6. D. Fensel, I. Horrocks, F. van Harmelen, S. Decker, M. Erdmann, and M. Klein. OIL in a nutshell. In R. Dieng and O. Corby, editors, *Knowledge Engineering and Knowledge Management; Methods, Models and Tools, Proceedings of the 12th International Conference EKAW 2000*, number 1937 in LNCS, pages 1–16, Juan-les-Pins, France, Oct. 2–6, 2000. Springer-Verlag.
7. D. Fensel and M. A. Musen. The semantic web: A new brain for humanity. *IEEE Intelligent Systems*, 16(2), 2001.
8. T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 1993.
9. M. Klein and D. Fensel. Ontology versioning for the Semantic Web. In *Proceedings of the International Semantic Web Working Symposium (SWWS)*, pages 75 – 91, Stanford University, California, USA, July 30 – Aug. 1, 2001.
10. O. Lassila and R. R. Swick. Resource Description Framework (RDF): Model and Syntax Specification. Recommendation, World Wide Web Consortium, Feb. 1999. See http://www.w3.org/TR/REC-rdf-syntax/.
11. D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *KR2000: Principles of Knowledge Representation and Reasoning*, pages 483–493, San Francisco, 2000. Morgan Kaufmann.
12. J. F. Roddick. A survey of schema versioning issues for database systems. *Information and Software Technology*, 37(7):383–393, 1995.
13. P. R. S. Visser, D. M. Jones, T. J. M. Bench-Capon, and M. J. R. Shave. An analysis of ontological mismatches: Heterogeneity versus interoperability. In *AAAI 1997 Spring Symposium on Ontological Engineering*, Stanford, USA, 1997.

Superimposed Schematics: Introducing E-R Structure for *In-Situ* Information Selections[*]

Shawn Bowers, Lois Delcambre, and David Maier

OGI School of Science and Engineering at OHSU
Beaverton OR 97006, USA
{shawn,lmd,maier}@cse.ogi.edu

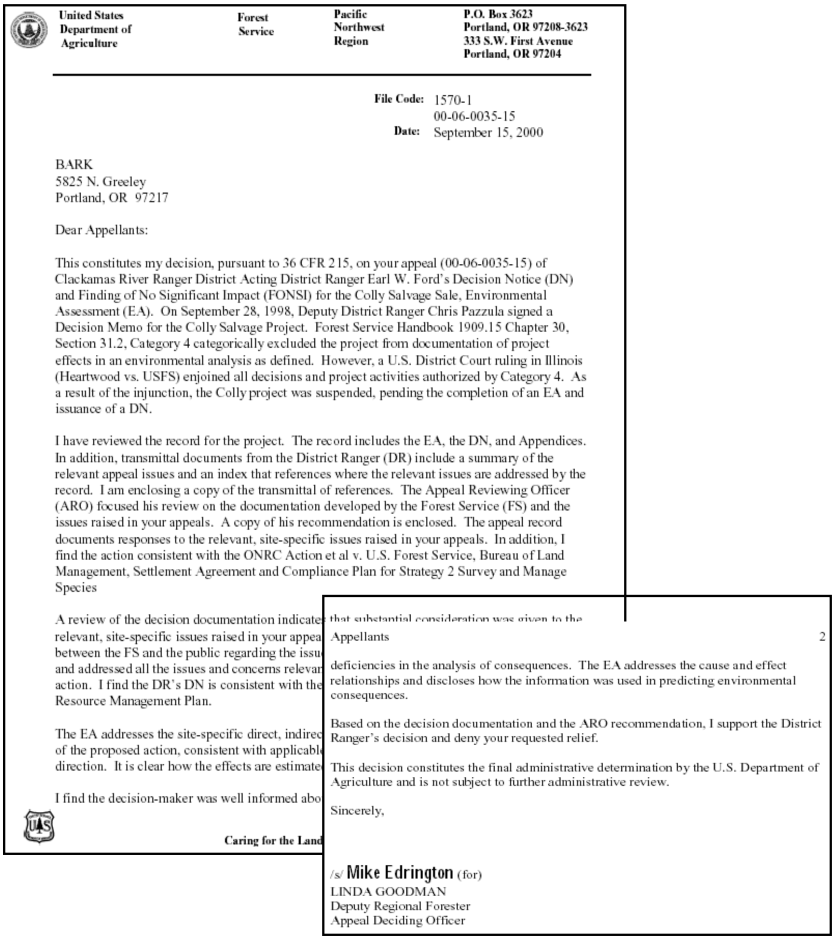
Abstract. Considerable research exists on providing structured access to unstructured information sources, primarily for search and query. Little attention has been placed on (1) keeping the by-products of the process (e.g., connections between structured and unstructured data are typically forgotten and context is lost), (2) developing a rich, conceptual structure for this new layer of information (e.g., the structured data is often represented in simple relational tables), and (3) further elaborating and linking the new, structured information. We propose superimposed schematics to address these issues. Superimposed schematics offer E-R modeling constructs integrated with marks, where a mark holds an address to an information element in an underlying source. A superimposed schematic enables a conceptual addressing scheme for the information it contains (directly) and for the information it references (through marks). This addressing scheme can then be used for marking from additional layers of superimposed information. We consider schematics as a useful model for superimposed information and discuss how it fits into our general research on superimposed information.

1 Introduction

The actions of the USDA Forest Service, such as selling timber or issuing or denying special use permits, are officially documented in *Decision Notices*, *Records of Decision*, and so forth. Members of the public then have the right to file an appeal requesting that the decision be changed. When the appeal deadline has passed, a *reviewing officer* from the Forest Service normally considers the entire set of appeals for a given decision and makes a recommendation. Finally, the *deciding officer* makes a determination for each issue raised (in one or more appeals). Each *appeal decision*[1] is typically represented in two standard letters, one from the reviewing officer and one from the deciding officer (see Figure 1 for an example decision memo).

[*] This work supported in part by the National Science Foundation through grants EIA-99-83518 and IIS-98-17492. Information and examples concerning the Forest Service and appeal decisions provided by Timothy Tolle of the USDA Forest Service

[1] The Pacific Northwest Region of the USDA Forest Service maintains appeal decisions for twenty national forests online at http://www.fs.fed.us/r6/plan/appeal.htm, dating back to 1997.


United States Department of Agriculture | Forest Service | Pacific Northwest Region | P.O. Box 3623 Portland, OR 97208-3623 333 S.W. First Avenue Portland, OR 97204

File Code: 1570-1
00-06-0035-15
Date: September 15, 2000

BARK
5825 N. Greeley
Portland, OR 97217

Dear Appellants:

This constitutes my decision, pursuant to 36 CFR 215, on your appeal (00-06-0035-15) of Clackamas River Ranger District Acting District Ranger Earl W. Ford's Decision Notice (DN) and Finding of No Significant Impact (FONSI) for the Colly Salvage Sale, Environmental Assessment (EA). On September 28, 1998, Deputy District Ranger Chris Pazzula signed a Decision Memo for the Colly Salvage Project. Forest Service Handbook 1909.15 Chapter 30, Section 31.2, Category 4 categorically excluded the project from documentation of project effects in an environmental analysis as defined. However, a U.S. District Court ruling in Illinois (Heartwood vs. USFS) enjoined all decisions and project activities authorized by Category 4. As a result of the injunction, the Colly project was suspended, pending the completion of an EA and issuance of a DN.

I have reviewed the record for the project. The record includes the EA, the DN, and Appendices. In addition, transmittal documents from the District Ranger (DR) include a summary of the relevant appeal issues and an index that references where the relevant issues are addressed by the record. I am enclosing a copy of the transmittal of references. The Appeal Reviewing Officer (ARO) focused his review on the documentation developed by the Forest Service (FS) and the issues raised in your appeals. A copy of his recommendation is enclosed. The appeal record documents responses to the relevant, site-specific issues raised in your appeals. In addition, I find the action consistent with the ONRC Action et al v. U.S. Forest Service, Bureau of Land Management, Settlement Agreement and Compliance Plan for Strategy 2 Survey and Manage Species

A review of the decision documentation indicates that substantial consideration was given to the
relevant, site-specific issues raised in your appea
between the FS and the public regarding the issu
and addressed all the issues and concerns relevar
action. I find the DR's DN is consistent with the
Resource Management Plan.

The EA addresses the site-specific direct, indirec
of the proposed action, consistent with applicabl
direction. It is clear how the effects are estimate

I find the decision-maker was well informed abo

Caring for the Land

Appellants 2

deficiencies in the analysis of consequences. The EA addresses the cause and effect relationships and discloses how the information was used in predicting environmental consequences.

Based on the decision documentation and the ARO recommendation, I support the District Ranger's decision and deny your requested relief.

This decision constitutes the final administrative determination by the U.S. Department of Agriculture and is not subject to further administrative review.

Sincerely,

/s/ Mike Edrington (for)
LINDA GOODMAN
Deputy Regional Forester
Appeal Deciding Officer

Fig. 1. The decision letter for an example appeal decision

Although not easy to discern from looking at the documents, an appeal decision is comprised of a fairly standard set of information items, such as: (1) which decision is being challenged, (2) which appellants have filed (and on behalf of which organization), (3) what issues were raised (across the set of appeals), and (4) what final determination was made for each issue. We observe that the standard items and relationships among them can be usefully represented in a *superimposed schematic*, an ER-style schema for superimposed information, as shown in Figure 2.

The key feature of a superimposed schematic is that it can be populated with *marks*, which are integral to superimposed information [4,5,6], in addition to regular attribute values. Each mark represents and holds the address for an information excerpt from the underlying information source (e.g., in a decision letter), as shown in Figure 3. We see, for example, that a particular appeal is mentioned in the first paragraph of the decision letter and that the issue raised in that appeal is described by the entire third paragraph of the review document.

Appeal decisions are well suited for superimposed schematics because they are unstructured, heterogeneous sources of information with important, conceptual content (e.g., decision and review letters have little physical structure other than a typical memo). Schematics provide structured access at “no additional cost” [14], i.e., the underlying information does not require modification since schematics do not add semantic markup directly to the document. Schematics also offer more than simple nested hierarchies of document structure; they model exactly the concepts of interest (regardless of the documents structure) through an E-R schema. In general, any given document may have many associated schematics.

We consider document authors, schematic designers, schematic populators, and users of populated schematics as (potentially) separate people. For example, a schematic populator may create a new schematic instance for each appeal decision allowing natural resource managers (the end-users) to easily browse appeal decisions. A resource manager might begin with an Appeal Decision entity, navigate to the determination(s) and their associated issues, and then browse to see if the issue was raised on behalf of an organization.

Besides navigation, schematics enable a powerful mechanism for queries, which can be answered across a collection of schematic instances. When a natural resource manager is pondering a decision, she might like to know what issues were raised for similar decisions. And at a more strategic level, the USDA Forest Service routinely analyzes appeals to track the most important issues and trends. Both tasks are tedious and labor-intensive, requiring appeal decisions to be read and searched manually. Superimposed schematics are designed for both purposes: information browsing (introducing structure of interest over an unstructured universe of information) and collection-based querying.

Finally, most documents and their associated tools (like editors or browsers) provide limited addressing capabilities (e.g., page and byte offset, or section, paragraph, sentence heirarchies). One of our goals is to use superimposed schematics for *enhanced addressing* over underlying documents, i.e., to create addresses based on the conceptual information (in the schematic) for accessing document content. Enhanced addressing can be used for marking information elements in one schematic from additional layers of superimposed information. A resource manager can develop a schematic (as a new layer of superimposed information) for tracking important actions, issues, appeals, and results; and populate instances by using enhanced addresses to environmental assessment, scientific study, report, and appeal decision schematics.

In this paper, we present the data model for superimposed schematics and describe how schematics provide enhanced addressing for further levels of superimposed information. Our next step, which we are still currently developing, is to exploit schematics for query. The rest of this paper is organized as follows. In Section 2, we describe the data model for superimposed schematics. Section

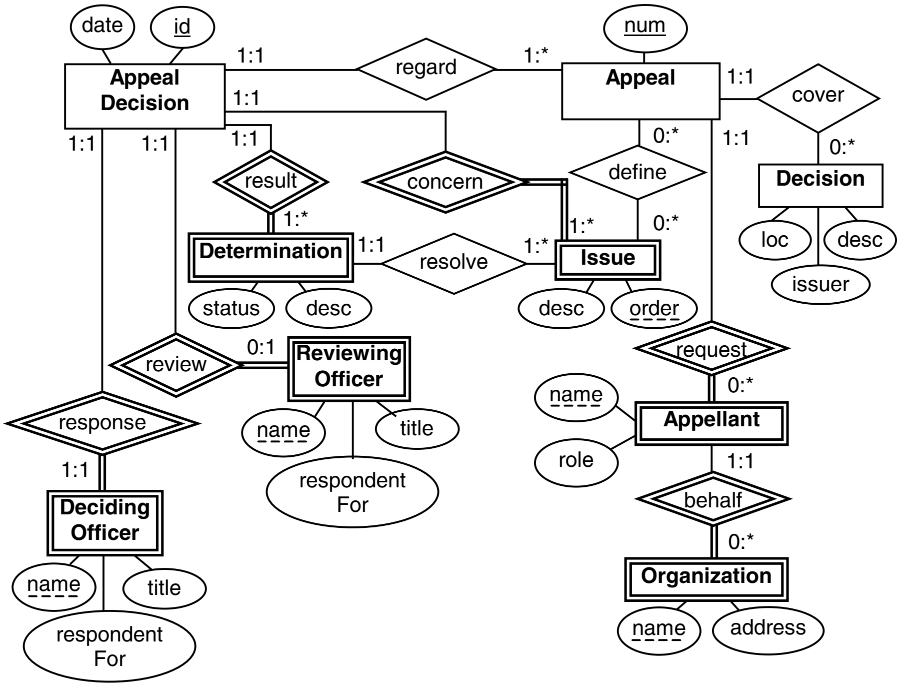


Fig. 2. A superimposed schematic representing the conceptual structure of the standard information elements in an appeal decision

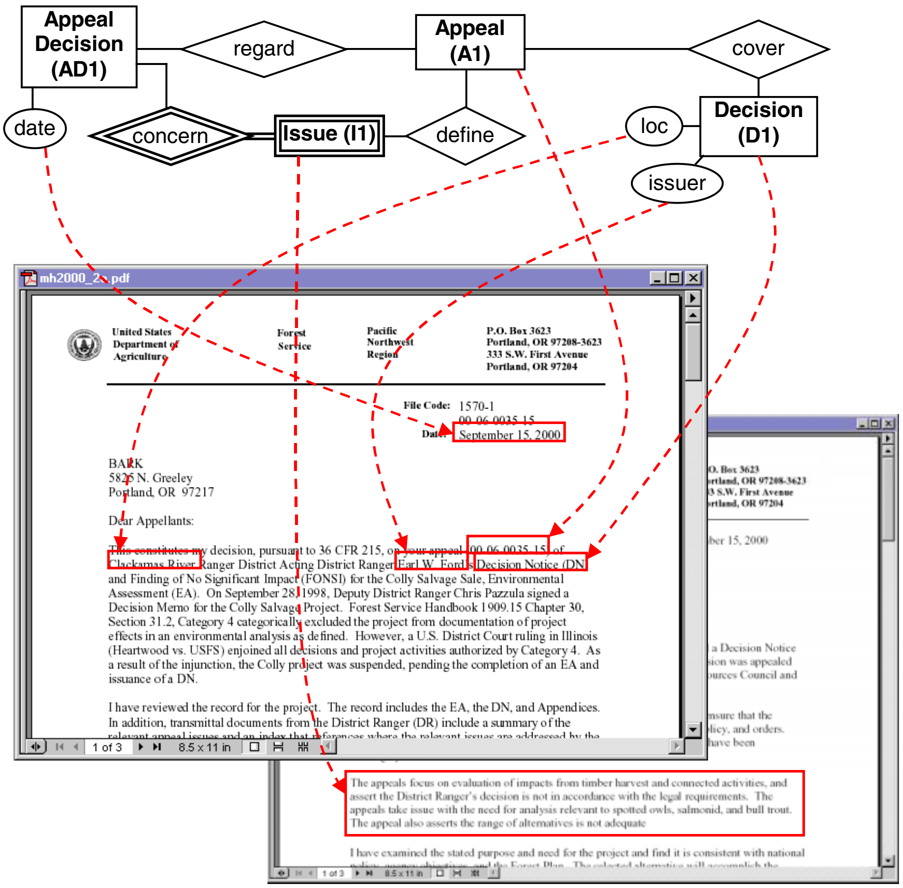
3 presents an addressing language for superimposed schematics for supporting enhanced addressing. We discuss related work in Section 4, and conclude with a summary and a discussion of future work in Section 5.

2 A Data Model for Superimposed Schematics

The schematics data model is fundamentally based on Chen's original E-R data model [3], however, to accomodate schematics, we permit multiple instances of a schema and most importantly, incorporate marks.

2.1 The Basic Schematics Data Model

Within a given domain, there may be multiple schematics of interest, which we maintain in a collection called the *schematic universe*. We define a schematic universe $\mathcal{U}$ as a logical collection of schematics represented as the triple $(\mathcal{S}, \mathcal{ET}, \mathcal{RT})$, consisting of a set of named superimposed schematics $\mathcal{S}$, a set of named schematic entity types $\mathcal{ET}$, and a set of named schematic relationship types $\mathcal{RT}$. Each schematic $s \in \mathcal{S}$ consists of a set of schematic entity types in $\mathcal{ET}$ such that $\mathcal{S}$



United States Department of Agriculture

Forest Service

Pacific Northwest Region

P.O. Box 3623
Portland, OR 97208-3623
333 S.W. First Avenue
Portland, OR 97204

File Code: 1570-1
00-06-0035-15
Date: September 15, 2000

BARK
5825 N. Greeley
Portland, OR 97217

Dear Appellants:

This constitutes my decision, pursuant to 36 CFR 215, on your appeal 00-06-0035-15 of Clackamas River Ranger District Acting District Ranger Earl W. Ford's Decision Notice (DN) and Finding of No Significant Impact (FONSI) for the Colly Salvage Sale, Environmental Assessment (EA). On September 28, 1998, Deputy District Ranger Chris Pazzula signed a Decision Memo for the Colly Salvage Project. Forest Service Handbook 1909.15 Chapter 30, Section 31.2, Category 4 categorically excluded the project from documentation of project effects in an environmental analysis as defined. However, a U.S. District Court ruling in Illinois (Heartwood vs. USFS) enjoined all decisions and project activities authorized by Category 4. As a result of the injunction, the Colly project was suspended, pending the completion of an EA and issuance of a DN.

I have reviewed the record for the project. The record includes the EA, the DN, and Appendices. In addition, transmittal documents from the District Ranger (DR) include a summary of the

O. Box 3623
rtland, OR 97208-3623
3 S.W. First Avenue
rtland, OR 97204

ber 15, 2000

a Decision Notice
sion was appealed
ources Council and

nsure that the
licy, and orders.
have been

The appeals focus on evaluation of impacts from timber harvest and connected activities, and assert the District Ranger's decision is not in accordance with the legal requirements. The appeals take issue with the need for analysis relevant to spotted owls, salmonid, and bull trout. The appeal also asserts the range of alternatives is not adequate

I have examined the stated purpose and need for the project and find it is consistent with national

Fig. 3. A portion of decision memo (on the left) and accompanying review memo (on the right) shown with a portion of the populated superimposed schematic (at the top)

partitions $\mathcal{ET}$. Therefore, every schematic entity type in $\mathcal{U}$ belongs to a superimposed schematic, and no two superimposed schematics contain the same entity type. Schematic relationship types are considered part of a schematic when all entity types they associate are within the schematic. Relationship types are not required to be within a schematic, i.e., we permit schematic relationship types that span schematics.

A schematic entity type $et \in \mathcal{ET}$ consists of a set of named attributes A. Each attribute in A has an associated domain represented by the function $dom : \mathcal{ET} \times \mathcal{A} \rightarrow \mathcal{VT}$, where $\mathcal{A}$ and $\mathcal{VT}$ represent the set of attribute names and value types in $\mathcal{U}$, respectively (note that if the attribute name is not defined for the entity type, dom returns $null$). We consider only single-valued attributes that have either atomic (e.g., strings and integers), complex (structured values such as dates and phone numbers), or mark values. Additionally, we introduce the *uninterpreted type* as a valid value type, which says that the attribute can contain a value of any type including an unknown type (which is useful for certain kinds of marks).

A schematic relationship type $rt \in \mathcal{RT}$ is a multi-directional association between one or more entity types and is represented as a set $\mathcal{L}$ of *roles*, each of which is a pair (rn, et) where rn is a role name and $et \in \mathcal{ET}$ an entity type. Roles represent the entity types associated by relationship types. For non-recursive binary relationship types, role names are optional (a missing role name is represented by $null$). However, all recursive and n-ary (where $n > 2$) relationships require role names. We do not permit attributes on relationships.

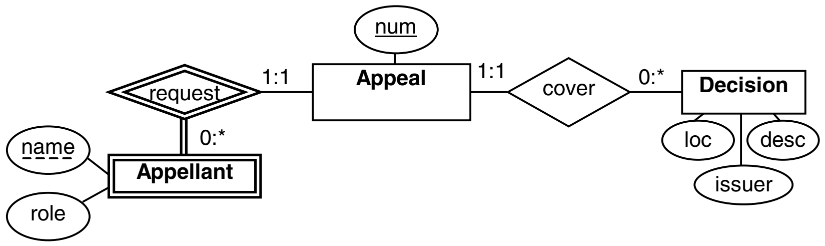
Instead of using the typical method of grouping entities into entity sets and relationships into relationship sets, we introduce an explicit extent for the schematic universe. The *universal schematic extent* $\mathcal{D}$ is a triple $(\mathcal{I}, \mathcal{E}, \mathcal{R})$ that contains a set of schematic instances $\mathcal{I}$, a set of schematic entity identifiers $\mathcal{E}$ (representing entities), and a set of schematic relationship identifiers $\mathcal{R}$ (representing relationships). Each schematic instance $i \in \mathcal{I}$ contains a set of entities in $\mathcal{E}$ such that $\mathcal{I}$ partitions $\mathcal{E}$.

A schematic entity identifier e is a pair (et, V), where $et \in \mathcal{ET}$ is the type of the entity and V a set of attribute-value pairs (att, v). (Note the definition of type is more restrictive than permitted in most E-R models, where an entity can belong to more than one entity set, i.e., type.) For $V = \{(att_1, v_1), (att_2, v_2), \ldots, (att_n, v_n)\}$, we require each att_i for $1 \leq i \leq n$ be unique such that if A is the set of attributes for et, $att_i \in A$ and $n \leq |A|$. That is, each attribute in V is unique and there is one attribute for each attribute of et.

A schematic relationship identifier r is a pair (rt, E) such that $rt \in \mathcal{RT}$ is the type of the relationship and E is a set of role name, schematic entity identifier pairs (rn, e) for $e \in \mathcal{E}$. Like entity attributes, each role name corresponds to a role name in rt's set of roles $\mathcal{L}$. We limit relationships such that only one relationship of a given type can occur between a set of entity identifiers.

A schematic entity and a schematic relationship may additionally contain at most one *anchor* that holds a single mark (e.g., see Figure 3). Anchors are representing with the function $anchor : \mathcal{E} \cup \mathcal{R} \rightarrow \mathcal{M} \cup \{null\}$, where $\mathcal{M}$ is the set of possible marks. The *anchor* function returns $null$ if the identifier does not have an anchor. Anchors serve to "locate" an entity in an underlying source, when such a location is appropriate.

Attributes of an entity type can be specified as either *required* or *optional.* A required attribute implies that corresponding entities have non-null values for the attribute; optional attributes permit null values. Relationship types can specify minimum (required or optional) and maximum (*one* or *many*) cardinalities for each of their roles. Figure 4 defines a portion of the appeal decision schematic of Figure 2.



$\mathcal{U} = (\{AppealDecision\}, \{Appeal, Appellant, Decision\}, \{request, cover\})$.
$AppealDecision = \{Appeal, Appellant, Decision\}$.
$Appeal = \{num\}\ [key = \{num\}]$.
$Appellant = \{name, role\}\ [partialkey = (\{name\}, request)]$.
$Decision = \{loc, issuer, desc\}$.
$request = \{(null, Appeal), (null, Appellant)\}$.
$cover = \{(null, Appeal), (null, Decision)\}$.
$dom(Appeal, num) = String$.
$dom(Appellant, name) = String$.
$dom(Appellant, role) = String$.
$dom(Decision, loc) = String$.
$dom(Decision, issuer) = String$.
$dom(Decision, desc) = String$.

Fig. 4. Part of the appeal decision schematic described using the model. Note that only keys and partial keys are assumed to be required

2.2 Marks and Constraints

A key feature of superimposed schematics is that it integrates marks into an E-R style model. We examine here the effect of marks on type conformance checking and on the defnition of keys.

To check whether an entity conforms to its corresponding entity type, we must take into account marks, since attributes can have marks as values. We require that all marks support the function *excerpt* : $\mathcal{M} \rightarrow \mathcal{V}$, where $\mathcal{V}$ is the set of all possible values. The default type of a mark's excerpt is the uninterpreted type. If an attribute has a mark as a value and the attribute's domain is not the uninterpreted type, we must *interpret* the type of the excerpt to check conformance. Note that this interpretation is done when the mark is assigned as the attribute value. There are a number of ways in which an excerpt can be interpreted, for example:

1. The base layer may support a function for supplying the excerpt's type (e.g., whether it is a text string, image, or some other type).
2. The base layer may be able to provide a value with a known type to represent the excerpt (e.g., a text string representing the text in an image).
3. An external or user-defined function could be used (e.g., to convert the excerpt into a structured value).
4. The creator of the mark could provide a typed value to represent the excerpt.

In most cases, the type of an excerpt will be a simple value like a string, which is the case for appeal decisions.

Once an excerpt's type has been interpreted, we test conformance in the normal way. That is, we ensure all required attributes are present and that each attribute value of the entity conforms to the domain of the corresponding attribute name of the entity type. Relationship conformance is also checked in the typical way, i.e., each entity associated by a relationship is required to have the same type as the corresponding role of the relationship type.

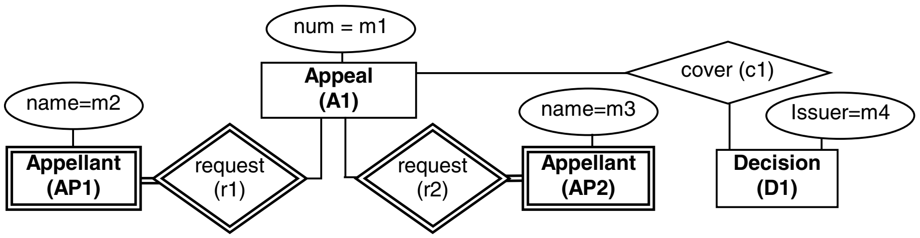
In the schematics model, key constraints on entity types are permitted as are weak entity types with (or without) partial keys. A key constraint is denoted by a set $K \subseteq A$, where A is the set of attribute names for the entity type. An entity type is not required to have a key constraint. Without a key, two entities with different entity identifiers and identical values can exist in the universal extent. A partial key is defined as a pair (K, rt), where K is the partial key and rt is a binary, identifying relationship type. A key definition constrains entities in the normal way, i.e., only one entity of the entity type in the universal extent $\mathcal{D}$ can contain the key value. However, like with entity type conformance, we must take into account marks that are defined to be part of the key. Testing equality between entities with mark-valued attribtues in the key requires determining if two marks (as values for an attribute of an entity) are equal. Normally, the base layer will provide a Boolean-valued function for testing equivalence of marks. However, if no such function exists, (as in conformance) we treat a mark's value as its excerpt's value. This approach considers two entities with identical key values, where at least one of the key values is an excerpt, but with different marks, as identical.

Keys in superimposed schematics have a subtle consequence when mixed with marks. For example, consider the appeal decision schematic of Figure 2. The Reviewing Officer entity type would normally have a key (perhaps its name attribute). However, we want to store exactly one mark (for the officer's name) for each appeal decision that the officer reviews. Further, each such "occurrence" of the officer's name could be spelled slightly differently (e.g., sometimes with a middle initial). We introduce *authoritative entity types* to represent key values in these situations. That is, an authoritative entity type serves as the domain for the key of one or more other entity types. Relationships connect entities to their corresponding authorities (via *authoritative relationship types*) to denote identity and enable queries with unique values.

As an example, we could define a new schematic to represent directory information that contains an authoritative entity type called *Person* (with a keyed attribute called "name"). In this way, we can add a relationship type from the Reviewing Officer entity type to the Person authoritative entity type to relate Officers to their corresponding Person. Thus, we can find all decisions where a particular person served as a Reviewing Officer.

2.3 Schematic Instances

A *schematic instance* consists of the entities and associated relationships that together represent one populated context for a schematic. For example, an appeal decision schematic instance consists of one Appeal Decision entity, its Reviewing



$\mathcal{D} = (\{I_1\}, \{A_1, AP_1, AP_2, D_1\}, \{r_1, r_2, c_1\})$.
$I_1 = \{A_1, AP_1, AP_2, D_1\}$.
$A_1 = (Appeal, \{(num, m_1)\})$.
$AP_1 = (Appellant, \{(name, m_2)\})$.
$AP_2 = (Appellant, \{(name, m_3)\})$.
$D_1 = (Decision, \{(issuer, m_4)\})$.
$r_1 = (request, \{(null, A_1), (null, AP_1)\})$.
$r_2 = (request, \{(null, A_1), (null, AP_2)\})$.
$c_1 = (cover, \{(null, A_1), (null, D_1)\})$.
$excerpt(m_1) =$ "00-06-0035-15".
$excerpt(m_2) =$ "John Rancher".
$excerpt(m_3) =$ "Gregory J. Dyson".
$excerpt(m_4) =$ "Earl W. Ford".

Fig. 5. Part of a populated appeal decision schematic. Note that each entity must be associated with an entry point (i.e., an Appeal Decision entity for this example)

and Deciding Officers, and a set of Appeals, each with a Decision, Appellants, and Issues with Determinations. A schematic can have zero or more instances. Part of a schematic instance for appeal decisions is shown in Figure 5.

To identify schematic instances, we introduce the notion of an *entry point*. Any entity type with a key can serve as an entry point. For entity type ET designated as an entry point in a schematic s, each schematic instance of s must contain *one* and *only one* entity of type ET. Entry points serve as schematic-instance keys—they allow schematic instances to be uniquely identified. For example, the Appeal Decision entity type serves as an entry point, thus allowing the *id* attribute (of the Appeal Decision entity type) to uniquely determine an instance of the appeal decision schematic.

We require every schematic to define at least one entry-point entity type. Whenever an entity whose type is an entry point is created, a new schematic instance is defined. Alternatively, each newly created entity whose type is not considered an entry point must be explicitly placed within a schematic instance. For example, when a new Determination entity is created, an existing Appeal Decision entity is given (using an *id* attribute value), and the Determination is then placed in the corresponding schematic instance. Note that we do not require every entity in a schematic instance to be related to another entity. Finally, we define the function $instance : \mathcal{E} \cup \mathcal{R} \rightarrow \mathcal{I}$, which takes an entity or relationship identifier and returns the identifier's corresponding schematic instance.

3 Schematic-Based Addressing

To address selections in underlying documents, superimposed schematics use marks. However, there are also cases where marking into a schematic is desired. Consider Figure 6, which shows a virtual document of selected issues, excerpted from a set of appeal decisions. Each excerpt is displayed along with a mark to the corresponding issue to enable user navigation into the associated schematic instance (which can then be further navigated, e.g., to find determinations or to see who the appellants are). In general, marks into schematics permit additional layers of superimposed information. This section describes our approach for supporting marks into superimposed schematics.

3.1 Opaque versus Transparent Marks

A mark is an address to an information selection made within a particular context. We require all marks to support the operation *resolve*, which dereferences a mark by opening and (possibly) highlighting the corresponding information selection. For example, for a particular issue in an appeal decision, we can call *resolve* on its anchor to view the appropriate sentence(s).

Just as an underlying document is considered a context, we also view a schematic instance as a context. For example, a schematic instance for an appeal decision groups exactly the information necessary to represent a single appeal decision (even though there may be many underlying documents). For marks into schematics, *resolve* should open the associated schematic instance and highlight the selection (i.e., entity, relationship, or attribute). In cases where multiple levels of superimposed information exist, we may also wish to have finer control over *resolve*, e.g., to resolve a mark into a schematic by unnesting the mark until we reach an anchor into a base layer document. We provide the *unnest-mark* : $\mathcal{M} \rightarrow \mathcal{M}$ function, which takes a mark and returns the mark it refers to (e.g., the entity or relationship anchor) or *null* if no such mark exists.

We consider two types of marks, *opaque* and *transparent*. An opaque mark contains an address that is application specific. For example, an address generated by MS Excel or MS Powerpoint is an opaque mark. By transparent mark, we mean marks whose addresses are semantically meaningful. Common examples of transparent marks include URLs and XPath/XPointer addresses. An advantage of transparent marks is that they can be created and usefully examined out of context. Also, opaque marks can only be created within the associated base-layer application, whereas transparent marks can be created without any application intervention, i.e., they can be created in the superimposed layer.

Our goal is to define an addressing scheme over schematics to support transparent marks. We take a conservative approach, namely, we restrict an address to refer to a single entity, relationship, or attribute. We impose this constraint since the model for schematics does not support arbitrary collections, e.g., we support single valued attributes (not multi-valued) and implied collections exist only as schematic instances and the universal schematic extent.

3.2 Addressability Constraints

Depending on the structure of a schematic, not all entities, relationships, and attributes can always be uniquely addressed. To determine which items can be transparently addressed, we introduce the *schematic addressability constraint.* Intuitively, to address an entity it must be identifiable. An entity cannot be uniquely identified if it does not have some form of key or is not reachable via a unique path from a keyed entity. This situation also applies to relationships, since a relationship is identified by the entities it associates. Only *identified* entity types are transparently addressable. We recursively define an identified entity type as follows:

1. An entity type is an identified entity type if it has a key (which includes entry points).
2. A weak entity type is an identified entity type if it has a partial key.
3. An entity type $e1$ is an identified entity type if it uniquely participates with an identified entity type $e2$. (By uniquely participates, we mean that there is a binary relationship type that associates $e1$ through role $r1$ to $e2$ through role $r2$ such that $r1$ has a maximum cardinality of one.)

To be a *completely addressable* schematic, every entity, relationship, and attribute must be transparently addressable. To be completely addressable, we must guarantee that all entity types be identified (via the addressability constraint). Second, we must guarantee that all entities with entity types that are identified only through a unique participation relationship type (part 3 above) are associated by such a relationship. In this way, we can uniquely address every entity through either its key, its identifying entity's key and its partial key, or through its unique participation entity's key (since the relationship is guaranteed to exist).

We do not require schematics to be compeletely addressable nor do we require that the addressability constraint be met. For some applications, enhanced addressing will not be a priority, e.g., applications focused on navigation and browsing. Opaque addresses, however, can always be defined for entities and relationships by using their identifiers. Therefore, all entities, relationships, and attributes can be (at least) opaquely addressed. Note that the appeal decision schematic in Figure 2 does not satisfy the addressability constraint since the *Decision* entity type is not an identified entity type. However, if *Decision* were an identified entity type, the schematic would satisfy the addressiblity constraint.

Finally, all marks are required to support the *excerpt* function, therefore, we must consider excerpts for marked entities, relationships, and attributes. For an attribute, we can use its value, or if it has a mark, its mark's excerpt. For an entity or relationship, the anchor's excerpt can be used (if such an anchor exists).

3.3 The Schematic Addressing Language

A transparent schematic address P is a list of address components p_i for $1 \leq i \leq n$ such that P has the form $p_1.p_2. \; \ldots \; .p_n$ (see Table 1 for examples). We distinguish between three kinds of schematic addresses, namely, entity addresses, attribute addresses, and relationship addresses, each of which are defined below.

Entity Address A valid entity address is defined recursively as:

1. *Key Entity.* The address $et(\kappa)$ is a valid address if et is an entity type with key K and κ is a key value of the form $k_1 = v_1, k_2 = v_2, \ldots, k_m = v_m$ for all $k_i \in K$ and $i \geq 1$ and $m = \mid K \mid$. The type of the entity addressed is et.
2. *Partial Key Entity.* The address $P.rt(\kappa)$ is a valid address if P is a valid entity address for an entity of type $et1$, rt is an identifying relationship between $et1$ and $et2$, and κ is a partial key value for $et2$. The type of the entity addressed is $et2$.
3. *Unique Participation Entity.* The address $P.rt$ is a valid address if P is a valid entity address for an entity of type $et1$ and rt is a unique participation relationship from $et1$ to $et2$. The type of the entity addressed is $et2$.

Relationship Address A valid relationship address is defined recursively as:

1. *Single Participation Address.* The address $P.rt\#$ is a valid address if P is a valid entity address of type et and rt is a binary relationship type connecting et to et' such that the role et' plays has a maximum cardinality of one.
2. *Multiple Participation Address.* The address $P.rt\#[P']$ is a valid address if P is a valid entity address of type et, rt is a binary relationship type between et and et' such that the role played by et' in rt has a cardinality of *many*, and P' is a valid entity address of type et'. Intuitively, to identify the *many* end of a many-to-many or one-to-many relationship type rt, we must identify the entity that the corresponding relationship connects.
3. *Non-Binary Address.* The address $P.rt\#/role_1[P'_1]/\ldots/role_m[P'_m]$ is a valid address if P is a valid entity address of type et, rt is an n-ary (for $n > 2$) relationship type from et to a list of entity types $[et_1, et_2, \ldots, et_m]$, $role_i$ for $1 \leq i \leq m$ is the role played by et_i, and P_i is an entity address of type et_i.

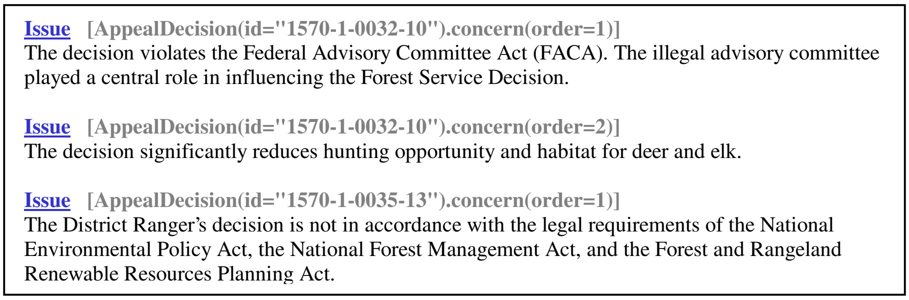
Attribute Address The address $P.a$ is a valid address if P is a valid entity address of type et and a is a valid attribute name for et.

To determine the schematic instance of an entity or relationship address, e.g., for implementing the *resolve* operation, we can use the *instance* function defined in the previous section. For attribute addresses $P.a$, we can call *instance* on P to determine the schematic instance. Examples of transparent addresses over the Appeal Decision schematic are given in Table 1.

Finally, Figure 6 shows an example virtual document serving as a new layer of superimposed information over the Appeal Decision schematic. The virtual document uses enhanced addressing—it stores excerpts of issues from schematic instances as well as transparent marks to Issue entities. The virtual document demonstrates a notional example of query, i.e., the virtual document results from

Table 1. Example enhanced addresses. Note constants x and y represent "1570-1-0032-10" and "00-06-0032-10," respectively, and `AD` stands for `AppealDecision`

Address	*Description*
`AD(id=`x`).concern(order=1)`	Address to an *Issue* entity.
`AD(id=`x`).concern(order=1).desc`	Address to an Issue *desc* attribute.
`AD(id=`x`).response#`	Addres to a *response* relationship.
`AD(id=`x`).result#[Appeal(num=`y`).define(order=1).resolve]`	Address to a *result* relationship (identified via its associated *Issue*, as shown in brackets).


Issue [AppealDecision(id="1570-1-0032-10").concern(order=1)]
The decision violates the Federal Advisory Committee Act (FACA). The illegal advisory committee played a central role in influencing the Forest Service Decision.

Issue [AppealDecision(id="1570-1-0032-10").concern(order=2)]
The decision significantly reduces hunting opportunity and habitat for deer and elk.

Issue [AppealDecision(id="1570-1-0035-13").concern(order=1)]
The District Ranger's decision is not in accordance with the legal requirements of the National Environmental Policy Act, the National Forest Management Act, and the Forest and Rangeland Renewable Resources Planning Act.

Fig. 6. A virtual document with enhanced addresses to corresponding Issues

the schematic query: "List all Issues concerning Decisions made in the Mount Hood National Forest within the last two years."

4 Related Work

Superimposed models differ in how they structure information. A number of untyped, hypertext models (such as HTML) are unstructured. The bundle/scrap model [6] along with most annotation models [14,16] provide very simple structures for organizing superimposed data. For example, in the bundle/scrap model, marks are held in named "scraps" and can be placed into (possibly) nested, named bundles. Semi-structured models are the most prevalent superimposed models, and include XLink [7] (for XML), RDF [10], Topic Maps [2], and hypertext models for emergent structure [11]. Superimposed schematics are richly structured, and to our knowledge the only approach based on the E-R model, which offers a standard, powerful conceptual data model. Other structured models include Structured Maps [4] (a simple, typed version of a Topic Map), typed hypertext systems [13], and models for wrappers [1,9,15].

Superimposed models also differ in their support for marks. Opaque marks are derived from the mark architecture in the bundle/scrap model [6], and support fine-grain, sub-document marks to arbitrary sources. Multivalent Documents [14] provide a document model (that various document formats are mapped to) for robust marks, but marks must be resolved using a special viewer.

Topic Maps, RDF, and HTML support marks through URLs, which are limited for sub-document addressing (using HTML document fragments). Although XPath/XPointer addresses can be used, they only support marks into XML documents. Schematics support transparent marks, similar in spirit to the motivation for XPath/XPointer. However, XPath/XPointer addresses are based on document structure (since they address XML) as opposed to conceptual structure. Note that wrappers offer a limited form of superimposed information since they support new layers of information, but not marks.

The constructs used for linking (i.e., marking) also differentiate superimposed models. Unique to the bundle/scrap model and superimposed schematics is the notion of context to separate layers of information. That is, a mark is a separate modeling construct used to navigate into a different context (both conceptually and operationally). While a number of approaches [2,4,7,10,12,14,16] store superimposed information separately from the base layer, they usually operate as if both layers were within the same context. For example, Structured Maps, XLink, typed hypertext [13], and most Topic Map applications integrate superimposed information into a single hypertext (using HTML).

Finally, some superimposed models support the use of excerpts, e.g., with virtual documents [12]. We have also encountered examples of special purpose, hard-wired schematics, such as in the Distributed Annotation Server [8] in which sequence "landmarks" are used to attach annotations to genetic information.

5 Summary and Future Work

Superimposed schematics contribute a rich conceptual layer for representing and elaborating exerpted information from underlying documents, while maintaining context through marks. We have presented a data model for schematics based on E-R structures with support for marks, multiple schematic instances (via entry points), and an accompanying addressing language.

We have implemented a browser for superimposed schematics, which can be used to navigate schematic instances and underlying information. We intend to enhance the browser for mixing navigation and query. We believe simple techniques can be used to help populate schematics semi-automatically. In addition, we believe integrating schematics into a document editor to tie population with actual document creation would also be useful. Finally, we intend to further explore enhanced addressing. Currently, we are working towards a model for layers of schematics over appeal decision packets. Appeal packets are used by deciding officers and others to assess an entire project, its issues, alternatives, and so on. We also wish to explore using enhanced addressing for standard E-R schemas, possibly to support tracking data lineage.

References

1. S. Abiteboul, S. Cluet, and T. Milo. Querying and updating the file. In *Proc. of the 19th VLDB Conf.*, pp. 73–84, Aug. 1993. 102
2. M. Biezunski, M. Bryan, S. Newcomb, eds. ISO/IEC 13250, *Topic Maps*, 2000. http://www.ornl.gov/sgml/sc34/document/0058.htm. 102, 103
3. P. P. Chen. The entity-relationship model—toward a unifed view of data. In *ACM Trans. on Database Systems*, 1(1):9–36, 1976. 93
4. L. Delcambre, et al. Structured maps: Modeling explicit semantics over a universe of information. In *Intl. Journal on Digital Libraries*, 1(1):20–35, 1997. 91, 102, 103
5. L. Delcambre and D. Maier. Models for superimposed information. In *Advances in Conceptual Modeling: ER '99 Workshop on the WWW and Conceptual Modeling*, LNCS 1727, pp. 264–280, Nov. 1999. 91
6. L. Delcambre, et al. Bundles in captivity: An application of superimposed information. In *Proc. of the IEEE Intl. Conf. on Data Eng. (ICDE)*, pp. 111–120, Apr. 2001. 91, 102
7. S. DeRose, E. Maler, D. Orchard, B. Trafford, eds. XML Linking Language (XLink), W3C Working Draft 21-Feb-2000. http://www.w3.org/TR/2000/WD-xlink-20000221. 102, 103
8. R. D. Dowell, et al. The distributed annotation system. In *BMC Bioinformatics*, 2:(7), Oct. 2001. http://www.biomedcentral.com/1471-2105/2/7 103
9. D. W. Embley, et al. A conceptual-modeling approach to extracting data from the web. In *Proc. of the 17th Intl. Conf. on Conceptual Modeling*, LNCS 1507, pp. 78–91, Nov. 1998. 102
10. O. Lassila and R. R. Swick (eds.). Resource Description Framework (RDF) Model and Syntax Specification, W3C Rec. 22-Feb-1999, http://www.w3.org/TR/REC-rdf-syntax. 102, 103
11. C. C. Marshall, F. Shipman, and J. Coombs. VIKI: Spatial hypertext supporting emergent structure. In *Proc. of the European Conf. on Hypertext Technology (ECHT)*, pp. 13–24, Sept. 1994. 102
12. S. H. Myaeng, et al. A digital library system for easy creation/manipulation of new documents from existing resources. In *Proc. of the 6th Conf. on Content-Based Multimedia and Information Access (RIAO)*, Apr. 2000. 103
13. J. Nanard and M. Nanard. Should anchors be typed too? An experiment with MacWeb. In *Hypertext'93 Proceedings*, pp. 51–62, Nov. 1993. 102, 103
14. T. Phelps and R. Wilensky. Multivalent documents. In *Communications of the ACM*, 43:(6):83–90, June 2000. 92, 102, 103
15. A. Rajaraman and J. D. Ullman. Querying websites using compact skeletons. In *Proc. of the 20th ACM Symposium on Principles of Database Systems*, pp. 16–27, May 2001. 102
16. M. Rosheisen, C. Mogensen, T. Winograd. Shared web annotations as a platform for third-party value-added information providers. STAN-CS-TR-97-1582, Stanford Integrated Digital Library Project, Nov. 1997. 102, 103

Web extensions to UML: Using the MVC Triad

David Lowe, Brian Henderson-Sellers, and Alice Gu

University of Technology, Sydney
PO Box 123 Broadway NSW 2007, Australia
david.lowe@uts.edu.au

Abstract. Current Web Modelling Languages (WMLs) fall short of the requirements for the modelling of web system development. In particular, those WMLs with a hypermedia basis are more closely focussed on the information architecture whereas software WMLs are more focussed on the functional architecture. Generally, modelling languages have failed to bridge the gap between these two areas, nor do they handle well the connection between different levels of abstraction and are largely unable to connect well with business models. Based on an analysis of existing modelling approaches, we propose a conceptual extension to modelling approaches that attempts to address these limitations. We show how it can implemented using UML modelling along with the addition of concepts taken from Web information modelling approaches, WebML in particular. The extensions are structured around the model-view-controller concept, which we argue provides an appropriate integrating modelling framework. We begin by discussing the scope and objectives of the extensions, followed by the extensions themselves. We then illustrate the extensions using a small case study.

1 Introduction

Current modelling languages for web system development were examined by Gu *et al.* [1] against a set of developed criteria or requirements for a Web Modelling Language (WML)[1]. Of the six current WMLs examined, some with hypermedia roots, others extensions to OO modelling languages such as the UML [2], all were found deficient in part. This is because those WMLs with a hypermedia basis are more closely focussed on the information architecture whereas modelling-derived WMLs are more closely focussed on the functional architecture. The spread of WMLs along these two important axes is shown in Figure 1, where it is seen that there are no candidates which address both the informational and functional architectures concurrently. This paper attempts to move WMLs towards this “target zone”.

[1] We acknowledge that the acronym WML is used in other situations: in the XML community for Wireless Markup Language and as the name for a freely distributed web tool for HTML generation. Here we use it simply as an abbreviation for web modelling language.

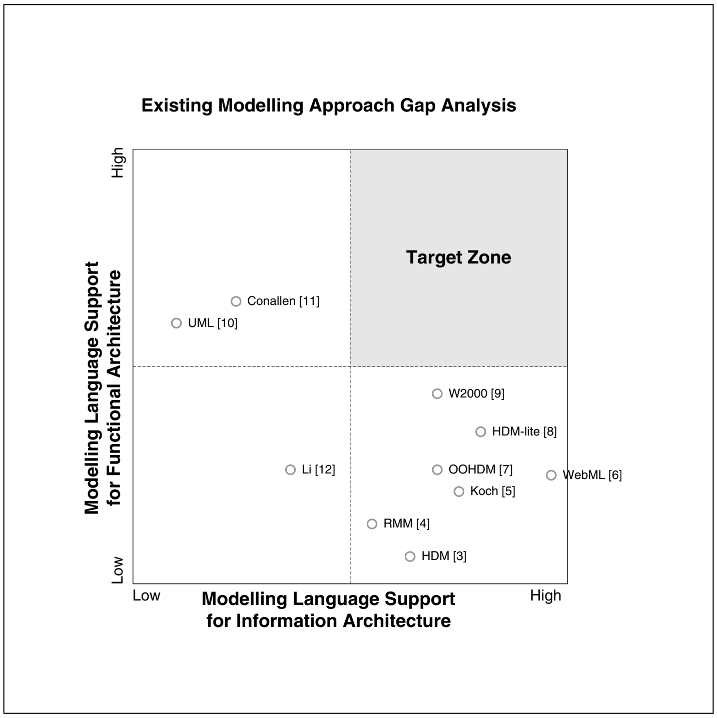


Fig. 1. Existing Modelling Approach Gap Analysis (after [1]).

Firstly, the scope and objectives of the extension are discussed (Section 2), followed by the proposed model extensions (Section 3) and UML diagram extensions (Section 4). These extensions are then discussed in terms of how they might be used in different design components of the overall software development process (Section 5). We also illustrate throughout the paper the key aspects (though not many of the details) of the proposed extension model structure and UML notation extensions using a simple case study based on an existing system. This system facilitates the online enquiry of policies, procedures and issues that apply to students who attend specific University courses. The system allows searching of issues and policies by category or keyword, and displaying of selected issues and policies. For the purpose of this paper, we will call this system UTSE-Guide.

2 Scope, Objectives and Approach

As discussed in [1], limitations and flaws can be identified in the existing modelling approaches that are currently used to develop Web systems. Some of the gaps, such as the inability to support system life cycle management and the potential misuses of UML extension mechanisms, need to be addressed in separate research projects and are beyond the scope of this present paper. Here we address the issues relating to the need to model increasingly sophisticated functionality and, most importantly, to integrate the functional architecture with the informational architecture.

The proposed extensions make use of UML extension mechanisms. The normal way to extend the UML is by the use of stereotypes. "A stereotype is not the same as a parent class in a parent/child generalization relationship. Rather, you can think of a stereotype as a metatype, because each one creates the equivalent of a new class in the UML's metamodel." [10, p80]. In other words, stereotypes extend the UML (M2) metamodel indirectly at the M1 (or model) level, wherein the relationship between a class and its stereotype is an "instance_of" rather than an "is_a_kind_of" relationship [13]. Whilst this extension mechanism makes it easy for users to extend the UML notation, it may introduce confusion and semantic problems because the (mis)use of the inheritance relationship in a stereotype does not truly reflect the intended instantiation relationship that is used in direct metamodelling [13–16]. However, despite the inherent problems of the current UML stereotype concept, for the purpose of this project and this paper, we will still use stereotypes as the extension mechanism, mostly due to the support given to them by CASE tools. To further improve the proposed extensions, direct modification to the UML metamodel may need to be considered as an alternative (see e.g. [17]).

The objectives of our proposed extensions are twofold. First, to address the deficiencies identified in the the gap analysis reported in [1]. These issues include the inability to model sophisticated functionality, the disconnection between the information architecture and the functional architecture, the disconnection between the business model and the technical architecture and the inability to support modelling at various abstraction levels. The second objective is to ensure the integrity of the resultant Web system architecture, whilst merging the business model with the functional and information aspects of the technical architecture and representing system structure at different abstraction levels.

Based on the analysis of the existing modelling approaches, we propose a UML extension with information modelling concepts taken from other modelling approaches, WebML in particular [6]. This option is chosen for several reasons. First, the UML notation is commonly used and accepted, and provides reasonable support for system functional architecture modelling. Second, approaches from a hypermedia background demonstrate reasonably rich and balanced support for the information architecture. Amongst them, WebML is a recent attempt that provides modelling capabilities for most of the critical aspects of Web system information architecture.

3 Extension Model Structure

To ensure the architectural integrity of Web systems, we propose an extension model structure that supports the integrated modelling of both information and functional architectures, and which is based on the MVC concept.

3.1 The MVC Concept

“The model-view-controller architecture, often known just by the letters MVC, has been a feature of Smalltalk since Smalltalk-80. It is based on the concept of separating out an application from its user interface” [18, pp266]. The responsibilities of model, view and controller are described as follows:

- Model — the information model that handles data storage and information processing. It manages the behaviour of the data in the application domain.
- View — handles how the information is displayed visually, which is the interface part of the system.
- Controller — provides user interaction to, or control of, the information models.

From its Smalltalk roots, the MVC concept has recently gained more recognition and been applied to the design level, such as in the design patterns in J2EE [19]. It should be noted that MVC is typically used as a specific architecture rather than a broader modelling framework and, as such, there may be concerns over the extent to which this limits its applicability to modelling a broader range of applications and systems. This issue is recognized but not addressed further in this paper.

A thorough study of the existing modelling approaches, especially the ones that support the information architecture reasonably well, such as OOHDM [7] and WebML [6], demonstrates that the separation of modelling entities in the conceptual model and the interface entities in the presentational and navigational models has been used to provide advantages to these well-established approaches, which include:

- Better understanding of system architectural issues brought by the separation of concerns; and
- The possibility of both flexibility and personalization provided by the match of the conceptual model with different presentational or navigational models.

Whilst we believe that these approaches can provide reasonable support for information architecture modelling, the functional architecture aspect is normally weak or even absent in these approaches. We therefore propose the extension model structure — a modified MVC architecture as shown in Figure 2 — which is explained in detail in the following three subsections.

3.2 Extended Conceptual Model

The extended conceptual model contains three types of elements:

- Model — the normal model element that represents business entities;
- View — defines the composition of models at the interface level; and
- Controller — defines object behaviours. These behaviours can occur either at the interface level, which are navigational behaviours, or at the back-end, which are system functions.

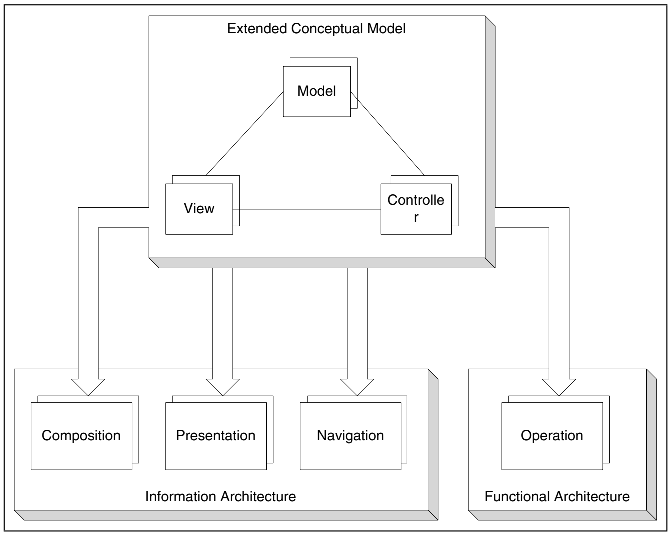


Fig. 2. Extension Model Structure

For example, business or application domain entities, such as "student" and "product", are models in this extended conceptual model. At the interface layer, though, these entities can be displayed in various forms. When "student" and "product" are displayed in the form of list, they can be view "List"; they can also be displayed as view "DataUnit" if detailed information is required. An example of controller is the index used to sort the "student" or "product" list views. If the "product" List needs to be displayed by category, then an "Index" controller "index by category" can be connected to the view to fulfil this requirement.

There are multiple models, views and controllers in the extended conceptual model. Whilst each model represents a business entity, the views and controllers, which are connected to the models, represent and specify the interfaces and behaviours of the models. The design of information and functional architectures is performed on the foundation of the extended conceptual model.

3.3 Information Architecture

Various aspects of the information architecture are modelled using different modelling artefacts. These aspects include:

- Composition: The model structure in terms of model, view and controller.
- Presentation: The interface level concepts. Presentation is based on the views in the composition model.
- Navigation: The navigational structure and behaviour. Navigation is performed by controllers. When activated, controllers pass control to one another and the application therefore flows from one part of the system to another. The result of navigation is the change of views.

3.4 Functional Architecture

Operations need to be modelled in the functional architecture by:

- Using existing UML concepts and diagrams, such as the statechart diagram and the sequence diagram.
- Using views and controllers defined in the extended conceptual model. For example, if, when activated, instead of passing control to another controller in the same or another view, the controller passes control to a controller that performs back-end functionality, then an operation is invoked. The end result of an operation can be a state change in the system, either at the user interface level or at the back-end level.

Due to the lack of sophistication in the UTSE-Guide system, no complex operation needs to be modelled separately using operation diagrams in this example. In fact, some simple operations are actually specified in the navigation diagrams.

4 UML Diagram Extensions

We propose several additional diagrams that are needed in the UML notation to support the proposed model structure. These diagrams are shown in Figure 3. We also introduce three new stereotypes of Classifier: «view», «controller» and «presentation». Of the diagrams shown in Figure 3, several need no modification (Use Case, Activity, Collaboration, Sequence, Statechart, Component and Deployment Diagrams). Extensions to existing diagrams and newly introduced diagrams are documented as following:

Conceptual Model

Extension Needs: This model uses those concepts from the problem domain that are independent of the technologies used to build the system. The proposed concept of extended conceptual model links these basic conceptual entities (models) to views and controllers.

Extensions: Although the concept of extended conceptual model is introduced, it does not necessarily mean that new modelling artefacts need to be defined in the conceptual model; instead, it only implies that the conceptual model contains models and their corresponding views and controllers, from a semantic perspective. The relationship between model, view and controller can be represented in the composition diagrams, which will be discussed later. The conceptual model for the case study, including the core business entities in UTSE-Guide and the relationships between them, is shown in Figure 4.

Composition Diagram

Extension Needs: This diagram does not exist in the UML. Coming from WebML, it covers one aspect of information architecture of Web applications.

Extensions: Stereotypes «view» and «controller» are defined on the UML class diagram. «view» is defined in order to show different components of

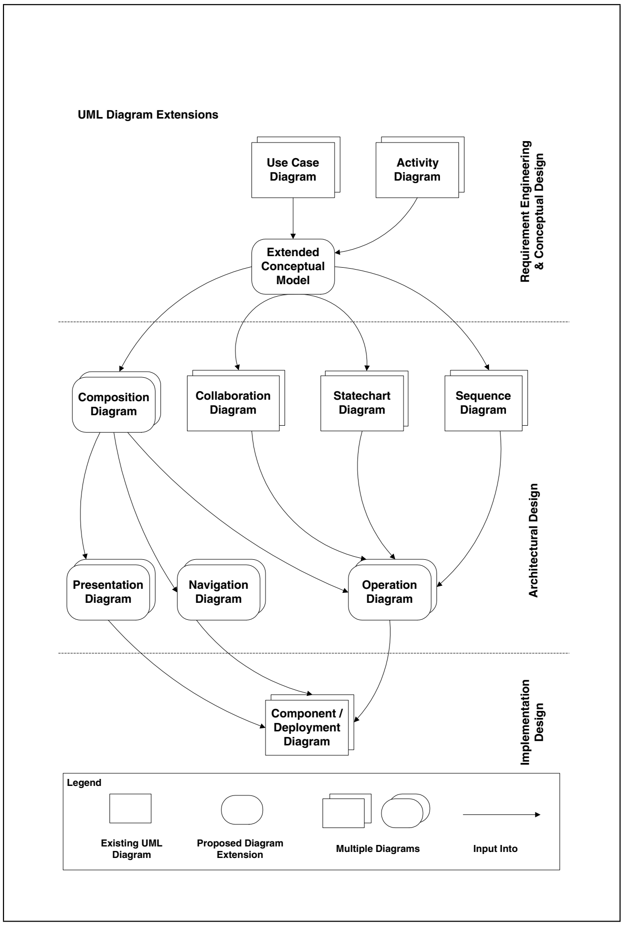


Fig. 3. UML Diagram Extensions

the interface at a relatively high level. «view»s can contain other «view»s, in which case it makes up a view – sub-view hierarchy. «view» or sub-«view» can be a Web page, part of a Web page, or a combination of several Web pages. «view»s can contain not only other «view»s, but also «controller»s. «controller»s perform functions — navigations or operations, either on the interface, or behind the screen. The «controller»s can be further defined as different classes e.g. Index, Filter, DataUnit, and Operation. Controller can navigate from one «view» to another, either in a contextual or non-contextual manner. When a «controller» initiates a function from the interface, it can then activate other «controller»s, either on the client side or on the server side. At the end of a function, control can be passed back to a «controller» on the interface, i.e., a DataUnit «controller», so that the

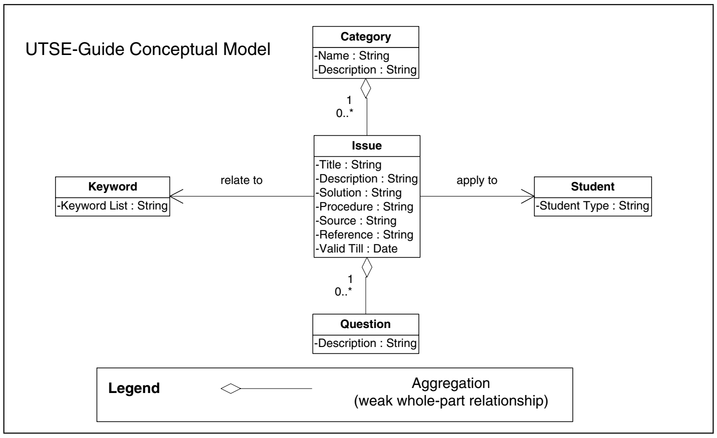


Fig. 4. Conceptual Model

user can interact with the system again.

The high level structure of the UTSE-Guide is shown in Figure 5. This diagram is called composition in-the-large. From an information structure perspective, the UTSE-Guide system can be stereotyped as a «view» class, with the two major components also represented as «view»s: Issue Search and Issue Display. There is a strong connection and coincident lifetime of the parts (the sub-«view»s) with the whole (the «view»). This high level composition demonstrates the information structure of the system at an abstract level, thus providing the developers with the big picture. To further specify system composition at a more detailed level, diagrams can be constructed to document the composition in-the-small. For example, with the UTSE-Guide system, the diagram would document the composition of the component "Issue Search" at a more detailed level, i.e., Web page level. View "Issue Search" is further broken down into views with associated controllers. There are various types and interconnections of views and controllers: Unactionable views (which simply display fields at the user interface layer, cannot be activated to trigger any navigation or operation and do not therefore link to controllers); Actionable view and controller pairs (a view-controller pair can provide navigational and/or operational capability); view and sub-views.

Presentation Diagram

Extension Needs: This diagram does not exist in UML and needs to be defined to support the representation of interface-level modelling, such as the components that make up a Web page.

Extensions: «presentation» elements, such as Page, Filter, DataUnit and Button, are defined to show screen mock-ups. These elements specify the presentational aspects of «view»s. «view»s are at a higher abstraction level than «presentation» elements, and represent a selection of data from the extended conceptual model. Instead of using the existing UML diagrams, the

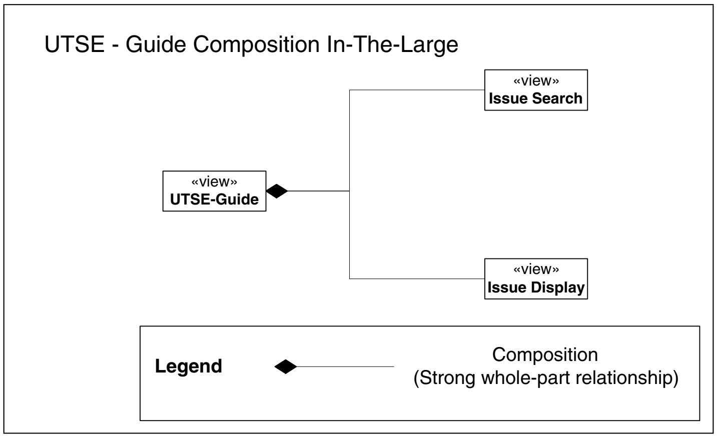


Fig. 5. Composition In-The-Large for UTSE-Guide

presentation diagram is defined as a new type of diagram. This is largely due to the lack of support for presentation level modelling in the existing UML notation. Modification and personalization can be done by matching different «presentation» elements to the same «view»s. Ideally, presentation diagrams should be generated automatically by a CASE tool, according to the semantics in the composition diagrams. Users can modify them manually if required. Style is defined as a class with the stereotype «presentation» to show the style or format on the interface level. It can be generic and linked to, and indeed reused by, many «presentation» elements; or it can also be specific and used to define particular presentational characteristics of some individual «presentation» elements. By defining a number of different «presentation»s for each «view» and defining and linking a number of Styles to each «presentation», flexibility and personalization can be achieved in Web system design.

A presentation diagram of the UTSE-Guide system is shown in Figure 6. It specifies the interface layer definition of the system. The presentational elements map to «view»s, and thus define the composition and format of «view»s when they are implemented in the user interface. The flexibility of mapping each «view» to a number of «presentation»s provides the capability to implementation personalization, and also makes future change to the system's presentational aspect relatively independent to the core architecture of the systems.

Navigation Diagram

Extension Needs: This diagram does not exist in the UML and needs to be defined.

Extensions: At a high level, navigations are performed by moving from one «view» to another. For each user or user group, navigation diagram(s) can be defined to show to which «view»s the user has access and how they are

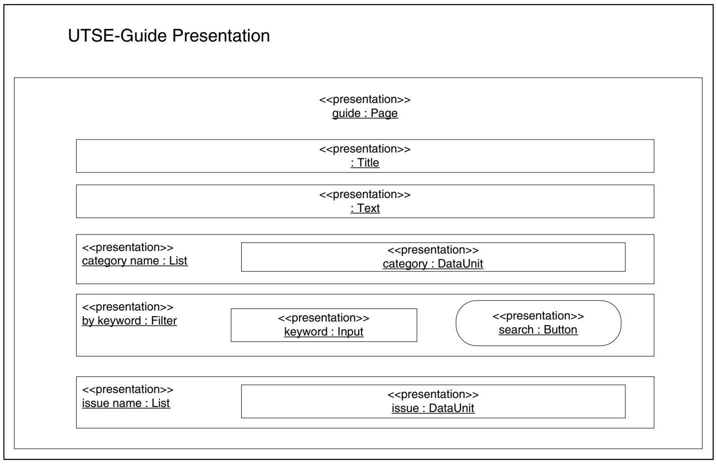

UTSE-Guide Presentation
<<presentation>> guide : Page
<<presentation>> : Title
<<presentation>> : Text
<<presentation>> category name : List
<<presentation>> category : DataUnit
<<presentation>> by keyword : Filter
<<presentation>> keyword : Input
<<presentation>> search : Button
<<presentation>> issue name : List
<<presentation>> issue : DataUnit

Fig. 6. Presentation Diagram

interconnected. At a lower level, navigations are performed through «controller»s. When a user invokes a «controller», the display changes, either by going to another «view» (i.e., another page) or to another part of the «view» (i.e., goes to the Top). The «controller»s can carry contextual information. When a «controller» is invoked to perform an operation, the user "loses" control of the system. The «controller» either performs a function itself or consequently activates another «controller» to complete the function. Once the function is finished, control can be passed back to an interface level «controller» and the user regains the control over the system.
The high level navigation of the UTSE-Guide system is shown in Figure 7. At a high abstraction level, the navigational structure of the UTSE-Guide system can be represented as the connections of the "Issue Search" «view» and "Issue Display" «view». Users can navigate from «view» "Issue Search" to «view» "Issue Display" by invoking «controller» "filter by issue" from «view» "Issue Search". The navigational behaviour is performed when the «controller» "filter by issue" is activated. As shown in Figure 5, the execute model of this «controller» is "invoke". This means that the «controller» will not be activated by the system automatically at times such as initiation; rather, it needs to be invoked by a user during interaction with the system.
A low-level specification of UTSE-Guide navigation would be shown in navigation in-the-small diagram. Three types of elements would be included in this diagram:

- «view»: A navigational behaviour results in the change of «view»s. In the scenario studied here, "Issue Display" «view» is displayed to replace "Issue Search" «view».

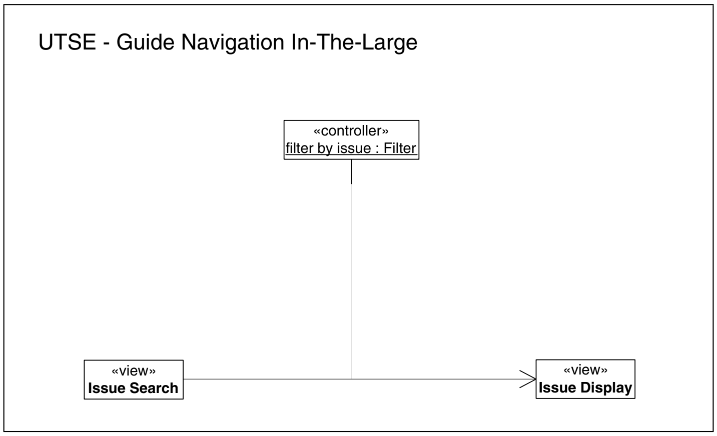


Fig. 7. Navigation Diagram — In-The-Large

- «controller»: As described earlier, the navigational activity is completed by one or many «controller»s, and the interconnections and execution sequence of these «controller»s are specified in the navigation diagram.
- Other resources: Other resources, such as legacy applications, databases, file servers and external Web sites, may be needed to perform the navigation. In this example, the «controller»s interact with "issues" database to obtain the information needed.

Operation Diagram

Extension Needs: Operations are currently modelled in the UML using diagrams such as the collaboration diagram and the Statechart diagram. However, in the proposed model structure, operations also need to be represented using view and controller. This is to ensure that the functional architecture is modelled using the same concepts as in the information architecture, so that the two aspects can be connected in a reliable and consistent fashion.

Extensions: Simple operations can be represented in detailed level navigation diagrams, whilst complex operations may need to be defined in further detail using operation diagrams. In an operation diagram, the flow of operations is represented by the passing of control amongst «controller»s. These «controller»s can reside either on the client side or on the server side.

5 Using the Extended UML Diagrams

In this section, the extended model structure (shown in Figure 3) and UML diagram extensions (described above) will be studied using a partial development process. This does not, in any way, imply that the proposed extensions need to be used in conjunction with any particular process; rather, the process described here is used as an example to demonstrate the usage of the extensions and diagrams during the course of Web system development.

5.1 Requirements Engineering and Conceptual Design

During the requirements engineering and conceptual design stages, business requirements are captured. Business requirements are normally captured using UML use cases and/or activity diagrams. These diagrams are used as input in the creation of the extended conceptual model, which is built from an understanding of the problem domain. The entities in the extended conceptual model are problem domain concepts, not computer system components.

To extend the conceptual model, «view»s and «controller»s are then defined based on the original conceptual model. «view»s define interface composition, at a relatively high abstraction level. «controller»s define object behaviours, either at the interface level or in the back-end. Whilst the concept of the model-view-controller structure is viewed as part of the extended conceptual model, the definition is represented in the composition diagrams, which describes aspects of the information architecture. To support modelling at various abstraction levels, the composition diagrams consist of two types: composition in-the-large and composition in-the-small. One conceptual model can connect to multiple composition diagrams and can therefore support various definitions on the interface and behaviour levels.

5.2 Architectural Design

Once the extended conceptual model is defined, it can be used as the foundation for the Web system architectural design. Other model elements and diagrams can be defined from the extended conceptual model and used to document the system structure at more detailed levels. These design activities may, and often do, occur more or less in parallel. During this stage, some existing UML diagrams, such as the collaboration diagram, the statechart diagram and the sequence diagram can be used to facilitate the modelling process [2]. Additional diagrams are used to better support the modelling of Web-specific aspects — particulary the system information architecture and functional architecture. Presentation diagrams (see Figure 6) are defined from the composition diagrams and show how views are displayed on the interface level. One view can be matched to more than one presentational definition. Personalization at the interface level is then supported.

To represent navigational design, «view»s and «controller»s interconnect with each other. From a navigational perspective, the user navigates between «view»s through the usage of «controller»s. Navigation contains two aspects: the static navigational structure (represented by interconnections between «view»s); and the dynamic navigational behaviour (represented by the connections between «view»s and «controller»s).

Because of the potential complexity of Web system navigation, the representation of navigational structure and navigational behaviour needs to be supported at different abstraction levels, so that a thorough understanding of the navigational aspect can be achieved. This is implemented by using two types of navigation diagrams: navigation in-the-large and navigation in-the-small.

Operational Design Operations are performed by controllers. This can happen either when a user invokes a controller by interacting with its related view, or when the system initiates a function by passing control to a controller. Some functions need only one controller to complete, whilst others require the collaboration between several controllers. In the latter case, control flows from controller to controller during the process of the function. When the control is not with the controller that relates to views at the interface layer.

To perform an operation, other resources such as legacy applications, database files or external links may be required. This is true with navigation as well. A user can navigate from a website to its related links and then return, as part of his/her normal navigational route.

5.3 Potential Improvements

As demonstrated by the case study, the proposed extensions to the UML notation can, if utilized properly, increase the modelling capability of the notation. These proposed extensions can be helpful in addressing the limitations in the existing modelling approaches that were identified in Gu *et al.* (2002). An analysis of this potential is shown in Table 1.

6 Summary

In this paper, we have proposed some extension directions to UML notation, with the aim of improving some aspects of the modelling language support for Web system development. The extensions to the conceptual model are based on the model-view-controller concept and the addition of several diagrams, such as composition diagrams, presentation diagrams, navigation diagrams and operation diagrams, can potentially increase the modelling capability offered by the existing UML notation.

Inspired by concepts taken from other modelling approaches, such as WebML, which support Web system information architecture reasonably well, and used in conjunction with the existing UML functional modelling capabilities, the proposed model structure aims to support both the functional and information architectures. The support for sophisticated functionality and the connection between the business model and the technical architecture is also addressed by the proposal. Furthermore, the proposed modelling approach can represent the system architecture at different abstraction levels and the linkage between these levels can be managed through the extended conceptual model.

A small case study has been used to illustrate the proposed extensions and their applications although, due to the small size and complexity of the case study system, not all aspects and features of the extension proposal could be fully demonstrated. Another, much larger case study has in fact been conducted using a real-world, commercially confidential, Web system, with excellent results.

Table 1. Extension Proposal — Addressing the Gaps

Gap in Existing Approaches	Potential Improvements
Inability to model sophisticated functionality	The introduction of the model-view-controller concept provides the capability to model navigational and functional behaviours using the definitions of controllers. Whilst some simple functions can be performed by individual controllers, more sophisticated functions may need the collaboration of several controllers. Some of these controllers also support the integration to both internal and external applications and information resources.Whilst the functional modelling aspect cannot be fully demonstrated in this paper due to the lack of sophistication of the case study.
Disconnection between functional architecture and information architecture	The introduction of the model-view-controller concept and the extended model structure provides the potential capability to connect the functional architecture and the information architecture. Since the two aspects of Web system architecture are both connected to, and indeed developed from, the extended conceptual model, consistency and integrity are more likely to be achieved in the proposed approach.
Disconnection between business model and technical architecture	In the proposed extension model structure, the extended conceptual model is built upon the business requirements captured during requirements engineering. This extended conceptual model is then used as the foundation for the design of both the information architecture and the functional architecture of the Web system. This close connection introduced by the MVC structure can help to translate business requirements into the two aspects of Web system technical architecture.
Inability to support modelling at various abstraction levels	With the modelling of composition, navigation and operation, diagrams can be constructed at two abstraction levels. For example, the composition of a Web system can be represented as logical elements at a high abstraction level, whilst it can also be modelled in term of pages and elements on the pages at a more detailed level. This is implemented by using the composition in-the-large and composition in-the-small diagrams.Although far from complete and thorough, this approach demonstrates the potential to address the issues related to modelling abstractions and interconnections between the abstraction levels.
Potential misuse of UML extension mechanisms	Although this issue was not directly addressed in our proposal here, attention was given to the proposed UML extensions with the aim of avoiding the potential misuse of this mechanism.
Inability to support system life cycle management	This issue was not directly addressed. However, the proposed support for a more complete and balanced Web system technical architecture can potentially expand the usage of the model during the system life cycle.

References

1. Gu, A., Henderson-Sellers, B., Lowe, D.: Web modelling languages: the gap between requirements and current exemplars. In: Australian Web Conference. (2002)
2. OMG: OMG Unified Modeling Language specification, version 1.3 (released to the general public as OMG document formal/00-03-01 in March 2000) (2000)
3. Garzotto, F., Mainetti, L., Paolini, P.: Hypermedia design, analysis, and evaluation issues. Communications of the ACM **38** (1995) 74–86
4. Isakowitz, T., Stohr, E., Balasubramanian, P.: RMM: A methodology for structured hypermedia design. Communications of the ACM **38** (1995) 34–44
5. Koch, N., Mandel, L.: Using uml to design hypermedia applications. Technical Report 9901, Ludwig-Maximilians-University, March 1999 (1999)
6. Ceri, S., Fraternali, P., Bongio, A.: Web modeling language (WebML): a modeling language for designing web sites. In: Proceedings of WWW9 Conference, Amsterdam (2000)
7. Schwabe, D., Rossi, G.: Developing hypermedia applications using OOHDM. In: Workshop on Hypermedia Development Processes, Methods and Models (Hypertext'98), Pittsburgh, USA (1998)
8. Fraternali, P., Paolini, P.: A conceptual model and a tool environment for developing more scalable, dynamic, and customizable web applications. In: Advances in Database Technology - EDBT'98, 6th International Conference on Extending Database Technology, Valencia, Spain (1998) 421–435
9. Baresi, L., Garzotto, F., Paolini, P.: Extending uml for modelling web applications. In: Proceedings of the 34th Hawaii International Conference on System Sciences, Hawaii, USA (2001)
10. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modelling Language User Guide. Addison-Wesley (1999)
11. Conallen, J.: Building Web Applications with UML. Addison Wesley Object Technology Series. Addison-Wesley (1999)
12. Li, J., Chen, J., Chen, P.: Modelling web application architecture with uml. In: Technology of Object-Oriented Languages and Systems 2000 (TOOLS - Asia 2000). (2000) 265–274
13. Atkinson, C., Kühne, T., Henderson-Sellers, B.: To meta or not to meta - that is the question. Journal of Object-Oriented Programming **13** (2000) 32–35
14. Atkinson, C.: Supporting and applying the UML conceptual framework. In: «UML»'98: Beyond the Notation, 1st International Workshop, Mulhouse, France (1998) 21–36
15. Atkinson, C., Kühne, T.: Strict profiles: Why and how. In: «UML»'2000: Advancing the Standard, 3rd International Conference, York, UK (2000) 309–322
16. Atkinson, C., Kühne, T.: The essence of multilevel metamodelling. In: «UML»'2001 (the 4th International Conference on the Unified Modelling Language), Toronto, Canada (2001) 19–33
17. Henderson-Sellers, B., Atkinson, C., Firesmith, D.: Viewing the OML as a variant of the UML. In France, R., Rumpe, B., eds.: «UML»'99 -The Unified Modeling Language. Beyond the Standard. Volume LNCS 1723., Springer-Verlag, Berlin, Germany (1999) 49–66
18. Hunt, J.: Smalltalk and Object Orientation: An Introduction. Springer (1997)
19. Sun: J2EE blueprints design pattern: Model view controller (MVC) (2001)

Wiccap Data Model: Mapping Physical Websites to Logical Views

Zehua Liu, Feifei Li, and Wee Keong Ng

Centre for Advanced Information Systems, School of Computer Engineering
Nanyang Technological University, Singapore, 639798, SINGAPORE
wkn@acm.org

Abstract. Information sources over the WWW contain a large amount of data organized according to different interests and values. Thus, it is important that facilities are there to enable users to extract information of interests in a simple and effective manner. To do this, information from the Web sources need to be extracted automatically according to users' interests. However, the extraction of information requires in-depth knowledge of relevant technologies and the extraction process is slow, tedious and difficult for ordinary users. We propose the Wiccap Data Model, an XML data model that maps Web information sources into commonly perceived logical models. Based on this data model, ordinary users are able to extract information easily and efficiently. To accelerate the creation of data models, we also define a formal process for creating such data model and have implemented a software tool to facilitate and automate the process of producing Wiccap Data Models.

1 Introduction

1.1 Motivation

In the past decade, the World Wide Web has completely reshaped the Internet and led to a wide range of applications and services that are available to people surfing the Internet. Net-surfers do not face the problem of information unavailability. Instead, *information overloading* is becoming a bottleneck for obtaining information from the Internet. People are not able to retrieve exact information of interest. Normal Internet search engines usually return massive irrelevant information along with some useful ones. Even in a given website, useful information is normally split into pieces and is highly coupled with other unrelated data such as advertisements and formatting styles. It's a time consuming and tedious process for users to finally get the right information that they are looking for. In addition, it is not easy for applications to access information from the web, because websites are designed specifically for human browsing.

One of the solutions to this information overloading problem is *Information Extraction* (IE) from the Web. Over the past couple of years, some IE systems, mostly wrappers and software agents, have been built to automatically retrieve information from various sources and extract only those that are of the users' interests.

However, the extraction of information requires a process of specifying data models or extraction rules that define the mapping between the actual information on the Web pages and the pieces that the users are interested. This process usually requires *in-depth knowledge* of relevant technologies, which ordinary users do not possess. As a result, ordinary users who do not possess such technical knowledge are prevented from using IE systems or are limited to use wrapper programs created by other people which may not extract the exact information that the users want.

Another common problem that most Information Extraction systems face is the *slow speed* at which the data models are created. The rate at which new websites are appearing and old websites are changing is much faster than what the current IE systems can handle. Thus, to have a tool to help users to quickly create the required data model for extracting information has become the top priority of IE system for the Web.

1.2 The WICCAP Approach

In this paper, we propose a data model, called the Wiccap Data Model (WDM), to map information from the Web into commonly perceived organization of logical concepts. The aim of WDM is to enable ordinary users to easily understand the data models and use them to perform extraction tasks without worrying about technical details. WDM is designed as part of the *WWW Information Collection, Collaging and Programming* (WICCAP) system [11], which is a Web-based information extraction system to enable people to obtain information of interest in a simple and efficient manner.

To allow ordinary users to perform information extraction, the WICCAP system explicitly separates the tasks of information modeling and information extraction. In WICCAP, expert users construct the WICCAP Data Model and specify how to extract the information; ordinary users indicate what to extract based on the given WDM and use the tools provided by the system to extract and view the information.

Besides being easy-to-understand and easy-to-use to ordinary users, WDM is designed to be *flexible* enough to work with heterogeneous categories of websites and *open* enough to allow other systems and applications to understand and make use of it.

To cope with the rapid changes of old websites and establishment of new websites, we formalize the process of creating a Wiccap Data Model for a given website to permit as much automation as possible. A software tool, called the Mapping Wizard, has been implemented to assist users to quickly generate the data model of a given website.

1.3 Paper Overview

The remaining sections of this paper are organized as follows. In Section 2, we give the definition of Wiccap Data Model and describe how it can be used to model websites. In Section 3, we define the formal process for creating WDM.

In Section 4, we present a software tool that helps to automate the data model creation process. We discuss some of the related work in Section 5. Finally, some concluding remarks are given in Section 6.

2 Wiccap Data Model

In most current Web Information Extraction systems, when defining data models, users usually have to directly specify which portion of a web page within a website constitutes the target information. The isolated pieces of target information are later re-organized to make them more readable and accessible. As a result, the data model produced is usually not intuitive to other users and is very specific to the particular website.

In WICCAP, we try to derive a logical data model (WDM) of the target website first and then extract information from the website based on this model. The role of the WICCAP Data Model is to relate information from a website in terms of *commonly perceived logical structure*, instead of physical directory locations. The logical structure here refers to people's perception on the organization of contents of a specific type of website.

For example, when a newspaper website, such as BBC Online News [16], is mentioned, people generally think of a site that consists of sections such as World News, or Sports News; each section may have subsections and/or a list of articles, each of which may have a number of attributes such as title, summary, the article itself, and maybe links to other related articles. This hierarchy of information is the commonly perceived structure of a newspaper website, which most users are quite familiar with. If we model and organize the information from a newspaper website in this way, the resulting data model will appear familiar to and be easily accepted by most users. In addition, since the model is applicable to the whole category of websites, e.g. all newspaper websites, the logical structure created can be re-used in the future when creating data model for websites of the same type.

WDM is designed to be flexible such that it can be used to model not only a single webpage, but also a set of webpages. This set of webpages may or may not have similar layout structures in terms of HTML syntax. It could be a small collection of several web pages located in the same directory, the whole website, or pages across several websites, so long as all pages together form an unified logical view.

The final logical view of a website is usually a tree structure. In the case of BBC Online News, as shown in Figure 1, we have a root node, under which there are different sections, such as "World" and "Business". There may be subsections under each section. Each node (Section or Subsection) may have a list of articles, denoted by the node "ArticleList" with a child node "Article". Attributes and contents of the article are represented as child nodes of "Article". The mapping information is stored as attributes of the elements to hide the technical details from ordinary users and to maintain the logical overview of the data model.

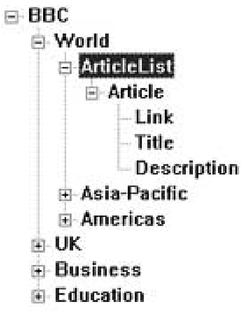


Fig. 1. Logical View of BBC Online News

2.1 WDM Schema and Mapping Rules

The Wiccap Data Model consists of two main components: WDM Schema and Mapping Rule. Wdm schema provides the skeletons of the logical view while Mapping Rule offers a detailed definition of the mapping between logical elements and the actual webpages.

A Wdm *Schema* concerns not only about a single website but a category of websites. It defines the basic data elements and how these elements are organized to form a logical view of a website. The Wiccap system categorizes websites into different types, such as online newspaper, digital library, and product information. For each category of website, a specific Wdm schema is defined. Each Wdm schema is defined using a XML Schema. Since a XML schema is itself an XML document, the Wdm schema is well structured, interoperable, and easy to process.

A *Mapping Rule* of a website refers to a specific Wiccap Data Model that describes that particular website's logical structure using the Wdm schema of the category of that website. A Mapping Rule is defined as a normal XML document, with the corresponding XML schema as its definition.

A Mapping Rule based on a Wdm schema could be viewed as an instance of that schema. The relationship between Wdm Schema and Mapping Rule is similar to the Class-Object relationship in object-oriented concept. This relationship is reinforced by using the Wdm schema as the XML Schema of Mapping Rule's XML document.

2.2 WDM Elements

We define a few *Basic WDM Schema Elements* in Wiccap to form the base of the Wiccap Data Model concept. Additional Wdm schema elements that are specific to certain categories of websites can be defined to achieve more accurate modeling of a particular type of website. A Wdm schema generally consists of the basic predefined elements and some other extended elements. Each Wdm schema describes a category of websites that share the same or similar logical structure. Mapping Rules for specific websites can be created according to the Wdm schema of the website.

The Wdm schema defines several basic elements, including *Locator*, *Link*, *Form*, *Mapping*, *Item*, *Record*, and *Region*. These basic elements themselves do not have much logical meaning. However, they are essential to Wdm schema because they provide the mechanisms for mapping from other elements in the logical data model to the actual Web pages in the website. This seciton describes the basic elements and Section 2.3 illustrates the use of these elements with an example.

Locator A *Locator* is the fundamental element in Wdm. It locates a portion within a webpage. The current Wiccap system only implements a ByPattern Locator that relies on the starting pattern and ending pattern to locate a specific portion that resides between these two text patterns. Since HTML pages are merely ASCII texts, with a Locator, we are able to retrieve any part within a given webpage, regardless of whether it is a link, attribute, element name, or text. The definitions of Locator and some of its relevant children are as follows.

```
<xsd:complexType name="LocatorType" content="elementOnly">
    <xsd:choice>
        <xsd:element name="LocatorPattern" type="LocatorPatternType"/>
    </xsd:choice>
    <xsd:attribute name="Type" use="required">
        <xsd:simpleType base="xsd:NMTOKEN">
            <xsd:enumeration value="ByPattern"/>
            <xsd:enumeration value="ByPath"/>
        </xsd:simpleType>
    </xsd:attribute>
</xsd:complexType>
<xsd:complexType name="LocatorPatternType" content="elementOnly">
    <xsd:sequence>
        <xsd:element name="BeginPattern" type="BeginPatternType"/>
        <xsd:element name="EndPattern" type="EndPatternType"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="BeginPatternType" base="xsd:string">
            ⋮   ⋮   ⋮
```

Link A *Link* corresponds to the concept of a hyperlink in the Web context. It gives Wdm the ability to following hyperlinks in Web pages; thus, traversing the entire website. When a Link appears in a Mapping Rule, it indicates that there is a hyperlink in the actual web page that should be followed. When the extraction agent follows a Link in a Mapping Rule, it attains the same effect as a user clicking on a hyperlink in a browser, and is brought to the new page pointed by that hyperlink.

A Link can be *Static*, *Dynamic* or a *Form*. A static link is just a simple fixed URL. It is used to indicate the base address of a Mapping Rule. For example, in the root element of the BBC Mapping Rule, we specify a static Link with a value “http://news.bbc.co.uk/”. A Dynamic type of Link’s link value must be extracted by the extraction agent at run-time from the Web page. The Link element contains a Locator that locates the hyperlink to be extracted. The agent will follow this link to reach another webpage to continue the extraction. This Dynamic type of Link is critical to a flexible and reliable extraction rule because the links in a webpage are likely to change while the positions where these links appear remain relatively stable. A good example of dynamic Link is the links of news headlines that point to the detailed news.

The definition of Link is shown as follows:

```
<xsd:complexType name="LinkType" content="mixed">
    <xsd:choice>
        <xsd:element name="Locator" type="LocatorType"/>
        <xsd:element name="Form" type="FormType"/>
    </xsd:choice>
    <xsd:attribute name="Type" use="required">
        <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
                <xsd:enumeration value="Dynamic"/>
                <xsd:enumeration value="Form"/>
                <xsd:enumeration value="Static"/>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:attribute>
</xsd:complexType>
```

Form *Form* caters to a special group of HTML tags: FORM and other related HTML elements. Form is defined as a type of Link because a FORM ACTION causes the server to perform some operation and return another page; this has the same effect as clicking on a hyperlink. Typical examples where the Form element may be useful are the search function and user login authentication.

When the type of a Link is Form, the URL is dynamically constructed by concatenating all the name and value pairs of the input elements contained in the specified form. The extraction agent will post this URL to the remote web server and obtain the returned webpage. This ability to handling Forms is important for a Web IE System due to the extensive use of Forms in websites. It is also one of the characteristics that distinguish WDM from other similar systems, as currently only few of these systems [2] [17] take Forms into consideration or directly incorporate them into their systems.

Mapping *Locator* allows us to access anywhere within a single page while *Link* enables us to jump from one page to another. *Mapping* combines these two basic elements to allow us to navigate throughout an entire website, or even the whole World Wide Web. It is the element that a Mapping Rule uses to relate the logical element to the actual physical webpage.

A Mapping is a simple container element that holds either a Link, a Locator or both. A Mapping may contain a child Mapping element. This recursive definition allows WDM elements to jump through multiple levels of links and essentially makes it possible to completely separate the logical structure from the physical structure defined by the hierarchy of links and physical directories. Every main element, which is shown in the logical tree view as a node, should have a Mapping as its attribute to indicate how to get to this node in the actual website.

Item The elements introduced before are more for mapping purpose. They are not the main elements that constitute the logical tree. The following three elements, Item, Record and Region, are the logical elements that represent information to be extracted.

An *Item*, which represents a piece of information that the user is interested in, is the basic main element of WDM. It is the atomic unit of all the information in the logical data model. Item has a Mapping attribute that indicates where data is to be extracted, and other attributes that describe its semantic meaning.

Strictly speaking, Item is not a basic WDM element as generally different types of websites will have different Item Types. The definition of Item element varies in different WDM schema. As they have the similar definition and only differ slightly, we treat it as a basic WDM element.

Record In most cases, we do not just specify a single Item to extract in an entire logical tree. Within a website, there is likely to be a lot of Items that the users are interested. Thus, we need to group related Items together, using *Record*.

A Record contains a group of Items, or a tuple of attributes. It is usually used to repeatedly retrieve data in a table or a list when its "Repeative" attribute is 'True'. It makes WDM more flexible and expressive by allowing it to describe iterative structure, which appears frequently in complex logical data model. As with other main elements, there is a Mapping attribute to map the Record to the actual webpages.

An example of Record is the Article element in the BBC News Mapping Rule, with the different attributes of the article, such as Title, Link, Short Description and Content, forming the group of Items under this Record. This article record is iterative since there is a number of articles under each section.

Region A *Region* is defined to help to identify the area where an iterative Record repeats itself. A Region typically contains only one Record if that Record is repeative. It can be considered as the rough area that we are interested within a webpage.

Complete WDM Schema With all the basic elements, we are now ready to define WDM Schemas for different types of websites. The process of producing a WDM schema consists of creating customized elements and assembling these new elements and the basic elements to form an organized structure that reflects the logical view of the website.

New WDM elements have to be defined for the type of website that the user is interested. What elements should be added depends on the specific type of Web site that the user is dealing with. All the basic and customized WDM elements will be organized into a certain hierarchical structure. This hierarchy of elements must reflect the logical structure of the Website that the user perceives or understands.

2.3 An Example on BBC Online News

In this section, we illustrate with an example of how WDM schemas and the Mapping Rules translate physical structure into logical view. The website that we will be looking at is the BBC Online News [16].

```
<!-- WiccapNewsBBC.xml -->
<?xml version="1.0" encoding="UTF-8"?>
<Wiccap Name="BBC" Group="News"
            xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="WiccapNews.xsd">
    <Mapping><Link Type="Static">http://news.bbc.co.uk</Link></Mapping>
    <Section Name="World">
        <Mapping>
            <Link Type="Dynamic"><Locator Type="ByPattern">
                <LocatorPattern>
                    <BeginPattern>
                        <![CDATA[<TD CLASS="rootsection" BGCOLOR="#FFFFFF"><A HREF="]]>
                    </BeginPattern>
                    <EndPattern><![CDATA[" CLASS="index">]]></EndPattern>
                </LocatorPattern>
            </Locator></Link>
        </Mapping>
        <Region Name="ArticleList">
            <Mapping><Locator Type="ByPattern">
                <LocatorPattern>
                    <BeginPattern><![CDATA[<SPAN CLASS="sectiontitle">]]></BeginPattern>
                    <EndPattern><![CDATA[<SCRIPT LANGUAGE=]]></EndPattern>
                </LocatorPattern>
            </Locator></Mapping>
            <Record Name="Article" NumOfItem="3">
                <Mapping><Locator Type="ByPattern"><LocatorPattern>
                    <BeginPattern><![CDATA[<DIV CLASS="]]></BeginPattern>
                    <EndPattern/>
                </LocatorPattern></Locator></Mapping>
                <Item Type="Link" TagFilter="Off" Description="This is the link to the news">
                    <Mapping><Locator Type="ByPattern"><LocatorPattern>
                        <BeginPattern><![CDATA[<a href="]]></BeginPattern>
                        <EndPattern><![CDATA[">]]></EndPattern>
                    </LocatorPattern></Locator></Mapping>
                </Item>
                <Item Type="Title" Description="This is the title of the news">
                    <Mapping><Locator Type="ByPattern"><LocatorPattern>
                        <BeginPattern><![CDATA[<B class="h***]]></BeginPattern>
                        <EndPattern><![CDATA[</B>]]></EndPattern>
                    </LocatorPattern></Locator></Mapping>
                </Item>
                <Item Type="Description" Description="This is the description of the news">
                    <Mapping><Locator Type="ByPattern"><LocatorPattern>
                        <BeginPattern><![CDATA[</A>]]></BeginPattern>
                        <EndPattern><![CDATA[<BR]]></EndPattern>
                    </LocatorPattern></Locator></Mapping>
                </Item>
            </Record>
        </Region>
        <Section Name="Asia-Pacific">
         .  .  .
         .  .  .
        </Section>
        <Section Name="Americas">
         .  .  .
         .  .  .
        </Section>
    </Section>
    <Section Name="Business">
     .  .  .
     .  .  .
</Wiccap>
```

Fig. 2. Logical View of BBC Online News

The WDM schema for this website is used as an example throughout the paper. Most of the definitions of the basic and customized elements are described in previous sections. This WDM schema simply redefines the Item type and add in two more new elements: Section and Wiccap. Item type is tailored to this specific type of website. Wiccap is the root node of the entire Mapping Rule XML document. Section, as described previously, is the key element in this logical structure because it not only defines a section of a newspaper, but also allows the existence of subsection by using recursive definition. This enables the WDM schema to represent the logical view of most of the online newspaper websites.

The Mapping Rule for BBC Online News, shown in Figure 2, is based on the logical data model established by the WDM schema defined above. Due to the limitation of space, only details of one Section and one subsection of the Mapping Rule are shown.

In the root element, the XML Schema file is specified; this is also the file that contains the definition of the WDM schema for online newspaper websites. The Mapping of the root element Wiccap is a static Link that points to "http://news.bbc.co.uk". This is true for most Mapping Rules because a website needs to have a base URL or home URL for any navigation to start from.

Under the root node "Wiccap", there are a few Sections. In this example, we specify "World", "Business" and so on. Again, the "World" Section has a Mapping, which extracts the link to the page containing the actual "World" section on the website. An ArticleList (Region) is included under the "World" Section to indicate that we are interested in some content in this section. The Article (Record) and Item elements under this ArticleList node give details of the contents to be extracted.

Two Sections "Asia-Pacific" and "Americas" are inserted after the ArticleList. The internal definition of these two sub-Sections are similar to the Section "World". All the Sections and ArticleLists together form a tree structure of the logical view (Figure 1), which is quite close to our common understanding of newspaper: a hierarchy of Sections and Sub-Sections with Articles in each section.

3 WDM Creation Process

As discussed in the introduction section, the manual process of creating data model for Web information sources, including the WICCAP Data Model, is tedious and slow. To allow the WICCAP system to handle large amount of new websites and changing websites, we define a formal process for the generation of Mapping Rule so that it can be made as automatic as possible and have implemented the Mapping Wizard as a tool to help facilitate and automate this process. The formal process of Mapping Rule generation using Mapping Wizard consists of four stages, as illustrated in Figure 3.

Basic Logical Structure Construction To start building a logical view of a website, the user supplies the home URL of the target website to the Mapping

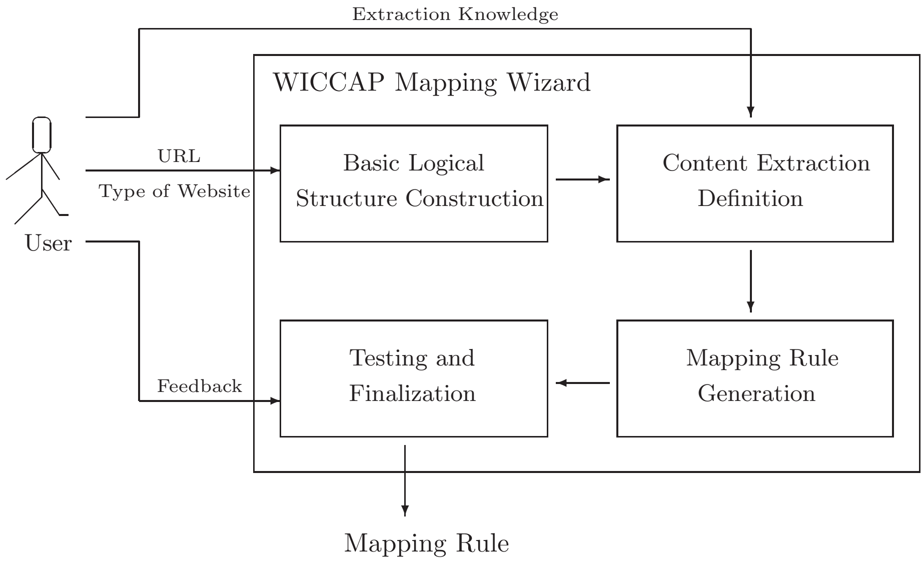


Fig. 3. Formal Process of Mapping Rule Creation

Wizard and use the Mini Web Browser to navigate the website. The user has to decide on the type of the website and select an appropriate WDM schema either by using his knowledge and understanding of the website or by navigating the website. A logical tree can then be constructed with the understanding of the organization of the contents provided by the website.

In this stage, it is important that the user concentrates on building the overall logical data model instead of details of how to map the logical tree nodes into the actual website. This is because confirming the logical structure first gives the user an overview of the whole website and makes it easy to find the target information that is distributed among a set of webpages.

The process of building up the logical structure of the website is usually quite straightforward. In the case of an online news website, what the user needs to figure out are the sections under the root node and the articles or subsections under those sections. This can be easily done by looking at the index pages of the main sections and subsections. Therefore, the user only needs to insert “World”, “UK”, “Business”, etc, one at a time as the children of the root node, and does the same for any subsection of these sections.

Content Extraction Definition After the basic logical tree is made clear, the user continues with the *Content Extraction Definition* stage, in which he or she spells out the specific information that is of interest and relates the nodes in the logical tree to the actual website.

As described earlier, Mapping includes Link and Locator and eventually points to certain portion of a webpage. To specify the Mapping attributes, it is the same as figuring out the patterns that enclose each piece of information.

Without the help of the Mapping Wizard, the user has to look at the HTML source to decide those patterns are.

The Mapping Wizard is equipped with the ability to figuring out the mapping information with certain manual guidance. This is done by using the assistant tools, which will be discussed in Section 4. The Mapping Wizard also provides a Graphical User Interface (GUI) for the user to modify all the properties to fine-tune and optimize the mapping information.

One point that is very important to the first two stages is that a certain amount of human intervention or feedback is crucial to the Mapping Wizard. Due to the complexity of HTML syntax, we do not believe that there is a single complete algorithm that is capable of fully automating the process of generating the Mapping Rule. What the Mapping Wizard can do is to try to reduce the human intervention as much as possible.

Mapping Rule Generation Once all the information is confirmed, the Mapping Wizard generates the Mapping Rule according to the tree structure and all the corresponding properties. This process is fully automated. The Mapping Wizard validates the information that has been specified and produces the Mapping Rule according to the syntax defined in the WDM schema.

Testing and Finalization In this stage, the user may test the generated Mapping Rule to see whether it represents the correct logical structure and whether it locates all the required information.

The testing is a trial and error process, similar to the process of debugging a program. the Mapping Wizard performs the actual extraction using the generated Mapping Rule. Details of how to perform the extraction using a Mapping Rule is discussed in [10]. If the user is satisfied with all the information retrieved from the website, he or she can finalize the Mapping Rule. Once finalized, the Mapping Rule can either be stored into some repository or be delivered to users of the extraction agent for the extraction of information.

4 Implementation

The current version of the *Mapping Wizard* has been implemented using Visual C++ 6.0. The programming of the Graphical User Interface (GUI) makes use of Microsoft Foundation Classes (MFC). This implementation is available on the Windows Operating System platform (Windows 98/ME/2000/XP). This section briefly discusses the assistant tools provided by Mapping Wizard and its GUI.

4.1 Assistant Tools

Mapping Wizard provides a set of assistant tools to help to automate the process of generating a Mapping Rule. These tools are the critical components that make the Mapping Wizard practically useful. Without the help of these automation tools, the Mapping Wizard is only something like a blind combination of the Internet Explorer, Notepad and an XML editor.

We believe that, in practice, it is not possible to have a single tool that fully automates the whole Mapping Rule generation process because this process is usually complex and requires human intervention from time to time. In the Mapping Wizard, instead of trying to provide a single tool, we build a set of tools that address different aspects of the problem. We identify the main difficulties and bottlenecks that slow down the overall process and develop assistant tools to accelerate those parts one by one. In this way, the efficiency of the overall process is improved and the time required to generate a Mapping Rule of a given website is greatly reduced. The following gives an overview of the tools provided.

The **Pattern Searching Tool** is similar to the "Search" function that most text editors provide. It allows searching of certain text pattern within the HTML source and the web browser view. Searching directly on the Mini Web Browser may be more helpful and straightforward to the users.

The **Pattern Induction Tools** are the most useful ones among all the assistant tools provided. They are developed to release users from manually studying the HTML Source to figure out the patterns that enclose the target information. To derive the Mapping properties using these tools, the user only needs to highlight the information on the Mini Web Browser and activates the command. In the background, the Mapping Wizard detects what portion on the Mini Web Browser is highlighted and determines the corresponding HTML source. It then applies an algorithm to derive the starting pattern and ending pattern that enclose this part of the HTML source. The algorithm is developed based on those described in [9].

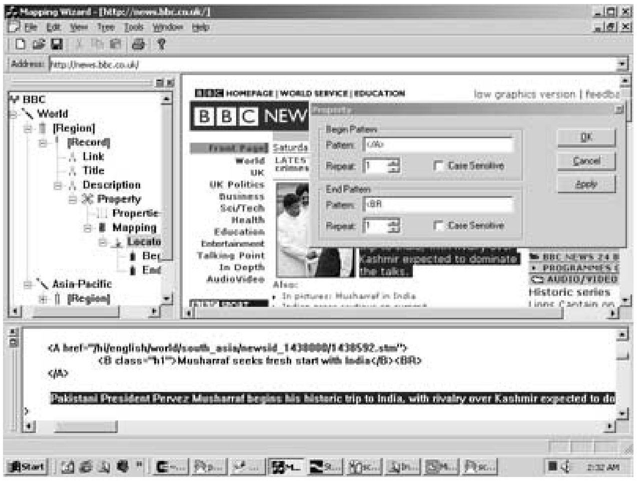
The **Section Extraction Tool** is developed to figure out all the possible Section nodes in the same level and derive the Mapping properties for them in a single click; while the **Structure Copier** is to copy the structure of the constructed Section to other empty Section nodes, hoping that the same Region, Record and Items structure will fit into those Sections with slight modification.

Combining all the assistant tools to create a Mapping Rule, the user first uses Section Extraction Tool to generate all the Sections, uses the Pattern Induction Tools to identify Region, Record and all Items under the first Section, then activates the Structure Copier to copy the structure of the first Section to other Sections. After some modification on the rest of the Sections, a Mapping Rule is generated.

4.2 Graphical User Interface

Figure 4 shows the main GUI of Mapping Wizard. The **Mini Web Browser** on the right panel is a small web browser for users to navigate the whole website. It has nearly all the functions of a normal web browser. Most assistant tools rely on the Mini Web Browser to perform operations.

The **HTML Source Viewer** displays the HTML source code of the web page that the user is viewing in the Mini Web Browser. It may yield higher accuracy in selecting the desired text string. However, it requires the user to have relevant knowledge, such as HTML and JavaScript.



```
<A href="/hi/english/world/south_asia/newsid_1438000/1438592.stm">
        <B class="h1">Musharraf seeks fresh start with India</B><BR>
</A>

Pakistani President Pervez Musharraf begins his historic trip to India, with rivalry over Kashmir expected to do
```

Fig. 4. Main GUI of Mapping Wizard with Property Dialog

The panel on the left is the **Logical Tree Editor**, which displays the logical model of the website that the user is modeling. The editor allows users to perform normal tree manipulation on the logical tree, including insertion and deletion of tree nodes and modification of their properties, using the **Property Dialog**.

5 Related Work

We compare our work with others related systems in several aspects. In the aspect of information modeling, most existing IE systems, including Tsimmis [6], Wien [9] and W4F [17] do not model information from the logical point of view. The organization of the extracted information has close tie to the physical structure of the website. The data models are schema-less and are difficult to understand. Other systems like Araneus [2] [14] and Stalker [15] build data models that separate the physical and logical structure to certain extent. But their data models still do not truly describe the logical structure that is hidden behind the physical appearance of websites and can not be understood easily by users other than the data model creator. On the contrary, the WICCAP Data Model models information in the way that most users perceive the websites and the resulting logical models are very close to these users' understanding.

In the aspect of data model creation, *manual generation* of data models or wrappers, used in Jedi [7], Tsimmis and Araneus, does not offer GUI support and require the use of programming language like grammar. These systems tend to be very difficult to use by ordinary users. Other systems, including Wien, Stalker and NoDoSe [1], take the *machine learning* approach to learn the extraction rules by examples. Some of these systems provide visual support to allow easy selection and labelling of training examples. However, the requirement for expensive computation power and large number of examples are their drawbacks. A third category of IE systems use *supervised wrapper generation*. Lixto [3], XWrap [12] and W4F provide GUI to interact with users and make use of the HTML DOM tree to locate the information. These systems are closest to WICCAP in terms of the data model generation process. However, they are still not intuitive to ordinary users and require users to have some technical background.

One important distinguishing characteristic of the WICCAP architecture is the separation of the tasks of information modeling and information extraction. This is essentially enable ordinary users to use the system without having special knowledge on Information Extraction.

In addition, systems such as Embley et al. [5], Lixto, RoadRunner [4], Wien and W4F deal with only a single webpage or a set of webpages with similar structure. This greatly limits the ability to modeling a website as a whole. Meanwhile, only few systems (Lixto and Wien) offer the similar functionality as Mapping Wizard to allow users to directly operate on the browser view of webpages instead of the HTML sources.

There are other works that are related in other aspects. Ariadne [8] focuses on modeling and extracting information for further integration and querying. XGrammar [13] takes a similar approach to WDM to use XML Schema to describe data semantics. SiteLang [18] is a language that allows specification of information services based on the concepts of story and interaction spaces.

6 Conclusion

In this paper, we investigated the problems with current Web Information Extraction systems as a solution to the information overloading problem. A new XML data model has been proposed to map websites from their physical structures to logical views. Based this intuitive data model, ordinary users are able to perform information extraction tasks that were previously only possible for people with in-depth technical knowledge. A formal process is presented to standardize the creation of Wiccap Data Model and a software tool has been implemented to automate the process. The process of generating a data model has been significantly accelerated.

References

1. Brad Adelberg. NoDoSE - A Tool for Semi-Automatically Extracting Semi-Structured Data from Text Documents. In *ACM SIGMOD International Conference on Management of Data*, pages 283–294, Seattle, Washington, June 1998. 132
2. Paolo Atzeni, Giansalvatore Mecca, and Paolo Merialdo. To Weave the Web. In *Proceedings of 23rd International Conference on Very Large Data Bases (VLDB 97)*, pages 206–215, Athens, Greece, August 25-29 1997. Morgan Kaufmann. 125, 132
3. Robert Baumgartner, Sergio Flesca, and Georg Gottlob. Visual Web Information Extraction with Lixto. In *Proceedings of 27th International Conference on Very*

Large Data Bases (VLDB 2001), pages 119–128, Roma, Italy, September 11-14 2001. Morgan Kaufmann. 133
4. Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. RoadRunner: Towards Automatic Data Extraction from Large Web Sites. In *Proceedings of 27th International Conference on Very Large Data Bases (VLDB 2001)*, pages 109–118, Roma, Italy, September 11-14 2001. Morgan Kaufmann. 133
5. David W. Embley, Yiu-Kai Ng, and Li Xu. Recognizing Ontology-Applicable Multiple-Record Web Documents. In *Proceedings of 20th International Conference on Conceptual Modeling (ER 2001)*, Lecture Notes in Computer Science, pages 555–570, Yokohama, Japan, November 27-30 2001. Springer. 133
6. Joachim Hammer, Héctor García-Molina, Junghoo Cho, Arturo Crespo, and Rohan Aranha. Extracting Semistructured Information from the Web. In *Proceedings of the Workshop on Management of Semistructured Data*, Tucson, Arizona, May 1997. 132
7. Gerald Huck, Peter Fankhauser, Karl Aberer, and Erich J. Neuhold. Jedi: Extracting and Synthesizing Information from the Web. In *Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems (CoopIS 98)*, pages 32–43, New York City, New York, USA, August 20-22 1998. IEEE-CS Press. 132
8. Craig A. Knoblock, Steven Minton, Jose Luis Ambite, Pragnesh Jay Modi Naveen Ashish, Ion Muslea, Andrew G. Philpot, and Sheila Tejada. Modeling Web Sources for Information Integration. In *Proceedings of Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 211–218, Madison, Wisconsin, July 1998. 133
9. Nicholas Kushmerick. Wrapper induction: Efficiency and expressiveness. In *AAAI-98 Workshop on AI and Information Integration*, pages 15–68, Madison, Wisconsin, July 1998. 131, 132
10. Feifei Li. Network Extraction Agent for WICCAP System. Technical report, Nanyang Technological University, November 2001. 130
11. Feifei Li, Zehua Liu, Yangfeng Huang, and Wee Keong Ng. An Information Concierge for the Web. In *Proceedings of the First International Workshop on Internet Bots: Systems and Applications (INBOSA2001), in conjunction with the 12th International Conference on Database and Expert System Applications (DEXA'2001)*, pages 672–676, Munich, Germany, September 3-8 2001. 121
12. Ling Liu, Calton Pu, and Wei Han. XWRAP: An XML-Enabled Wrapper Construction System for Web Information Sources. In *Proceedings of 16th International Conference on Data Engineering (ICDE 2000)*, pages 611–621, San Diego, California, USA, 28 February - 3 March 2000. IEEE Computer Society. 133
13. Murali Mani, Dongwon Lee, and Richard R. Muntz. Semantic Data Modeling Using XML Schemas. In *Proceedings of 20th International Conference on Conceptual Modeling (ER 2001)*, Lecture Notes in Computer Science, pages 149–163, Yokohama, Japan, November 27-30 2001. Springer. 133
14. Giansalvatore Mecca and Paolo Atzeni. Cut and Paste. *Journal of Computer and System Sciences*, 58(3):453–482, 1999. 132
15. Ion Muslea, Steve Minton, and Craig Knoblock. A Hierarchical Approach to Wrapper Induction. In *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 190–197, Seattle, WA, USA, 1999. ACM Press. 132
16. BBC Online News. http://news.bbc.co.uk/. 122, 126
17. Arnaud Sahuguet and Fabien Azavant. Wysiwyg web wrapper factory (w4f). In *Proceedings of World Wide Web Conference*, Orlando, October 1999. 125, 132

18. Bernhard Thalheim and Antje Dústerhóft. SiteLang: Conceptual Modeling of Internet Sites. In *Proceedings of 20th International Conference on Conceptual Modeling (ER 2001)*, Lecture Notes in Computer Science, pages 179–192, Yokohama, Japan, November 27-30 2001. Springer. 133

Representing and Querying Semistructured Web Data Using Nested Tables with Structural Variants*

Altigran S. da Silva[1,†], Irna M.R.Evangelista Filha[1], Alberto H. F. Laender[1], and David W. Embley[2]

[1]Department of Computer Science
Federal University of Minas Gerais
31270-901 - Belo Horizonte MG - Brazil
{alti,imre,laender}@dcc.ufmg.br

[2]Department of Computer Science
Brigham Young University
Provo, Utah 84602 USA
embley@cs.byu.edu

Abstract. This paper proposes an approach to representing and querying semistructured Web data. The proposed approach is based on nested tables, which may have internal nested structural variations to accommodate semistructured data. Our motivation is to reduce the complexity found in typical query languages for semistructured data and to provide users with an alternative for quickly querying data obtained from multiple-record Web pages. We show the feasibility of our proposal by developing a prototype for a graphical query interface called QSByE (*Querying Semistructured data By Example*). For QSByE, we define a particular variation of nested tables and propose a set of QBE-like operations that extends typical nested-relational-algebra operations to handle semistructured data. We show examples of how users can pose interesting queries using QSByE.

1 Introduction

In recent years, the demand for technologies to manage data available on the Web has increased considerably. Motivated by such a demand, several researchers have proposed data models for representing [3, 8, 16] and query languages for manipulating [1, 2] Web data. These proposals generally adapt database techniques for dealing efficiently and flexibly with the particularities of this kind of data [8]. The advent of XML has also offered a perspective on Web data management. Indeed, XML has been largely used as a data model for representing semistructured data in general [15], and Web data in particular. This has led to the development of several systems, query languages, and tools, all to deal with XML objects as data containers.

Unfortunately, the inherent freedom common in generic semistructured data models and XML inhibits the deployment of metaphors and paradigms like the ones that have been used extensively in typical data management tasks. For instance, the problem of storing arbitrary XML objects in relational databases

*This work was partially supported by Project SIAM (MCT/CNPq/PRONEX grant number 76.97.1016.00) and by CNPq (grant number 467775/00-1). The first and second authors are supported by scholarships from CAPES. The fourth author is supported by NSF (grant number IIS-0083127).

†On leave from the University of Amazonas, Brazil.

involves finding regular structures in XML files—a problem known to be NP-complete [5]. More to the point of this paper, XML query languages [1] have become much more complex, making it even more difficult for less skilled users to tap into the newly available and rapidly growing body of data on the Web.

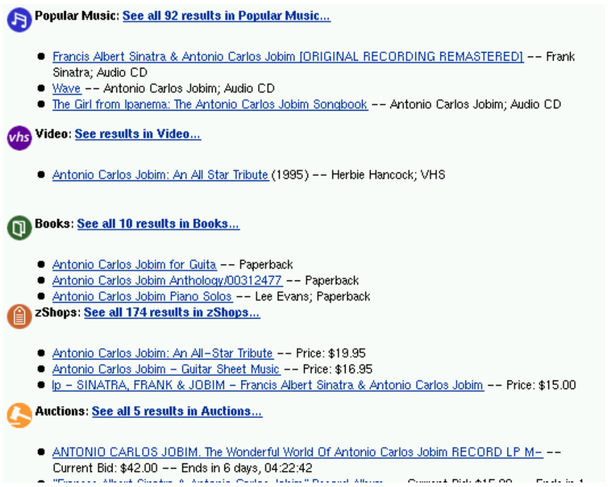
These problems have motivated work in the management of semistructured data where flexibility is restrained in favor of simplicity. STORED [5], for example, is a query language that maps a semistructured data model to the relational data model. Since semistructured data does not usually fit in a regular structure, STORED uses an "overflow" graph to accommodate irregular portions of the data. Buneman et al. propose in [3] a data model that is more limited than the usual semistructured data models. Their goal is to provide a data modeling process for semistructured data that is as close as possible to conventional database systems. Dataguides [9], an interactive tool for querying OEM databases [16], presents users with a dynamic structural summary of semistructured data. The goal is to reduce the level of complexity in query formulation by exempting users from dealing (possibly unnecessarily) with the entire structure of the data to be queried.

In harmony with the work of others that favors simplicity over flexibility, we propose in this paper the use of nested tables allowing internal structural variations for representing and querying semistructured Web data. We show throughout the paper how nested tables can be nicely adapted for this kind of data and can provide a simple and intuitive representation close to record-based database representations. As Makinouchi [14] explains, nested tables naturally accommodate regular hierarchical data. The distinction, and main contribution, of our proposal is that it also allows the representation of variations and irregularities typical of semistructured data.

As an example, consider the data implicitly present in Web pages generated as a result of a query in the main page of the Amazon.com Web site. Fig. 1 shows an example of such a page, for the Brazilian composer Antonio Carlos Jobim. In Fig. 2, we present a nested table containing data of interest extracted from this page. Observe that the information about items from each Amazon.com store is different. For instance, in row 1, where the value of Store is "Popular Music", the information consists of Item, By, and Format, whereas in row 5, where the value of Store is "Auctions", the information consists of Item, Bid, and Time.

In addition to formally defining nested tables with internal variations, this paper also formally defines a set of query operations for semistructured data represented as an internally varying nested table. We base this set of operations on several proposals found in the literature [4, 10, 17], particularly on the algebra proposed by Colby [4]. As a further contribution, this paper also describes QSByE (*Querying Semistructured data By Example*) [7], which shows the feasibility of using nested tables for representing and querying semistructured Web data[1]. QSByE combines features of QBE (*Query By Example*) [18] with typical features of query languages for semistructured data. As implemented, QSByE

[1] Previous query languages for nested tables, such as QBEN [10], have been defined, but none deal with semistructured data as does QSByE, as described in this paper.



Popular Music: See all 92 results in Popular Music...

- Francis Albert Sinatra & Antonio Carlos Jobim [ORIGINAL RECORDING REMASTERED] -- Frank Sinatra; Audio CD
- Wave -- Antonio Carlos Jobim; Audio CD
- The Girl from Ipanema: The Antonio Carlos Jobim Songbook -- Antonio Carlos Jobim; Audio CD

Video: See results in Video...

- Antonio Carlos Jobim: An All Star Tribute (1995) -- Herbie Hancock; VHS

Books: See all 10 results in Books...

- Antonio Carlos Jobim for Guita -- Paperback
- Antonio Carlos Jobim Anthology/00312477 -- Paperback
- Antonio Carlos Jobim Piano Solos -- Lee Evans; Paperback

zShops: See all 174 results in zShops...

- Antonio Carlos Jobim: An All-Star Tribute -- Price: $19.95
- Antonio Carlos Jobim - Guitar Sheet Music -- Price: $16.95
- lp - SINATRA, FRANK & JOBIM - Francis Albert Sinatra & Antonio Carlos Jobim -- Price: $15.00

Auctions: See all 5 results in Auctions...

- ANTONIO CARLOS JOBIM. The Wonderful World Of Antonio Carlos Jobim RECORD LP M- -- Current Bid: $42.00 -- Ends in 6 days, 04:22:42

Fig. 1. Excerpt of a page from the Amazon.com Web site.

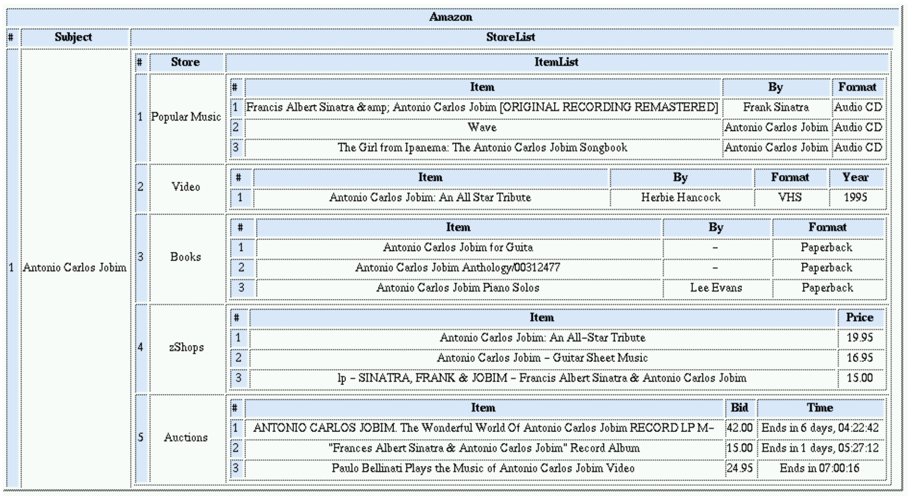
provides the structure of the data as a nested table skeleton [13] so that users do not have to uncover the structure of the data by themselves. It also lets users restructure provided nested tables to their own liking.

Despite the relative simplicity of dealing with semistructured data in the form of nested tables, it is easy to see that such a representation is not as expressive as general semistructured data models or XML. We cannot, for example, have different structures at the top level. Thus, we sacrifice some flexibility for greatly increased simplicity. However, in this work we are mainly concerned with representing data in Web pages, like the one in Fig. 1. This kind of page is said to be a *data-rich*, *ontologically narrow*, *multiple-record* Web page [6]. Examples of such pages are found in Web sites such as bookstores, electronic catalogs, travel agencies, and classified ads and include pages composed of data whose overall structure is naturally hierarchical, but exhibits a modest degree of variation. In particular, QSByE makes it possible to manipulate data resulting from the extraction of data from these kinds of Web pages by DEByE (*Data Extraction By Example*) [11], a tool that extracts data from multiple-record Web pages and organizes the extracted data in the form of nested tables.

The remainder of the paper is organized as follows. In Section 2 we present the terminology and modeling concepts used to represent semistructured data in the form of nested tables. In Section 3, we discuss the expressiveness of nested tables as a model for representing semistructured Web data. In Section 4 we define the set of operations used to query nested tables that comply with our definitions. In Section 5, we describe QSByE and illustrate how it can be used by showing some examples. Finally, we present our conclusions and an interesting direction for future work in Section 6.

2 Basic Concepts and Notation

In this section, we present the data-modeling concepts we adopt for representing semistructured Web data. These concepts are based on the notion of *nested ta-*



Amazon

#	Subject	StoreList
1	Antonio Carlos Jobim	

#	Store	ItemList
1	Popular Music	
2	Video	
3	Books	
4	zShops	
5	Auctions	

#	Item	By	Format
1	Francis Albert Sinatra &amp; Antonio Carlos Jobim [ORIGINAL RECORDING REMASTERED]	Frank Sinatra	Audio CD
2	Wave	Antonio Carlos Jobim	Audio CD
3	The Girl from Ipanema: The Antonio Carlos Jobim Songbook	Antonio Carlos Jobim	Audio CD

#	Item	By	Format	Year
1	Antonio Carlos Jobim: An All Star Tribute	Herbie Hancock	VHS	1995

#	Item	By	Format
1	Antonio Carlos Jobim for Guita	-	Paperback
2	Antonio Carlos Jobim Anthology/00312477	-	Paperback
3	Antonio Carlos Jobim Piano Solos	Lee Evans	Paperback

#	Item	Price
1	Antonio Carlos Jobim: An All-Star Tribute	19.95
2	Antonio Carlos Jobim - Guitar Sheet Music	16.95
3	lp - SINATRA, FRANK & JOBIM - Francis Albert Sinatra & Antonio Carlos Jobim	15.00

#	Item	Bid	Time
1	ANTONIO CARLOS JOBIM. The Wonderful World Of Antonio Carlos Jobim RECORD LP M-	42.00	Ends in 6 days, 04:22:42
2	"Frances Albert Sinatra & Antonio Carlos Jobim" Record Album	15.00	Ends in 1 days, 05:27:12
3	Paulo Bellinati Plays the Music of Antonio Carlos Jobim Video	24.95	Ends in 07:00:16

Fig. 2. Example of a nested table.

bles [14], augmented with the concept of *variants* [12]. As we discuss in Section 5, this kind of representation allows us, for instance, to adapt the QBE paradigm for querying this kind of data.

The main problem we have in achieving this goal is dealing with irregularities typical of semistructured data. As a solution, we propose a generalization of regular nested tables in which we allow a column to have two or more distinct substructures. An example of this solution is presented in the nested table in Fig. 2. Note that the internal structures of the objects in the column ItemList are distinct for each of the Store rows.

In what follows, we characterize these ideas more precisely. We begin by defining the notion of a *table scheme.*

Definition 1 *A* ***table scheme*** *τ is defined using the notation $\tau = (C_1 : [\tau_1^1; \ldots; \tau_1^{n_1}], C_2 : [\tau_2^1; \ldots; \tau_2^{n_2}], \ldots, C_m : [\tau_m^1; \ldots; \tau_m^{n_m}]), (m \geq 2,\ n_k \geq 1,\ k = 1, \ldots, m)$, where C_j is called a* ***column*** *and τ_j^i denotes exactly one of the following: (i) an atomic value, represented by* atom*, (ii) a set of atomic values, represented by* {atom} *or (iii) a table scheme. For the sake of simplifying the notation, if $n_j = 1$, we can use $C_j : \tau_j^1$ instead of $C_j : [\tau_j^j]$.*

Intuitively, a table scheme describes the structure of a kind of nested table in which a column C_j may store "values" or objects with distinct structure in distinct tuples. The structures of the possible objects are given by the alternatives $\tau_j^1, \ldots, \tau_j^{n_j}$ which can be either atomic values, lists of atomic values, or other nested tables.

Consider the page excerpt illustrated in Fig. 1. The structure of the objects implicitly present can be described as follows:

$$\tau = (\mathsf{Subject{:}atom, StoreList{:}(Store{:}atom, ItemList{:}}[\tau_2^1;\tau_2^2;\tau_2^3]))$$
$$\tau_2^1 = (\mathsf{Item{:}atom, By{:}atom; Format{:}atom; Year{:}atom})$$
$$\tau_2^2 = (\mathsf{Item{:}atom, Price{:}atom})\ \tau_2^3 = (\mathsf{Item{:}atom, Bid{:}atom, Time{:}atom})$$

The nested table Amazon in Fig. 2 is an instance of the table scheme τ defined above. In this table scheme, for the first level, two columns are defined: Subject and StoreList. For the column Subject, only atomic values are allowed, while the column StoreList stores nested tables. For these inner tables, we have a column Store, whose values are atomic, and a column ItemList with three distinct possible structures (nested tables), each one corresponding to a type of store in Fig. 1. We next define precisely the notion of an instance of a table scheme.

Definition 2 *Let* $\tau = (C_1 : [\tau_1^1; \ldots; \tau_1^{n_1}],\ C_2 : [\tau_2^1; \ldots; \tau_2^{n_2}], \ldots, C_m : [\tau_m^1; \ldots; \tau_m^{n_m}])$, $(m \geq 2,\ n_k \geq 1,\ k = 1 \ldots, m)$, *be a table scheme. An* ***instance*** T *of* τ, *denoted by* $T : \tau$, *is a set* $T = \{\langle C_1 : v_1^1, C_2 : v_2^1, \ldots, C_m : v_m^1\rangle, \ldots, \langle C_1 : v_1^n, C_2 : v_2^n, \ldots, C_m : v_m^n\rangle\}, (n \geq 0)$, *where* v_j^k *is: (i) an atomic value, if any* $\tau_j^i = \mathsf{atom}$, *(ii) a list of atomic values, if any* $\tau_j^i = \{\mathsf{atom}\}$, *or (iii) an instance of any* τ_j^i *that is a table scheme. An instance of a table scheme is referred to as a* ***table****.*

According to the notation introduced in Definition 2, a possible instance of our example table scheme is as follows:

$$\mathsf{Amazon} = \{\langle \mathsf{Subject} : \text{“Antonio Carlos Jobim”}, \langle \mathsf{StoreList} : S\rangle\}$$
$$S = \{\langle \mathsf{Store} : \text{“Popular Music”}, \mathsf{ItemList} : I_1\rangle, \ldots, \langle \mathsf{Store} : \text{“Auctions”}, \mathsf{ItemList} : I_5\rangle\}$$
$$I_1 = \{\langle \mathsf{Item} : \text{“Francis Albert...”}, \mathsf{By} : \text{“Frank Sinatra”}, \mathsf{Format} : \text{“Audio CD”}\rangle, \ldots\}$$
$$\ldots$$
$$I_5 = \{\langle \mathsf{Item} : \text{“ANTONIO CARLOS ...”}, \mathsf{Bid} : \text{“42.00”}, \mathsf{Time} : \text{“Ends in 6 days, 04:22:42”}\rangle, \ldots\}$$

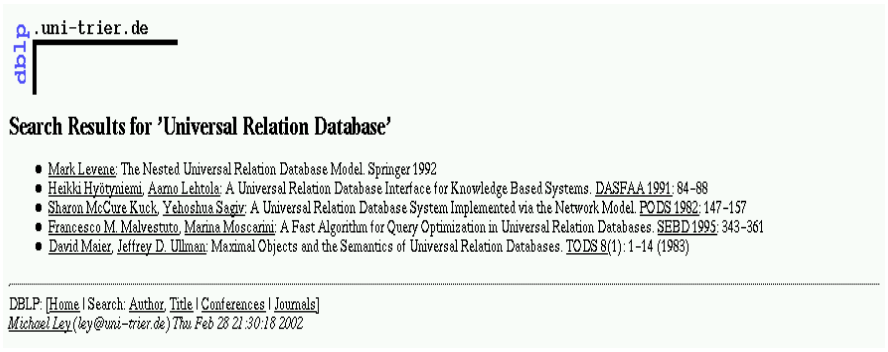
Observe that the notation above incorporates structural information along with the data itself; thus we have a self describing representation for semistructured data. As a consequence, instead of using this notation, we could easily describe such data by means of XML, as we actually do in our data extraction tool DEByE and in the QSByE interface discussed in Section 5. A formal description of a possible XML representation, however, is beyond the scope of this paper, and we omit it.

3 Expressiveness of Nested Tables for Representing Semistructured Web Data

In this section, we discuss the expressiveness of nested tables as a data model for representing semistructured Web data. In particular, we make a brief comparison between our nested-tables model and typical semistructured data models. For the discussion that follows, consider the Web page resulting from the query “Universal Relation Database” in the DBLP Web site, which is shown in Fig. 3.

Fig. 4(a) and Fig. 4(b) show the data extracted from this page organized into two distinct labeled trees according to OEM, a well known semistructured data model [16]. In the following discussion, we refer to these trees as M and N, respectively.

Trees M and N can be considered as semistructured databases and, intuitively, they are equivalent, since the relationship between atomic values is maintained. However, while in M each Publication subtree is composed of distinct atomic components, in N we introduce two additional nodes (AuthorList


dblp.uni-trier.de

Search Results for 'Universal Relation Database'

- Mark Levene: The Nested Universal Relation Database Model. Springer 1992
- Heikki Hyötyniemi, Aarno Lehtola: A Universal Relation Database Interface for Knowledge Based Systems. DASFAA 1991: 84-88
- Sharon McCure Kuck, Yehoshua Sagiv: A Universal Relation Database System Implemented via the Network Model. PODS 1982: 147-157
- Francesco M. Malvestuto, Marina Moscarini: A Fast Algorithm for Query Optimization in Universal Relation Databases. SEBD 1995: 343-361
- David Maier, Jeffrey D. Ullman: Maximal Objects and the Semantics of Universal Relation Databases. TODS 8(1): 1-14 (1983)

DBLP: [Home | Search: Author, Title | Conferences | Journals]
Michael Ley (ley@uni-trier.de) Thu Feb 28 21:30:18 2002

Fig. 3. A sample Web page from DBLP.

and PublishedIn) in Publication sub-trees with the goal of making these sub-trees uniform in their first levels. This alternative representation preserves the semantics of the objects, but it is less concise than the first one. On the other hand, for our purposes, N presents an important property: it can be directly mapped into a nested table, such as the one presented in Fig. 5.

The table in Fig. 5 makes explicit an interesting characteristic of nested tables for the representation of semistructured Web data. Traditionally, nestings have the role of representing in a single column complex objects, i.e. non-atomic values. In our approach, we "overload" this structural feature by using it to also accommodate structural variations. This is what happens for the column PublishedIn, in which rows 1 and 5 store tables that have a structure distinct from the structure of the tables stored in rows 2, 3 and 4. Notice that, for the case of this specific example, each row corresponds to a single publication. Thus, this representation can be considered imprecise, since all tables stored under PublishedIn will actually have one single tuple each. From this simple example, we can conclude that nested tables are indeed less expressive than OEM for representing semistructured data, but in situations where typical Web data is to be represented, nested tables are a viable alternative.

To go further in this discussion, we now present a brief comparison with XML, which is currently the predominant formalism for representing Web data. We notice that most of the discussion we have presented so far in this section also applies to XML, since it is essentially a notation for representing labeled trees.

In Fig. 6(a) we present a DTD that declares the structure of an XML document corresponding to the labeled tree M of Fig. 4(a). Similarly, in Fig. 6(b) we present a DTD that declares the structure of an XML document corresponding to the labeled tree N of Fig. 4(b). Let us refer to these DTDs as D_M and D_N, respectively.

Notice that D_M and D_N define XML documents that are equivalent in the same sense as trees M and N are. Considering that DTDs are indeed context-free grammars and that XML documents (or OEM labeled trees) are derivations of such grammars, we can see D_N as the grammar that results from including

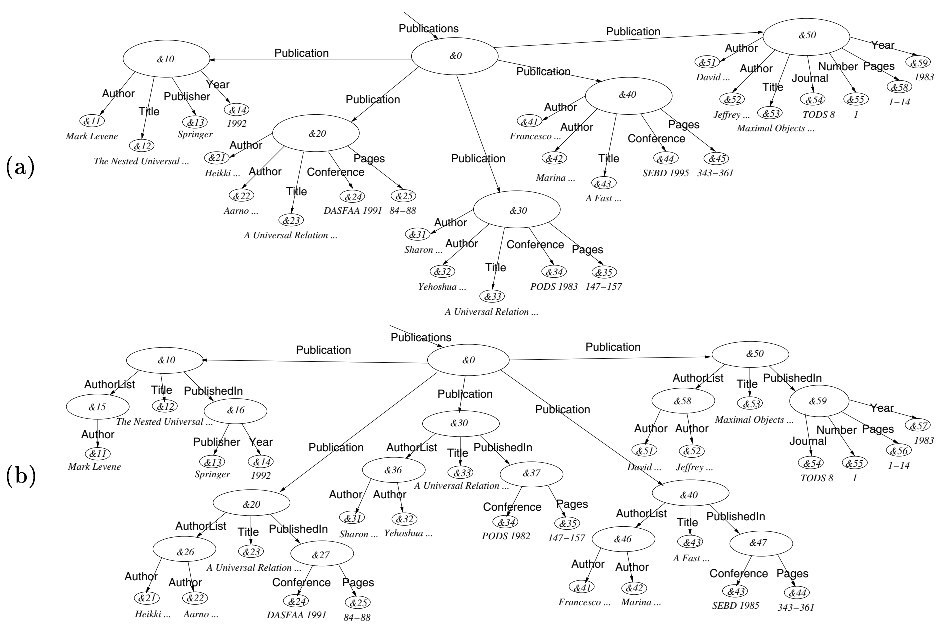
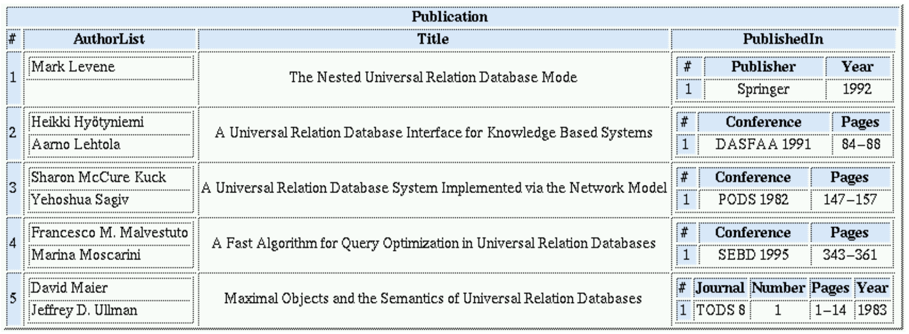


Fig. 4. Alternative OEM trees for the data in the page of Fig. 3.

in D_M a number of productions (or ELEMENT declarations) to ensure that the resulting documents or trees take a form similar to N and thus can be mapped to nested tables. More precisely, such trees would be considered as tables according to Definition 2.

Indeed, nested tables are less expressive than XML for representing Web data precisely because they can be described by a sub-class of DTDs such as D_N, which we refer to as *Tabular DTDs* or *TDTDs*. In DTDs, ELEMENT declarations are restricted to some pre-defined forms that guarantee that XML documents (or labeled trees) correspond to nested tables. In particular, TDTD's non-terminal ELEMENT declarations are restricted to be of one of the following forms.



Publication			
#	AuthorList	Title	PublishedIn
1	Mark Levene	The Nested Universal Relation Database Mode	# / Publisher / Year: 1 / Springer / 1992
2	Heikki Hyötyniemi; Aarno Lehtola	A Universal Relation Database Interface for Knowledge Based Systems	# / Conference / Pages: 1 / DASFAA 1991 / 84–88
3	Sharon McCure Kuck; Yehoshua Sagiv	A Universal Relation Database System Implemented via the Network Model	# / Conference / Pages: 1 / PODS 1982 / 147–157
4	Francesco M. Malvestuto; Marina Moscarini	A Fast Algorithm for Query Optimization in Universal Relation Databases	# / Conference / Pages: 1 / SEBD 1995 / 343–361
5	David Maier; Jeffrey D. Ullman	Maximal Objects and the Semantics of Universal Relation Databases	# / Journal / Number / Pages / Year: 1 / TODS 8 / 1 / 1–14 / 1983

Fig. 5. Data from the DBLP page of Fig. 3 organized into a nested table.

```
<!DOCTYPE dpub [
 <!ELEMENT Publications (Publication*)>
 <!ELEMENT Publication  (Author*, Title,
          ((Publisher,Year)|
          (Conference,Pages)|
          (Journal,Number,Pages,Year))>
 <!ELEMENT Publisher    (#PCDATA)>
 <!ELEMENT Author       (#PCDATA)>
 <!ELEMENT Title        (#PCDATA)>
 <!ELEMENT Conference   (#PCDATA)>
 <!ELEMENT Journal      (#PCDATA)>
 <!ELEMENT Number       (#PCDATA)>
 <!ELEMENT Pages        (#PCDATA)>
 <!ELEMENT Year         (#PCDATA)>
]>
```

(a)

```
<!DOCTYPE dpub [
 <!ELEMENT Publications (Publication*)>
 <!ELEMENT Publication  (AuthorList,Title,
                         PublishedIn)>
 <!ELEMENT Authorlist   (Author*)>
 <!ELEMENT PublishedIn  (PublishedIn1|
                         PublishedIn2|
                         PublishedIn3)>
 <!ELEMENT PublishedIn1 (Publisher,Year)>
 <!ELEMENT PublishedIn2 (Conference,Pages)>
 <!ELEMENT PublissheIn3 (Journal,Number,Pages,
                         Year)>
 <!ELEMENT Title        (#PCDATA)>
 <!ELEMENT Author       (#PCDATA)>
 <!ELEMENT Publisher    (#PCDATA)>
 <!ELEMENT Year         (#PCDATA)>
 <!ELEMENT Conference   (#PCDATA)>
 <!ELEMENT Pages        (#PCDATA)>
 <!ELEMENT Journal      (#PCDATA)>
 <!ELEMENT Number       (#PCDATA)>
]>
```

(b)

Fig. 6. DTDs for XML documents storing data from the DBLP page of Fig. 3.

- An ***aggregating*** (or ***tuple-generating***) declaration has the form `<!ELEMENT` X_0 $(X_1 \dots X_n)$`>` $(n \geq 2)$, where $X_i \neq X_j$, for every $0 \leq i, j \leq n$ except $i = j$. Further, each X_k, $k = 1 \dots n$, must appear on the left-hand side of exactly one iterating or terminal declaration;
- An ***iterating*** (or ***list-generating***) declaration has the form `<!ELEMENT` X $(Y*)$`>`, where $X \neq Y$. Further, Y must appear on the left-hand side of exactly one aggregating, varying, or terminal declaration;
- A ***varying*** (or ***variant-generating***) declaration has the form `<!ELEMENT` X_0 $(X_1| \dots |X_n)$`>` $(n \geq 2)$, where $X_i \neq X_j$, for every $0 \leq i, j \leq n$ except $i = j$. Further, each X_k, $k = 1 \dots n$, must appear on the left-hand side of exactly one aggregating or iterating declaration.

It is easy to see that limiting the possible `ELEMENT` declarations as described above considerably restricts the possible derivations (i.e. labeled trees or XML documents) that can be generated. However, it must be observed that formats such as XML are intentionally non-restrictive, since they do not aim at any application in particular. Indeed, XML can be used to represented typical Web data, such as the data found in the pages of Fig. 1 and Fig. 3, but it is also flexible enough to describe, for instance, DNA sequences, communication protocols or stylesheets. In this paper, we claim that for representing data typically found in *data-rich*, *ontologically narrow*, *multiple-record* Web pages, it is possible to use nested tables as we define them, without compromising the accuracy of the representation. Indeed, despite their relative lack of expressiveness, nested tables are expressive enough to represent a vast collection of different data available in Web pages, such as those in Fig. 1 and Fig. 3.

4 Query Operations for Nested Tables

In this section we present a set of operations for querying semistructured Web data represented as nested tables. Other operations for manipulating table schemes are omitted here but are part of a preliminary proposal we made and are described in [7].

The query operations presented here are *selection*, *projection*, *nest*, and *unnest*. The particular version of these operators can be regarded as extensions of the recursive query operations proposed by Colby [4] to query data in regular nested tables. We extended these operations to deal with semistructured data so that these operations adequately deal with irregularities such as structural variations and absence of values.

In the following, we describe each operation and present its formal definition. For the remainder of this section, we assume that all operations apply to an instance of a table scheme: $\tau = (C_1 : [\tau_1^1; \ldots; \tau_1^{n_1}], C_2 : [\tau_2^1; \ldots; \tau_2^{n_2}], \ldots, C_m : [\tau_m^1; \ldots; \tau_m^{n_m}])$ $(m \geq 2,\ n_k \geq 1,\ k = 1, \ldots, m)$. Before proceeding, we define two necessary auxiliary functions.

Definition 3 *We denote by $\Upsilon(T)$ a function that when applied to table $T : \tau$ returns its table scheme τ. Otherwise, if T is not a table, it returns a null value.*

Definition 4 *Let $t = \langle C_1 : v_1, ..., c_m : v_m \rangle$ be a tuple of a table $T : \tau$. Let $\{D_1 : u_1, ..., D_p : u_p\} \subseteq \{C_1 : v_1, ..., c_m : v_m\}$ be a subset of the columns of T. We use the notation $t[D_1, ..., D_p]$ to refer to tuple $\langle D_1 : u_1, ..., D_p : u_p \rangle$, composed of the values of the columns specified in t.*

Informally, Definition 4 captures the notion of a "sub-tuple" or "restriction" of a given tuple. This notation is common in the context of the traditional relational model.

Selection (σ). The selection operation allows selection conditions to be specified at any level of nesting. A selection condition is a boolean expression defined over the values in a column or over the existence of a column in an internal subtable (i.e. a *structural condition*). The operators used in selection conditions are $\{\supset, \supseteq, \in, <, >, =, \sim, *\}$. The symbols $\sim$ and $*$ are used to denote the pattern matching operator and the structural condition, respectively. The other symbols correspond to the operators of nested relational algebra.

The general form of the selection operation is $\sigma_{sel-cond}(T)$, where *sel-cond* represents a selection condition specified on an instance T of a table scheme τ. To formally define the selection operation, we first consider the following definition.

Definition 5 *A selection condition S specified on a table scheme τ is of the form $\mathrm{s} + \mathsf{L}$, where s represents the condition specified over the columns of τ and L is a list. L is called an* **internal list of conditions** *of τ if and only if (i) L is empty, or (ii) L is of the form $\{s_1 + C_1'.\mathsf{L}_1;\ s_2 + C_2'.\mathsf{L}_2; \ldots; s_l + C_l'.\mathsf{L}_l\}$, where $\{C_1', C_2', \ldots, C_l'\} \subseteq \{C_1, C_2, \ldots, C_m\}$ and $\forall C_i'$ $(0 \leq i \leq l)$, $\Upsilon(C_i')$ is a table scheme, each s_i is a condition specified on $\Upsilon(C_i')$, and L_i is an internal list of conditions of $\Upsilon(C_{i'})$.*

Intuitively, each element $s_i + C_i'.\mathsf{L}_i$ of an internal list of conditions limits the scope of a selection condition to C_i'.

As an example, assume that we are interested in instances of the table in Fig. 2 that represent popular music items by "Antonio Carlos Jobim". A possible internal list for conditions of the table Amazon is Store $=$ *Popular Music* $+$ $\{$ItemList.By $\sim$ *Antonio Carlos Jobim*$\}$, where the condition involves the column Store and an internal list of conditions is recursively specified on the column ItemList.

For the condition s_i to be satisfied by a row of the table, the row must contain all the columns involved in s_i and the corresponding boolean expression must evaluate to true. Based on this, we present the definition of the selection operation as follows:

Definition 6 *Let T be a table such that $T : \tau$, L an internal list of conditions for τ, and S a selection condition. The selection operation $\sigma_{\mathsf{S}}(T)$ is defined as follows:*

(i) if L is empty, then $\sigma_s(T) = \{t \in T \mid s(t) = \mathsf{true}\}$;

(ii) if L is not empty, then $\sigma_{s+\{s_1+C'_1.\mathsf{L}_1;\ldots;s_l+C'_l.\mathsf{L}_l\}}(T) = \{t \mid \exists t_r \in T \wedge$ $(t[\{C_1,\ldots,C_m\} - \{C'_1,\ldots,C'_l\}] = t_r[\{C_1,\ldots,C_m\} - \{C'_1,\ldots,C'_l\}] \wedge$ $(s(t_r) = \mathsf{true} \wedge ((t[C'_1] = \sigma_{s_1+C'_1.\mathsf{L}_1}(t_r[C'_1])) \wedge (t[C'_1] \neq \emptyset)) \wedge \ldots \wedge$ $((t[C'_l] = \sigma_{s_l+C'_l.\mathsf{L}_l}(t_r[C'_l])) \wedge (t[C'_l] \neq \emptyset)))\}$.

According to Definition 6, selection conditions are evaluated throughout the nested table, so that tables at any level of nesting can be addressed. In general, the selection operation generates a table with the same structure as the original table. Because of the semistructured nature of the queried data, however, the selection operation can generate a table whose structure is distinct from the structure of the original table. For instance, if we have specified a selection operation on the attribute Bid for the table of Fig. 2, only rows of the nested table ItemList that have this attribute would be considered. All other structural variations for the column ItemList would be discarded. As a consequence, the resultant table would have a structure distinct from the original table.

Projection (π). The projection operation is similar to the projection operation defined in relational algebra. We can say that this operation horizontally reduces a table by maintaining only the columns specified by the user (referred to as *projected columns*). The general form of the projection operation is $\pi_{col-set}(T)$, where *col-set* represents the set of projected columns of an instance T of τ. To formally define the projection operation, we first consider the following definition.

Definition 7 *A set of projected columns A is of the form $\mathsf{A}_\tau + \mathsf{P}$, where A_τ is a subset of the columns of τ and P is a list. P is called an* **internal list of projections** *of τ, if and only if, (i) P is empty, or (ii) P is of the form $\{\mathsf{A}_{C'_1} + C'_1.\mathsf{P}_1;\ \mathsf{A}_{C'_2} + C'_2.\mathsf{P}_2;\ldots;\mathsf{A}_{C'_l} + C'_l.\mathsf{P}_l\}$, where $\forall C'_i$ $(0 \leq i \leq l)$ $\{C'_1, C'_2, \ldots, C'_l\} \subseteq \{C_1, C_2, \ldots, C_m\}$, $\Upsilon(C'_i)$ is a table scheme, each $\mathsf{A}_{C_i'}$ is a set of columns in $\Upsilon(C'_i)$, and P_i is an internal list of projections of $\Upsilon(C_{i'})$.*

Similar to the internal list of conditions, we can say that each element $\mathsf{A}_{C'_i} + C'_i.\mathsf{P}_i$ of an internal list of projections determines that the instances of the columns represented by $\mathsf{A}_{C'_i}$ will be retrieved from the internal table defined in column C'_i. As an example, suppose that we want to project on the columns

Store, Item, and By of the table in Fig. 2. A possible internal list for these projected columns is $\{\{\mathsf{Store}\} + \mathsf{ItemList}.\{\mathsf{Item,By}\}\}$, where A_τ is a set composed only of the column Store and a list of internal projections is recursively specified on the column ItemList.

Definition 8 *Let T be a table such that $\Upsilon(T) = \tau$ and P be an internal list of projections of τ. The projection operation $\pi_{\mathsf{A}}(T)$ is defined as follows:*

(i) if P *is empty , then* $\pi_{\mathsf{A}_\tau}(\mathsf{T}) = \{t \mid (\exists t_r \in T \wedge \mathsf{A}_\tau \subseteq \tau \wedge t = t_r[\mathsf{A}])\}$;
(ii) if P *is not empty, then* $\pi_{\mathsf{A}_\tau} + \{\mathsf{A}_{C'_1 + C'_1.\mathsf{P}_1};\ldots;\mathsf{A}_{C'_l + C'_l.\mathsf{P}_l}\}(\mathsf{T}) =$

$$\{t \mid \exists t_r \in T \wedge (t[\mathsf{A}_\tau] = \pi_{\mathsf{A}_\tau}(\mathsf{T}) \wedge t[C'_1] = \pi_{\mathsf{A}_{C'_1} + C'_1.\mathsf{P}_1}(t[C'_1]) \wedge \ldots \wedge$$
$$t[C'_l] = \pi_{\mathsf{A}_{C'_l} + C'_l.\mathsf{P}_l}(t[C'_l]))\}.$$

According to Definition 8, to deal with the semistructured nature of the data, the projection operation recursively verifies the existence of each projected column in all rows of the table. As a consequence of the irregularities that might occur in a nested table, the result of a projection operation sometimes includes an "artificial" nesting level to accommodate the structural variations inherent in the projected columns. For instance, consider the nested table in Fig. 2. If we project on the column ItemList, we cannot group the resulting instances in a table because of the different structure in each variation. We would thus need an additional level of nesting to create a single table.

Nest (ν). Similar to the operation proposed by Thomas and Fischer [17], our nest operation has two distinct semantics. When applied to a single column, this operation groups the values in the column that are associated with equal values occurring in all other columns. Otherwise, when the nest operation is applied to a set of columns, it creates a new table scheme that groups the values of these columns and consequently generates a new level of nesting in the table. The general form of the nest operation is $\nu_{col-set}(T)$, where *col-set* represents a set of columns of T that will be nested.

Definition 9 *Let T be a table such that $\Upsilon(T) = \tau$ and A be a set of columns to be nested in T. The nest operation $\nu_{\mathsf{A}}(T)$ is defined as follows:*

$$\nu_{\mathsf{A}}(T) = \{t \mid \exists t_r \in T \wedge (((\mathsf{A} = \{c_1, \ldots, c_o\},\ (1 \le o \le m) \wedge \mathsf{A} \subseteq \{C_1, \ldots, C_m\})$$
$$\wedge\ \tau_{\mathsf{A}} = (c_1 : [\tau^1_{\mathsf{A}_1}; \ldots; \tau^{n_1}_{\mathsf{A}_1}], c_2 : [\tau^1_{\mathsf{A}_2}; \ldots; \tau^{n_2}_{\mathsf{A}_2}], \ldots, c_o : [\tau^1_{\mathsf{A}_o}; \ldots; \tau^{n_o}_o]) \wedge$$
$$((t[\{C_1, \ldots, C_m\} - \{c_1, \ldots, c_o\}] = t_r[\{C_1, \ldots, C_m\} - \{c_1, \ldots, c_o\}]) \wedge$$
$$(t[\tau_{\mathsf{A}}] = \{s[c_1, \ldots, c_o] \mid s \in T \wedge s[\{C_1, \ldots, C_m\} - \{c_1, \ldots, c_o\}] =$$
$$t_r[\{C_1, \ldots, C_m\} - \{c_1, \ldots, c_o\}]))) \vee (\exists C'_k\ (1 \le k \le m)$$
$$\wedge\ \Upsilon(C'_k) \text{ is a table scheme } \wedge \Upsilon(C'_k) \supseteq \mathsf{A} \wedge \nu_{\mathsf{A}}(C'_k)))\}.$$

According to Definition 9, the nest operation is recursively applied to a nested structure. This recursion navigates the nesting structure of the table until the columns to be nested are reached. Both of the two semantics are addressed by this definition. It is important to note that when a single column is nested, the table structure does not change, but all the values of this column are grouped, eliminating repeated values in other columns. In addition, this definition shows that when the nest operation is applied to a set of columns, it can only be executed if all of the chosen columns are defined in the table scheme τ.

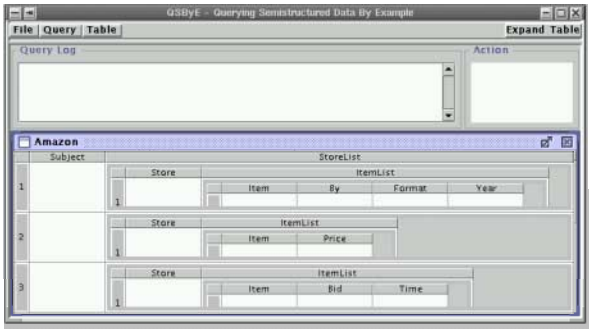


Fig. 7. Skeleton for the repository used in the query examples.

Unnest (μ). The unnest operation is the inverse of the nest operation. It also has two distinct semantics according to [17], and it must always be applied to a column whose contents is either a list of atoms or an internal table. If it is applied to a list of atoms, it will "ungroup" its elements, i.e. it will split them into different rows of the table. Otherwise, if it is applied to an internal table, it will eliminate a level of nesting. The general form of the unnest operation is $\mu_{col}(T)$, where *col* is the column to be unnested in T.

Definition 10 *Let T be a table such that $\Upsilon(T) = \tau$ and* a *be a column to be unnested in T. The unnest operation $\mu_{\mathsf{a}}(T)$ is defined as follows:*

$\mu_{\mathsf{a}}(T) = \{t \mid \exists t_r \in T \wedge \mathsf{a} = C_i \ (0 \leq i \leq m)$ *such that C_i is not an atom* $\wedge$ $(t[C_1, C_{i-1}, C_{i+1}, C_m] = t_r[C_1, C_{i-1}, C_{i+1}, C_m] \wedge \tau_i^j = \Upsilon(C_i) \ (0 \leq j \leq n)$, *such that* $\tau_i^j = (c_1 : [\gamma_1^1; \ldots; \gamma_1^{n_1}], \ldots, c_l : [\gamma_l^1; \ldots; \gamma_l^{n_l}]) \wedge t[c_1, \ldots, c_l] = t_s$, *where* $t_s \in C_i \vee \exists C_k'(1 \leq k \leq m) \wedge \Upsilon(C_k')$ *is a table scheme* $\wedge \ \Upsilon(C_k') \supseteq \mathsf{A} \wedge \mu_{\mathsf{a}}(C_k'))\}$.

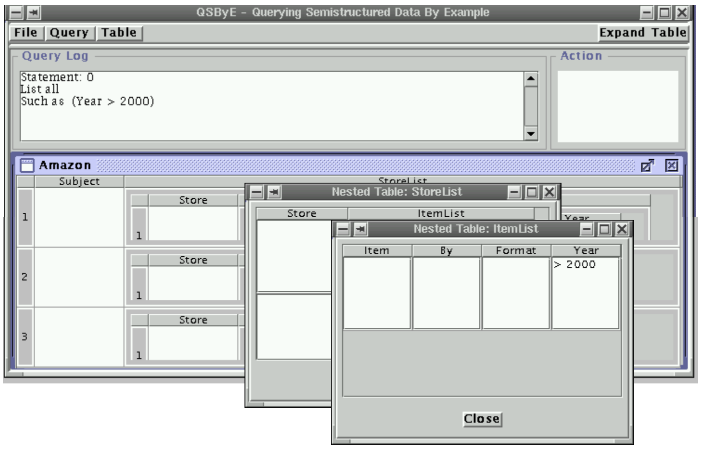
According to Definition 10, the unnest operation is recursively defined in a way similar to the nest operation. The only restriction refers to the situation where the chosen column presents internal tables with different structures. In this case, the unnest operation cannot be executed because it would generate a result whose scheme would not conform with Definition 1.

5 QSByE

In this section, we present QSByE (*Querying Semistructured data By Example*) [7], a query interface that adopts a variation of the QBE paradigm [18]. QSByE combines features of query languages for semistructured data, such as type coercion and path expressions, with features of the original QBE language to query semistructured Web data extracted by the DEByE tool [11]. When using QSByE, users formulate their queries by means of graphical facilities provided by the interface, which makes the syntax of the operations defined in Section 4 completely transparent to them.

To show the suitability of QSByE for semistructured data, and to illustrate the process of query formulation, we use three examples. We assume that these queries are executed over a repository containing data extracted from a set of Web pages similar to the page of Fig. 1. The data in this repository are structured according to the example in Fig. 2. Our three example queries are as follows:

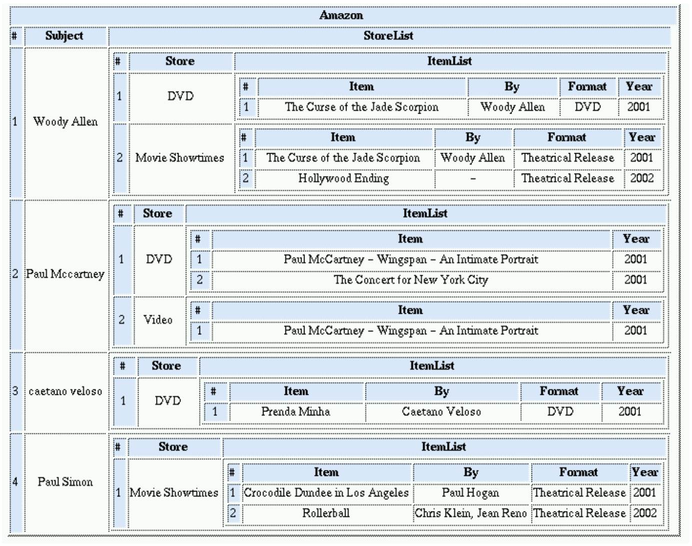
(a)



```
Statement: 0
List all
Such as (Year > 2000)
```

Year: > 2000

(b)



Amazon

#	Subject	#	Store	#	Item	By	Format	Year
1	Woody Allen	1	DVD	1	The Curse of the Jade Scorpion	Woody Allen	DVD	2001
		2	Movie Showtimes	1	The Curse of the Jade Scorpion	Woody Allen	Theatrical Release	2001
				2	Hollywood Ending	-	Theatrical Release	2002
2	Paul Mccartney	1	DVD	1	Paul McCartney – Wingspan – An Intimate Portrait			2001
				2	The Concert for New York City			2001
		2	Video	1	Paul McCartney – Wingspan – An Intimate Portrait			2001
3	caetano veloso	1	DVD	1	Prenda Minha	Caetano Veloso	DVD	2001
4	Paul Simon	1	Movie Showtimes	1	Crocodile Dundee in Los Angeles	Paul Hogan	Theatrical Release	2001
				2	Rollerball	Chris Klein, Jean Reno	Theatrical Release	2002

Fig. 8. Specification and result of Query $\mathbf{Q}_1$.

$\mathbf{Q}_1$: *List all information on items released after 2000.*
$\mathbf{Q}_2$: *Show information on all items related to “Garfunkel”.*
$\mathbf{Q}_3$: *For each store, list the subjects for whom products are available.*

To formulate queries using QSByE, the user first selects a repository containing the data of interest. This immediately provides the structure of the stored data in this repository as a nested table skeleton. Fig. 7 shows the skeleton for the repository used in our examples.

Using the skeleton provided as an aid to query formulation, users just have to choose the operations required to formulate the query. For query $\mathbf{Q}_1$, the user specifies the selection condition on the column Year, as Fig. 8(a) shows.

This yields the result in Fig. 8(b). Notice that, for this query, no projection conditions are specified. Thus, the result shows the complete objects satisfying the selection condition.

By looking into the contents of the "Query Log" window in the screen shown in Fig. 8(a), we see a textual description of the operation executed. Such descriptions are generated automatically by QSByE for each query executed, as a way of logging the queries. They can be also observed in the execution of the next example queries.

An important requirement of query tools and languages for semistructured data is allowing the specification of selection conditions disregarding the details of the scheme. We use query $\mathbf{Q}_2$ to show how this is accomplished with QSByE.

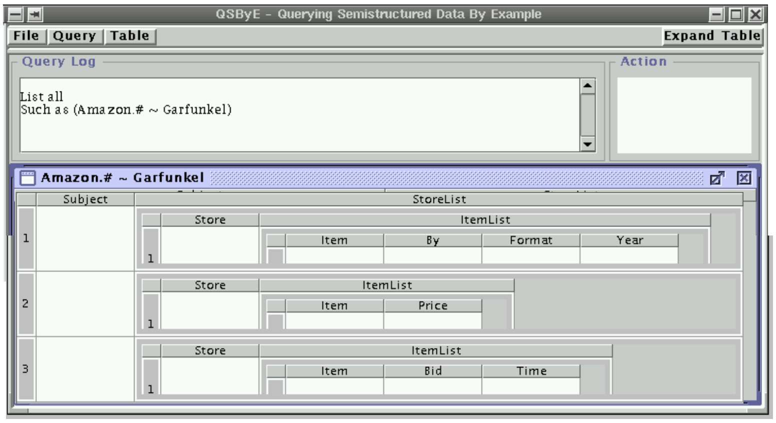
As Fig. 9(a) illustrates, to formulate this query, we specify the selection condition on the outermost table header (labeled Amazon), meaning that this selection condition must be applied to all columns under this header. This feature explores the whole nested structure of the table in a way similar to what is done in regular path expressions on typical semistructured query languages. Graphically, in QSByE, this condition is represented by the symbol "#", as indicated in the header of the table presented in Fig. 9(a). This flexibility is suitable for semistructured data because of possible distinct occurrences of the same value in different places in a data source. Also, in this query selection condition we have used the $\sim$ operator, which provides the flexibility of pattern matching and allows errors, therefore making it possible to retrieve the required data even if the condition value has been misspelled. Fig. 9(b) shows the query result. Our next query, $\mathbf{Q}_3$, described in Fig. 10(a) through Fig. 10(f), exemplifies the use of the projection and nest/unnest operations. The result is the table in Fig. 10(d), where each subject is associated with the list of stores having its products.

However, the statement of the query requires this table to be rearranged. For this, we first apply an unnest operation over the table of Fig. 10(d). This is accomplished by pressing the button "Table" to select the the unnest operation and then clicking on the header of the Store column, which causes it to be marked with a "(u)". The result is the table in Fig. 10(e). Finally, we apply a nest operation over this table by again pressing the button "Table" to select the nest operation and then clicking on the header of the Subject column, which causes it to be marked with an "(n)". The result of this last operation is shown in Fig. 10(f).

6 Conclusions and Directions for Future Work

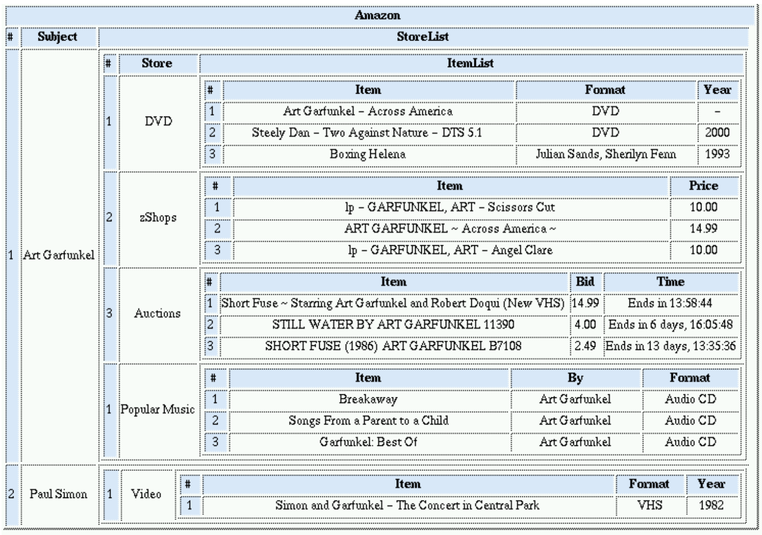
We have formalized nested tables with varying internal structure. This formalization generalizes nested tables for semistructured Web data by allowing distinct rows to contain objects with distinct structures. We have shown how to use these nested tables to represent and query semistructured data extracted from the Web. We extended nested relational operators to manage the variable nestings we defined, and we implemented QSByE to allow a user to query tables with variable internal structure in a QBE-like manner. Preliminary experiments with QSByE [7] demonstrate that with a small amount of training even unskilled

(a)



```
List all
Such as (Amazon.# ~ Garfunkel)
```

(b)



Amazon

#	Subject	#	Store	#	Item	Format	Year
1	Art Garfunkel	1	DVD	1	Art Garfunkel - Across America	DVD	-
				2	Steely Dan - Two Against Nature - DTS 5.1	DVD	2000
				3	Boxing Helena	Julian Sands, Sherilyn Fenn	1993

#	Subject	#	Store	#	Item	Price
1	Art Garfunkel	2	zShops	1	lp - GARFUNKEL, ART - Scissors Cut	10.00
				2	ART GARFUNKEL ~ Across America ~	14.99
				3	lp - GARFUNKEL, ART - Angel Clare	10.00

#	Subject	#	Store	#	Item	Bid	Time
1	Art Garfunkel	3	Auctions	1	Short Fuse ~ Starring Art Garfunkel and Robert Doqui (New VHS)	14.99	Ends in 13:58:44
				2	STILL WATER BY ART GARFUNKEL 11390	4.00	Ends in 6 days, 16:05:48
				3	SHORT FUSE (1986) ART GARFUNKEL B7108	2.49	Ends in 13 days, 13:35:36

#	Subject	#	Store	#	Item	By	Format
1	Art Garfunkel	1	Popular Music	1	Breakaway	Art Garfunkel	Audio CD
				2	Songs From a Parent to a Child	Art Garfunkel	Audio CD
				3	Garfunkel: Best Of	Art Garfunkel	Audio CD

#	Subject	#	Store	#	Item	Format	Year
2	Paul Simon	1	Video	1	Simon and Garfunkel - The Concert in Central Park	VHS	1982

Fig. 9. Specification and result of Query $\mathbf{Q}_2$.

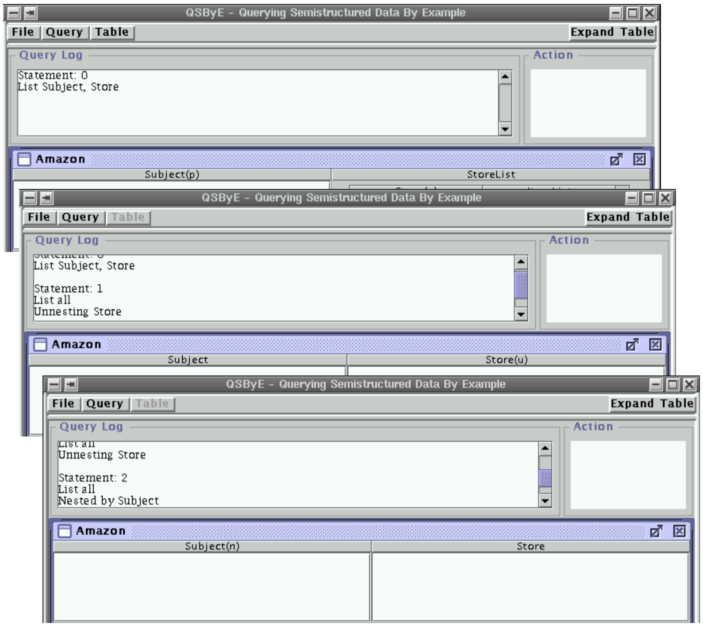
users can use our interface to query semistructured data extracted from Web pages.

An exciting direction for future work is to rearrange our query interface to allow a user to directly query a semistructured, multiple-record Web document. Instead of first extracting data into nested tables using DEByE [11], we can allow a user to pose a query directly to a Web page. To pose a direct query, a user would specify a nested table, choosing header names and a nesting structure to suit the query and would fill in selection conditions as explained in this paper. To complete the query, the user would specify *real examples* from the Web page by highlighting data values on the page with a mouse and dragging them into the nested query table. This action would invoke DEByE, causing it to analyze the Web page and extract data values as explained in [11], and then invoke the backend selection processing described in this paper to limit the extracted values

(a)

(b)

(c)



(a)

```
Statement: 0
List Subject, Store
```

Subject(p) | StoreList

(b)

```
List Subject, Store

Statement: 1
List all
Unnesting Store
```

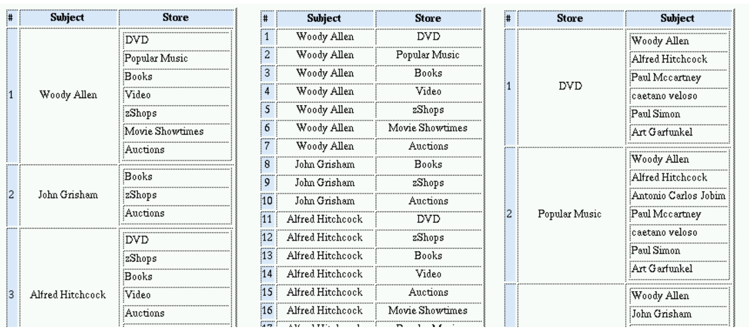
Subject | Store(u)

(c)

```
List all
Unnesting Store

Statement: 2
List all
Nested by Subject
```

Subject(n) | Store



(d)

#	Subject	Store
1	Woody Allen	DVD
		Popular Music
		Books
		Video
		zShops
		Movie Showtimes
		Auctions
2	John Grisham	Books
		zShops
		Auctions
3	Alfred Hitchcock	DVD
		zShops
		Books
		Video
		Auctions

(e)

#	Subject	Store
1	Woody Allen	DVD
2	Woody Allen	Popular Music
3	Woody Allen	Books
4	Woody Allen	Video
5	Woody Allen	zShops
6	Woody Allen	Movie Showtimes
7	Woody Allen	Auctions
8	John Grisham	Books
9	John Grisham	zShops
10	John Grisham	Auctions
11	Alfred Hitchcock	DVD
12	Alfred Hitchcock	zShops
13	Alfred Hitchcock	Books
14	Alfred Hitchcock	Video
15	Alfred Hitchcock	Auctions
16	Alfred Hitchcock	Movie Showtimes

(f)

#	Store	Subject
1	DVD	Woody Allen
		Alfred Hitchcock
		Paul Mccartney
		caetano veloso
		Paul Simon
		Art Garfunkel
2	Popular Music	Woody Allen
		Alfred Hitchcock
		Antonio Carlos Jobim
		Paul Mccartney
		caetano veloso
		Paul Simon
		Art Garfunkel
		Woody Allen
		John Grisham

(d) (e) (f)

Fig. 10. Specification and result of Query $\mathbf{Q}_3$.

to those satisfying the query specification. Thus, users would be able to directly specify a QBE-like query over a Web page and obtain an answer. Carrying this future work a bit further, we could obtain the results of a specified QBE-like query from several Web pages containing similar application data and merge obtained results into one final answer.

References

1. Bonifati, A., and Ceri, S. Comparative analysis of five XML query languages. *SIGMOD Record 29*, 1 (2001), 68–79.
2. Buneman, P., Davidson, S. B., Hillebrand, G. G., and Suciu, D. A Query Language and Optimization Techniques for Unstructured Data. In *Proceedings*

of the 1996 ACM SIGMOD International Conference on Management of Data (Quebec, Canada, 1996), pp. 505–516.

3. Buneman, P., Deutsch, A., and Tan, W. A Deterministic Model for Semistructured Data. In *Proceedings of the Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats* (Jerusalem, Israel, 1999).
4. Colby, L. S. A Recursive Algebra and Query Optimization for Nested Relations. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data* (Portland, Oregon, 1989), pp. 273–283.
5. Deutsch, A., Fernandez, M. F., and Suciu, D. Storing Semistructured Data with STORED. In *Proceedings the 1999 ACM SIGMOD International Conference on Management of Data* (Philadephia, Pennsylvania, 1999), pp. 431–442.
6. Embley, D., Campbell, D., Jiang, Y., Liddle, S., Lonsdale, D., Ng, Y.-K., and Smith, R. Conceptual-model-based data extraction from multiple-record Web pages. *Data & Knowledge Engineering 31*, 3 (1999), 227–251.
7. Evangelista-Filha, I. M. R., Laender, A. H. F., and Silva, A. S. Querying Semistructured Data By Example: The QSByE Interface. In *Proceedings of the International Workshop on Information Integration on the Web* (Rio de Janeiro, Brazil, 2001), pp. 156–163.
8. Florescu, D., Levy, A., and Mendelzon, A. Database Techniques for the World-Wide Web: A Survey. *SIGMOD Record 27*, 3 (1998), 59–74.
9. Goldman, R., and Widom, J. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In *Proceedings of the 23rd International Conference on Very Large Data Bases* (Athens, Greece, 1997), pp. 436–445.
10. Jaeschke, G., and Schek, H.-J. Remarks on the Algebra of Non First Normal Form Relations. In *Proceedings of the ACM Symposium on Principles of Database* (Los Angeles, California, 1982), pp. 124–138.
11. Laender, A. H. F., Ribeiro-Neto, B., and da Silva., A. S. DEByE – Data Extraction By Bxample. *Data and Knowledge Engineering 40*, 2 (2002), 121–154.
12. Libkin, L. A Relational Algebra for Complex Objects Based on Partial Information. In *Proceedings of the 3rd Symposium on Mathematical Fundamentals of Database and Knowledge Bases Systems* (Rostock, Germany, 1991), pp. 29–43.
13. Lorentzos, N. A., and Dondis, K. A. Query by Example for Nested Tables. In *Proceedings of the 9th International Conference on Database and Expert Systems Applications* (Vienna, Austria, 1998), pp. 716–725.
14. Makinouchi, A. A Consideration on Normal Form of Not-Necessarily-Normalized Relation in the Relational Data Model. In *Proceedings of the 3rd International Conference on Very Large Data Bases* (Tokyo, Japan, 1977), pp. 447–453.
15. McHugh, J., Abiteboul, S., Goldman, R., Quass, D., and Widom, J. Lore: A Database Management System for Semistructured Data. *SIGMOD Record 26*, 3 (1997), 54–66.
16. Papakonstantinou, Y., Garcia-Molina, H., and Widom, J. Object Exchange Across Heterogeneous Information Sources. In *Proceedings of the 11th International Conference on Data Engineering* (Taipei, Taiwan, 1995), pp. 251–260.
17. Thomas, S. J., and Fischer, P. C. Nested Relational Structures. *Advances in Computing Research 3* (1986), 269–307.
18. Zloof, M. M. Query-by-Example: A Data Base Language. *IBM Systems Journal 16*, 4 (1977), 324–343.

On the Transformation of Object-Oriented Conceptual Models to Logical Theories

Pieter Bekaert*, Bert Van Nuffelen, Maurice Bruynooghe, David Gilis, and Marc Denecker

Katholieke Universiteit Leuven, Department of Computer Science
Celestijnenlaan 200A, 3001 Leuven, Belgium
{pieter.bekaert,bert.vannuffelen,maurice.bruynooghe, david.gilis,marc.denecker}@cs.kuleuven.ac.be
http://www.cs.kuleuven.ac.be/~dtai/kt/btw/

Abstract. This paper describes a semi-automatic transformation from object-oriented conceptual models to logical theories. By associating a logical theory with a conceptual model, we are able to combine the best of both worlds. On one hand, the object-oriented software development paradigm is recognized to be well-suited to build maintainable and communicable conceptual models. On the other hand, the logical programming paradigm offers very powerful and semantically founded concepts to represent knowledge and the use of logical inference systems makes it possible to prototype solutions to computational tasks. Since our method makes this mapping from conceptual models to logical theories traceable, dealing with the evolution of the problem domain and requirements becomes more manageable. Moreover a path is offered towards building prototypes for object-oriented conceptual models.

1 Introduction

Recently, there is a lot of research interest for providing formal semantics for Object-Oriented modeling languages, in particular for the Unified Modeling Language [16]. Most OO modeling languages lacked formal semantics, often introducing ambiguity and imprecision in the meaning of a model. Defining a transformation to another formalism is one approach to define semantics. For example, a translation from UML models to Z [6] has been investigated as part of the Precise UML project. Executable specifications, allowing validation and rapid prototyping, is another important issue of investigation [11]. It is however important to retain the qualities of good OO conceptual modeling, such as declarativity.

We believe that a mapping of conceptual models to logical theories, if done properly, gives both a firm semantics to the OO conceptual modeling language in question and creates an executable specification. The logic programming

* Research Assistant of the Fund for Scientific Research - Flanders (Belgium) (F.W.O. - Vlaanderen)

paradigm, in particular, offers very powerful and semantically founded concepts to represent knowledge about a problem domain. The construction and manipulation of the logical theories, however, can be a complex activity. OO analysis methods are better suited to build maintainable and communicable conceptual models of the problem domain.

A key quality of such a transformation is traceability. This offers an explicit connection between the conceptual model and the executable specification. The lack of traceability between different views of a system in development or maintenance creates a lot of problems and inconsistencies, especially when the requirements evolve. In a traceable process, changes to models and decisions are propagated in a manageable way.

Our approach starts with building a conceptual model with EROOS, an OO conceptual modeling method. This model then is transformed into an ID-Logic logical theory that forms the basis for prototyping the implementation of the software via the use of a logical inference system, the $\mathcal{A}$system. In Sect. 2, we give an overview of the EROOS method for OO conceptual modeling and introduce an example model that will be used in the rest of the paper. In Sect. 3, ID-Logic is presented as a language for representing logical theories. Section 4 describes the transformation. Section 5 introduces some typical tasks and gives an outline of how they can be solved using the $\mathcal{A}$system general solver. Finally, we conclude and set the directions for further research.

2 Object-Oriented Conceptual Modeling in EROOS

The EROOS method (An Entity Relationship based Object-Oriented Specification method [20, 18, 17]) is an OO analysis method for building conceptual models (augmented with behavior). EROOS uses a notation with a formal and declarative nature. The modeling constructs introduced by the method offer strong guidance to the modeler, imposing a number of methodological rules. These restrict the expressivity of the modeling language, ruling out inferior ways to model reality. In this way, EROOS differs significantly from most other notations and methods such as UML [16] and RUP [12] or as in e.g. [14].

The example model used throughout the paper represents a simplified part of an educational resource management system [4]. The purpose of the system is amongst others to administer which teaching assistant has to teach which exercise sessions. The graphical representation of the model is given in Fig. 1.

An EROOS conceptual model introduces a number of classes corresponding with the concepts that are identified in the problem domain, e.g. COURSE, SESSION, EXERCISE TEACHER. A class can be refined at most once with a (unary or binary) relation involving (one or two) participating classes.[1] This introduces an existential dependence for each object of the refined class upon a single object of each participating class. A relationship is thus reified, unlike in most other methods, where relations or associations are modeled between classes as such.

[1] This methodological rule is a driving force in finding and organizing classes, leading to better models.

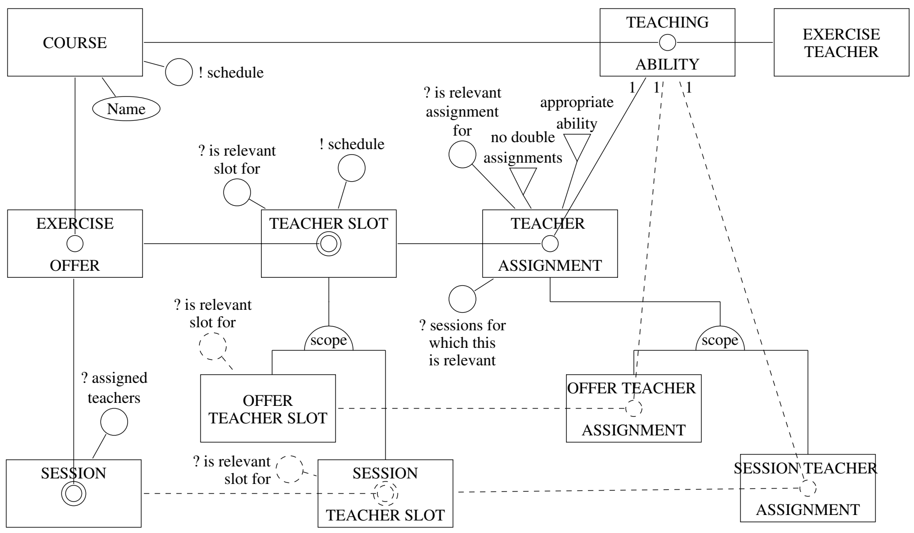


Fig. 1. An ERŒS conceptual model for an exercise teacher assignment system

Relations are represented by means of a circle in the refined class, with lines towards the participating classes. A single circle indicates a multiplicity restricted to one duplicate: e.g. at most one *teaching ability* can exist having the same participant tuple, i.e. the same *exercise teacher* and the same *course*. Unrestricted multiplicity is indicated with two circles (e.g. TEACHER SLOT). A "1" at the end of a participant line of a binary relation denotes a restricted connectivity: e.g. given a *teacher slot*, this object can be related to *at most* one *teaching ability* through *teacher assignments*.

Attributes can be defined for classes, introducing an existential dependence for their objects upon a value of a given domain. In the example, each *course* must have a *name* with the domain COURSE NAME as type.

ERŒS emphasizes the formal specification of all constraints observable in the problem domain [19]. This can be done implicitly through the semantics of the constructs (e.g. existential dependence) or integrated in the definition of model elements (e.g. multiplicity and connectivity of relations). If not possible otherwise, explicit constraints can be specified on top of one or more classes, introducing a universally quantified logical assertion. The population of a model must at any instance of time obey all constraints.

Explicit constraints are represented by means of triangles connected to the classes to which they apply. The constraint *appropriate ability* states that for each *teacher assignment*, the *course* upon which the participating *teaching ability* depends must be the same as the *course* that is reached via the *teacher slot* and *exercise offer*. The constraint *no double assignments* states that the same *exercise teacher* cannot be assigned more than once to teach a certain *session* (via different *teacher slots*).

Building class hierarchies using generalization/specialization is an important aspect in OO conceptual modeling. In ERŒS every characteristic of a generalization class is inherited in a specialization class, introducing an *is a* relationship and thus supporting polymorphic reasoning. Characteristics can be strengthened in specialization classes. Specialization classes are always grouped in a (mathematical) partition of the generalization class, turning the latter into an abstract class. ERŒS supports multiple specialization through the definition of multiple orthogonal partitions [2].

A partition is indicated using a half circle. Dashed lines are used to represent that inherited characteristics are strengthened. In the model, two partitions are introduced. They model that a *teacher slot*, and consequently a *teacher assignment*[2] can exist for the entire *exercise offer* (and thus for all it's sessions) or for a particular *session*. Note that these *slots* and *assignments* are additive.

Queries are defined for modeling derived information. They are denoted using circles with question marks, connected to the class on which they are defined. Queries can be applied to individual objects of this class. The query *assigned teachers* specifies which *teachers* teach a given *session* in terms of the other queries. (The queries with a name ending with "for" take a *session* as an argument.)

The evolution of the state in the problem domain is modeled by the occurrence of events upon objects. The specification of these events is part of an ERŒS model. Events have a transactional, declarative nature. The *schedule* events in the figure, create assignments for all resp. a given vacant slot for a course. This can be specified elegantly using non-determinism [3]. In this paper, we will not describe the formalism for specifying queries, events and constraints.

Semantics for ERŒS conceptual models, including non-deterministic behavior, have been defined in terms of populations and population evolutions, using set-theory and logic [3].

3 Logical Theories

3.1 The Language and Its Semantics

Inductive Definition Logic (ID-Logic [7, 8]) is an integration of first order logic and Logic Programming[3], interpreting the logic program as a non-monotone inductive definition. An ID-Logic theory is defined as a pair $(\mathcal{D},\mathcal{F})$ where $\mathcal{F}$ is a set of first order statements, expressing true assertions of the domain of discourse. $\mathcal{D}$ is a logic program and represents a (non-monotone) inductive definition of a subset of the predicates.[4] These are the *defined* predicates. Only defined predicates

[2] This second partition is not strictly needed in the simplified model, but is kept for illustrative purposes.

[3] Consequently ID-Logic generalizes all subclasses of Logic Programming, e.g. datalog.

[4] In the context of this paper, an ID-Logic theory contains only one definition (the logic program). In general [7], it can contain an arbitrary number of independent definitions.

appear in rule heads. The other predicates are called *open*. Their interpretation is free.

The semantics of an ID-Logic theory are given by an interpretation M of $\mathcal{D}$ satisfying the following conditions. Let M_O be an arbitrary two-valued interpretation for the open predicates in $\mathcal{D}$, fixing their meaning. This interpretation M_O turns $\mathcal{D}$ in a standard logic program. The unique two-valued well-founded model[5] [21] of this resulting logic program is M. M is only a model of the whole ID-Logic theory if it also is a model for $\mathcal{F}$.

As a consequence, the formulas in $\mathcal{F}$ constrain the possible interpretations of the open predicates.

$$\left\{\begin{array}{l}
\text{defined(teacher\_assigment(\_,\_)).} \\
\text{defined(teacher(\_)).} \\
\text{open(offer\_teacher\_assignment(\_,\_)).} \\
\text{open(session\_teacher\_assignment(\_,\_)).} \\
\hline
\text{teacher\_assigment(Slot,Teacher)} \leftarrow \text{ offer\_teacher\_assignment(Slot,Teacher).} \\
\text{teacher\_assigment(Slot,Teacher)} \leftarrow \text{ session\_teacher\_assignment(Slot,Teacher).} \\
\text{teacher(bert).} \\
\text{teacher(pieter).} \\
\hline
\forall \text{ Slot,Teacher: teacher\_assignment(Slot,Teacher)} \rightarrow \text{teacher(Teacher).}
\end{array}\right\}$$

Fig. 2. An ID-Logic theory

Figure 2 shows an example of an ID-Logic theory. For readability, the different parts are separated. The distinction between defined and open predicates is made explicit by the declarations in the upper part. In the middle, rules are given for the defined predicates. One is given in terms of other predicates and the other as a full enumeration. The bottom part contains the FOL formula that the second argument of `teacher_assigment/2` must be a teacher. Remark that this theory is not the result of the transformation that will be defined later on.

3.2 Computing Models Using the $\mathcal{A}$system

Given an ID-Logic theory, a basic computational task is to find an interpretation Δ for the open predicates that is a solution. (This Δ can be specified by the set of atoms that are true in the interpretation, the atoms outside the set Δ are false.) This task is known as abduction and is studied in abductive logic programming [10]. However, an ID-Logic theory is also a generalization of standard logic programming. There, the basic computational task is deduction: Given a query $\leftarrow Q$ (with Q a conjunction of literals), the task is to find a substitution σ such that $Q\sigma$ is true in the unique model of the program.

The $\mathcal{A}$system [13, 1] is able to combine both computational tasks, because it is an implementation of the SLDNFA procedure [9]. That SLDNFA procedure

[5] This is exactly the least Herbrand model in the case of a definite program.

is a sound and complete abductive extension of the SLDNF procedure that is at the basis of any implementation of logic programming. Given an ID-Logic theory $(\mathcal{D},\mathcal{F})$ and a query $\leftarrow Q$, SLDNFA computes a Δ and a substitution σ (if they exist, otherwise it fails) such that (1) the well-founded model of $(\mathcal{D} \cup \Delta)$ is two-valued and (2) $Q\sigma$ and $\mathcal{F}$ are true in the well-founded model of $(\mathcal{D} \cup \Delta)$.

The $\mathcal{A}$system has been used in a number of experiments [13], including computationally hard problems like AI-planning.

3.3 Extensions of ID-Logic

In the above discussion, we have formulated a basic version of ID-Logic and omitted a number of aspects which are of minor importance for the understanding of this paper. Besides the formulated core, ID-Logic is extended with a number of features that allow its users to express certain knowledge in a more concise and more readable way. Two features we make use of are type declarations and function declarations. These declarations enhance the readability of the language and they are exploited by the solvers to optimize the reasoning process.

In ID-Logic, a type declaration introduces a unary predicate. Figure 3 shows the unary predicate `course/1`. The actual definition of the predicate is as any defined predicate and belongs to the $\mathcal{D}$ part of the theory. It can be defined by an enumeration of facts, in terms of other predicates or declared open.

$$\left\{ \begin{array}{l} \text{type course.} \\ \text{function number\_of\_attendees(course):int.} \\ \text{predicate follows\_course(student,course).} \end{array} \right\}$$

Fig. 3. Type and function declarations in ID-Logic

Figure 3 also declares a function. On one hand, a function introduces an alternative notation for a predicate, here a predicate `number_of_attendees/2`, on the other hand it also introduces a formula in $\mathcal{F}$, namely the functional dependence between the arguments of the function and its result. It also introduces a formula restricting the domain and range of the function to the declared types. Also predicate arguments can be restricted to a declared type as the example shows.

Remark that both extensions can be compiled away to an equivalent type and function free theory. It is this theory that the $\mathcal{A}$system will take as input.

4 Transformation of ERŒS Conceptual Models to ID-Logic Theories

In this section we describe the transformation of ERŒS conceptual models to ID-Logic theories. A tool has been developed that performs a large part of this transformation automatically.

EROOS conceptual models are built using a conceptually rich set of constructs. In the process of the transformation to ID-Logic, we must define how to represent them in terms of definitions and FOL formulae. The theory that results is restricted to the structure of the problem domain, and does not contain information about the actual objects that are present. Neither does it declare predicates as defined or open. These aspects are only introduced in the context of tasks (see Sect. 5).

The EROOS events are currently not taken into account in the transformation, since the approach in logical programming for behavior is significantly different. The relation between both approaches is under investigation.

The transformation is presented in an informal way and illustrated in the context of the example introduced in Sect. 2. The transformation of all concepts is however clearly described and covered by the example.

4.1 Transformation of Classes and Domains

For each class in the conceptual model, a type is defined in the theory. This introduces a unary predicate that in a certain state of the system evaluates to true for those atoms that actually represent an object of that class. In the same way, types are defined for the domains of the attributes introduced in the conceptual model. Figure 4 shows the types generated for the conceptual model in Fig. 1.[6] To reduce the length of the lines, we abbreviated the symbol names.

$$\left\{\begin{array}{l}\text{type courseName.}\\ \text{type course.}\\ \text{type session.}\\ \text{type teachAbil.}\\ \text{type teachSlot.}\\ \text{type offTeachSlot.}\\ \ldots\end{array}\right\}$$

Fig. 4. The ID-Logic theory: types

The type system introduced in Sect. 3, is very simple, allowing us to explicitly state the properties of the type system of EROOS (see Sect. 4.3).

4.2 Transformation of Relations and Attributes

Each refinement of a class with a relation gives rise to one or two functions (for unary resp. binary relations) expressing the existential dependences introduced by the refinement. A function is also introduced for each attribute. The domain of these functions is the type for the class that is refined with the relation or decorated with the attribute. The range is the type for the participating class resp. the domain of the attribute. The functions are shown in Fig. 5.

[6] We will only present some partial fragments. The full theory can be found in [5].

$$\left\{\begin{array}{l}
\text{function name(course):courseName.}\\
\text{function exerOffer\_course(exerOffer):course.}\\
\text{function session\_exerOffer(session):exerOffer.}\\
\text{function teachAbil\_exerTeacher(teachAbil):exerTeacher.}\\
\text{function teachAbil\_course(teachAbil):course.}\\
\text{function teachSlot\_exerOffer(teachSlot):exerOffer.}\\
\text{function sessTeachSlot\_session(sessTeachSlot):session.}\\
\ldots
\end{array}\right\}$$

Fig. 5. The ID-Logic theory: functions for existential dependences

For binary relations, an alternative would be to introduce a ternary predicate instead of two functions. We would then need to impose functional dependences explicitly. The current representation is closer to ERŒS.

The multiplicity and connectivity constraints integrated in the definition of the relation, are also added to the theory. This is shown in Fig. 6.

$$\left\{\begin{array}{l}
\text{\% Multiplicity constraint for binary refined TEACHING ABILITY}\\
\forall\ \text{X1,X2} :: \text{teachAbil(X1)} \land \text{teachAbil(X2)} \rightarrow\\
\qquad (\text{teachAbil\_exerTeacher(X1)} = \text{teachAbil\_exerTeacher(X2)} \land\\
\qquad \text{teachAbil\_course(X1)} = \text{teachAbil\_course(X2))}\\
\qquad \rightarrow \text{X1} = \text{X2).}\\
\ldots\\
\text{\% Connectivity constraint for TEACHER ASSIGNMENT}\\
\forall\ \text{X1,X2} :: \text{teachAsgmt(X1)} \land \text{teachAsgmt(X2)} \rightarrow\\
\qquad (\text{teachAsgmt\_teachSlot(X1)} = \text{teachAsgmt\_teachSlot(X2)} \rightarrow\\
\qquad \text{teachAsgmt\_teachAbil(X1)} = \text{teachAsgmt\_teachAbil(X2)).}\\
\ldots
\end{array}\right\}$$

Fig. 6. The ID-Logic theory: integrated constraints for relations

4.3 Transformation of Generalization/Specialization

The primary characteristic of Generalization/Specialization, is that it introduces an *is a* relationship between the objects of a specialization class and a generalization class. This is also indicated in the ID-Logic theory. In ERŒS however, generalization/specialization is a rich concept, that introduces a lot more. In our approach, we fully transform each class in isolation, and then relate them in the theory. An alternative is to define all generalization classes in terms of their specializations. The transformation presented below equally applies to multiple generalization/specialization hierarchies, as elaborated in [4].

Partitioning Constraints. As explained in Sect. 2, specialization classes are grouped in mathematical partitions of generalization classes. The implicit disjointness and completeness constraints are made explicit in the theory. This is shown, together with the *is a* relationship, in Fig. 7.

$$\left\{\begin{array}{l} \forall\ \mathrm{X} :: \mathrm{offTeachSlot(X)} \rightarrow \mathrm{teachSlot(X)}. \\ \forall\ \mathrm{X} :: \mathrm{sessTeachSlot(X)} \rightarrow \mathrm{teachSlot(X)}. \\ \\ \forall\ \mathrm{X} :: \mathrm{teachSlot(X)} \rightarrow (\mathrm{offTeachSlot(X)} \vee \mathrm{sessTeachSlot(X)}). \\ \\ \forall\ \mathrm{X} :: \neg(\mathrm{offTeachSlot(X)} \wedge \mathrm{sessTeachSlot(X)}). \end{array}\right\}$$

Fig. 7. The ID-Logic theory: the "is a" relationship, completeness and disjointness for the partition *scope* of TEACHER SLOT

Finally, an object in ERŒS can only belong directly to a single most-specialized class. In other words, the populations of each couple of concrete classes is disjoint. Due to the limitations on the form of the class hierarchy, this is equivalent to stating that each couple of most general classes is disjoint, as done in Fig. 8.

$$\left\{\begin{array}{l} \forall\ \mathrm{X} :: \neg(\mathrm{course(X)} \wedge \mathrm{exerOffer(X)}). \\ \forall\ \mathrm{X} :: \neg(\mathrm{course(X)} \wedge \mathrm{session(X)}). \\ \ldots \end{array}\right\}$$

Fig. 8. The ID-Logic theory: each couple of most general classes is disjoint

Strengthening. The notion of specialization incorporates the very powerful possibility of strengthening characteristics of a generalization class at the level of the specialization class. This is done in the example for the refinement of classes with a relation. In the previous paragraph, functions and formulae were generated for all refined classes, independent of generalization/specialization and strengthening. The functions generated for the strengthened definition of the relation in the specialization class, must be related with those for the relation in the generalization class. Since objects of the specialization class must be interpretable as objects of the generalization class(es). This relation is defined in Fig. 9. The first formula, states that the participating *teaching ability* for a *session teacher slot* through the function at the specialized level must be the same as through the function at the generalized level. This is an unstrengthened participation, but the formula is needed. The second formula essentially does the same for the other participant. However, the range of the function introduced at the specialized level is restricted to a specialization of TEACHER SLOT (cfr. Fig. 5).

$$\left\{\begin{array}{l}
\forall\ \mathrm{X1,X2} :: \\
\qquad \mathrm{sessTeachAsgmt\_teachAbil(X1)} = \mathrm{X2} \\
\qquad\qquad \rightarrow \mathrm{teachAsgmt\_teachAbil(X1)} = \mathrm{X2}. \\
\forall\ \mathrm{X1,X2} :: \\
\qquad \mathrm{sessTeachAsgmt\_sessTeachSlot(X1)} = \mathrm{X2} \\
\qquad\qquad \rightarrow \mathrm{teachAsgmt\_teachSlot(X1)} = \mathrm{X2}. \\
\forall\ \mathrm{X1,X2,X3} :: \\
\qquad \mathrm{sessTeachSlot\_session(X1)} = \mathrm{X2} \wedge \\
\qquad \mathrm{session\_exerOffer(X2)} = \mathrm{X3} \\
\qquad\qquad \rightarrow \mathrm{teachSlot\_exerOffer(X1)} = \mathrm{X3}. \\
\ldots
\end{array}\right\}$$

Fig. 9. The ID-Logic theory: strengthening of relations

The third formula is more complicated. Here an additional link is inserted in the chain of existential dependence. The formula states that the function composition at the specialized level is equal to the function at the generalized level. Notice that in these formulae, no explicit domain is set for the variables of the universal quantification. The equations will be false for those X's out of range, thus not intervening with the truth of the formula.

4.4 Transforming Explicit Constraints

Since the formalism of ERŒS is based on set theory and first order logic, we expect this transformation to pose no conceptual problems. It is however not yet worked out and implemented.

A possible formulation of a formula for the constraint *appropriate ability* on TEACHER ASSIGNMENT is straight-forward from its specification in ERŒS and is given in Fig. 10.

$$\left\{\begin{array}{l}
\forall\ \mathrm{X} :: \mathrm{teachAsgmt(X)} \rightarrow \\
\qquad \mathrm{exerOffer\_course(techSlot\_exerOffer(teachAsgmt\_teachSlot(X)))} = \\
\qquad \mathrm{teachAbil\_course(teachAsgmt\_teachAbil(X))}.
\end{array}\right\}$$

Fig. 10. The ID-Logic theory: The constraint *appropriate ability* on TEACHER ASSIGNMENT

4.5 Transforming ERŒS Queries

The definition of a deterministic query in an ERŒS model introduces information that can be derived from the current population by applying the query (with arguments) to an object. ERŒS queries do not have preconditions and are thus always defined. In the ID-Logic theory, a deterministic, single-valued EROOS query can be represented by means of a function, with as domain the Cartesian product of the type introduced for the class of objects the query can be applied to and the types for the arguments, and with as range the type of the result. A query returning a set gives rise to a predicate. Figure 11 outlines the predicate generated for the query *assigned teachers* on SESSION. As for constraints, queries

$$\left\{\begin{array}{l}\text{predicate assigned\_teachers(session,exerTeacher).}\\ \hline \text{defined assigned\_teachers(\_,\_).}\\ \hline \text{assigned\_teachers(Session,ExerTeacher)} \leftarrow \ldots.\end{array}\right\}$$

Fig. 11. The ID-Logic theory: The query *assigned teachers* on SESSION

are not yet dealt with automatically.

A conceptual model with recursive queries can introduce problems, since the semantic equivalence between the EROOS query and a definition in the theory is still to be investigated in that context. Non-deterministic queries also need to be further investigated.

For the full specification of the queries in the example, we refer to a forthcoming technical report [5].

5 Solving Tasks

The above transformation is the first step towards a prototyping system. This section shows the use of the $\mathcal{A}$system to compute solutions for a given task. We present it by means of two important classes of tasks: namely database querying and scheduling. Traditionally, solutions for these applications use very different techniques.

5.1 General Notes on Tasks

Before presenting our task representations, we have to refine the view depicted above on the input of the $\mathcal{A}$system. In OO modeling the distinction between an actual instance of the conceptual model (the population) and the conceptual model itself is commonly made. In a logical context this distinction is not always made. However in the context of deductive databases, it corresponds to the distinction between the extentional database (EDB) and the intentional database (IDB). In ID-Logic this EDB has to be provided for each task. In what follows, we represent it by $\mathcal{P}$ as it corresponds to the population of the conceptual model.

The description of a task consists on one hand of the theory ($\mathcal{D}$,$\mathcal{F}$), the population $\mathcal{P}$ and the set $\mathcal{O}$ containing the open predicates. On the other hand it consists of the query Q that triggers the computation. We define the triple

($\mathcal{D}\cup\mathcal{P}$,$\mathcal{F}$,$\mathcal{O}$) as the representation for the theory which is given to the $\mathcal{A}$system to resolve the task.

To illustrate the reasoning process, we use the following population $\mathcal{P}$. The unmentioned classes are assumed to be empty. Also we have ignored the partitioning in the conceptual model to simplify the example population.

$$\left\{\begin{array}{ll}
\text{course(ai).} & \text{course(oo).} \\
\text{exerOffer(aioffer).} & \text{exerOffer(oooffer).} \\
\text{exerOffer\_course(aioffer)=ai.} & \text{exerOffer\_course(oooffer)=oo.} \\
\text{teachSlot(slot1).} & \text{teachSlot(slot2).} \\
\text{teachSlot\_exerOffer(slot1)=aioffer.} & \text{teachSlot\_exerOffer(slot2)=oooffer.} \\
\text{exerTeacher(bert).} & \text{exerTeacher(pieter).} \\
\text{teachAbil(a1).} & \text{teachAbil(a2).} \\
\text{teachAbil\_course(a1)=ai.} & \text{teachAbil\_course(a2)=oo.} \\
\text{teachAbil\_exerTeacher(a1)=bert.} & \text{teachAbil\_exerTeacher(a2)=pieter.} \\
\text{teachAssign(ass1).} & \\
\text{teachAssign\_teachSlot(ass1)=slot1.} & \\
\text{teachAssign\_teachAbil(ass1)=a1.} &
\end{array}\right\}$$

Furthermore when referring to $\mathcal{D}$ and $\mathcal{F}$ from now on, the result from the above transformation applied on the example conceptual model is meant.

5.2 Database Querying

A good characterization of this class of tasks is that of posing queries over a fully known database. In our formulation the population $\mathcal{P}$ is the database.

E.g. for the query:*who teaches the exercises of course ai* in our example, the answer is given by resolving the query `assigned_teachers(ai,Teacher)` in the context of the theory ($\mathcal{D}\cup\mathcal{P}$,$\mathcal{F}$,$\emptyset$). Here, the $\mathcal{A}$system will return the answer substitution σ={Teacher/bert} and an empty Δ as answer.

Note that no open predicates exists (the set $\mathcal{O}$ is empty) because the database is complete and no hypotheses should be made about any of the predicates. This reduces the inference process of the $\mathcal{A}$system to deductive reasoning.

5.3 Scheduling

This class differs from database querying in different points. Mainly because the expected answer is not a simple substitution for the query, but a set of objects and their dependences.

Let us explain this by example: suppose that we are looking for an assignment of teachers to exercises in the context of the population $\mathcal{P}$. $\mathcal{P}$ contains already a partial assigment: namely `bert` for `slot1`, however `slot2` is still vacant. Then an answer to the scheduling task is an extension of the set of teacherAssignments (and its dependences) that will fill up this gap.

In order to allow the system to generate this set, the expected answer predicates are declared open. For our example, the set $\mathcal{O}$ contains {teachAsgmt(_), teachAsgmt_teachAbil(_,_), teachAsgmt_teachSlot(_,_)}[7].

[7] The partial information about these predicates in the population is now part of $\mathcal{F}$.

Secondly for this task, we have to add the requirement that each slot must have a teacher assigned. This constraint is part of the scheduling event and does not appear in the conceptual model (and consequently in the derived theory), since it is not to be fulfilled during the entire system life-cycle. But we have to take it into account, because it enforces the search for a schedule. Formally the requirement is $\forall\ X :: teachSlot(X) \rightarrow\ \exists Y : teachAsgmt\_teachSlot(X) = Y$.

Bringing everything together we obtain the input theory $(\mathcal{D} \cup \mathcal{P}, \mathcal{F} \cup \{\forall\ X :: teachSlot(X) \rightarrow\ \exists Y : teachAsgmt\_teachSlot(X) = Y\}, \mathcal{O})$. The query for a scheduling task is typically *true* because the interest is in the set teacherAssignments.

Note that solving this problem cannot be done by deductive reasoning, but needs abductive reasoning. The computed answer is here an empty answer substitution and the set Δ ={teachAsgmt(ID1), teachAsgmt_teachSlot(ID1)=slot2, teachAsgmt_teachAbil(ID1)=a2} in which ID1 is a new object identifier belonging to the set teacherAssignments.

5.4 Some Remarks

The presented approach contains all the ingredients needed to build a prototype system out of it. It is a very flexible approach to describe a task and is capable to deal in a simple, uniform way with different kinds of problems (namely by declaring open and defined predicates). Hence making it well suited to be the kernel of a prototype system.

As far as efficiency is concerned, the current prototype executes the whole transformed program, a rather time consuming task. Obviously, when knowing which predicates are defined and open and the population, many constraints could be simplified. We expect this to be possible with off the shelve logic program specializers such as ECCE [15] but that has not yet been explored.

6 Conclusion and Future Work

In this paper we have presented a transformation from object-oriented conceptual models in ERŒS to logical theories in ID-Logic. This is a first step towards a method that provides a modeling tool for logical theories. At the same time the transformation is a first step towards a complete formal semantics for ERŒS conceptual models in a well established formalism. By following a transformative approach, we try to build a method that supports the evolutive aspects of the software development process. Moreover this approach makes it possible to build a prototype of the system, and even to perform computational tasks in the context of the generated theory. This has been worked out for two important classes of tasks: database querying and scheduling.

The work presented here is only a beginning. First of all, the transformation is only partial. Certain concepts of the ERŒS method, e.g. the archive and object life-cycles, are left out. Explicit constraints and queries are not yet translated automatically and events are not yet taken into account at all. Non-deterministic constructs in ERŒS allow complex planning functionalities to be specified in an elegant way [3]. Important work is to be done to integrate the ERŒS approach to modeling behavior using queries and events with the (Abductive) Logic Programming approach using tasks. We expect that this will result in a deeper understanding of both paradigms.

Using theorem provers, we can verify if the given conceptual model entails a certain property, starting from the generated logical theory. An interesting application of this approach consists in proving for each class, that a non-empty population can exist. If not possible, this may for instance indicate malformed (multiple) generalization/specialization hierarchies.

The transformation generates a theory that is entirely tailored to the conceptual model at hand. An alternative approach would be to see a conceptual model as an instance of a meta-model. If for this meta-model a corresponding meta-theory is made in logic, it is possible to transform a conceptual model to an instantiation of this meta-theory. All semantics of the ERŒS constructs are then expressed once at the level of this meta-theory. Moreover in logical knowledge representation, meta-theories are built for specific types of knowledge and reasoning patterns. Bringing this back to OO conceptual modeling can introduce interesting modeling constructs and patterns.

References

[1] The $\mathcal{A}$system. Obtainable via `http://www.cs.kuleuven.ac.be/~dtai/kt/systems-E.shtml`. 156

[2] Pieter Bekaert, Geert Delanote, Frank Devos, and Eric Steegmans. Specialization/Generalization in Object-Oriented Analysis: Strengthening and Multiple Partitioning. In *Workshop MASPEGHI: MAnaging SPEcialization/Generalization HIerarchies, OOIS 2002*, September 2002. 155

[3] Pieter Bekaert and Eric Steegmans. Non-Determinism in Conceptual Models. In *Proceedings of the Tenth OOPSLA Workshop on Behavioral Semantics, OOPSLA'01*, October 2001. 155, 165

[4] Pieter Bekaert, Bert Van Nuffelen, Maurice Bruynooghe, David Gilis, and Marc Denecker. Specifying An Educational Resource Management System: A Transformative Approach. Report, Department of Computer Science, K.U.Leuven, Leuven, Belgium, in preparation, draft at `http://www.cs.kuleuven.ac.be/~dtai/kt/btw/`. 153, 159

[5] Pieter Bekaert, Bert Van Nuffelen, Maurice Bruynooghe, David Gilis, and Marc Denecker. Transforming EROOS Conceptual Models to ID-Logic Theories: A Case Study. Report, Department of Computer Science, K.U.Leuven, Leuven, Belgium, in preparation, draft at `http://www.cs.kuleuven.ac.be/~dtai/kt/btw/`. 158, 162

[6] Jean-Michel Bruel and Robert B. France. Transforming UML models to formal specifications. In Pierre-Alain Muller and Jean Bézivin, editors, *Proc. International Conference on the Unified Modeling Language (UML): Beyond the Notation.* Springer-Verlag, 1998. 152

[7] Marc Denecker. Extending classical logic with inductive definitions. In John Lloyd, Veronica Dahl, Ulrich Furbach, Manfred Kerber, Kung-Kiu Lau, Catuscia Palamidessi, Luis Moniz Pereira, Yehoshua Sagiv, and Peter J. Stuckey, editors, *Computational Logic - CL 2000, First International Conference, London, UK, July 2000, Proceedings*, volume 1861 of *Lecture Notes in Artificial Intelligence*, pages 703–717. Springer, 2000. 155

[8] Marc Denecker, Maurice Bruynooghe, and Victor Marek. Logic programming revisited: logic programs as inductive definitions. *ACM Transactions on Computational Logic*, 2(4):623–654, October 2001. 155

[9] Marc Denecker and Danny De Schreye. SLDNFA: an abductive procedure for normal abductive programs. *Journal of Logic Programming*, 34(2):111–167, February 1998. 156

[10] Marc Denecker and A.C. Kakas, editors. *Special Issue : Abductive Logic Programming*. Elsevier, North-Holland, 2000. Special issue Journal of Logic Programming, Vol. 44(1-3), July/August 2000. 156

[11] N. E. Fuchs. Specifications are (preferably) executable. *IEE/BCS Software Engineering Journal*, 7(5):323–334, 1992. 152

[12] Ivar Jacobson, Grady Booch, and James Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, 1999. 153

[13] A.C. Kakas, Bert Van Nuffelen, and Marc Denecker. A-system : Problem solving through abduction. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 591–596. International Joint Conferences on Artificial Intelligence, Inc. and American Association for Artificial Intelligence, 2001. 156, 157

[14] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design*. Prentice Hall, 1997. 153

[15] Michael Leushel. The ECCE partial deduction system and the dppd library of benchmarks. Obtainable via `http://www.ecs.soton.ac.uk/~mal`. 164

[16] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1998. 152, 153

[17] Eric Steegmans, Pieter Bekaert, Frank Devos, Jan Dockx, Bart Swennen, and Stefan Van Baelen. Object-Oriented Analysis. Course Notes, Department of Computer Science, K.U.Leuven, 2001. 153

[18] Eric Steegmans, Johan Lewi, M. D'Haese, Jan Dockx, D. Jehoul, Bart Swennen, Stefan Van Baelen, and P. Van Hirtum. EROOS Reference Manual. Version 1.0. Report CW 208, Department of Computer Science, K.U.Leuven, Leuven, Belgium, January 1995. 153

[19] Stefan Van Baelen, Johan Lewi, Eric Steegmans, and Bart Swennen. Constraints in object-oriented analysis. In S. Nishio and A. Yonezawa, editors, *Object Technologies for Advanced Software*, Lecture Notes in Computer Science, Vol. 742, pages 393–407. Springer-Verlag, Berlin, 1993. ISBN 3-540-57342-9. 154

[20] Stefan Van Baelen, Johan Lewi, Eric Steegmans, and H. Van Riel. EROOS : An Entity-Relationship based Object-Oriented Specification Method. In B. Magnusson G. Heeg and B. Meyer, editors, *Technology of Object-Oriented Languages and Systems TOOLS 7*, pages 103–117. Prentice-Hall, Hertsfordshire, UK, 1992. 153

[21] A. Van Gelder, K.A. Ross, and J.S. Schlipf. The Well-Founded Semantics for General Logic Programs. *Journal of the ACM*, 38(3):620–650, 1991. 156

Reasoning with Goal Models[⋆]

Paolo Giorgini[1], John Mylopoulos[2], Eleonora Nicchiarelli[1], and Roberto Sebastiani[1]

[1] Department of Information and Communication Technology
University of Trento - Italy
{pgiorgini,eleonora,rseba}@science.unitn.it
[2] Computer Science Department - University of Toronto - Canada
jm@cs.toronto.edu

Abstract. Over the past decade, goal models have been used in Computer Science in order to represent software requirements, business objectives and design qualities. Such models extend traditional AI planning techniques for representing goals by allowing for partially defined and possibly inconsistent goals. This paper presents a formal framework for reasoning with such goal models. In particular, the paper proposes a qualitative and a numerical axiomatization for goal modeling primitives and introduces label propagation algorithms that are shown to be sound and complete with respect to their respective axiomatizations. In addition, the paper reports on preliminary experimental results on the propagation algorithms applied to a goal model for a US car manufacturer.

1 Introduction

The concept of goal has been used in many areas of Computer Science for quite some time. In AI, goals have been used in planning to describe desirable states of the world since the 60s [5]. More recently, goals have been used in Software Engineering to model early requirements [2] and non-functional requirements [4] for a software system. For instance, an early requirement for a library information system might be "*Every book request will eventually be fulfilled*", while "*The new system will be highly reliable*" is an example of a non-functional requirement. Goals are also useful for knowledge management systems which focus on strategic knowledge, e.g., "*Increase profits*", or "*Become the largest auto manufacturer in North America*" [3].

Traditional goal analysis consists of decomposing goals into subgoals through an AND- or OR-decomposition. If goal G is AND-decomposed (respectively, OR-decomposed) into subgoals $G_1, G_2, \ldots, G_n$, then if all (at least one) of the subgoals are satisfied so is goal G. Given a goal model consisting of goals and AND/OR relationships among them, and a set of initial labels for some nodes

[⋆] We would like to thank Greg McArthur for sharing with us a version of his goal model for car manufacturing. We also thank the anonymous reviewers and Greg McArthur for helpful feedback on earlier drafts of this paper.

of the graph (S for Satisfied, D for Denied) there is a simple label propagation algorithm which can generate labels for all other nodes of the graph [6].

Unfortunately, this simple framework for modeling and analyzing goals won't work for many domains where goals are not formalizable and the relationships among them can't be captured by semantically well-defined relations such as AND/OR ones. For example, goals such as "*Highly reliable system*" have no formally defined predicate which prescribes their meaning, though one may want to define necessary conditions for such a goal to be satisfied. Moreover, such a goal may be related to other goals, such as "*Thoroughly debugged system*", "*Thoroughly tested system*" in the sense that the latter obviously contribute to the satisfaction of the former, but this contribution is partial and qualitative. In other words, if the latter goals are satisfied, they certainly contribute to the satisfaction of the former goal, but don't guarantee its satisfaction. The framework will also not work in situations where there are contradictory contributions to a goal. For instance, we may want to allow for multiple decompositions of a goal G into sets of subgoals, where some decompositions suggest satisfaction of G while others suggest denial.

We are interested in a modeling framework for goals which allows for other, more qualitative goal relationships and can accommodate contradictory situations. We accomplish this by introducing goal relationships labelled "+" and "–" which model respectively a situation where a goal contributes positively or negatively towards the satisfaction of another goal.

A major problem that arises from this extension is giving a precise semantics to the new goal relationships. This is accomplished in two different ways in this paper. In section 3 we offer a qualitative formalization and a label propagation algorithm which is shown to be sound and complete with respect to the formalization. In section 4, we offer a quantitative semantics for the new relationships which is based on a probabilistic model, and a label propagation algorithm that is also shown to be sound and complete. Section 5 presents preliminary experimental results on our label propagation algorithms, while section 6 summarizes the contributions of the paper and sketches directions for further research.

2 An Example

Suppose we are modeling the strategic objectives of a US car manufacturer, such as Ford or GM. Examples of such objectives are increase return on investment or increase customer loyalty. Objectives can be represented as goals, and can be analyzed using goal relationships such as AND, OR, "+" and "−". In addition, we will use "++" (respectively "−−") a binary goal relationship such that if ++(G,G') (−−(G,G')) then satisfaction of G implies satisfaction (denial) of G'.

For instance, increase return on investment may be AND-decomposed into increase sales volume and increase profit per vehicle. Likewise, increase sales volume might be OR-decomposed into increase consumer appeal and expand markets. This decomposition and refinement of goals can continue until we have goals that are tangible (i.e., someone can satisfy them through an appropriate course

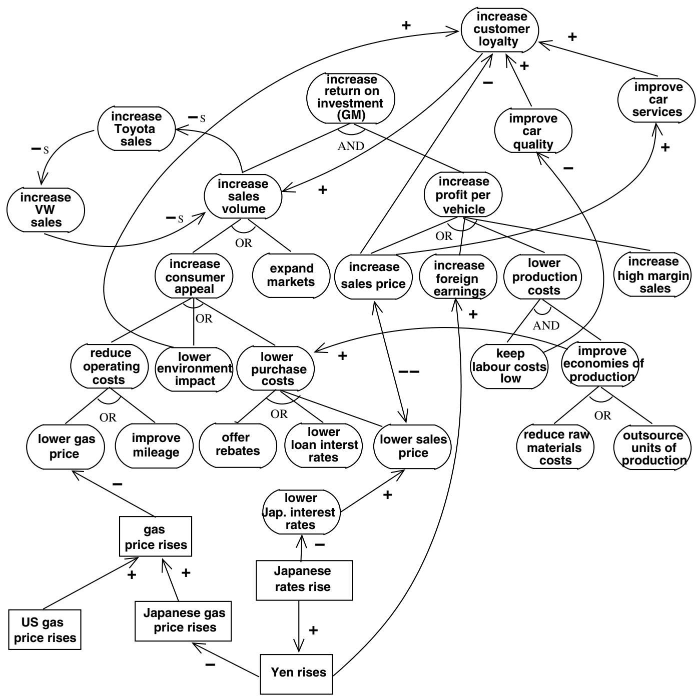


Fig. 1. A partial goal model for GM.

of action) and/or observable (i.e., they can be confirmed satisfied/denied by simply observing the application domain).

For vaguely stated goals, such as increase customer loyalty we may want to simply model other relevant goals, such as improve car quality, improve car services and relate them through "+" and "–" relationships, as shown in Figure 1. These goals may influence positively or negatively some of the goals that have already been introduced during the analysis of the goal increase return on investment. Such lateral goal relationships may introduce cycles in our goal models.

Examples of observable goals are Yen rises, gas prices rise etc. When such a goal is satisfied, we will call it an *event* (the kind of event you may read about in a news story) and represent it in our graphical notation as a rectangle (see lower portion of Figure 1).

Figure 1 shows a partial and fictitious goal model for GM focusing on the goal increase return of investment. In order to increase return of investment, GM has to satisfy both goals increase sales and increase profit per vehicle. In turn, increase sales volume is OR-decomposed into increase consumer appeal and expand markets, while the goal increase profit per vehicle is OR-decomposed into increase sales price, lower production costs, increase foreign earnings, and increase high margin sales. Additional decompositions are shown in the figure. For instance, the goal increase consumer appeal can be satisfied by satisfying lower environment impact, trying to lower purchase costs, or reducing the vehicle operating costs (reduce operating costs).

The graph shows also lateral relationships among goals. For example, the goal increase customer loyalty has positive (+) contributions from goals lower environment impact, improve car quality and improve car services, while it has a negative (−) contribution from increase sales price. The root goal increase return on investment (GM) is also related with goals concerning others auto manufacturer, such as Toyota and VW. In particular, if GM increases sales, then Toyota loses a share of the North American market; if Toyota increases sales increase Toyota sales), it does so at the expense of VW; finally, if VW increases sales (increase VW sales), it does so at the expense of GM.

So far, we have assumed that every goal relationship treats S and D in a dual fashion. For instance, if we have +(G,G'), then if G is satisfied, G' is partially satisfied, and (dually) if G is denied G' is partially denied. Note however, that sometimes a goal relationship only applies for S (or D). In particular, the – contribution from increase sales (GM) to increase sales (Toyota) only applies when increase sales (GM) is satisfied (if GM hasn't increased sales, this doesn't mean that Toyota has.) To capture this kind of relationship, we introduce $-_S$, $-_D$, $+_S$, $+_D$ (see also Figure 1). Details about the semantics of these are given in the next section.

3 Qualitative Reasoning with Goal Models

Formally, a *goal graph* is a pair $\langle \mathcal{G}, \mathcal{R} \rangle$ where $\mathcal{G}$ is a set of goals and $\mathcal{R}$ is a set of goal relations over $\mathcal{G}$. If $(G_1, ..., G_n) \overset{r}{\longmapsto} G$ is a goal relation in $\mathcal{R}$, we call $G_1...G_n$ *source goals* and G the *target goal* of r. To simplify the discussion, we consider only binary OR and AND goal relations. This is not restrictive, as all the operators we consider in this section and in Section 4 —i.e., $\wedge$, $\vee$, min, max, $\otimes$, $\oplus$— are associative and can be thus trivially used as n-ary operators.

3.1 Axiomatization of goal relationships

Let $G_1, G_2, ...$ denote goal labels. We introduce four distinct predicates over goals, $FS(G)$, $FD(G)$ and $PS(G)$, $PD(G)$, meaning respectively that there is (at least) *full* evidence that goal G is satisfied and that G is denied, and that there is at least *partial* evidence that G is satisfied and that G is denied. We also use the proposition $\top$ to represent the (trivially true) statement that there is at least null evidence that the goal G is satisfied (or denied). Notice that the predicates state that there is *at least* a given level of evidence, because in a goal graph there may be multiple sources of evidence for the satisfaction/denial of a goal. We introduce a total order $FS(G) \geq PS(G) \geq \top$ and $FD(G) \geq PD(G) \geq \top$, with the intended meaning that $x \geq y$ iff $x \rightarrow y$.

We want to allow the deduction of *positive* ground assertions of type $FS(G)$, $FD(G)$, $PS(G)$ and $PD(G)$ over the goal constants of a goal graph. We refer to externally provided assertions as *initial conditions*. To formalize the propagation of satisfiability and deniability evidence through a goal graph $\langle \mathcal{G}, \mathcal{R} \rangle$, we introduce the axioms described in Figure 2. (By "dual" we mean that we invert satisfiability with deniability.)

Goal	**Invariant Axioms**	
$G:$	$FS(G) \rightarrow PS(G)$	(1)
	$FD(G) \rightarrow PD(G)$	(2)
Goal relation	**Relation Axioms**	
$(G_2, G_3) \xmapsto{and} G_1:$	$(FS(G_2) \wedge FS(G_3)) \rightarrow FS(G_1)$	(3)
	$(PS(G_2) \wedge PS(G_3)) \rightarrow PS(G_1)$	(4)
	$FD(G_2) \rightarrow FD(G_1), \quad FD(G_3) \rightarrow FD(G_1)$	(5)
	$PD(G_2) \rightarrow PD(G_1), \quad PD(G_3) \rightarrow PD(G_1)$	(6)
$G_2 \xmapsto{+_S} G_1:$	$PS(G_2) \rightarrow PS(G_1)$	(7)
$G_2 \xmapsto{-_S} G_1:$	$PS(G_2) \rightarrow PD(G_1)$	(8)
$G_2 \xmapsto{++_S} G_1:$	$FS(G_2) \rightarrow FS(G_1),$	(9)
	$PS(G_2) \rightarrow PS(G_1)$	(10)
$G_2 \xmapsto{--_S} G_1:$	$FS(G_2) \rightarrow FD(G_1),$	(11)
	$PS(G_2) \rightarrow PD(G_1)$	(12)

Fig. 2. Ground axioms for the invariants and the propagation rules in the qualitative reasoning framework. The (or), $(+_D)$, $(-_D)$, $(++_D)$, $(--_D)$ cases are dual w.r.t. (and), $(+_S)$, $(-_S)$, $(++_S)$, $(--_S)$ respectively.

As indicated in Section 2, the propagation of goal satisfaction through a $++$, $--$, $+$, $-$ may or may not be symmetric w.r.t. that of denial. Thus, for every relation type $r \in \{++, --, +, -\}$, it makes sense to have three possible labels: "r_S", "r_S", and "r", meaning respectively that satisfaction is propagated, that denial is propagated, and that both satisfaction and denial are propagated. (We call the first two cases *asymmetric*, the latter *symmetric*.) For example, $G_2 \xmapsto{-_S} G_1$ means that if G_2 is satisfied, then there is some evidence that G_1 is denied, but if G_2 is denied, then nothing is said about the satisfaction of G_1; $G_2 \xmapsto{-_D} G_1$ means that if G_2 is denied, then there is some evidence that G_1 is satisfied, but if G_2 is satisfied, then nothing is said about the denial of G_1; $G_2 \xmapsto{-} G_1$ means that, if G_2 is satisfied [denied], then there is some evidence that G_1 is denied [satisfied]. In other words, a symmetric relation $G_2 \xmapsto{r} G_1$ is a shorthand for the combination of the two corresponding asymmetric relationships $G_2 \xmapsto{r_S} G_1$ and $G_2 \xmapsto{r_D} G_1$.

(1) and (2) state that full satisfiability and deniability imply partial satisfiability and deniability respectively. For an AND relation, (3) and (4) show that the full and partial satisfiability of the target node require respectively the full and partial satisfiability of all the source nodes; for a "$+_S$" relation, (7) show that only the partial satisfiability (but not the full satisfiability) propagates through a "$+_S$" relation. Combining (1) with (3), and (1) with (7), we have, respectively,

$$(G_2, G_3) \xmapsto{and} G_1 : \ (FS(G_2) \wedge PS(G_3)) \rightarrow PS(G_1) \tag{13}$$

$$G_2 \xmapsto{+_S} G_1 : \ FS(G_2) \rightarrow PS(G_1). \tag{14}$$

Thus, an AND relation propagates the minimum satisfiability value (and the maximum deniability one), while a "$+_S$" relation propagates at most a partial satisfiability value.

	$(G_2, G_3) \xrightarrow{and} G_1$	$G_2 \xrightarrow{+_S} G_1$	$G_2 \xrightarrow{-_S} G_1$	$G_2 \xrightarrow{++_S} G_1$	$G_2 \xrightarrow{--_S} G_1$
$Sat(G_1)$	$min \left\{ \begin{array}{l} Sat(G_2), \\ Sat(G_3) \end{array} \right\}$	$min \left\{ \begin{array}{l} Sat(G_2), \\ P \end{array} \right\}$	N	$Sat(G_2)$	N
$Den(G_1)$	$max \left\{ \begin{array}{l} Den(G_2), \\ Den(G_3) \end{array} \right\}$	N	$min \left\{ \begin{array}{l} Sat(G_2), \\ P \end{array} \right\}$	N	$Sat(G_2)$

Table 1. Propagation rules in the qualitative framework. The (or), $(+_D)$, $(-_D)$, $(++_D)$, $(--_D)$ cases are dual w.r.t. (and), $(+_S)$, $(-_S)$, $(++_S)$, $(--_S)$ respectively.

From now on, we implicitly assume that axioms (1) and (2) are always applied whenever possible. Thus, we say that $PS(G_1)$ is deduced from $FS(G_2)$ and $FS(G_3)$ by applying (3) —meaning "applying (3) and then (1)"— or that $PS(G_1)$ is deduced from $FS(G_2)$ and $PS(G_3)$ by applying (4) —meaning "applying (1) and then (4)".

We say that an atomic proposition of the form $FS(G)$, $FD(G)$, $PS(G)$ and $PD(G)$ *holds* if either it is an initial condition or it can be deduced via modus ponens from the initial conditions and the ground axioms of Figure 2. We assume conventionally that $\top$ always holds. Notice that all the formulas in our framework are propositional Horn clauses, so that deciding if a ground assertion holds not only is decidable, but also it can be decided in polynomial time.

We say that there is a *weak conflict* if either $PS(G)$ and $PD(G)$, $FS(G)$ and $PD(G)$, $PS(G)$ and $FD(G)$ hold for some goal G. We say that there is a *strong conflict* if $FS(G)$ and $FD(G)$ hold for some G.

3.2 The label propagation algorithm

Based on the logic framework of Section 3.1, we have developed an algorithm for propagating through a goal graph $\langle \mathcal{G}, \mathcal{R} \rangle$ labels representing evidence for the satisfiability and deniability of goals. To each node $G \in \mathcal{G}$ we associate two variables $Sat(G), Den(G)$ ranging in $\{F, P, N\}$ (full, partial, none) such that $F > P > N$, representing the current evidence of satisfiability and deniability of goal G. For example, $Sat(G_i) \geq P$ states that there is at least partial evidence that G_i is satisfiable. Starting from assigning an initial set of input values for $Sat(G_i)$, $Den(G_i)$ to (a subset of) the goals in $\mathcal{G}$, we propagate the values through the goal relations in $\mathcal{R}$ according to the propagation rules of Table 1.

The schema of the algorithm is described in Figure 3. *Initial*, *Current* and *Old* are arrays of $|\mathcal{G}|$ pairs $\langle Sat(G_i), Den(G_i) \rangle$, one for each $G_i \in \mathcal{G}$, representing respectively the initial, current and previous labeling states of the graph. We call the pair $\langle Sat(G_i), Den(G_i) \rangle$ a *label* for G_i. Notationally, if W is an array of labels $\langle Sat(G_i), Den(G_i) \rangle$, by $W[i].sat$ and $W[i].den$ we denote the first and second field of the ith label of W.

The array *Current* is first initialized to the initial values *Initial* given as input by the user. At each step, for every goal G_i, $\langle Sat(G_i), Den(G_i) \rangle$ is updated by propagating the values of the previous step. This is done until a fixpoint is reached, in the sense that no further updating is possible ($Current == Old$).

The updating of $\langle Sat(G_i), Den(G_i) \rangle$ works as follows. For each relation R_j incoming in G_i, the satisfiability and deniability values sat_{ij} and den_{ij} derived

```
1   $label\_array\ Label\_Graph(graph\ \langle \mathcal{G}, \mathcal{R} \rangle, label\_array\ Initial)$
2       $Current{=}Initial;$
3       do
4           $Old{=}Current;$
5           for each $G_i \in \mathcal{G}$ do
6               $Current[i] = Update\_label(i, \langle \mathcal{G}, \mathcal{R} \rangle, Old);$
7       until not $(Current{==}Old);$
8       return $Current;$
9
10  $label\ Update\_label(int\ i,\ graph\ \langle \mathcal{G}, \mathcal{R} \rangle,\ label\_array\ Old)$
11      for each $R_j \in \mathcal{R}$ s.t. $target(R_j) == G_i$ do
12          $sat_{ij} = Apply\_Rules\_Sat(G_i, R_j, Old);$
13          $den_{ij} = Apply\_Rules\_Den(G_i, R_j, Old);$
14      return $\langle\ max(max_j(sat_{ij}), Old[i].sat), max(max_j(den_{ij}), Old[i].den)\ \rangle$
```

Fig. 3. Schema of the label propagation algorithm.

from the old values of the source goals are computed by applying the rules of Table 1. The result is compared with the old value, and the maximum is returned as new value for G_i.

3.3 Termination and complexity

Theorem 1. *Label_Graph($\langle \mathcal{G}, \mathcal{R} \rangle$, Initial) terminates after at most* $6|\mathcal{G}|+1$ *loops.*

Proof. First, from lines 6 and 14 in Figure 3 we have that, for every goal G_i,

$$Current[i].sat = max(..., Old[i].sat),$$
$$Current[i].den = max(..., Old[i].den)$$

so that their values are monotonically non-decreasing. In order not to terminate, at least one value per step should monotonically increase. Each of the $2|\mathcal{G}|$ variables $Sat(G_i)$ and $Den(G_i)$ admits 3 possible values and at each non-final loop at least one value increases. Thus the procedure must terminate after at most $6|\mathcal{G}| + 1$ loops. □

Notice that the upper bound is very pessimistic, as many value updates are done in parallel. In Section 5, we report on experiments we have conducted which suggest that the algorithm generally converges after a few loops.

3.4 Soundness and completeness

We call a *value statement* an expression of the form $(v \geq c)$, $v \in \{Sat(G_i), Den(G_i)\}$ for some goal G_i and $c \in \{F, P, N\}$, with the intuitive meaning "there is at least evidence c for v". Thus from now on we rewrite the assertion $FS(G)$, $PS(G)$, $\top$ as $(Sat(G) \geq F)$, $(Sat(G) \geq P)$, $(Sat(G) \geq N)$ and $FD(G)$, $PD(G)$, $\top$ as $(Den(G) \geq F)$, $(Den(G) \geq P)$, $(Sat(G) \geq N)$ respectively.

We say that $(v_1 \geq c_1)$ *is deduced* from $(v_2 \geq c_2)$ [and $(v_3 \geq c_3)$] by a relation axiom meaning that the corresponding assertions are deduced. For instance, $(Sat(G_1) \geq P)$ is deduced from $(Sat(G_2) \geq F)$ by axiom (7) as $PS(G_1)$ is deduced from $FS(G_2)$ by axiom (7).

We say that $(v_1 \geq c_1)$ *derives* from $(v_2 \geq c_2)$ [and $(v_3 \geq c_3)$] by means of one propagation rule $x = f(y, z)$ [$x = f(y)$] of Table 1 if $c_1 = f(c_2, c_3)$ [$c_1 = f(c_2)$]. For instance, $(Sat(G_1) \geq P)$ derives from $(Sat(G_2) \geq P)$ and $(Sat(G_3) \geq F)$ by means of the first propagation rule. Notice that this is possible because the operators *min* and *max* are *monotonic*, that is, e.g., $max(v_1, v_2) \geq max(v_1', v_2')$ iff $v_1 \geq v_1'$ and $v_2 \geq v_2'$.

To this extent, the rules in Table 1 are a straightforward translation of the axioms (1)-(12), as stated by the following result.

Lemma 1. *$(v_1 \geq c_1)$ derives from $(v_2 \geq c_2)$ [and $(v_3 \geq c_3)$] by means of the propagation rules of Table 1 if and only if $(v_1 \geq c_1)$ is deduced from $(v_2 \geq c_2)$ [and $(v_3 \geq c_3)$] with the application of one of the relation axioms (3)-(12).*

Proof. For short, we consider only the AND and $+_S$ cases for $Sat(G_1)$, as the other cases are either analogous or trivial and can be verified by the reader.

$(G_2, G_3) \xmapsto{and} G_1$**:** If either $(Sat(G_2) \geq N)$ or $(Sat(G_3) \geq N)$, then $(Sat(G_1) \geq N)$ is derived. This matches one-to-one the fact that from $\top$ nothing else is deduced;
otherwise, if either $(Sat(G_2) \geq P)$ or $(Sat(G_3) \geq P)$, then $(Sat(G_1) \geq P)$ is derived. This matches one-to-one the fact that from $(Sat(G_2) \geq P)$ and $(Sat(G_3) \geq P)$ axiom (4) is applied, so that $(Sat(G_1) \geq P)$ is deduced;
finally, if both $(Sat(G_2) \geq F)$ and $(Sat(G_3) \geq F)$, then $(Sat(G_1) \geq F)$. This matches one-to-one the fact that from $(Sat(G_2) \geq F)$ and $(Sat(G_3) \geq F)$, axiom (3) is applied, so that $(Sat(G_1) \geq F)$ is deduced.

$G_2 \xmapsto{+_S} G_1$**:** If $(Sat(G_2) \geq N)$, then $(Sat(G_1) \geq N)$. Again, this matches one-to-one the fact that from $\top$ nothing else is deduced;
otherwise, if either $(Sat(G_2) \geq P)$ or $(Sat(G_2) \geq F)$, then we have that $Sat(G_1) \geq min(Sat(G_2), P) \geq P$. This matches one-to-one the fact that from either $(Sat(G_2) \geq P)$ or $(Sat(G_2) \geq F)$ axiom (7) is applied and $(Sat(G_1) \geq P)$ is deduced. □

Given an array of labels W, we say that $(Sat(G_i) \geq c)$ [resp. $(Den(G_i) \geq c)$] *is true in* W if and only if $W[i].sat \geq c$ [resp. $W[i].den \geq c$]. This allows us to state the correctness and completeness theorem for *Label_Graph()*.

Theorem 2. *Let $Final$ be the array returned by Label_Graph($\langle \mathcal{G}, \mathcal{R} \rangle$, $Initial$). $(v \geq c)$ is true in $Final$ if and only if $(v \geq c)$ can be deduced from $Initial$ by applying the relation axioms (3)-(12).*

Proof. First we define inductively the notion of "deduced from $Initial$ in k steps": (i) an assertion in $Initial$ can be deduced from $Initial$ in 0 steps; (ii) an assertion can be deduced from $Initial$ in up to $k + 1$ steps if either it can be deduced from *Initial* in up to k steps or it can be deduced by applying a relation axiom to some assertions, all of which can be deduced from *Initial* in up to k steps.

Let $Current_k$ be the value of *Current* after k loops. We show that $(v \geq c)$ is true in $Current_k$ if and only if $(v \geq c)$ can be deduced from *Initial* in up to k steps. The thesis follows from the fact that $Final = Current_k$ for some k.

We reason by induction on k. The base case $k = 0$ is obvious as $Current_0 = Initial$. By inductive hypothesis, we assume the thesis for k and we prove it for $k + 1$.

If. Consider $(v \geq c)$ such that $(v \geq c)$ is deduced from *Initial* in up to $k+1$ steps. Thus, $(v \geq c)$ is obtained by applying some relation axiom AX to some assertion(s) $(v_1 \geq c_1)$ [and $(v_2 \geq c_2)$], which can be both deduced from *Initial* in up to k steps. Thus, by inductive hypothesis, $(v_1 \geq c_1)$ [and $(v_2 \geq c_2)$] occur in $Current_k$. Then, by Lemma 1, $(v \geq c)$ derives from $(v_1 \geq c_1)$ [and $(v_2 \geq c_2)$] by means of the propagation rules of Table 1. Thus, if v is $Sat(G_i)$ [resp $Den(G_i)$], then c is one of the values sat_{ij} [resp. den_{ij}] of lines 14, 15 in Figure 3, so that $Current_{k+1}[i].sat \geq c$ [$Current_{k+1}[i].den \geq c$]. Thus $(v \geq c)$ is true in $Current_{k+1}$.

Only if. Consider a statement $(v \geq c)$ true in $Current_{k+1}$. If $(v \geq c)$ is true also in $Current_k$, then by inductive hypothesis $(v \geq c)$ can be deduced from *Initial* in up to k steps, and hence in $k+1$ steps. Otherwise, let R_j be the rule and let $(v_2 \geq c_2)$ [and $(v_3 \geq c_3)$] the statements(s) true in $Current_k$ from which v has been derived. By inductive hypothesis $(v_2 \geq c_2)$ [and $(v_3 \geq c_3)$] can be deduced from *Initial* in up to k steps. By Lemma 1 $(v \geq c)$ can be deduced from $(v_2 \geq c_2)$ [and $(v_3 \geq c_3)$] by the application of one relation axiom, and thus can be deduced from *Initial* in up to $k + 1$ steps. □

Thus, from Theorem 2, the values returned by *Label_Graph*($\langle \mathcal{G}, \mathcal{R} \rangle$, *Initial*) are the maximum evidence values which can be deduced from *Initial*.

4 Quantitative Reasoning with Goal Models

The qualitative approach of Section 3 allows for setting and propagating partial evidence about the satisfiability and deniability of goals and the discovery of conflicts.

We may want to provide a more fine-grained evaluation of such partial evidence. For instance, when we have $G_2 \xrightarrow{+S} G_1$, from $PS(G_2)$ we can deduce $PS(G_1)$, whilst one may argue that the satisfiability of G_1 is in some way less evident than that of G_2. For example, in the goal graph of Figure 1, the satisfaction of the goal Lower environment impact may not necessarily imply satisfaction of Increase customer loyalty, so it may be reasonable to assume "less evidence" for the satisfaction of the latter compared to the former. Moreover, the different relations which mean partial support – i.e., $+_S$, $-_S$, $+_D$, $-_D$ – may have different strengths. For instance, in the goal graph of Figure 1, `US Gas price rises`

	$(G_2, G_3) \xmapsto{and} G_1$	$G_2 \xmapsto{w+_S} G_1$	$G_2 \xmapsto{w-_S} G_1$	$G_2 \xmapsto{++_S} G_1$	$G_2 \xmapsto{--_S} G_1$
$Sat(G_1)$	$Sat(G_2) \otimes Sat(G_3)$	$Sat(G_2) \otimes w$		$Sat(G_2)$	
$Den(G_1)$	$Den(G_2) \oplus Den(G_3)$		$Sat(G_2) \otimes w$		$Sat(G_2)$

Table 2. Propagation rules in the quantitative framework. The (or), $(+_D)$, $(-_D)$, $(++_D)$, $(--_D)$ cases are dual w.r.t. (and), $(+_S)$, $(-_S)$, $(++_S)$, $(--_S)$ respectively.

may have a bigger impact on `Gas price rises` than Japanese gas price rises if the manufacturer's market is mainly in the US.

To cope with these facts, we need a way for representing different *numerical* values of partial evidence for satisfiability/deniability and for attributing different weights to the $+_S$, $-_S$, $+_D$, $-_D$ relations. And, of course, we also need a formal framework to reason with such quantitative information.

4.1 An extended framework

In our second attempt, we introduce a *quantitative* framework, inspired by [1]. We introduce two real constants inf and sup such that $0 \leq inf < sup$. For each node $G \in \mathcal{G}$ we introduce two *real* variables $Sat(G), Den(G)$ ranging in the interval $\mathcal{D} =_{def} [inf, sup]$, representing the current evidence of satisfiability and deniability of the goal G. The intended meaning is that inf represents no evidence, sup represents full evidence, and different values in $]inf, sup[$ represent different levels of partial evidence.

To handle the goal relations we introduce two operators $\otimes, \oplus : \mathcal{D} \times \mathcal{D} \longmapsto \mathcal{D}$ representing respectively the evidence of satisfiability of the conjunction and that of the disjunction [deniability of the disjunction and that of the conjunction] of two goals. $\otimes$ and $\oplus$ are associative, commutative and monotonic, and such that $x \otimes y \leq x, y \leq x \oplus y$; there is also an implicit unary operator $inv()$, representing negation, such that $inf = inv(sup)$, $sup = inv(inf)$, $inv(x \oplus y) = inv(x) \otimes inv(y)$ and $inv(x \otimes y) = inv(x) \oplus inv(y)$.

We also attribute to each goal relation $+_S$, $-_S$, $+_D$, $-_D$ a weight $w \in]inf, sup[$ stating the strength by which the satisfiability/deniability of the source goal influences the satisfiability/deniability of the target goal. The propagation rules are described in Table 2. As in the qualitative approach, a symmetric relation —such as, $G_2 \xmapsto{w+} G_1$— is a shorthand for the combination of the two corresponding asymmetric relationships sharing the same weight w —e.g., $G_2 \xmapsto{w+_S} G_1$ and $G_2 \xmapsto{w+_D} G_1$.

There are a few possible models following the schema described above. In particular, here we adopt a *probabilistic* model, where the evidence of satisfiability $Sat(G)$ [resp. deniability $Den(G)$] of G is represented as the probability that G is satisfied (respectively denied). As usual, we adopt the simplifying hypothesis that the different sources of evidence are independent. Thus, we fix $inf = 0$, $sup = 1$, and we define $\otimes, \oplus, inv()$ as:

$$p_1 \otimes p_2 =_{def} p_1 \cdot p_2, \quad p_1 \oplus p_2 =_{def} p_1 + p_2 - p_1 \cdot p_2, \quad inv(p_1) = 1 - p_1$$

Goal relation	Axioms	
$(G_2, G_3) \xmapsto{and} G_1 :$	$(Sat(G_2) \geq x \wedge Sat(G_3) \geq y) \rightarrow Sat(G_1) \geq (x \otimes y)$	(15)
	$(Den(G_2) \geq x \wedge Den(G_3) \geq y) \rightarrow Den(G_1) \geq (x \oplus y)$	(16)
$G_2 \xmapsto{w+_S} G_1 :$	$Sat(G_2) \geq x \rightarrow Sat(G_1) \geq (x \otimes w)$	(17)
$G_2 \xmapsto{w-_S} G_1 :$	$Sat(G_2) \geq x \rightarrow Den(G_1) \geq (x \otimes w)$	(18)
$G_2 \xmapsto{++_S} G_1 :$	$Sat(G_2) \geq x \rightarrow Sat(G_1) \geq x$	(19)
$G_2 \xmapsto{--_S} G_1 :$	$Sat(G_2) \geq x \rightarrow Den(G_1) \geq x$	(20)

Fig. 4. Axioms for the propagation rules in the quantitative reasoning framework. The x, y variables are implicitly quantified universally. The (or), $(+_D)$, $(-_D)$, $(++_D)$, $(--_D)$ cases are dual w.r.t. (and), $(+_S)$, $(-_S)$, $(++_S)$, $(--_S)$ respectively.

that is, respectively the probability of the conjunction and disjunction of two independent events of probability p_1 and p_2, and that of the negation of the first event. To this extent, the propagation rules in Table 2 are those of a Bayesian network, where, e.g., in $G_2 \xmapsto{w+_S} G_1$ w has to be interpreted as the conditional probability $P[G_1\ is\ satisfied \mid G_2\ is\ satisfied]$. Notice that the qualitative framework of Section 3 can be seen as another such model with $\mathcal{D} = \{F, P, N\}$, $\otimes = min()$ and $\oplus = max()$. [3]

4.2 Axiomatization

As with the qualitative case, we call a *value statement* an expression of the form $(v \geq c)$, $v \in \{Sat(G_i), Den(G_i)\}$ for some G_i and $c \in [0, 1]$, with the intuitive meaning "there is at least evidence c of v". We want to allow the user to state and deduce non-negated value statements of the kind $(Sat(G) \geq c)$ and $(Den(G) \geq c)$ over the goal constants of the graph. We call externally provided assertions about the satisfaction/denial of goals *initial conditions.*

To formalize the propagation of satisfiability and deniability evidence values through a goal graph, for every goal and goal relation in $\langle \mathcal{G}, \mathcal{R} \rangle$, we introduce the axioms (15)-(20) in Figure 4. Unlike those of Figure 2, the relation axioms in Figure 4 are not ground Horn clauses —thus, propositional—but rather first-order closed Horn formulas, so that they require a first-order deduction engine.

We say that a statement $(v \geq c)$ holds if either it is an initial condition or it can be deduced from the initial conditions and the axioms of Figure 4. We implicitly assume that $(Sat(G_i) \geq 0)$ and $(Den(G_i) \geq 0)$ hold for every G_i, and that the deduction engine —either human or machine— can semantically evaluate $\otimes$ and $\oplus$ and perform deductions deriving from the values of the evaluated terms and the semantics of $\geq$. For instance, we assume that, if $(G_2, G_3) \xmapsto{and} G_1$ is in $\mathcal{R}$, then $(Sat(G_1) \geq 0.1)$ can be deduced from $(Sat(G_2) \geq 0.5)$, $(Sat(G_3) \geq 0.4)$, as from (15) it is deduced $(Sat(G_1) \geq 0.5 \otimes 0.4)$, which is evaluated into $(Sat(G_1) \geq 0.2)$, from which it can be deduced $(Sat(G_1) \geq 0.1)$.

[3] Another model of interest is that based on a serial/parallel resistance model, in which $inf = 0$, $sup = +\infty$, $x \oplus y = x + y$, $x \otimes y = \frac{x \cdot y}{x+y}$ and $inv(x) = \frac{1}{x}$ [1].

We say that there is a *weak conflict* if both $(Sat(G) \geq c_1)$ and $(Den(G) \geq c_2)$ hold for some goal G and constants $c_1, c_2 > 0$. We say that there is a *strong conflict* if there is a weak conflict s.t. $c_1 = c_2 = 1$.

4.3 The label propagation algorithm

Starting from the new numeric framework, we have adapted the label propagation algorithm of Figure 3 to work with numeric values. The new version of the algorithm differs from that of Section 3 in that: the elements in $Initial$, $Current$ and Old range in $[0, 1]$; the input graph contain also weights to the $+_S$, $-_S$, $+_D$, $-_D$ goal relations; and, the propagation rules applied are those of Table 2.

4.4 Soundness and completeness

We say that $(v_1 \geq c_1)$ *derives* from $(v_2 \geq c_2)$ [and $(v_3 \geq c_3)$] by means of one propagation rule $x = f(y, z)$ [$x = f(y)$] of Table 2 if $c_1 = f(c_2, c_3)$ [$c_1 = f(c_2)$]. For instance, $(Sat(G_1) \geq 0.4)$ derives from $(Sat(G_2) \geq 0.8)$ and $(Sat(G_3) \geq 0.5)$ by means of the first and propagation rule. Again, this is possible because the the operators $\otimes$ and $\oplus$ are monotonic.

Lemma 2. *$(v_1 \geq c_1)$ derives from $(v_2 \geq c_2)$ [and $(v_3 \geq c_3)$] by means of the propagation rules of Table 1 if and only if $(v_1 \geq c_1)$ is deduced from $(v_2 \geq c_2)$ [and $(v_3 \geq c_3)$] with the application of one of the relation axioms (15)-(20).*

Proof. Trivial observation that the axioms (15)-(20) are straightforward translation of the propagation rules in Table 1. □

Given an array of labels W, we say that $(Sat(G_i) \geq c)$ [resp. $(Den(G_i) \geq c)$] *is true in W* if and only if $W[i].sat \geq c$ [resp. $W[i].den \geq c$]. This allows for stating the correctness and completeness theorem for *Label_Graph()*.

Theorem 3. *Let $Final$ be the array returned by Label_Graph($\langle \mathcal{G}, \mathcal{R} \rangle, Initial$). $(v \geq c)$ is true in $Final$ if and only if $(v \geq c)$ can be deduced from $Initial$ by applying the relation axioms (15)-(20).*

Proof. Identical to that of Theorem 2, substituting the axioms (15)-(20) for the axioms (3)-(12), the rules in Table 2 for those in Table 1 and Lemma 2 for Lemma 1. □

Again, from Theorem 3, the values returned by *Label_Graph($\langle \mathcal{G}, \mathcal{R} \rangle, Initial$)* are the maximum evidence values which can be deduced from $Initial$.

4.5 Termination

To guarantee termination, the condition $(Current == Old)$ of line 7 in Figure 3 is implemented as:

$$\begin{array}{l} max_i\,(|Current[i].sat - Old[i].sat|) < \epsilon \textbf{ and} \\ max_i\,(|Current[i].den - Old[i].den|) < \epsilon, \end{array} \tag{21}$$

ϵ being a sufficiently small real constant. (This is a standard practice to avoid numeric errors when doing computations with real numbers.) Thus, the algorithm loops until all satisfiability or deniability value variations have become negligible.

Theorem 4. *Label_Graph($\langle \mathcal{G}, \mathcal{R} \rangle$, Initial) terminates in a finite number of loops.*

Proof. For the same reason as in Theorem 1, the values $Current[i]_k.sat$ and $Current[i]_k.den$ are monotonically non-decreasing with k. As every monotonically non-decreasing upper-bounded sequence is convergent, both $Current[i]_k.sat$ and $Current[i]_k.den$ are convergent. Thus they are also Cauchy-convergent. [4] It follows that condition (21) becomes true after a certain number of loops. □

Remark 1. In the proof of Theorem 3, we have assumed the terminating condition ($Current == Old$), whilst in our implementation we use condition (21). Thus, whilst *Label_Graph()* stops when (21) is verified, the corresponding deductive process might go on, and thus deduce a value slightly bigger than the one returned by the algorithm. Since we can reduce ϵ at will, this "incompleteness" has no practical consequences.

5 Experimental Results

Both qualitative and quantitative algorithms have been implemented in Java and a series of tests were conducted on a Dell Inspiron 8100 laptop with a Pentium III CPU and 64 MB RAM (OS: GNU/Linux, kernel 2.4.7-10). The tests were intended to demonstrate the label propagation algorithms, also to collect some experimental results on how fast the algorithms converge.

The first set of experiments was carried out in order to demonstrate the qualitative label propagation algorithm. For each experiment, we assigned a set of labels to some of the goals and events of the auto manufacturer graph (Figure 1) and see their consequences for other nodes. The label propagation algorithm reached a steady state after at most five iterations.

In a second set of experiments we assigned numerical weights to "+", "−" and "$-_S$" lateral relationships as reported in Table 3. For instance, the goal increase sales volume contributes negatively to the goal increase Toyota sales with a weight 0.6, while the goal increase car quality contributes positively to the goal increase customer loyalty with weight 0.8. Table 4 reports the results of four different experiments. For each goal/event, the table shows the initial (Init) label assignment to the variables S and D, also their final (Fin) value after label propagation. For instance, in the first experiment, the initial assignment for the goal expand markets is S=.3 and D=0, while the final values for increase return on investment (GM) are S=0 and D=.4.

The results of these experiments confirm those obtained with the qualitative algorithm. However, the numeric approach allows us to draw more precise conclusions about the final value of goals. This is particularly helpful for evaluating contradictions. For instance, in the second experiment (Exp 2), even though we

[4] An infinite sequence a_n is Cauchy-convergent if and only if, for every $\epsilon > 0$, there exist an integer N s.t. $|a_{n+1} - a_n| < \epsilon$ for every $n \geq N$. a_n is convergent if and only if it is Cauchy-convergent.

Goal/Event	Relationship	Goal/Event
increase sales volume	$\xmapsto{0.6-S}$	increase Toyota sales
increase Toyota sales	$\xmapsto{0.6-S}$	increase VW sales
increase VW sales	$\xmapsto{0.6-S}$	increase sales volume
increase customer loyalty	$\xmapsto{0.4+}$	increase sales volume
increase sales prices	$\xmapsto{0.5-}$	increase customer loyalty
increase car quality	$\xmapsto{0.8+}$	increase customer loyalty
improve car services	$\xmapsto{0.7+}$	increase customer loyalty
lower environment impact	$\xmapsto{0.4+}$	increase customer loyalty
increase sales prices	$\xmapsto{0.3+}$	improve car services
keep labour costs low	$\xmapsto{0.7-}$	increase car quality
improve economies of production	$\xmapsto{0.8+}$	lower purchase costs
Yen rises	$\xmapsto{0.8+}$	increase foreign earnings
lower Japanese interest rates	$\xmapsto{0.4+}$	lower sales price
Japanese rates rises	$\xmapsto{0.8-}$	lower Japanese interest rates
Japanese rates rises	$\xmapsto{0.6+}$	Yen rises
Yen rises	$\xmapsto{0.4-}$	Japanese gas price rises
Japanese gas price rises	$\xmapsto{0.6+}$	gas price rises
US gas price rises	$\xmapsto{0.6+}$	gas price rises
gas price rises	$\xmapsto{0.8-}$	lower gas price

Table 3. Quantitative relationships for the auto manufacturer example of Figure 2

have a contradiction for the final values of the root goal increase return on investment (GM) (i.e., S=0.8 and D=0.4), we have more evidence for its satisfaction than its denial. With the qualitative approach we had S=P and D=P, and there was nothing else to say about this contradiction. Analogous comments apply for the fourth experiment.

The threshold ϵ (equation 21) used in the experiments has been chosen as the smallest positive value of type float (i.e., `Float.MIN_VALUE`). With this threshold, the algorithm converged in five iterations for all four experiments.

6 Conclusions

We have presented a formal framework for reasoning with goal models. Our goal models use AND/OR goal relationships, but also allow more qualitative relationships, as well as contradictory situations. A precise semantics has been given for all goal relationships which comes in a qualitative and a numerical form. Moreover, we have presented label propagation algorithms for both the qualitative and the numerical case that are shown to be sound and complete with respect to their respective axiomatization. Finally, the paper reports some preliminary experimental results on the label propagation algorithms applied to a goal model for a US car manufacturer.

Future research directions include applying different techniques, such as Dempster Shafer theory [7], to take into consideration the sources of information in the propagation algorithms. This will allow us to consider, for instance, the reliability and/or the competence of a source of evidence . We also propose to apply our framework to more complex real cases to confirm its validity.

Goals/Events	Exp 1				Exp 2				Exp 3				Exp 4			
	Init		Fin		Init		Fin		Init		Fin		Init		Fin	
	S	D	S	D	S	D	S	D	S	D	S	D	S	D	S	D
increase return on investment (GM)	0	0	0	**.4**	0	0	**.8**	**.4**	0	0	**.9**	**.2**	0	0	**.9**	**.6**
increase sales volume	0	0	**1**	**.1**	0	0	**1**	**.1**	0	0	**1**	**.2**	0	0	**1**	**.6**
increase profit per vehicle	0	0	0	**.4**	0	0	**.8**	**.4**	0	0	**.9**	0	0	0	**.9**	0
increase customer appeal	0	0	**1**	0	0	0	**1**	0	0	0	**1**	0	0	0	**1**	0
expand markets	**.3**	0	**.3**	0	**.3**	0	**.3**	0	**.3**	0	**.3**	0	**.3**	0	**.3**	0
increase sales price	0	**.5**	0	**.8**	0	**.5**	0	**.8**	0	**.5**	0	**.8**	0	**.5**	0	**.8**
increase foreign earnings	0	**.9**	0	**.9**	0	**.9**	**.8**	**.9**	0	**.9**	**.8**	**.9**	0	**.9**	**.8**	**.9**
lower production costs	0	0	0	**.9**	0	0	0	**.9**	0	0	**.6**	0	0	0	**.6**	0
increase high margin sales	0	**.6**	0	**.6**	0	**.6**	0	**.6**	0	**.6**	0	**.6**	0	**.6**	0	**.6**
reduce operating costs	0	0	**.8**	0	0	0	**.8**	0	0	0	**.8**	0	0	0	**.8**	0
lower environmental impact	**.9**	0	**.9**	0	**.9**	0	**.9**	0	**.9**	0	**.9**	0	**.9**	0	**.9**	0
lower purchase costs	0	0	**.9**	0	0	0	**.9**	0	0	0	**.9**	0	0	0	**.9**	0
keep labour costs low	0	**.9**	0	**.9**	0	**.9**	0	**.9**	**.9**	0	**.9**	0	**.9**	0	**.9**	0
improve economies of production	0	0	0	0	0	0	0	0	0	0	**.7**	0	0	0	**.7**	0
improve mileage	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
lower gas price	**.8**	0	**.8**	0	**.8**	0	**.8**	0	**.8**	0	**.8**	0	**.8**	0	**.8**	0
offer rebates	**.3**	0	**.3**	0	**.3**	0	**.3**	0	**.3**	0	**.3**	0	**.3**	0	**.3**	0
lower loan interest rates	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
lower sales price	**.8**	0	**.8**	0	**.8**	0	**.8**	0	**.8**	0	**.8**	0	**.8**	0	**.8**	0
reduce raw materials costs	0	0	0	0	0	0	0	0	**.7**	0	**.7**	0	**.7**	0	**.7**	0
outsource units of production	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
gas price rises	0	0	0	0	0	0	0	**.2**	0	0	0	**.2**	0	0	0	**.2**
lower Japanese interest rates	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
US gas price rises	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Japanese gas price rises	0	0	0	0	0	0	0	**.4**	0	0	0	**.4**	0	0	0	**.4**
Yen rises	0	0	0	0	**1**	0	**1**	0	**1**	0	**1**	0	**1**	0	**1**	0
Japanese rates rise	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
improve car quality	0	0	**.6**	0	0	0	**.6**	0	0	0	0	**.6**	0	0	0	**.6**
improve car services	0	0	0	**.2**	0	0	0	**.2**	0	0	0	**.2**	0	0	0	**.2**
improve customer loyalty	0	0	**.5**	**.2**	0	0	**.5**	**.2**	0	0	**.5**	**.2**	0	0	**.4**	**.5**
increase Toyota sales	0	0	0	**.6**	0	0	0	**.6**	0	0	0	**.6**	**1**	0	**1**	**.6**
increase VW sales	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	**1**	**.6**

Table 4. Results with the quantitative approach

References

1. A. Bundy, F. Giunchiglia, R. Sebastiani, and T. Walsh. Calculating Criticalities. *Artificial Intelligence*, 88(1-2), December 1996.
2. A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1–2):3–50, 1993.
3. R. Jarvis, G. McArthur, J. Mylopoulos, P. Rodriguez-Gianolli, and S. Zhou. Semantic Models for Knowledge Management. In *Proc. of the Second International Conference on Web Information Systems Engineering (WISE'01)*, 2001.
4. J. Mylopoulos, L. Chung, and B. Nixon. Representing and Using Non-Functional Requirements: A Process-Oriented Approach. *IEEE Transactions on Software Engineering*, 6(18):483–497, June 1992.
5. A. Newell and H. Simon. GPS: A Program that Simulates Human Thought. In E. Feigenbaum and J. Feldman, editors, *Computers and Thought*. McGraw Hill.
6. N. Nilsson. *Problem Solving Methods in Artificial Intelligence*. McGraw Hill, 1971.
7. G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press., 1976.

Registering Scientific Information Sources for Semantic Mediation*

Amarnath Gupta[1], Bertram Ludäscher[1], and Maryann E. Martone[2]

[1] San Diego Supercomputer Center, University of California
San Diego
{gupta,ludaesch}@sdsc.edu
[2] Department of Neurosciences, University of California
San Diego
mmartone@ucsd.edu

Abstract. In a conventional information mediation scenario it is assumed that all sources, including their schemas, are known before the integrated view is defined. We have found this assumption to be unrealistic for scientific information integration – new relevant sources are discovered quite frequently, and need to be integrated *incrementally* with an existing federation. In this paper, we address the issue of *source registration*, the mechanism by which a new information source "registers" its semantics with the mediator, such that not only new views can be defined with the newly joining source, but existing views can benefit from the source without any redefinition. We approach the problem in the framework of *semantic* (a.k.a. *knowledge-based* or *model-based*) *mediation*, a version of information integration where the sources cannot be integrated solely based on their own logical schema, but need additional domain knowledge at the mediator to "glue" them together. We solve the problem by introducing a process called *contextualization*, whereby a source specifies a set of axioms to express its own conceptual model relative to the mediator's knowledge base. To this end, we present a *context specification language* $\mathcal{CSL}$ that allows the user to specify this mapping, and illustrate how the mediator interprets a $\mathcal{CSL}$ specification to update its knowledge schema and preexisting views. The examples are derived from a real-world scenario involving an ongoing collaboration with several neuroscience groups.

1 Introduction

Information integration refers to the problem of combining multiple information sources such that they appear to a user as a single (virtually) integrated source over a single global schema. A mediator is a data integration software that allows one to define such an integrated schema over schemas of the individual sources. In doing so, it hides from the the various heterogeneities arising from differences in data source types, data models and query capabilities among different

* Work partially supported by NIH BIRN-CC 3 P41 RR08605-08S1, NSF/NPACI Neuroscience Thrust ASC-975249, and DOE SciDAC/SDM DE-FC02-01ER25486.

sources. Given a user query against the global schema, the mediator transparently decomposes it into constituent local subqueries against the appropriate the sources, collects partial query results from the sources, and after due post-processing, reports the combined results to the user. There are two predominant techniques to map the source schemas to the the global schema [FLM98]. In the *global-as-view* (GAV) model, the global schema is defined as a view over local schemas. Hence mediated data objects are "fused" together from parts obtained from data objects in one or more sources. In contrast, in the *local-as-view* (LAV) model, a global schema is defined first by modeling the application domain. Then the source schemas and their data objects are defined as views over the global schema. For query evaluation, the rules have to be "inverted" or "folded" (if possible) [LRO96, Hal01, GM02].

Recently, research has been devoted to the problem of *semantic* (also called knowledge-based or model-based) mediation [GLM00, LGM00, LGM01]. Semantic mediation adds a few additional considerations to the logical information integration problem as described above. In this scenario, sources not only export their logical schema, but also their *conceptual model* to the mediator, thus exposing their concepts, roles, classification hierarchies, and other high-level semantic constructs to the mediator. The mediator, in turn, exposes a conceptual schema to the user, based on the conceptual schemata of individual sources. Semantic mediation allows information sources to export their schema at an appropriate level of abstraction to the mediator. If the individual sources have different *local ontologies* or namespaces, the mediator needs to reconcile their differences, or establish some well-defined relationships among them. To enable this reconciliation, the mediator use a *global ontology* to correlate the terminology from different sources. The mediator's ontology includes general facts and rules that hold in the specific domain of application (or common-sense knowledge as for a dictionary [MWK00]), and some additional rules that explicitly specify the relationships among the conceptual models of the different sources.

In this paper, we continue our prior research [GLM00, LGM00, LGM01] on semantic integration of scientific information, and address the *source registration problem*. In a state-of-the-art information mediation scenario (semantic or otherwise), it is presumed that all sources, including their schemas, are known before the integrated view is defined. In our experience with scientific information sources, this is not a realistic assumption – new relevant sources are discovered quite frequently, and need to be integrated *incrementally* with an existing federation already operational among a number of previously known sources. The registration problem refers to the mechanism by which a new information source "semantically registers" with the mediator, such that not only new views can be defined with the newly joining source, but existing views can benefit from the source without any redefinition. We approach the problem by introducing a process called *contextualization*, whereby a source declares, in addition to its conceptual model, an extra set of axioms to express its own conceptual model in terms of the the mediator's knowledge base. The axioms are expressed in what [RTU01] calls an *interschema language*. The mediator, in turn, processes the contextualization axioms, by compiling them from the interschema language into its own internal formalism, and updating its knowledge base and views.

The organization and main contributions of the paper are as follows. In the remainder of this section, we present related work. In Section 2, we describe how an information source can specify its local semantics without explicitly linking to the mediator's ontology. In Section 3, we present $\mathcal{CSL}$, our *context specification language* that serves as the primary vehicle for source registration. Source registration is accomplished by specifying mappings between the source's ontology and the mediator's ontology using $\mathcal{CSL}$ declarations. This is followed in Section 4 by a brief description of how $\mathcal{CSL}$ statements are processed in the mediator to complete the registration procedure. We conclude in Section 5, with a brief discussion and an outlook on future work.

Related Work

The general problem of defining a global schema over a set of local schema is not a new problem, see, *e.g.*, [SL90, PS98]. In [RR97] a seven-step methodology for schema integration based on semantic integrity constraints is presented; see [Tür99] for a comprehensive treatment of semantic integrity constraints in federated databases. For this paper, we focus on related research in three areas.

Conceptual Schema Integration. Research in schema integration techniques have been undertaken since the mid 80's [SL90]. The primary focus in information integration is in resolving relation and attribute conflicts between to-be-integrated schemas. A fraction of schema integration research has investigated the problem of conceptual schema integration. For example, the use of "integration operators" *copy*, *generalize*, *join*, and *simplify* has been proposed [CE93] to integrate conceptual schemas. In contrast, constraint based approaches have been proposed for operations like *generalization*, *type assignment*, and *exclusion constraints* to correlate two conceptual graphs [EJ95]. Benn et al. [BCG96] first transform the relational schema of each source to an object schema consisting of classes, attributes and semantic constraints. Schema integration is achieved using two groups of first-order rules. With the first group the integrator *identifies subgraph isomorphisms* between the schemas based on their semantic similarity. The second group of rules provide *merger rules*, including rules that state when two schemas, although similar, cannot be merged.

Semantic Mediation. Significant progress has been made in the general area of data mediation in recent years, and several prototype mediator architectures have been designed by projects like TSIMMIS [GMPQ+95], SIMS [KMA+98], Information Manifold [LRO96], Garlic [HKWY97], and MIX [BGL+99]. While these approaches focus mostly on structural and schema aspects, the problem of *semantic mediation* has also been addressed:

In the DIKE system [PTU00], the focus is on automatic extraction of mappings between semantically analogous elements from different schemas. A global schema is defined in terms of a conceptual model (SDR network) where the nodes represent concepts and the (directed) edge labels represent their semantic distances and a score called *semantic relevance* that measures the number of instances of the target node that are also instances of the source node. The correspondence between objects are defined in terms of *synonymies*, *homonymies* and *sub-source similarities*, defined by finding maximal matching between the two graphs.

ODB-Tools [BB01] is a system developed on top of the MOMIS [BCV99] system for modeling and reasoning about the common knowledge between two to-be-integrated schemas. They present the object-oriented language ODL_{I^3} derived from a description logic (OCDL). The language allows a user to create complex objects with finite nesting of values, union and intersection types, integrity constraints and quantified paths. These constructs are used to define a class in one schema as a *generalization*, *aggregation*, or *equivalent* with respect to another; *subsumption* of a class by another can be inferred. An integrated schema is obtained by clustering schema elements that are close to one another in terms of an affinity metric.

Calvanese et al. [CCG+01] perform semantic information integration using an LAV approach by expressing the conceptual schema by a description logic language called $\mathcal{DLR}$, and subsequently defining non-recursive Datalog views to express source data elements in terms of the conceptual model. The language $\mathcal{DLR}$ represents concepts C, relations R, and a set of assertions of the form $C_1 \sqsubset C_2$ or $R_1 \subset R_2$, where R_1, R_2 are $\mathcal{DLR}$ relations with the same arity. Mediation is accomplished by defining "reconciliation correspondences", specifications that a query rewriter uses to match a conceptual level term to data in different sources.

Recently Peim et al. [PFPG02] have proposed to extend the well-known TAMBIS system [GSN+01]. Their approach is similar to ours [GLM00, LGM01] in that a logic-based ontology (in their case the $\mathcal{ALCQI}$ description logic) interfaces with an "object-wrapped" source. Their work focuses on how a query on the ontology is transformed to monoid comprehensions for semantic query optimization. In contrast, this paper addresses the issue of how to *dynamically* register a new object source with pre-existing ontologies at the mediator.

Ontology Merging. The problem of ontology matching and merging stems from AI and KRDB research, and is now revisited by the Semantic Web community. Work on the Cyc Upper Ontology [Hov97], the Ontomorph system [Cha00], the Chimaera system [MFRW00], the PROMPT algorithm [NM00] and the FCA-MERGE algorithm [SM01] are all different techniques to represent and find term-matching relationships so that they can be put into a concept lattice.

The ONION system [MWK00] performs *algebraic composition of ontologies*. They use a special "semantic implication" relation $P \Rightarrow Q$ which relates graph patterns P and Q by making the assertion that the object Q *semantically belongs to* the class P. Bridge rules for semantic implications are typically expressed as simple Horn clauses, and are translated to graph operations. The system also admits functional rules, permitting simple functions to be executed as part of the ontology correlation process.

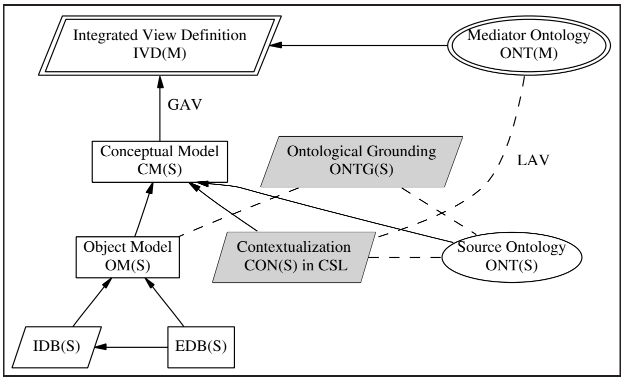


Fig. 1. Semantic Source Registration: logical components and their dependencies

Summary. We note that most of the semantic information integration methods and systems described above essentially correlate schema elements from different sources with some common relationships including class-instance, class-subclass, relation-specialization, part-whole, class equivalence, class-subsumption, and algebraically composable classes. We posit that the source registration problem, introduced above, requires a more general *rule-definable* approach compared to current semantic correlation methods and should make the mappings between models "first-class citizens" [BHP00].

2 Modeling Source Semantics

To enable semantic mediation, an information source needs to be wrapped in such a way that it exports a *conceptual model* CM of the source rather than its logical schema. This requires the CM wrapper to provide the mediator with additional information about its object class structure and constraints that its logical schema may not provide. We call this the "lifting" of the source to a conceptual level. More details about our semantic mediation strategy can be found in [LGM01]. In the sequel, we describe in detail the conceptual model of the source after it has been "lifted" by the CM wrapper. In Section 3 we show how to specify the mapping between this lifted model and the ontology of the mediator.

2.1 Conceptual Model of the Source

The core components of the *conceptual model* $\mathsf{CM}(S)$ of a source S are:

$$\mathsf{CM}(S) = \mathsf{OM}(S) \cup \mathsf{ONT}(S) \cup \mathsf{CON}(S)$$

The logical components an their dependencies are depicted in Fig. 1:

- $\mathsf{OM}(S)$ is the *object model* of the source S and provides signatures for *classes*, *associations* between classes, and *functions*. $\mathsf{OM}(S)$ structures can be defined extensionally by facts (EDB), or intensionally via rules (IDB).
- $\mathsf{ONT}(S)$ is the *local ontology* of the source S, *i.e.*, defines *concepts* and their *relationships* from the source's perspective.
- $\mathsf{ONTG}(S)$ is the *ontological grounding* of $\mathsf{OM}(S)$ in $\mathsf{ONT}(S)$; it links the object model $\mathsf{OM}(S)$ (classes, attributes, associations) to the concepts and relationships of $\mathsf{ONT}(S)$.
- $\mathsf{CON}(S)$ is the *contextualization* of the local source ontology relative to a mediator ontology $\mathsf{ONT}(M)$.
- $\mathsf{IVD}(M)$ is the mediator's *integrated view definition* and comprises logic view definitions in terms of the sources' object models $\mathsf{OM}(S)$ and the mediator's ontology $\mathsf{ONT}(M)$. By posing queries against the mediator's $\mathsf{IVD}(M)$, the user has the illusion to interact with a single, semantically integrated source instead of interacting with independent, unrelated sources.

In the following, we present the local parts of $\mathsf{CM}(S)$, *i.e.*, $\mathsf{OM}(S)$, $\mathsf{ONT}(S)$, and $\mathsf{ONTG}(S)$ through a running example. The contextualization $\mathsf{CON}(S)$ is described in Section 3.

Example 1 (Cell-Centered Database: CCDB). Fig. 2 shows pieces of a simplified version of the conceptual model $\mathsf{CM}(\mathsf{CCDB})$ of a real-world scientific information source called the *Cell-Centered Database*, [MGW$^+$02]. The database consists of a set of EXPERIMENT objects. Each experiment collects a number of cell IMAGES from one or more instruments. For each image, the scientists mark out cellular STRUCTURES in the image and perform measurements on them [MGW$^+$02]. They also identify a second set of regions, called DEPOSITs, in images that show the deposition of molecules of proteins or genetic markers. In general, a region marked as deposit does not necessarily coincide with a region marked as a structure.

Note that $\mathsf{OM}(\mathsf{CCDB})$ in Fig. 2 includes classes that are instantiated with observed data, *i.e.*, the extensional database $\mathsf{EDB}(\mathsf{CCDB})$. In addition to classes, $\mathsf{OM}(\mathsf{CCDB})$ stores *associations*, *i.e.*, n-ary relationships between object classes. The association co_localizes_with specifies which pairs of substances occur together in a specific structure. The object model also contains *functions*, such as the domain specific methods that can be invoked by a user as part of a query. For example, when the mediator or another client calls the function CCDB.deposit_in_structure(), and supplies the *id* of a deposit object, the function returns a set of STRUCTURE objects that spatially overlap with the specified deposit object.

Next, we describe the source's local ontology, $\mathsf{ONT}(\mathsf{CCDB})$. In our world, an ontology $\mathsf{ONT}(S)$ consists of a set of *concepts* and inter-concept *relationships*, possibly augmented with additional inference rules and constraints.[1] The onto-

[1] *E. g.*, ONT4, ONT5 in Fig. 2 define virtual relations such as *transitive closure* over the base relations.

Classes in OM(CCDB)

EXPERIMENT(<u>id</u>:id, date:date, cell_type:string, images:SET(image)).
IMAGE(<u>id</u>:id, instrument:ENUM{c_microscope, e_microscope}, resolution:float, size_x:int, size_y:int,
  depth:int, structures:SET(structure), regions:SET(deposit)).
STRUCTURE(<u>id</u>:id, name:string, length:float, surface_area:float, volume:float, bounding_box:Cube).
DEPOSIT(<u>id</u>:id, substance_name:string, deposit_type:string, relative_intesity:ENUM{dark,normal,bright},
  amount:float, bounding_box:Cube).
. . .

Associations in OM(CCDB)

co_localizes_with(DEPOSIT.substance_name, DEPOSIT.substance_name, STRUCTURE.name).
surrounds(s1:STRUCTURE, s2:STRUCTURE).
. . .

Functions in OM(CCDB)

deposit_in_structure(DEPOSIT.id) $\longrightarrow$ SET(STRUCTURE.name)
. . .

Source Ontology – ONT(CCDB)

brain $\xrightarrow{has(co)}$ cerebellum $\xrightarrow{has(co)}$ cerebellar cortex $\xrightarrow{has(co)}$ vermis (ONT1)
dendrite $\xrightarrow{has(co)}$ spine process $\xrightarrow{has(pm)}$ spine (ONT2)
cell $\xrightarrow{projects\_to}$ brain_region
globus_pallidus$\xrightarrow{isa}$brain_region. . . . denaturation$\xrightarrow{isa}$process. (ONT3)
$tc\_has(co)$:= **transitive_closure**($has(co)$). $tc\_has(pm)$:= **transitive_closure**($has(pm)$). (ONT4)
has_co_pm := **chain**($tc\_has(co), tc\_has(pm)$) (ONT5)
. . .

Ontological Grounding – ONTG(CCDB)

domain(STRUCTURE.volume) **in** [0,300]
domain(STRUCTURE.name) **in** $tc\_has(co)$(cerebellum) (OG1)
domain(EXPERIMENT.cell_type) **in** $tc\_has(co)$(cerebellum) (OG2)
EXPERIMENT.cell_type $\xrightarrow{projects\_to}$ globus_pallidus (OG3)
DENATURED_PROTEIN$\xrightarrow{exhibits}$denaturation. (OG4)
. . .

Fig. 2. Conceptual Model for Registering the Cell-Centered Database [MGW+02]

logical grounding ONTG(S) links the object model OM(S) to the source ontology ONT(S).

The source ontology serves a number of different purposes:

Creating a Terminological Frame of Reference. For defining the terminology of a specific scientific information source, the source declares its own controlled vocabulary through ONT(S). More precisely, ONT(S) comprises the terms (*i.e.*, *concepts*) of this vocabulary and the *relationships* among them. The concepts and relationships are often represented as nodes and edges of a directed graph, respectively. Two examples of interconcept relations are has(co) and has(pm) which are different kinds of part-whole relationships [2]. In Fig. 2, items ONT1 and ONT2 show fragments of such a concept graph. Once a concept graph is

[2] By standards of meronyms, there are different kinds of the has relation: component-object has(co), portion-mass has(pm), member-collection has(mc), stuff-object has(so), place-area has(pa) etc. [AFGP96]

created for a source, one may use it to define additional constraints on object classes and associations.

Semantics of Relationships. The edges in the concept graph of the source ontology represent inter-concept relationships. Often these relationships have their own semantics that have to be specified within ONT(S). Item ONT4 declares two new relationships tc_has(co) and tc_has(pm). After registration, the mediator interprets this declaration and creates the new (possibly materialized) transitive relations on top of the base relations has(co) and has(pm) provided by the source S. Similarly, the item ONT5 is interpreted by the mediator using a higher-order rule for chaining binary relations:

– chain(R1,R2)(X,Y) **IF** R1(X,Z), R2(Z,Y)

With this, ONT5 creates a new relationship has_co_pm(X,Y) provided that there is a Z such that tc_has(co)(X,Z), *and* tc_has(pm)(Z,Y).

Ontological Grounding of OM(S). A local domain constraint specifies additional properties of the given extensional database, and thereby establishes an *ontological grounding* ONTG(S) between the local ontology ONT(S) and the object model OM(S) (Fig. 1). Items (OG1–OG2) in Fig. 2 refines the domains of the attributes EXPERIMENT.cell_type and STRUCTURE.name from the original type declaration (STRING). The refinement constrains them to take values from those nodes of the concept graph that are *descendants* of the concept cerebellum through the has(co) relationship.

This constraint illustrates an important role of the local ontology in a "conceptually lifted" source. By constraining the domain of an attribute to be concept name C, the corresponding object instance o is "semantically about" C. In addition, this also implies that o is about any ancestor concept C' of C where ancestor is defined via has(co) edges only. Similarly, if a specific instance, STRUCTURE.name has the value 'spine process', it is also about 'dendrite' (ONT2 in Fig. 2).

In addition to linking attributes to concept names, a constraint may also involve inter-concept relationships. Let us assume projects_to(cell, brain_region) is a relationship in the source ontology ONT(CCDB). A constraint may assert that for all instances e of class EXPERIMENT, projects_to(e.cell_type, 'globus_pallidus') holds (OG3). The constraint thus *refines* the original relationship projects_to to suit the specific semantics of OM(CCDB). In Section 3, we will use these constraint-defined correspondences between OM(S) and ONT(S) in the contextualization process.

Intensional Definitions. In the CM wrapper of a source S, we can define virtual classes and associations that can be exported to the mediator as first-class, queriable items by means of an intensional database (view definition) IDB(S). For example, we can create a new virtual class called DENATURED_PROTEIN in IDB(CCDB) via the rule:

```
DENATURED_PROTEIN(ProtName) IF
  DEPOSIT(ID, ProtName, protein, dark, _, _), deposit_in_structure(ID) $\neq \emptyset$
```

Thus, an instance of a DENATURED_PROTEIN is created when a “dark” protein deposit is recorded in an instance of DEPOSIT, and there is some structure in which this deposit is found. As a general principle of creating a CM wrapper, such a definition will be supplemented by additional constraints to connect it to the local ontology. For example, assume that ONT(CCDB) already contains a concept called process. Item (ONT3) defines denaturation as a specialization of process. We can now add the constraint (OG4) to complete the semantic specification about the new DENATURED_PROTEIN object.

Contextual References. It is a standard practice for scientific data sources to tag object instances with controlled vocabulary from a public standard. In (CON1) of Fig. 3, the source states the following mapping rule: The domain of the DEPOSIT.id field can be accessed through an internal method get_expasy_protein_id, which, given a protein name, gets its id from the SWISS-PROT database on the web.[3] How the source enforces this integrity constraint is internal to the source and not part of its conceptual export schema.

2.2 Mediator Information from the Source’s Viewpoint

In order to address the source registration issue, we have to specify which components of an existing n-source federation can be “seen”, *i.e.*, accessed by the new, $n+1^{st}$ source. A federation at the mediator consists of:

1. currently *registered conceptual models* CM(S) of each participating source S,
2. one or more *global ontologies* ONT(M) residing at the mediator that have been used in the federation, and
3. *integrated views* IVD(M) defined in a global-as-view (GAV) fashion.

Typical mediator ontologies ONT(M) are *public*, *i.e.*, serve as domain-specific expert knowledge and thus can be used to “glue” conceptual models from multiple sources. Examples of such ontologies are the Unified Medical Language System (UMLS) from the National Library of Medicine[4] and the Biological Process Ontology from the GeneOntology Consortium[5].

In the presence of multiple ontologies, *articulations*, *i.e.*, mappings between different source ontologies [MWK00] can be used to register with the mediator information about inter-source relationships.

Note that a source S usually cannot “see” all of the above components (1–3) when defining its conceptual model: While S sees the mediator’s ontologies ONT(M) and thus can define its own conceptual model CM(M) relative to the

[3] http://www.expasy.ch

[4] http://www.nlm.nih.gov/research/umls/

[5] http://www.geneontology.org/process.ontology

mediator's ontology in a *local-as-view* (LAV) fashion, it cannot (i) directly employ *another* source's conceptual model $\mathsf{CM}(S')$, nor (ii) can it query the mediator's integrated view $\mathsf{IVD}(M)$ which is defined *global-as-view* (GAV) on top of the sources. The former is no restriction, since S' can register $\mathsf{CM}(S')$, in particular $\mathsf{ONT}(S')$ with the mediator, at which point S can indirectly refer to registered concepts of S' via $\mathsf{ONT}(M)$. The latter guarantees that query processing in this setting does not involve "recursion through the web", *i.e.*, between a source S and the mediator M (the dependency graph in Fig. 1 is acyclic).[6]

3 Context Specification Language $\mathcal{CSL}$

A contextualization $\mathsf{CON}(S)$ "situates" a source's conceptual model $\mathsf{CM}(S)$ in the context given by the mediator's ontology $\mathsf{ONT}(M)$. This is accomplished by mappings between the source ontology $\mathsf{ONT}(S)$ and the mediator ontology $\mathsf{ONT}(M)$. In the following, we present the context specification language $\mathcal{CSL}$ that allows us to express such mappings.

First, observe that a source's object model $\mathsf{OM}(S)$ can be described in terms of special "built-in" predicates C:$\mathsf{classes}(S)$ ("C is a class of S"), A:$\mathsf{assocs}(S)$ ("A is an association of S"), A:$\mathsf{attributes}(S, C)$ ("A is an attribute of class C in S"), and O:$\mathsf{instances}(S, C)$ ("O is an object instance of C in S"). Similarly, the local ontology $\mathsf{ONT}(S)$ can be described by $\mathsf{concepts}(S)$, $\mathsf{relationships}(S)$ and $\mathsf{constraints}(S)$, where the latter are first-order constraints over $\mathsf{concepts}(S)$ and $\mathsf{relationships}(S)$. Analogously, $\mathsf{ONT}(M)$ is described via $\mathsf{concepts}(M)$, $\mathsf{relationships}(M)$ and $\mathsf{constraints}(M)$. We call these special predicates the *model elements* of the source and mediator respectively, and use them to specify source-to-mediator mappings.

$\mathcal{CSL}$ allows one to specify element mappings and access mappings. An *element mapping* is a $\mathcal{CSL}$ expression that specifies how an element of the source's conceptual model relates to that of the mediator. An *access mapping* is a $\mathcal{CSL}$ expression that specifies how an element from source's conceptual model can be physically accessed from the mediator.[7] In this paper, we focus on the element mapping part of $\mathcal{CSL}$ and present the language through examples from our CCDB scenario (Fig. 3).

Let us assume the CCDB source intends to inform the mediator that all concepts in ONT(CCDB) are identical to those concepts in the mediator that have the same name. This is expressed in $\mathcal{CSL}$ (CON2 in Fig. 3) as:

```
map (equivalent)(X, Y) IF
        X:concepts(CCDB), Y:concepts(mediator),
        X.name = Y.name
```

The general form of a $\mathcal{CSL}$ statement is

[6] At the cost of loss of efficiency, the restriction "no recursion through the web" could be lifted.

[7] *E. g.*, for an SQL source the access mapping is an SQL query

```
CONTEXTUALIZATION of CCDB – CON(CCDB)
domain(DEPOSIT.id) in get_expasy_protein_id(DEPOSIT.substance_name)
    IF DEPOSIT.deposit_type='protein'                                              (CON1)
map (equivalent)(X,Y)
    IF X:concepts(CCDB), Y:concepts(mediator), X.name = Y.name                     (CON2)
map (subconcept)(brain, organ)
    IF brain:concepts(CCDB), organ:concepts(mediator)                              (CON3)
map (subconcept)(axon, compartment)
    IF axon:concepts(mediator), compartment:concepts(CCDB)                         (CON4)
map (concept_concept)(regulates('nejire','CREB'))
    IF 'nejire':concepts(mediator), 'CREB':concepts(CCDB)                          (CON5a)
map (concept_concept)(exists G, regulates('nejire', G), regulates(G, 'CREB'))
    IF 'nejire':concepts(mediator), 'CREB':concepts(CCDB)                          (CON5b)
map (concept_concept)(tc_regulates('nejire','CREB'))
    IF 'nejire':concepts(mediator), 'CREB':concepts(CCDB)                          (CON5c)
map (subrelation)(has(co), has_part)
    IF has(co):relationships(CCDB), has_part:relationships(mediator)               (CON6)
map (instance_concept)(X,ultrastructure)
    IF X:instances(STRUCTURE, CCDB), ultrastructure:concepts(mediator),
    has_part:relationships(mediator), dendrite:concepts(mediator),
    X.name in transitive_closure(part_of)(dendrite)                                (CON7)
map (assoc_rel)(surrounds(s1, s2), inverse(inside(s3,s4))
    IF surrounds(s1, s2):assoc(CCDB), inside(s3,s4):relationships(mediator),
    not has_part(s1, s2).                                                          (CON8)
map (concept_concept)(new evidence_of(regulates)('cfos','CREB'))
    IF 'cfos':concepts(mediator), 'CREB':concepts(CCDB).                           (CON9a)
map (holds) evidence_of(_)(X,Y)
    IF X:concepts(mediator), Y:concepts(CCDB), Z:concepts(mediator),
    evidence_of(_)(X,Z), not opposes(Z,Y).                                         (CON9b)
...
```

Fig. 3. Context Specification for the Cell-Centered Database [MGW+02]

map $(\textit{correspondence relation})(X_1, \ldots, X_n)$ **IF**
type declarations,
body

where the *correspondence relation* (*e.g.*, equivalent) specifies which kind of mapping is being defined, thereby instructing the mediator how to compile the statement into a logic program during registration. The *type declarations* specify the kind of model element each variable represents (*e.g.*, X above is of type concepts). The type declaration also specifies whether an X_i belongs to the source or to the mediator. The system ensures that the $X_1, \ldots, X_n$ include both source and mediator model elements (since **map** links *between* source and mediator model elements). Furthermore, *correspondence relations* are themselves typed, *e.g.*, equivalent expects its arguments to be either both concepts or both relationships. The *body* of the $\mathcal{CSL}$ statement is like the body of a logic rule and specifies additional conditions that the mapping must satisfy. All variables in the head of the statement are universally quantified, unless otherwise mentioned. In the following paragraphs we present informal examples of different forms of mapping relations that can be described in $\mathcal{CSL}$.

Subconcept Mapping. Consider C_1:concepts(source) and C_2:concepts(mediator). The correspondence relation "subconcept" defines an *isa* relation between them. (CON3) in Fig. 3 states that brain, a concept defined in ONT(CCDB) *isa* organ, defined at the mediator. As discussed in Section 4, after registration, pre-existing integrated views will "see" the CCDB's *isa* relation through the subconcept mapping established via **map**(subconcept) declarations. In this example, the source concept brain *specializes* the mediator concept organ. Similarly, a source can also *generalize* a mediator concept. Assume, *e.g.*, that ONT(CCDB) has a concept called compartment (not shown in Fig. 2), and the mediator has the concept axon. Item (CON4) states that axons *are* compartments. The mediator has translation rules for both uses of the subconcept mapping.

Concept-Concept Mapping. Subconcept mapping is a special case of inter-concept mapping across the source and the mediator. In general, a concept of the source will be related to a concept at the mediator through a user-specified relationship R. For example, assume that ONT(M) contains the information that 'nejire' *isa* gene, and CCDB contains the relation 'CREB' *isa* protein (not shown in Fig. 2). Item (CON5a) shows declaration that states that 'nejire' bears the relationship *regulates* with 'CREB'. Since the relation *regulates* is known to the mediator, it translates the above mapping to enable any integrated view that accesses 'nejire' via the *regulates* relationship, to have access to 'CREB' in CCDB.

The concept-concept mapping allows a number of variations:
Often in the domain of scientific information, direct relationships between two concepts are not known. Assume for simplicity, that 'nejire' regulates exactly one unknown gene G, which in turn regulates 'CREB'. To express this, the $\mathcal{CSL}$ expression in (CON5a) will be modified to (CON5b), with an existential quantifier in the head.

If there were an *unknown number* of intermediate genes in the regulation path between 'nejire' and 'CREB', we would express this fact in $\mathcal{CSL}$ by placing 'CREB' in *tc_regulates* of 'nejire', where *tc_regulates* is the transitive closure relation built on *regulates* as in item (CON5c).

Subrelation Mapping. The "subrelation" mapping declares a relation in ONT(S) to be a special case of a relationship in ONT(M) (or vice versa). Consider that the mediator uses a relationship called *has_part*. CCDB uses more refined relationships $has(co)$ and $has(pm)$. Item (CON6) declares $has(co)$ to be a specialization of the mediator's *has_part* relationship. We omit the arguments of the relationships if the arguments of one relationship corresponds exactly to the positionally identical element of the second. The mediator processes this mapping by declaring $has(co)$ as one possible substitution of *has_part* for the source CCDB.

Concept-Instance Mapping. In the last section we showed how a **domain** declaration is used to connect a concept in the local ontology to instances of a local object class. Our idea there was to make the statement that the qualified instances of the object class were "semantically about" the concept. The concept-instance mapping is a similar idea to connect the instances of a local object class to a concept at the mediator. Let ultrastructure be a concept defined at the mediator. Let us also assume that *has_part* is a relationship defined at the mediator. We use item (CON7) to state that every instance of the class STRUCTURE in CCDB whose name has a value that can be found in the *has_part* tree of the mediator's ontology is "semantically about" the concept ultrastructure. So, if the mediator's ontology has the fragment:

$$\mathsf{dendrite} \overset{has\_part}{\longrightarrow} \mathsf{SER}$$

and CCDB had an object instance STRUCTURE(50, 'SER', 20.2, 45.5, ...), then this instance is "about" an ultrastructure.

Relation-Association Mapping. The "assoc_rel" mapping relates an inter-object association A in $\mathsf{OM}(S)$ to an inter-concept relationship in R in $\mathsf{ONT}(M)$. Let us assume $\mathsf{A}(\mathsf{X}_1, \mathsf{X}_2)$ and $R(Y_1, Y_2)$ are both binary. For the "assoc_rel" mapping to hold, X_1 and X_2 are *implicitly considered to be* "semantically about" Y_1 and Y_2 respectively. If $\mathsf{A}(\mathsf{a}_1, \mathsf{a}_2)$ is an instance of the association in the extension of $\mathsf{OM}(S)$, then one can construct a relation $R(a_1, a_2)$ at the mediator. Assume, *e.g.*, $\mathsf{ONT}(M)$ contains the spatial relationship $inside(s_3, s_4)$ meaning that structure s_3 is physically inside s_4. Now consider the association $\mathsf{surrounds}(s_1 : \text{STRUCTURE}, s_2 : \text{STRUCTURE})$ in OM(CCDB). In (CON8) of Fig. 3, we use the reserved word **inverse** to associate $\mathsf{surrounds}.s_1$ with $inside.s_4$ and $\mathsf{surrounds}.s_2$ with $inside.s_3$. If we find the instance surrounds('caudate_putamen', 'fiber_bundle'), the mediator can create a new relationship *inside*(fiber_bundle, caudate_putamen).

New Relationship Mapping. We repeat the $\mathcal{CSL}$ expression in item (CON5a):

```
map (concept_concept)(regulates('nejire', 'CREB'))
    IF 'nejire':concepts(mediator), 'CREB':concepts(CCDB)
```

where the relationship *regulates* is declared as part of a concept-concept mapping. $\mathcal{CSL}$ assumes that the name of the relationship is known to the mediator, otherwise allows one to declare unknown relationships via the reserved word **new**. For example, consider the statement of item (CON9a):

```
map (concept_concept)(new evidence_of(regulates)('cfos', 'CREB'))
        IF 'cfos':concepts(mediator), 'CREB':concepts(CCDB)
```

where *evidence_of*(*regulates*) is a new relationship. Typically, the declaration of a new relationship will be accompanied by additional constraints that specify its properties.

Mapping Constraints. Constraints are specified in $\mathcal{CSL}$ using the "*holds*" mapping element. Item (CON9b) shows an axiom about the relation *evidence_of*(_) for any parameter. The axiom assumes that the mediator knows the relation $opposes(X, Y)$ (*i.e.*, X contradicts Y), and states that no concept of CCDB can be an evidence of two opposing concepts of the mediator.

4 Registration Process

In the following, we outline how $\mathcal{CSL}$ specifications are handled by the mediator to complete the source registration process. The registration process involves the following steps:

– *Store*: At runtime, the mediator receives $\mathcal{CSL}$ statements sent by the source and stores them in a global registry.
– *Index*: Based on CON(S), the mediator updates ONT(M) to include new local concepts and relationships introduced by ONT(S). Then mediator updates its global concept index to keep track of which concepts have been used and referred to by the registered sources.
– *Assimilate Local Semantics*: The ontological grounding ONTG(S) and local integrity constraints of a source S are translated into an executable specification at the mediator. For example, the following statement from Fig. 2
    domain(STRUCTURE.volume) **in** [0,300]
  is translated into a logic rule encoding an integrity constraint in the form of a denial:
    false :– X:structure[volume→V], $\neg(0 \leq$V$\leq 300)$
  Similarly, the ontological grounding rule (OG1)
    domain(STRUCTURE.name) **in** *tc_has*(*co*)(cerebellum)
  is translated into the logic rule
    false :– X:structure[name→N], ¬tc_has(co)(cerebellum)
– *Assimilate Context*: The contextualization CON(S) is assimilated at the mediator. Consider, for example, item (CON6) which states that the CCDB relation *has*(*co*) is a "subrelation" of the mediator's relation *has_part*:
    map (subrelation)(*has*(*co*), *has_part*) (CON6)
        IF *has*(*co*):relationships(CCDB), *has_part*:relationships(mediator)
  This is translated into the logic rules
    has_part(X,Y) :– CCDB.has(co)(X,Y) (*derive*)
    false :– CCDB.has(co)(X,Y), ¬has_part(X,Y) (*denial*)
  The first rule is used to *populate* and *query* the has_part relation at the mediator, while the second, logically equivalent rule specifies the integrity constraint as a denial and is used for *reasoning* about contextualizations.[8]
– IVD *extension*: The final step is to augment the view definitions IVD(M) to reflect model elements such as equivalent and subconcept. For example, the declaration (CON2) states that source and mediator concepts should be considered equivalent if they are syntactically equal; (CON3) states that what CCDB calls brain is a subconcept of what the mediator calls organ. Logically, this corresponds to extending the concept hierarchy by asserting the equivalence or subconcept relationship between the respective terms. Note that the logic view definitions IVD(M) do not have to be rewritten but can automatically access the newly asserted concepts (equivalent or subconcepts), provided that inheritance rules have been asserted.[9]

[8] This is similar to subsumption testing in description logics; the details of this are beyond the scope of this paper.

[9] See [KLW95] for monotonic inheritance rules in F-Logic: the implementation language of our semantic mediation prototype.

5 Conclusion

In this paper we have investigated the problem of source registration in the context of semantic information mediation. We have shown how a source can export its schema and information semantics to the mediator by specifying its *object model*, *local ontology*, and *ontological grounding* that relates the local ontology with elements of the object model. This explicit modeling of the source's semantics to facilitate mediation is a novel contribution of our work. Further, we have developed $\mathcal{CSL}$ a context specification language by which the source maps its local ontology in the context of the mediator's ontology. The language allows a mediation engineer to perform fine-grained mapping between the modeling constructs of the source and those of the mediator. We are currently in the process of implementing a more complete version of the language.

We have outlined how the mediator can interpret the source's declarations and internalize these definitions to complete the process of registration. However, there are several difficult and unresolved problems in assimilation. For example, how should the mediator deal with contradictions between its own ontological definitions and the source's local ontology? Also, how does the mediator's query engine evaluate the views that have been affected by the newly joining source? We plan to address these issues in the future.

References

[AFGP96] A. Artale, E. Franconi, N. Guarino, and L. Pazzi. Part-whole Relations in Object-Centered Systems: An Overview. *Data & Knowledge Engineering*, 20:347–383, 1996. 188

[BB01] D. Beneventano and S. Bergamaschi. Extensional Knowledge for semantic query optimization in a mediator based system. In *Int. Workshop on Foundations of Models for Info. Integ. (FMII-2001)*, 2001. 185

[BCG96] B. Benn, Y. Chen, and I. Gringer. A rule-based strategy for schema integration in a heterogeneous information environment, 1996. 184

[BCV99] S. Bergamaschi, S. Castano, and M. Vincini. Semantic Integration of Semistructured and Structured Data Sources. *SIGMOD Record*, 28(1):54–59, 1999. 185

[BGL$^+$99] C. Baru, A. Gupta, B. Ludäscher, R. Marciano, Y. Papakonstantinou, P. Velikhov, and V. Chu. XML-Based Information Mediation with MIX. In *Intl. Conf. on Management of Data (SIGMOD)*, pp. 597–599, 1999. 184

[BHP00] P. A. Bernstein, A. Y. Halevy, and R. A. Pottinger. A vision for management of complex models. *SIGMOD Record*, 29(4):55–63, 2000. 186

[CCG$^+$01] D. Calvanese, S. Castano, F. Guerra, D. Lembo, M. Melchiori, G. Terracina, D. Ursino, M. Vincini. Towards a Comprehensive Methodological Framework for Semantic Integration of Heterogeneous Data Sources. *Intl. Workshop on Knowledge Representation meets Databases (KRDB)*, 2001. 185

[CE93] P. N. Creasy and G. Ellis. A Conceptual Graph Approach to Conceptual Schema Integration. In *Conceptual Graphs for Knowledge Representation: ICCS*, pp. 126–141, Quebec, Canada, 1993. 184

[Cha00] H. Chalupsky. OntoMorph: A Translation System for Symbolic Knowledge. In *Principles of Knowledge Representation and Reasoning*, 2000. 185

[EJ95] L. Ekenberg and P. Johannesson. Conflictfreeness as a Basis for Schema Integration. In *Conference on Information Systems and Management of Data (CISMOD)*, pp. 1–13, 1995. 184

[FLM98] D. Florescu, A. Levy, and A. Mendelzon. Database Techniques for the World-Wide Web: A Survey. *SIGMOD Record*, 27(3), September 1998. 183

[GLM00] A. Gupta, B. Ludäscher, and M. E. Martone. Knowledge-Based Integration of Neuroscience Data Sources. In *Intl. Conference on Scientific and Statistical Database Management (SSDBM)*, 2000. 183, 185

[GM02] J. Grant and J. Minker. A Logic-Based Approach to Data Integration. *Theory and Practice of Logic Programming (TPLP)*, 2(3):323–368, 2002. 183

[GMPQ$^+$95] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. The TSIMMIS Approach to Mediation: Data Models and Languages. In *Next Generation Information Technologies and Systems*, 1995. 184

[GSN$^+$01] C. Goble, R. Stevens, G. Ng, S. Bechhofer, N. Paton, P. Baker, M. Peim, and A. Brass. Transparent Access to Multiple Bioinformatics Information Sources. *IBM Systems Journal*, 40(2):534–551, 2001. 185

[Hal01] A. Y. Halevy. Answering Queries Using Views: A Survey. *VLDB Journal*, 10(4):270–294, 2001. 183

[HKWY97] L. M. Haas, D. Kossmann, E. L. Wimmers, and J. Yang. Optimizing Queries Across Diverse Data Sources. In *Intl. Conf. on Very Large Databases (VLDB)*, pp. 276–285, Athens, Greece, 1997. 184

[Hov97] E. Hovy. A Standard for Large Ontologies. In *Workshop on Research & Development Opportunities in Federal Information Services*, 1997. 185

[KLW95] M. Kifer, G. Lausen, and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM*, 42(4):741–843, 1995. 195

[KMA$^+$98] C. A. Knoblock, S. Minton, J. L. Ambite, P. J. M. N. Ashish, I. Muslea, A. G. Philpot, and S. Tejada. Modeling Web Sources for Information Integration. In *15th National Conference on Artificial Intelligence*, 1998. 184

[LGM00] B. Ludäscher, A. Gupta, and M. E. Martone. Model-Based Information Integration in a Neuroscience Mediator System. In *Intl. Conf. on Very Large Data Bases (VLDB)*, pp. 639–642, Cairo, Egypt, 2000. 183

[LGM01] B. Ludäscher, A. Gupta, and M. E. Martone. Model-Based Mediation with Domain Maps. In *17th Intl. Conf. on Data Engineering (ICDE)*, Heidelberg, Germany, 2001. 183, 185, 186

[LRO96] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In *Intl. Conference on Very Large Data Bases (VLDB)*, pp. 251–262, 1996. 183, 184

[MFRW00] D. L. McGuinness, R. Fikes, J. Rice, S. Wilder. The Chimaera Ontology Environment. *17th Natl. Conf. on Artificial Intelligence (AAAI)*, 2000. 185

[MGW$^+$02] M. E. Martone, A. Gupta, M. Wong, X. Qian, G. Sosinsky, S. Lamont, B. Ludäscher, and M. H. Ellisman. A Cell-Centered Database for Electron Tomographic Data. *Journal of Structural Biology*, 2002. to appear; see also http://ncmir.ucsd.edu/CCDB/. 187, 188, 192

[MWK00] P. Mitra, G. Wiederhold, and M. L. Kersten. A Graph-Oriented Model for Articulation of Ontology Interdependencies. In *Extending Database Technology*, pp. 86–100, 2000. 183, 185, 190

[NM00] N. F. Noy and M. A. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *17th National Conference on Artificial Intelligence (AAAI)*, pp. 450–455, 2000. 185

[PFPG02] M. Peim, E. Franconi, N. Paton, and C. Goble. Query Processing with Description Logic Ontologies Over Object-Wrapped Databases. In *Intl. Conf. on Scientific and Statistical Database Management (SSDBM)*, 2002. 185

[PS98] C. Parent and S. Spaccapietra. Issues and Approaches of Database Integration. *Communications of the ACM*, 41(5):166–178, 1998. 184

[PTU00] L. Palopoli, G. Terracina, and D. Ursino. The System DIKE: Towards the Semi-Automatic Synthesis of Cooperative Information Systems and Data Warehouses. In *Proc. ADBIS-DASFAA Symposium*, pp. 108–117, 2000. 184

[RR97] V. Ramesh and S. Ram. Integrity Constraint Integration in Heterogeneous Databases: An Enhanced Methodology for Schema Integration. *Information Systems*, 22(8):423–446, 1997. 184

[RTU01] D. Rosaci, G. Terracina, and D. Ursino. A Semi-automatic Technique for Constructing a Global Representation of Information Sources Having Different Formats and Structure. In *DEXA*, pp. 734–743, 2001. 183

[SL90] A. P. Sheth and J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, 1990. 184

[SM01] G. Stumme and A. Maedche. FCA-MERGE: Bottom-Up Merging of Ontologies. In *IJCAI*, pp. 225–234, 2001. 185

[Tür99] C. Türker. *Semantic Integrity Constraints in Federated Database Schemata*. DISDBIS 63, infix-Verlag, 1999. Ph.D. thesis, Fakultät für Informatik, Universität Magdeburg. 184

Multidimensional Modeling with UML Package Diagrams*

Sergio Luján-Mora[1], Juan Trujillo[1], and Il-Yeol Song[2]

[1] Dept. de Lenguajes y Sistemas Informàticos, Universidad de Alicante (Spain)
{slujan,jtrujillo}@dlsi.ua.es
[2] College of Information Science and Technology, Drexel University (USA)
songiy@drexel.edu

Abstract. The Unified Modeling Language (UML) has become the *de facto* standard for object-oriented analysis and design, providing different diagrams for modeling different aspects of a system. In this paper, we present the development of multidimensional (MD) models for data warehouses (DW) using UML package diagrams. In this way, when modeling complex and large DW systems, we are not restricted to use flat UML class diagrams. We present design guidelines and illustrate them with various examples. We show that the correct use of the package diagrams using our design guidelines will produce a very simple yet powerful design of MD models. Furthermore, we provide a UML extension by means of stereotypes of the particular package items we use. Finally, we show how to use these stereotypes in Rational Rose 2000 for MD modeling.

Keywords: UML, multidimensional modeling, data warehouses, UML extension, UML packages

1 Introduction

Multidimensional (MD) modeling is the foundation of data warehouses (DW), MD databases, and OLAP applications. These systems provide companies with many years of historical information for the decision making process. MD modeling structures information into facts and dimensions. A fact table contains interesting measures of a business process (sales, deliveries, etc.), whereas a dimension table represents the context for analyzing a fact (product, customer, time, etc.). Various approaches for the conceptual design of MD systems have been proposed in the last few years [1][2][3][4] to represent main MD structural and dynamic properties. However, none of them has been accepted as a standard conceptual model for MD modeling. Due to space constraints, we refer the reader to [5] for a detailed comparison and discussion about most of these models.

On the other hand, the Unified Modeling Language (UML) [6] has been widely accepted as the standard object-oriented (OO) modeling language for

* This paper has been partially supported by the Spanish Ministry of Science and Technology, project number TIC2001-3530-C02-02.

describing and designing various aspects of software systems. Therefore, any approach using the UML will minimize the efforts of developers in learning new diagrams or methodologies for every subsystem to be modeled. Following this consideration, we have previously proposed in [7] an OO conceptual MD modeling approach, based on the UML, for a powerful conceptual MD modeling. This proposal considers major relevant MD properties at the conceptual level in an elegant and easy way.

In this paper, we start from our previously presented approach [7] and show how to apply the grouping mechanism called *package* provided by the UML. A package groups classes together into higher level units creating different levels of abstraction, and therefore, simplifying the final model. In this way, when modeling complex and large data warehouse systems, we are not restricted to use flat UML class diagrams.

Furthermore, based on our experience in developing real world cases, we provide design guidelines to properly and easily apply packages to MD modeling. These design guidelines are extremely relevant for two reasons. First, the UML Specification does not formally define how to apply packages, and therefore, different people may use them in different ways. Second, these guidelines are very close to the natural way both designers and analyzers understand and accomplish MD modeling. Our experience indicates that these guidelines produce a very simple yet powerful design of large and complex MD models.

To facilitate the MD modeling using our package diagrams, we provide a UML extension by using stereotypes of the particular package items we define. Our extension uses the Object Constraint Language (OCL) [6] for expressing well-formedness rules of the new defined elements, thereby avoiding an arbitrary use of our extension. Finally, we use these package stereotypes in Rational Rose 2000 for MD modeling to show the applicability of our proposal.

The remainder of the paper is structured as follows: Section 2 briefly presents other works related to grouping mechanisms in conceptual modeling. Section 3 summarizes how we have previously used the UML for proper conceptual MD modeling. Section 4 presents our design guidelines for using packages in MD modeling. Section 5 presents a case study to show how our guidelines are properly applied for MD modeling. Section 6 presents the definition of our UML extension in terms of package stereotypes. Section 7 shows how to apply our package extension in Rational Rose. Finally, Section 8 presents the main conclusions and introduces our immediate future work.

2 Related Work

The benefits of layering modeling diagrams have been widely recognized. Different modeling techniques, such as Data Flow Diagrams, Functional Modeling (IDEF0), Entity Relationship Model (ER) and the UML make use of some kind of layering mechanism.

Focusing on the aim of this paper, i.e. data modeling, several approaches have been proposed to provide grouping mechanisms to the ER to simplify complex diagrams. In [8], the *Clustered Entity Model*, one of the early attempts at layering ER diagrams, is presented. In this approach, an ER diagram at a lower level appears as an entity on the next level. In [9], a model and a technique for clustering entities in an ER diagram is described. This modeling technique refines ER diagrams into higher-level objects that lead to a description of the conceptual database on a single page. The great benefit of this proposal is that it increases the clarity of the database description, and therefore, facilitates a better communication between end-users and the database designer. In [10], the *Leveled Entity Relationship Model* presents another layering formalism for ER diagrams.

Regarding the UML, and as it is stated in [11], "There are different views about how to use UML to develop software systems". The UML is a complex modeling language and it lacks a systematic way of guiding the users through the development of systems. With respect to *packages*, the UML defines them as a mechanism to simplify complex UML diagrams. However, no formal design guidelines regarding how to properly use them are clearly stated. In this context, many text books and authors have provided guidelines to apply packages in general or in a specific domain. For example, Fowler states [12]: "I use the term package diagram for a diagram that shows packages of classes and the dependencies among them". In a specific domain such as web applications, Conallen states [13] that "A package is merely a mechanism to divide the model into more manageable pieces" and "Try to avoid making the package hierarchy match the semantics of the business, and instead use packages as a means of managing the model". The author proposes to make packages "*comprehensible*, *cohesive*, *loosely coupled* and *hierarchically shallow*". To the best of our knowledge, no work has been presented showing the benefits of using UML packages for MD modeling.

On the other hand, there have lately been proposed several approaches to accomplish the conceptual design of data warehouses following the MD paradigm. Due to space constraints, we will only make a brief reference to those works which are close to the research topic of this paper [1][2][3][4]. These MD models are mainly conceived to gain user requirements and provide an easy to be used but yet powerful set of graphical elements to facilitate the task of conceptual modeling as well as the specification of queries. Both [2] and [3] extend the ER model to use it for MD modeling providing specific items such as facts, dimension levels and so on. [1] and [4] propose different graphical notations for data warehouse conceptual design.

Motivation: All the above-commented MD approaches use "flat design" in the sense that all the elements that form a MD model (e.g. facts, dimensions, classification hierarchies and so on) are represented in the same diagram at the same level. Therefore, these approaches are not often suitable for huge and complex MD models in which several facts share many dimensions and their classification hierarchies, thereby leading to cluttered diagrams that are very difficult to read. Based on our experience in designing real-world cases, we argue that in most cases we need an approach that structures the design of complex data warehouses at different levels.

3 Object-Oriented Multidimensional Modeling

In this section, we summarize[1] how an OO MD model can represent main structural aspects of MD modeling. The main features considered are the many-to-many relationships between facts and dimensions, degenerate dimensions, multiple and alternative path classification hierarchies, and non-strict and complete hierarchies. In this approach, the main structural properties of MD models are specified by means of a UML class diagram in which the information is clearly separated into facts and dimensions.

Dimensions and facts are represented by *dimension classes* and *fact classes*, respectively. Then, fact classes are specified as composite classes in shared aggregation relationships of n dimension classes. The flexibility of shared aggregation in the UML allows us to represent *many-to-many* relationships between facts and particular dimensions by indicating the 1..* cardinality on the dimension class role. In our example in Fig. 1 (a), we can see how the fact class Sales has a many-to-one relationship with both dimension classes.

By default, all measures in the fact class are considered additive. For non-additive measures, additive rules are defined as constraints and are included in the fact class. Furthermore, derived measures can also be explicitly considered (indicated by $/$) and their derivation rules are placed between braces near the fact class, as shown in Fig. 1 (a).

This OO approach also allows us to define identifying attributes in the fact class, by placing the constraint $\{OID\}$ next to an attribute name. In this way we can represent *degenerate dimensions* [14][15], thereby representing other fact features in addition to the measures for analysis. For example, we could store the ticket number (ticket_num) and the line number (line_num) as degenerate dimensions, as reflected in Fig. 1 (a).

With respect to dimensions, every *classification hierarchy* level is specified by a class (called a *base class*). An association of classes specifies the relationships between two levels of a classification hierarchy. The only prerequisite is that these classes must define a Directed Acyclic Graph (DAG) rooted in the dimension class (constraint $\{dag\}$ placed next to every dimension class). The DAG structure can represent both alternative path and multiple classification hierarchies. Every classification hierarchy level must have an *identifying* attribute (constraint $\{OID\}$) and a *descriptor* attribute[2] (constraint $\{D\}$). These attributes are necessary for an automatic generation process into commercial OLAP tools, as these tools store this information in their metadata. The multiplicity *1* and *1..** defined in the target associated class role addresses the concepts of *strictness* and *non-strictness*, respectively. Strictness means that an object at a hierarchy's lower level belongs to only one higher-level object (e.g., as one month can be

[1] We refer the reader to [7] for a complete description of our approach.

[2] A descriptor attribute will be used as the default label in the data analysis.

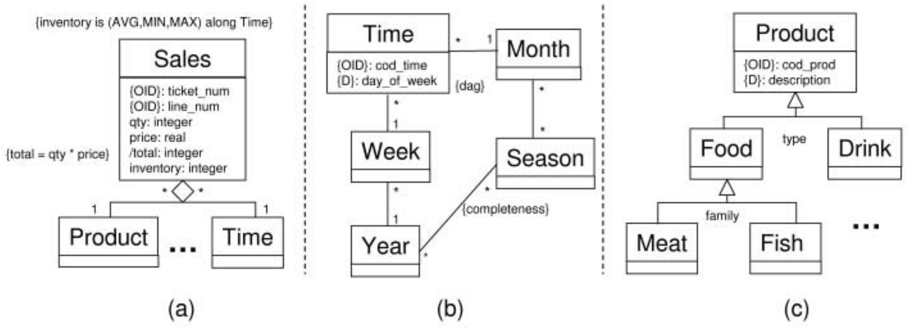


Fig. 1. Multidimensional modeling using UML

related to more than one season, the relationship between them is non-strict). Moreover, defining the {*completeness*} constraint in the target associated class role addresses the completeness of a classification hierarchy (see an example in Fig. 1 (b)). By completeness we mean that all members belong to one higher-class object and that object consists of those members only. For example, all the recorded seasons form a year, and all the seasons that form the year have been recorded. Our approach assumes all classification hierarchies are non-complete by default.

The *categorization of dimensions*, used to model additional features for a class's subtypes, is represented by means of generalization-specialization relationships. However, only the dimension class can belong to both a classification and specialization hierarchy at the same time. An example of categorization for the Product dimension is shown in Fig. 1 (c).

4 Package Design Guidelines for Multidimensional Modeling

In this section, based on our experience in real-world cases, we present our design guidelines for using UML packages in MD modeling following the approach presented in Section 3. We believe these guidelines are very close to the natural way that both designers and analyzers accomplish and understand MD modeling. Our experience indicates that these guidelines produce a very simple yet powerful design of MD models. We summarize all the design guidelines in Table 1.

`Guideline 0` is the foundation of the rest of the guidelines and summarizes our overall approach. This guideline closely resembles how data analyzers understand MD modeling. We have divided the design process into three levels (Fig. 2 shows a summary of our proposal):

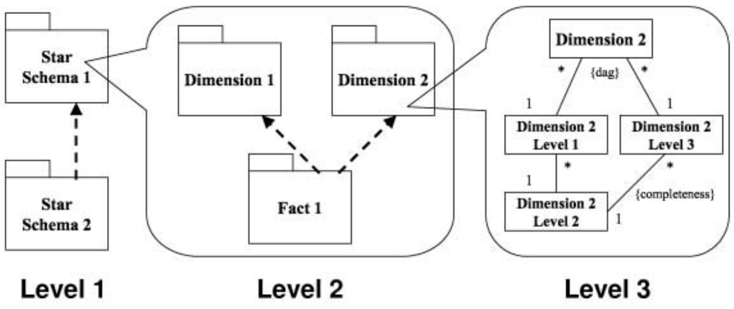


Fig. 2. The three levels of a MD model explosion using packages

Level 1 : Model definition. A package represents a star schema of a conceptual MD model. A dependency between two packages at this level indicates that the star schemas share at least one dimension.

Level 2 : Star schema definition. A package represents a fact or a dimension of a star schema. A dependency between two dimension packages at this level indicates that the packages share at least one level of a dimension hierarchy.

Level 3 : Dimension/fact definition. A package is exploded into a set of classes that represent the hierarchy levels in a dimension package, or the whole star schema in the case of the fact package.

The MD model is designed in a top-down fashion by further decomposing a package. We have limited our proposal to three levels because "deep hierarchies tend to be difficult to understand, since each level carries its own meanings" [13].

5 Applying Package Design Guidelines: a Case Study

In this section, we use a case study to show how our guidelines[3] are properly applied to MD modeling. We use the *supply value chain* example taken from Chapter 5 of [15]. As Kimball states, "The supply side of the business consists of the steps needed to manufacture the products from original ingredients or parts...". Typical DWs that support the supply value chain include seven facts (Purchase Orders, Deliveries, Materials Inventory, Process Monitoring, Bill of Materials, Finished Goods Inventory, and Manufacturing Plans) and nine dimensions (Time, Ingredient, Supplier, Deal, Plant, Ship Mode, Process, Product, and Warehouse).

In Fig. 3, we show the supply value chain example modeled by our approach presented in [7]. The fact classes have been filled in a dark colour, while the dimension classes in a light colour, and the base classes of the dimension hierarchies in white. As seen in Fig. 3, the Time dimension is the only one connected

[3] For the sake of comprehensibility, we explicitly indicate when we apply each guideline.

Table 1. Multidimensional modeling guidelines

N°	**Level**	**Guideline**
0a		At the end of the design process, the MD model will be divided into three levels: model definition, star schema definition, and dimension/fact definition
0b		Before starting the modeling, define facts and dimensions and remark the shared dimensions and dimensions that share some hierarchy levels
1	1	Draw a package for each star schema, i.e., for every fact considered
2a	1	Decide which star schemas will host the definition of the shared dimensions; according to this decision, draw the corresponding dependencies
2b	1	Group together the definition of the shared dimensions in order to minimize the number of dependencies
3	2	Draw a package for the fact (only one in a star package) and a package for each dimension of the star schema
4a	2	Draw a dependency from the fact package to each one of the dimension packages
4b	2	Never draw a dependency from a dimension package to a fact package
5	2	Do not define a dimension twice; if a dimension has been previously defined, import it
6	2	Draw a dependency between dimension packages in order to indicate that the dimensions share hierarchy levels
7	3	In a dimension package, draw a class for the dimension class (only one in a dimension package) and a class for every classification hierarchy level
8	3	In a fact package, draw a class for the fact class (only one in a fact package) and import the dimension classes with their corresponding hierarchy levels
9	3	In a dimension package, if a dependency from the current package has been defined at level 2, import the corresponding shared hierarchy levels
10	3	In a dimension package, when importing hierarchy levels form another package, it is not necessary to import all the levels

to all the facts. For the sake of clearness, we have omitted the definition of the classification hierarchy levels for both the Process and Ship Mode dimensions. Moreover, we have not represented the attributes and methods of the classes either.

Even though we have tried to obtain a clear model, the model is confusing because there are a lot of lines that cross each other. Furthermore, it is difficult to see at a glance the different dimensions connected to a fact. In short, when the scale of a model is large and includes a large number of interconnections among its different elements, it may be very difficult to understand and manage it, especially for end users.

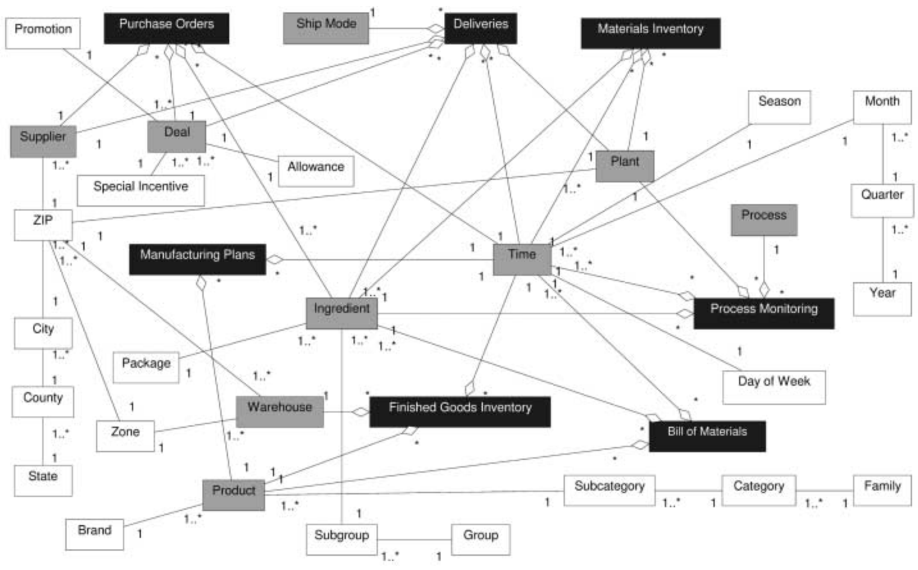


Fig. 3. A partial complex MD model

Guideline 1 and Guideline 2. Fig. 4 shows the first level of the model that is formed by seven packages that represent the different star schemas that form our case study (`G.1`). A dashed arrow from one package to another one denotes a dependency between packages, i.e., the packages have some dimensions in common (`G.2a`). The direction of the dependency indicates that the common dimensions shared by the two packages were first defined in the package pointed to by the arrow (to start with, we have to choose a star schema to define the dimensions, and then, the other schemas can use them with no need to define them again). If the common dimensions had been first defined in another package, the direction of the arrow would have been different. In any case, it is better to group together the definition of the common dimensions in order to reduce the number of dependencies (`G.2b`).

Guideline 3 and Guideline 4. A package that represents a star schema is shown as a simple icon with names. The package contents can be dynamically accessed by "zooming" to a detailed view. For example, Fig. 5 shows the content of the package Purchase Orders Star (level 2). The fact package Purchase Orders Fact is represented in the middle of Fig. 5, while the dimension packages are placed around the fact package (`G.3`). As it can be seen, a dependency is drawn from the fact package to each one of the dimension packages, because the fact package comprises the whole definition of the star schema (`G.4a`). At level 2, it is possible to create a dependency from a fact package to a dimension package or between dimension packages, but we do not allow a dependency from a dimension

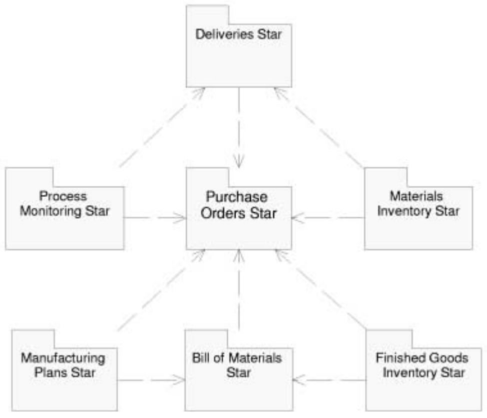


Fig. 4. Level 1: different star schemas of the supply value chain example

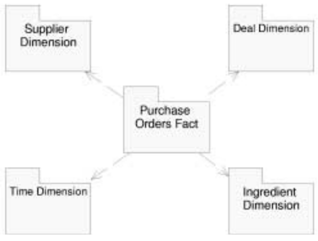


Fig. 5. Level 2: Purchase Orders Star

package to a fact package, since it is not semantically correct in our proposal (G.4b).

Guideline 5 and Guideline 6. Fig. 6 shows the content of the package Deliveries Star (level 2). As in the previous package, the fact package is placed in the middle of the figure and the dimension packages are placed around the fact package in a star fashion. The three dimension packages (Deal Dimension, Supplier Dimension, and Time Dimension) have been previously defined in the Purchase Orders Star (Fig. 5), so they are imported in this package[4] (G.5). Because of this, the name of the package where they have been previously defined appears below the package name (from Purchase Orders Star). Since Plant Dimension and Ship Mode Dimension have been defined in the current package, they do not show a package name. At this level, a dependency between dimension packages indicates that they share some hierarchy levels (G.6). For example, a dependency between Plant Dimension and Supplier Dimension is represented because there is a shared hierarchy[5] (ZIP, City, ...), as shown in Fig. 3.

In a similar way, Materials Inventory Star is further decomposed as shown in Fig. 7 (level 2). All the dimensions that this package contains have been previously defined in other packages. We can notice that it is possible to import packages defined in different star packages.

[4] However, our approach does not forbid defining the same dimension twice but with different names defined by views. For example, the designer can define two dimensions, such as Shipment Date and Reception Date with the same structure instead of defining only one Date dimension.

[5] We have decided to share a hierarchy for both dimensions to obtain a clearer design, although the designer may have decided not to do it if such sharing is not totally feasible.

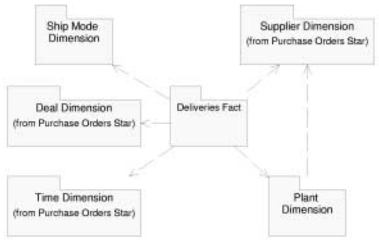

Ship Mode Dimension
Supplier Dimension (from Purchase Orders Star)
Deal Dimension (from Purchase Orders Star)
Deliveries Fact
Time Dimension (from Purchase Orders Star)
Plant Dimension

Fig. 6. Level 2: Deliveries Star

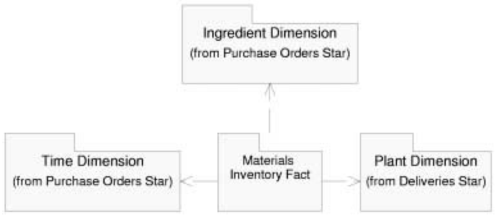

Ingredient Dimension (from Purchase Orders Star)
Time Dimension (from Purchase Orders Star)
Materials Inventory Fact
Plant Dimension (from Deliveries Star)

Fig. 7. Level 2: Materials Inventory Star

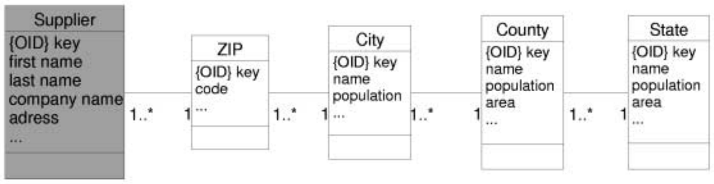


Fig. 8. Level 3: Supplier Dimension

Guideline 7. The content of the dimension and fact packages is represented at level 3. The diagrams at this level are only comprised of classes and associations among them. For example, Fig. 8 shows the content of the package Supplier Dimension (level 3), that contains the definition of the dimension (Supplier) and the different hierarchy levels (ZIP, City, County, and State) (`G.7`). The hierarchy of a dimension defines how the different OLAP operations (roll up, drill down, etc.) can be applied [15].

Guideline 8. Regarding fact packages, Fig. 9 shows the content of the package Purchase Orders Fact (level 3). In this package, the whole star schema is displayed: the fact class (filled in a dark colour) is defined (we show some of its attributes) and the dimensions with their corresponding hierarchy levels are imported (`G.8`). To avoid unnecessary detail, we have hidden the attributes and methods of dimensions and hierarchy levels.

Guideline 9 and Guideline 10. Fig. 10 shows the content of the package Plant Dimension (level 3). This dimension shares some hierarchy levels with Supplier Dimension (Fig. 8). Therefore, we notice that the shared hierarchy levels have been imported (the name of the package where they have been defined appears below the class name) (`G.9`). Furthermore, we also notice a salient feature of our approach: two dimensions, that share hierarchy levels, do not need to share the whole hierarchy (`G.10`). For example, the hierarchy of Plant Dimension does not include State level defined in Supplier Dimension. Other proposals, such as [1][2], when sharing dimension hierarchy levels, share all the hierarchy path from the

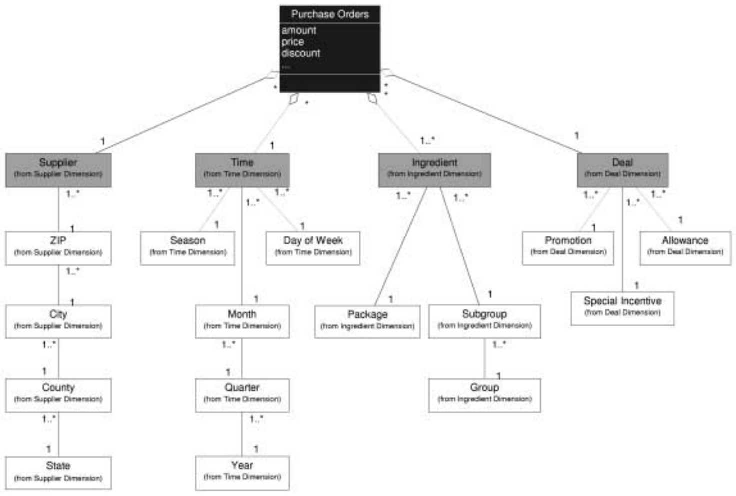

Purchase Orders
amount
price
discount
Supplier (from Supplier Dimension)
ZIP (from Supplier Dimension)
City (from Supplier Dimension)
County (from Supplier Dimension)
State (from Supplier Dimension)
Time (from Time Dimension)
Season (from Time Dimension)
Day of Week (from Time Dimension)
Month (from Time Dimension)
Quarter (from Time Dimension)
Year (from Time Dimension)
Ingredient (from Ingredient Dimension)
Package (from Ingredient Dimension)
Subgroup (from Ingredient Dimension)
Group (from Ingredient Dimension)
Deal (from Deal Dimension)
Promotion (from Deal Dimension)
Allowance (from Deal Dimension)
Special Incentive (from Deal Dimension)

Fig. 9. Level 3: Purchase Orders Fact

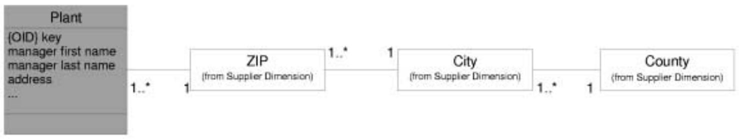

Plant
{OID} key
manager first name
manager last name
address
...
ZIP (from Supplier Dimension)
City (from Supplier Dimension)
County (from Supplier Dimension)
1..*
1

Fig. 10. Level 3: Plant Dimension

shared level. However, the package mechanism allows us to import only the required levels, thereby providing a higher level of flexibility.

6 Definition of Package Stereotypes

In Section 4 and 5, we have shown how the complexity of a MD model is managed by organizing it into logical packages. We have used three different kinds of packages: star package, dimension package, and fact package. In this section, we provide a UML extension by means of stereotypes of the particular packages we have defined.

For the definition of stereotypes, we follow the examples included in the UML Specification [6]:

– Name: The name of the stereotype.
– Base class (also called Model class): The UML metamodel element that serves as the base for the stereotype.
– Description: An informal description with possible explanatory comments.
– Icon: It is possible to define a distinctive visual cue for the stereotype.

– Constraints: A list of constraints defined by OCL expressions associated with the stereotype, with an informal explanation of the expressions.
– Tagged values: A list of all tagged values that may be associated with the stereotype.

We have defined three stereotypes that specialize the Package model element:

– Name: StarPackage
– Base class: Package
– Description: Packages of this stereotype represent MD star schemas
– Icon: Fig. 11 (a)
– Constraints:
  • A StarPackage can only contain FactPackages or DimensionPackages:[6]
    self.contents-¿forAll(oclIsTypeOf(FactPackage) or oclIsTypeOf(DimensionPackage))
  • A StarPackage can only contain one FactPackage:
    self.contents-¿select(oclIsTypeOf(FactPackage))-¿size ¡= 1
  • There are no cycles in the dependency structure[7]:[8]
    not self.allSuppliers-¿includes(self)
– Tagged values: None

– Name: DimensionPackage
– Base class: Package
– Description: Packages of this stereotype represent MD dimensions
– Icon: Fig. 11 (b)
– Constraints:
  • It is not possible to create a dependency from a DimensionPackage to a FactPackage (only to a DimensionPackage):
    self.clientDependency-¿forAll(supplier-¿forAll(oclIsTypeOf(DimensionPackage)))
  • There are no cycles in the dependency structure:
    not self.allSuppliers-¿includes(self)
  • A DimensionPackage cannot contain Packages:
    self.contents-¿forAll(not oclIsKindOf(Package))
– Tagged values: None

– Name: FactPackage
– Base class: Package
– Description: Packages of this stereotype represent MD facts
– Icon: Fig. 11 (c)

[6] contents is an additional operation defined in the UML Specification [6]: "The operation contents results in a Set containing the ModelElements owned by or imported by the Package".
[7] Fowler states: "As a rule of thumb, it is a good idea to remove cycles in the dependency structure" [12].
[8] allSuppliers is an additional operation defined in the UML Specification [6]: "The operation allSuppliers results in a Set containing all the ModelElements that are suppliers of this ModelElement, including the suppliers of these ModelElements. This is the transitive closure".

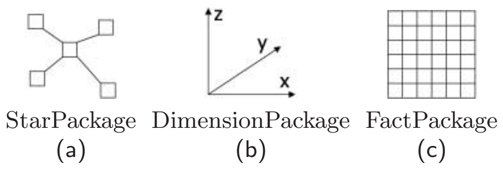


Fig. 11. Stereotype icons of the MD extension

- Constraints:
  - There are no cycles in the dependency structure:
    not self.allSuppliers-¿includes(self)
  - A DimensionPackage cannot contain Packages:
    self.contents-¿forAll(not oclIsKindOf(Package))
- Tagged values: None

7 Using Multidimensional Modeling in Rational Rose

Rational Rose (RR) is one of the most well-known visual modeling tools. As RR supports the UML, it is becoming the common modeling tool for OO modeling. RR is extensible by means of add-ins, that allows us to group together customizations and automation of several RR features through the Rose Extensibility Interface (REI) [16] into one component. An add-in allows us to customize main menu items, data types, stereotypes, etc. In this section, we present an add-in we have developed, that allows us to use the stereotypes we have previously presented in RR. Our add-in customizes the following elements:

- Menu item: We have added the new menu item MD Validate in the menu Tools. This menu item runs a Rose script that validates a MD model: our script checks all the constraints we have presented in Section 6.
- Stereotypes: We have defined the stereotypes we have previously presented in Section 6.

We use our stereotypes in RR by means of a stereotype configuration file. To graphically distinguish model elements from different stereotypes, each stereotype can have a graphical representation. Thus, for each stereotype, there may be four different icons: a diagram icon, a small diagram toolbar icon, a large diagram toolbar icon, and a list view icon.

The best way to understand our extension is to show a tangible example. Fig. 12 shows the level 1 of the supply value chain example (the same level is displayed in Fig. 4 without stereotypes). We can notice the list view icons of our stereotypes in the list of the browser (left hand panel in Fig. 12). Besides, the vertical toolbar in the middle of Fig. 12 contains three new buttons that correspond to our stereotypes.

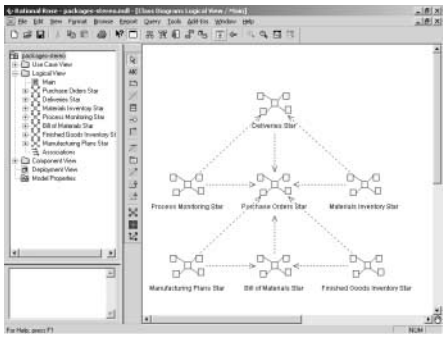


Fig. 12. Multidimensional modeling using Rational Rose (level 1)

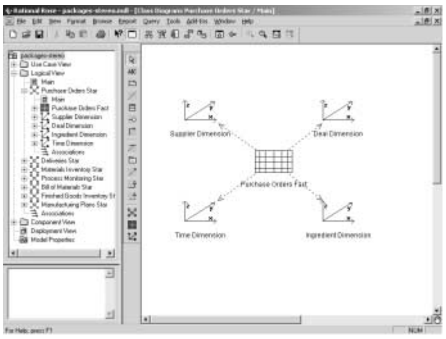


Fig. 13. Multidimensional modeling using Rational Rose (level 2)

Furthermore, we also show how the level 2 of a MD model is displayed in RR. In Fig. 13, the content of Purchase Orders Star is shown (the same level is displayed in Fig. 5 without stereotypes). The icons for the FactPackage and DimensionPackage can be observed.

8 Conclusions and Future Work

Existing multidimensional (MD) modeling approaches use "flat design" in that all the modeling elements are represented in the same diagram. These approaches are not often suitable for huge and complex MD models. Therefore, in this paper, we have presented how UML package mechanisms can be successfully used for MD modeling at three levels of complexity. We have also provided design guidelines, based on our experience in designing real cases, that allow the correct use of the UML packages for simplifying a conceptual design when modeling large and complex data warehouses. We have also illustrated the benefits of these guidelines by applying them to a case study. These guidelines are extremely useful as they allow us to obtain conceptual MD models that can be understood by both designers and analyzers, facilitating the communication between them. Furthermore, we have also provided a UML extension by means of stereotypes of the different package items we use. Finally, to show the applicability of our proposal, this UML extension has been defined for a well-known modeling tool such as Rational Rose 2000, which allows us to put in practice all ideas developed throughout the paper.

Future works are concerned with providing a UML extension including stereotypes for main structural properties of MD modeling (fact class, dimension class, etc.). Further future work refers to extending our approach to allow us to cover all life cycle of MD systems, which involves implementation of MD models into OO and object-relational databases.

References

[1] Golfarelli, M., Rizzi, S.: A methodological Framework for Data Warehouse Design. In: Proc. of the ACM 1st Intl. Workshop on Data warehousing and OLAP (DOLAP'98), Washington D.C., USA (1998) 3–9 199, 201, 208

[2] Sapia, C., Blaschka, M., Höfling, G., Dinter, B.: Extending the E/R Model for the Multidimensional Paradigm. In: Proc. of the 1st Intl. Workshop on Data Warehouse and Data Mining (DWDM'98). Volume 1552 of LNCS., Springer-Verlag (1998) 105–116 199, 201, 208

[3] Tryfona, N., Busborg, F., Christiansen, J.: starER: A Conceptual Model for Data Warehouse Design. In: Proc. of the ACM 2nd Intl. Workshop on Data warehousing and OLAP (DOLAP'99), Kansas City, Missouri, USA (1999) 199, 201

[4] Husemann, B., Lechtenborger, J., Vossen, G.: Conceptual Data Warehouse Design. In: Proc. of the 2nd. Intl. Workshop on Design and Management of Data Warehouses (DMDW'2000), Stockholm, Sweden (2000) 3–9 199, 201

[5] Abelló, A., Samos, J., Saltor, F.: A Framework for the Classification and Description of Multidimensional Data Models. In: Proc. of the 12th Intl. Conference on Database and Expert Systems Applications (DEXA'01), Munich, Germany (2001) 668–677 199

[6] Object Management Group (OMG): Unified Modeling Language Specification 1.4. Internet: http://www.omg.org/cgi-bin/doc?formal/01-09-67 (2001) 199, 200, 209, 210

[7] Trujillo, J., Palomar, M., Gómez, J., Song, I.: Designing Data Warehouses with OO Conceptual Models. IEEE Computer, special issue on Data Warehouses **34** (2001) 66–75 200, 202, 204

[8] Feldman, P., Miller, D.: Entity Model Clustering: Structuring a Data Model by Abstraction. The Computer Journal **29** (1986) 348–360 201

[9] Teorey, T., Wei, G., Bolton, D., Koenig, J.: ER Model Clustering as an Aid for User Communication and Documentation in Database Design. Communications of ACM **32** (1989) 975–987 201

[10] Gandhi, M., Robertson, E., Gucht, D.V.: Leveled Entity Relationship Model. In: Proc. of the 13th Intl. Conference on Entity-Relationship Approach (ER'94). Volume 881 of LNCS., Springer-Verlag (1994) 420–436 201

[11] Siau, K., Cao, Q.: Unified Modeling Language (UML) - A Complexity Analysis. Journal of Database Management **12** (2001) 26–34 201

[12] Fowler, M.: UML Distilled. Applying the Standard Object Modeling Language. Object Technology Series. Addison-Wesley (1998) 201, 210

[13] Conallen, J.: Building Web Applications with UML. Object Technology Series. Addison-Wesley (2000) 201, 204

[14] Giovinazzo, W.: Object-Oriented Data Warehouse Design. Building a star schema. Prentice-Hall, New Jersey, USA (2000) 202

[15] Kimball, R.: The Data Warehouse Toolkit. 2 edn. John Wiley & Sons (1996) 202, 204, 208

[16] Rational Software Corporation: Using the Rose Extensibility Interface. Rational Software Corporation (2001) 211

Comparative Evaluation of Large Data Model Representation Methods: The Analyst's Perspective

Daniel L. Moody

Department of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway N-7491
dmoody@idi.ntnu.no
School of Business Systems, Monash University
Melbourne, Australia 3800
dmoody@infotech.monash.edu.au

Abstract. One of the most serious limitations of the Entity Relationship (ER) Model in practice is its inability to cope with complexity. A number of approaches have been proposed in the literature to address this problem, but so far there has been no systematic empirical research into the effectiveness of these methods. This paper describes a laboratory experiment which compares the effectiveness of different representation methods for documentation and maintenance of large data models (analyst's viewpoint). The methods are compared using a range of performance-based and perception-based variables, including time taken, documentation correctness, consistency, perceived ease of use, perceived usefulness and intention to use. An important theoretical contribution of this paper is the development and empirical testing of a theoretical model (the Method Evaluation Model) for evaluating IS design methods. This model may help to bridge the gap between research and practice in IS design research, as it addresses the issue of method adoption in practice, which has largely been ignored by IS design researchers.

1 Introduction

1.1 The Problem of Complexity in Data Models

One of the most serious practical limitations of the Entity Relationship (ER) Model in practice is its inability to cope with complexity [1, 2, 3, 13, 18, 20, 40, 41, 43]. The two major practical problems with large data models are:

- Ease of understanding (end user's perspective): when data models exceed a certain size, they become difficult for end users to understand.

- Ease of documentation and maintenance (analyst's perspective): when data models exceed a certain size, they become difficult to document and maintain.

Neither the standard ER Model or the Extended Entity Relationship (EER) model provide explicit abstraction mechanisms for managing the size and complexity of real world data models [45].

A number of methods have been proposed to address this issue [e.g. 2, 3, 13, 18, 20, 25, 40, 41], but so far none of these have been widely adopted in practice. Consequently, it remains an open research issue [42]. The greatest weakness in the existing literature is the lack of empirical validation of the methods proposed. So far, there has been no systematic empirical research into the effectiveness of these methods. The authors of the methods argue that their approaches are effective but in most cases, no empirical evidence is provided. Most evidence of successful use of these methods is anecdotal and in many cases reports the direct experience of the author [36].

1.2 Levelled Data Models

A previous paper [27] defined a method for representing large data models based on the organisation of a street directory. A Levelled Data Model consists of the following components (**Fig. 1**):

- A high level diagram, called the *Context Data Model*, provides an overview of the model and how it is divided into subject areas. This corresponds to the key map in a street directory.
- A set of named *Subject Area Data Models* show a subset of the data model (a single subject area) in full detail. These correspond to detail maps in a street directory. *Foreign entities* are used to show cross-references between subject areas—these correspond to inter-map references in a street directory.
- An *Entity Index* are used to help locate individual entities within each subject area.

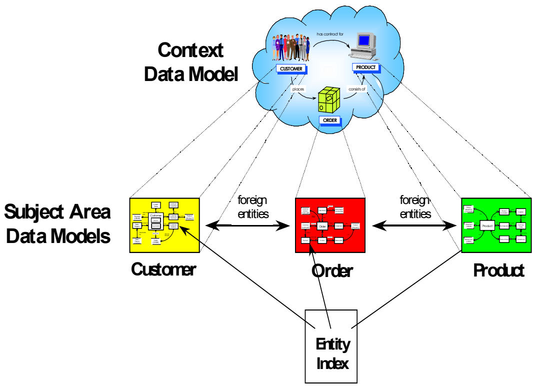


Fig. 1. Levelled Data Model Architecture

The model may be organised into any number of levels, depending on the size of the underlying data model.

1.3 Research Questions

This paper describes a laboratory experiment which evaluates the effectiveness of the Levelled Data Modelling method for documentation and maintenance of large data models (analyst's perspective). A previous experiment evaluated the effectiveness of the method for end user understanding [30]. The broad research questions addressed by this experiment are:

- How effective is the Levelled Data Models method compared to methods previously proposed in the literature ?
- What is the likelihood of the method being adopted in practice?

The second research question addresses an issue that is rarely addressed or even considered in IS design research, despite the fact that this is critical for research to have an impact on practice.

2 The Method Evaluation Model

2.1 Adoption of IS Design Methods

The issue of practitioner acceptance of methods is something which has been largely ignored in IS design research. However, usage in practice is clearly an important pragmatic measure of the "success" of a method and also of the impact of research on practice [15]. Regardless of the potential benefits of IS design methods published in the literature, unless they are used in practice, these benefits cannot be realised.

2.2 A Theoretical Model for Evaluating IS Design Methods

The Method Evaluation Model [28] is a theoretical model for evaluating IS design methods, which incorporates two dimensions of method "success": actual efficacy and adoption in practice. It combines Rescher's theory of Methodological Pragmatism, a theory for validating methodological knowledge [34] and the Technology Acceptance Model [12], a theoretical model for explaining and predicting user acceptance of information technology.

Until now, there seems to have been an implicit belief by researchers that if a method is effective in achieving its objectives, then it will be adopted in practice. In reality however, *perceptions* of efficacy by practitioners are far more important in determining whether a method will be used. According to the Method Evaluation Model, actual efficacy only has an indirect effect on decisions to use a method, through perceptions of efficacy. This relationship was hypothesised based on the Technology Acceptance Model [12] and the Theory of Reasoned Action [14].

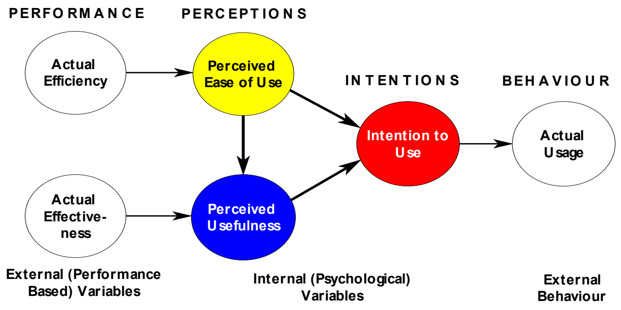


Fig. 2. Method Evaluation Model

3 Research Design

3.1 Research Method Selection

There are a wide variety of research methods which may be used in conducting IS research [5, 16, 17, 33, 35, 46]. Different research methods are appropriate in different situations, depending on the research question and the stage of knowledge in the area being studied [16, 35, 46]. Prior to this study, the proposed method had been extensively tested in practice using an action research approach [28, 29]. It was applied in eight different organisations in eight different industries. The method was refined significantly as a result of use in practice, and reached a point where it was a stable and mature approach (as evidenced by the lack of change from one action research cycle to the next). While action research was an appropriate research method when the method was in its developmental phases, it is clearly less suitable in evaluating the method once it had become stable. A controlled experiment provides the most effective way to evaluate the effectiveness of the proposed method because:

- It allows direct comparisons to be made between different methods under controlled conditions through manipulation of experimental treatments.
- It enables the method to be evaluated using objective and quantitative data.
- It enables the method to be evaluated using independent participants.

3.2 Identifying Methods for Comparison

In order to make the experiment manageable, it was decided to limit the number of methods evaluated to the two leading methods proposed in the literature. The selection of methods was based on the following criteria, which represent a balance between rigour and relevance:

- Academic credibility (rigour): publication in a refereed academic journals
- Practical credibility (relevance): evidence of successful use in practice

Only two methods satisfy both of these criteria: Clustered Entity Models (Feldman and Miller, 1986) and Structured Data Models (Simsion, 1989), so these are the methods that are evaluated. These methods also represent the two predominant paradigms for clustering models: *aggregation* and *generalisation*.

3.3 Experimental Design

A *three group, post-test only* design was used, with one active between-groups factor (representation method). All experimental groups are treatment groups (i.e. no control group). The experimental design is summarised in Fig. 3.

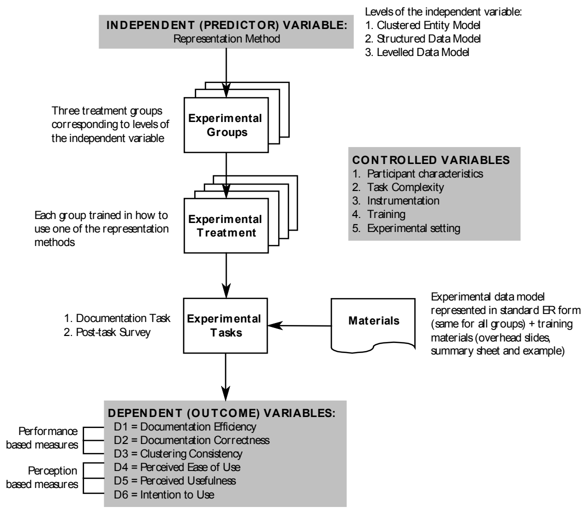

INDEPENDENT (PREDICTOR) VARIABLE: Representation Method
Levels of the independent variable: 1. Clustered Entity Model 2. Structured Data Model 3. Levelled Data Model
Three treatment groups corresponding to levels of the independent variable
Experimental Groups
CONTROLLED VARIABLES 1. Participant characteristics 2. Task Complexity 3. Instrumentation 4. Training 5. Experimental setting
Each group trained in how to use one of the representation methods
Experimental Treatment
1. Documentation Task 2. Post-task Survey
Experimental Tasks
Materials
Experimental data model represented in standard ER form (same for all groups) + training materials (overhead slides, summary sheet and example)
DEPENDENT (OUTCOME) VARIABLES:
D1 = Documentation Efficiency
D2 = Documentation Correctness
D3 = Clustering Consistency
D4 = Perceived Ease of Use
D5 = Perceived Usefulness
D6 = Intention to Use
Performance based measures
Perception based measures

Fig. 3. Experimental Design

3.4 Independent Variable

The independent variable has three levels, corresponding to the different representation methods being compared:

- Treatment Group 1: Levelled Data Models [27]
- Treatment Group 2: Clustered Entity Models [13]
- Treatment Group 3: Structured Data Models [40]

3.5 Dependent Variables

We distinguish between two types of dependent measures:

- Performance based (objective) measures: How effectively are subjects able to perform the experimental task?
- Perception based (subjective) measures: How effective do subjects perceive the method to be?

Performance Based Measures. Three performance based dependent variables were used to evaluate the methods:

- D1: Documentation efficiency: measured by the time taken to complete the experimental task
- D2: Documentation correctness: measured by the level of completeness of the model (whether any information was lost in the transformation process).
- D3: Clustering consistency: measured by the similarity in clustering produced by different participants.

Perception Based Measures. From a scientific viewpoint, objective measures generally provide much more convincing evidence than subjective measures. However in decisions about whether to use a particular method, perceptions play a much more important role, because of the element of free will or *intentionality* in human behaviour [14]. According to the Method Evaluation Model, it is not so much how effective a method *is* but how effective people *think it is* that determines whether they will use it. We define three perception based variables for evaluating the methods:

- D4: Perceived Ease of Use: the degree to which a person believes that using a particular method would be free of effort. This was measured using six items on the post-task survey (Questions 1, 4, 5, 9, 11 and 14).
- D5: Perceived Usefulness: the degree to which a person believes that a particular representation method will be effective in achieving its objectives. This was measured using eight items on the post-task survey (Questions 1, 4, 5, 9, 11 and 14).
- D6: Intention to Use: the degree to which an individual intends to use a particular method. This was measured using two items on the post-task survey (Q10 and Q16).

Actual Usage is not evaluated in this study, as this is not possible in an experimental context. Instead, we assume that like computer usage behaviour, usage of methods can be predicted reasonably well from behavioural intentions [e.g. 22, 26, 37].

3.6 Hypotheses

Each of the research questions defined in Section 1 is broken down into several *hypotheses*, each relating to a particular combination of independent and/or dependent variables.

Research Question 1: Comparison Between Methods

- H1: Participants will perform the documentation task faster using the Levelled Data Models approach than the other two methods.
- H2: Participants will make fewer errors using the Levelled Data Model approach than using the other two methods.
- H3: Participants will produce more consistent clustering using the Levelled Data Model approach than using the other two methods.
- H4: Participants will perceive the Levelled Data Model approach to be easier to use than the other two methods. This follows from H1 and the relationship

between actual efficiency and Perceived Ease of Use defined in the Method Evaluation Model (H10).

- H5: Participants will perceive the Levelled Data Model approach to be more useful than the other two methods. This follows from H4 and the relationship between Perceived Ease of Use and Perceived Usefulness in the Method Evaluation Model (H11).
- H6: Participants will be more likely to use the Levelled Data Model approach than the other two methods. This follows from H4, H5 and the relationships between Perceived Ease of Use, Perceived Usefulness and Intention to Use defined in the Method Evaluation Model (H12, H13).

Research Question 2. Adoption in Practice:

- H7: Participants will perceive the Levelled Data Models method to be easy to use.
- H8: Participants will perceive the Levelled Data Models method to be useful
- H9: Participants will intend to use the Levelled Data Models method

H7-H9 all follow from the results of the field testing, where participants in the action research studies found the method to be easy to use, useful and the method was adopted widely throughout the organisations.

Validation of Causal Relationships. Finally, we posit a number of relationships between dependent variables, based on the causal relationships defined in the Method Evaluation Model.

- H10: Perceived Ease of Use will be determined by Documentation Efficiency. Documentation Efficiency represents a performance based measure of efficiency, while Perceived Ease of Use represents a perception based measure of efficiency. Following the Method Evaluation Model, perceptions of efficiency should be determined by actual efficiency.
- H11: Perceived Usefulness will be determined by Perceived Ease of Use. This is one of the causal relationships defined in the Method Evaluation Model.
- H12: Intention to Use will be determined by Perceived Ease of Use. This is one of the causal relationships defined in the Method Evaluation Model.
- H13: Intention to Use will be determined by Perceived Ease of Use. This is one of the causal relationships defined in the Method Evaluation Model.

3.7 Participants

There were 41 participants in the experiment, all of whom were final year Information Systems students at the University of Melbourne. They had completed two database subjects and were expected to enter the work force 2-3 months after the experiment. They could thus be considered as entry-level IS professionals [7, 31]. All subjects participated voluntarily and were paid $25 on completion of the experiment. Subjects were randomly assigned to experimental groups.

3.8 Experimental Treatment

Each experimental group was given a forty-five minute training session in one of the representation methods being evaluated. To ensure the provision of equivalent training for the four experimental groups, the same worked example and similar instructional materials (overhead transparencies and handouts) were used for all groups.

3.9 Materials

Experimental Data Model. The experimental data model used was an application data model for an airline reservation and scheduling system. This contained 54 entities, so was around the size of a typical application data model. A study by Maier [24] found that the median size of application data models was 60 entities. The same data model was given to each experimental group.

Post-Task Survey. The post-task survey consisted of 16 closed questions, which were the items used to operationalise Perceived Ease of Use (D4), Perceived Usefulness (D5) and Intention to Use (D6). Each item was measured using a 5-point Likert scale, using the opposing statements question format. The order of the items was randomized and half the questions negated to avoid monotonous responses [22].

3.10 Experimental Task

Each experimental group was given the experimental data model represented in ER form and asked to document it using the representational method they had been taught. Subjects were allowed to refer to the training materials while performing the experimental task. A time limit of 1 hour and 15 minutes was allowed, but this was not strictly enforced (participants were allowed to go overtime if they wished, but were not paid any additional amount).

4 Results and Discussion

4.1 Research Question 1: Comparison of Methods

A one-way analysis of variance (ANOVA) was used to analyse differences between experimental groups on all dependent variables. Planned comparisons were carried out using predefined contrasts as part of the ANOVA analysis. Post hoc comparisons were carried out using Tukey's Honestly Significant Difference (HSD) test.

Documentation Efficiency. Table 1 shows the summary statistics for Documentation Time for each experimental group. Subjects in Treatment Group 1 took the longest time, and most did not even complete the task. Qualitative analysis showed that participants were on average only 65% through the task. In contrast, only two participants in Group 2 failed to complete the task (mean = 97% complete) while all participants in Group 3 completed the task.

Table 1. Documentation Time Statistics (minutes)

EXPERIMENTAL GROUP	MEAN (μ)	STDEV (δ)
Clustered Entity Model	95.39	4.68
Structured Data Model	71.34	9.81
Levelled Data Model	**55.96**	11.54

Because Documentation Time was found to be not normally distributed, the Kruskal-Wallis H test was used to test for differences between groups. All of the comparisons were significant with $\alpha < .01$. This means that H1 was strongly supported.

Documentation Correctness. This variable could only be evaluated for two of the experimental groups because most of the participants in Treatment Group 1 did not finish the experimental task. Table 2 shows the summary statistics for the other two treatment groups.

Table 2. Model Completeness Statistics

EXPERIMENTAL GROUP	MEAN (μ)	STDEV (δ)
2. Structured Data Model	58.63%	23.93%
3. Levelled Data Model	**93.67%**	9.95%

A Mann-Whitney U test was used to test for a difference between Treatment Groups 2 and 3. This is used in place of a t-test when the populations being compared are not normal [11]. A significant difference was found between the two groups, which partially confirms H2 ($\alpha < .01$). The likely reason for the loss of information using the Structured Data Modelling method was that participants could not think of an appropriate supertype for some entities, so simply left them out.

Clustering Consistency. Table 3 shows the summary statistics for Clustering Deviation for each experimental group. Subjects in Treatment Group 1 had the largest variation in the number of clusters produced. This is surprising, as considerable effort is required in this method to carry out the clustering process in a rigorous way.

Table 3. Clustering Deviation Statistics

EXPERIMENTAL GROUP	MEAN (μ)	STDEV (δ)
1. Clustered Entity Model	30.77%	35.40%
2. Structured Data Model	22.53%	18.92%
3. Levelled Data Model	**2.04%**	5.18%

Because Clustering Deviation was not normally distributed, the Kruskal-Wallis test was used to test for differences between groups. The analysis showed that Treatment

Group 3 was significantly different to both the other groups, but there was no difference between the other two groups. This strongly confirmed H2 ($\alpha < .01$).

Validity and Reliability Analysis. To evaluate the results for Perceived Ease of Use, Perceived Usefulness and Intention to Use, it is necessary first to evaluate the validity and reliability of their empirical indicators.

Construct Validity. Factor analysis is the preferred technique among researchers for evaluating construct validity. However in this experiment, the sample size was too small, so *inter-item correlation analysis* was carried out instead. Q13 was found to have low convergent validity, and was therefore removed from the analysis.

Reliability. Reliability analysis was conducted on the items used to measure Perceived Ease of Use, Perceived Usefulness and Intention to Use (excluding Item 13). As shown in Table 4, high levels of reliability were found for all constructs, with Cronbach's alpha > .8 in all cases. In the literature, alphas of 0.7 or above are considered by most authors to be acceptable [32].

Table 4. Item Reliabilities for Each Construct

CONSTRUCT	CRONBACH'S α
Perceived Ease of Use	.88
Perceived Usefulness	.85
Intention to Use	.83

Perceived Ease of Use. Table 5 shows the summary statistics for Perceived Ease of Use for each experimental group. Significant differences were found between all experimental groups for Perceived Ease of Use. This means that H3 was strongly supported ($\alpha < .01$).

Table 5. Perceived Ease of Use Statistics

EXPERIMENTAL GROUP	MEAN (μ)	STDEV (δ)
1. Clustered Entity Model	2.51	0.72
2. Structured Data Model	3.54	0.84
3. Levelled Data Model	4.10	0.59

Perceived Usefulness. Table 6 shows the summary statistics for Perceived Usefulness for each experimental group.

Table 6. Perceived Usefulness Statistics

EXPERIMENTAL GROUP	MEAN (μ)	STDEV (δ)
1. Clustered Entity Model	2.93	1.01
2. Structured Data Model	3.29	0.62
3. Levelled Data Model	3.82	0.78

Significant differences were found between Treatment Groups 1 and 3 ($\alpha < .01$), but neither of the other comparisons were significant. This means that hypothesis H4 was partially supported.

Intention to Use. Table 7 shows the summary statistics for Intention to Use for each experimental group. In this case, there was a significant difference between Treatment Groups 1 and 3 ($\alpha < .05$), but no difference between any of the other groups. This means that hypothesis H5 was partially supported.

Table 7. Intention to Use Statistics

EXPERIMENTAL GROUP	MEAN (μ)	STDEV (δ)
1. Clustered Entity Model	2.69	1.11
2. Structured Data Model	3.18	0.87
3. Levelled Data Model	3.61	1.00

4.2 Research Question 2: Adoption in Practice

This research question requires comparing the values of the constructs of the Method Evaluation Model for each experimental group with the "zero point" of the measurement scale (in this case 3). This is used to determine whether responses are significantly positive or negative. Table 8 summarises the results of the one sample t-tests. All comparisons were found to be significant for the Levelled Data Models method, which confirms H7, H8 and H8. The only significant results for the other two methods were for Perceived Ease of Use. The Structured Data Model method was found to be easy to use while the Clustered Entity Model method was found to be difficult to use.

The results show that there is a high likelihood that participants will use the Levelled Data Models method ($\alpha < .05$). Neither of the other methods was perceived to be useful, and participants did not intend to use them, which explains why neither of these methods have been widely adopted in practice.

4.3 Validation of Method Evaluation Model

Regression analysis was used to validate the causal relationships hypothesised in the Method Evaluation Model.

Table 8. Significance of Responses (Static Comparisons)

VARIABLE	1. Clustered Entity Model	2. Structured Data Model	3. Levelled Data Model
Perceived Ease of Use	**No (0.031)**	**Yes (0.033)**	**Yes (0.000)**
Perceived Usefulness	Undecided (0.834)	Undecided (0.103)	**Yes (0.002)**
Intention to Use	Undecided (0.337)	Undecided (0.455)	**Yes (0.041)**

H10: Documentation Efficiency → Perceived Ease of Use. The regression equation resulting from the analysis is:

Equation 1. *Perceived Ease of Use = – .04 * Documentation Time + 6.11*

The regression was found to be significant with $\alpha < .005$, which means that H10 was strongly confirmed. The r^2 statistic showed that Documentation Efficiency accounts for 50% of the variance in Perceived Ease of Use.

H11: Perceived Ease of Use → Perceived Usefulness. The regression equation resulting from this analysis is:

Equation 2. *Perceived Usefulness = .49 * Perceived Ease of Use + 1.65*

The regression was found to be significant with $\alpha < .005$, which means that H11 was strongly confirmed. The r^2 statistic showed that Perceived Ease of Use accounts for 29% of the variance in Perceived Usefulness.

H12 and H13: Perceived Ease of Use + Perceived Usefulness → Intention to Use. The regression equation resulting from this analysis was:

Equation 3. *Intention to Use = .23 * Perceived Ease of Use + .71 * Perceived Usefulness +.01*

The regression was found to be highly significant, with $\alpha < .005$. The adjusted r^2 showed that Perceived Ease of Use and Perceived Usefulness together account for 51% of the variance in Intention to Use. The t-statistics and significance levels for each variable are shown in Table 9. These measure the separate effects of each variable on Intention to Use, while controlling for the effect of the other variable.

Table 9. Multiple Regression Results

INDEPENDENT VARIABLE	T-STATISTIC	SIGNIFICANCE (p)
Perceived Ease of Use	1.41	(.117)
Perceived Usefulness	4.89	**.000**

The relationship between Perceived Usefulness and Intention to Use was found to be highly significant ($\alpha < .005$). This means that H13 was strongly confirmed. However the relationship between Perceived Ease of Use and Intention to Use was *not* statistically significant, which means that H12 was not supported. This result is consistent with many of the studies of the Technology Acceptance Model, which have found that the relationship between Perceived Ease of Use and Intention to Use is not significant after controlling for the effects of Perceived Usefulness [10, 12].

5 Conclusion

5.1 Summary of Findings

This experiment has conducted an empirical comparison of the efficacy (both actual and perceived) of methods proposed in the literature for documenting and maintaining large data models (analyst's viewpoint). The results for each research question are:

- Research Question 1: Out of six hypotheses, three were supported and three were partially supported. The Levelled Data Models method was found to be superior to both of the other methods in documentation efficiency, clustering consistency and perceived ease of use. It was also found to be superior to Structured Data Modelling for documentation correctness and Clustered Entity Modelling for perceived usefulness and intention to use.
- Research Question 2: All three hypotheses were supported. The results show that the Levelled Data Models method is perceived to be easy to use, useful and is likely to be adopted in practice.

5.2 Practical Significance

The most significant outcome of this experiment from a practical viewpoint is the positive results from Research Question 2. This suggests that there is a high likelihood of the Levelled Data Models method being adopted in practice. There have been many extensions proposed to the ER Model in the literature, but very few have been adopted in practice. However practitioners can only decide to use the method if they are actually aware that it exists. The channels for the diffusion of innovations in practice are not well understood, and requires more than publication in scholarly journals [19].

5.3 Theoretical Significance

This is the first experimental evaluation of large data model representational techniques from the analyst's perspective, so represents a significant contribution to knowledge in this area. However the most significant theoretical contribution of this experiment is the development of a theoretical model (the Method Evaluation Model) and associated measurement instrument for evaluating the likelihood of methods being adopted in practice. The results of this experiment showed that the measurement instrument was reliable and valid (with the exclusion of one item), and three out of four causal relationships between constructs evaluated were confirmed.

While the other results of this research are likely to be of interest only to data modelling researchers and practitioners, this model is of relevance to all IS design researchers. It provides a general theoretical framework for evaluating IS design methods, which may help to address the lack of empirical research into the effectiveness of IS design methods [9, 15, 46]. It may also help to bridge the gap between research and practice in IS design, as it addresses the issue of adoption of methods in practice. Unless methods are used by practitioners, they cannot possibly lead to improved IS design practices.

5.4 Strengths and Limitations of the Research

Internal Validity. To guard against problems of internal invalidity, all variables other than the independent variable should be held constant between groups. The following variables were controlled as part of this experiment:

- Participant Characteristics: individual differences between participants in different experimental groups were controlled by the randomisation procedure.
- Task Complexity: the same example data model was used by participants in all groups in the documentation task.
- Instrumentation: the same instruments were used to measure dependent variables for all experimental groups.
- Training: the same amount of training and similar training materials were given to each experimental group.
- Experimental Setting: the location, time of day, time of year, the experimenter and instructions given to subjects were consistent across experimental groups.

External Validity. The greatest threat to the generalisability of the findings of this study was the use of students as experimental subjects. In general, the population from which one selects subjects for the experiment should be representative of the population to which the researchers wishes to generalise results [4]. Because the participants in this study had completed two database units and were about to enter the workforce, they were considered as reasonable proxies for practitioners. Most previous experimental studies on data modelling have used undergraduate students as proxies for analysts [e.g. 6, 7, 8, 21, 23, 31, 38, 39, 44]. However, clearly their level of knowledge and expertise in ER modelling would be significantly less than practitioners (the target population). While level of expertise and experience may have affected overall performance on the task, the fact that this was equalised between experimental groups means that comparative findings (Research Question 1) should still be valid [4]. However the findings with respect to adoption in practice (Research Question 2) should be interpreted with caution.

5.5 Further Research

As discussed under External Validity, the greatest weakness of this experiment was the nature of the sample population used. For this reason, a field experiment using experienced practitioners has subsequently been conducted as a check on the external validity of the results of this experiment [28]. This partially confirmed the results of this experiment.

References

[1] Akoka, J. and I. Comyn-Wattiau (1996): Entity Relationship And Object Oriented Model Automatic Clustering, *Data And Knowledge Engineering*, **20** (1).

[2] Allworth, S. (1996): Using Classification Structures To Develop And Structure Generic Industry Models, In D.L. Moody (Ed.) *Proceedings Of The First Australian Data Management Conference,* Melbourne, Australia: Australian Data Management Association (DAMA), December 2-3.

[3] Allworth, S. (1999): Classification Structures Encourage the Growth of Generic Industry Models, In D.L. Moody (Ed.) *Proceedings of the Eighteenth International Conference on Conceptual Modelling (Industrial Track),* Paris, France: Springer, November 15-18.

[4] Babbie, E.R. (1998): *The Practice of Social Research*, Belmont, California: Wadsworth Publishing.

[5] Baskerville, R.L. and T. Wood-Harper (1996): A Critical Perspective on Action Research as a Method for Information Systems Research, *Journal of Information Technology*, **3** (11): p. 235-246.

[6] Batra, D., J.A. Hoffer, and R.P. Bostrom (1990): Comparing Representations with Relational and EER Models, *Communications of the ACM*, **33** (2), February: p. 126-139.

[7] Bock, D.B. and T. Ryan (1993): Accuracy in Modelling with Extended Entity Relationship and Object Oriented Data Model, *Journal Of Database Management*, **4** (4): p. 30-39.

[8] Bodart, F., A. Patel, M. Sim, and R. Weber (2001): Should the Optional Property Construct be used in Conceptual Modelling: A Theory and Three Empirical Tests, *Information Systems Research*, **12** (4).

[9] Bubenko, J.A. (1986): Information Systems Methodologies - A Research View, In T.W. Olle, H.G. Sol, and A.A. Verrijn-Stuart (Eds.), *Information Systems Design Methodologies: Improving The Practice*: North-Holland,

[10] Chau, P.Y.K. (1996): An Empirical Assessment of a Modified Technology Acceptance Model, *Journal of Management Information Systems*, **13** (2).

[11] Conover, W.J. (1980): *Practical Nonparametric Statistics (2nd edition)*, New York: John Wiley & Sons.

[12] Davis, F.D., R.P. Bagozzi, and P.R. Warshaw (1989): User Acceptance of Computer Technology: A Comparison of Two Theoretical Models, *Management Science*, **35** (8): p. 982-1003.

[13] Feldman, P. and D. Miller (1986): Entity Model Clustering: Structuring A Data Model By Abstraction, *The Computer Journal*, **29** (4).

[14] Fishbein, M. and I. Ajzen (1975): Belief, Attitude, Intention and Behaviour: An Introduction to Theory and Research, Reading, MA: Addison-Wesley.

[15] Fitzgerald, G. (1991): Validating New Information Systems Techniques: A Retrospective Analysis, In H.E. Nissen, H.K. Klein, and R. Hirschheim (Eds.), *Information Systems Research: Contemporary Approaches And Emergent Traditions*: North-Holland,

[16] Galliers, R.D. (1991): Choosing Information Systems Research Approaches, In H.E. Nissen, H.K. Klein, and R. Hirschheim (Eds.), *Information Systems Research: Contemporary Approaches And Emergent Traditions*: North-Holland,

[17] Galliers, R.D. (1992): Information Systems Research: Issues, Methods and Practical Guidelines: Blackwell Scientific Publications.

[18] Gandhi, M., E.L. Robertson, and D. Van Gucht (1994): Levelled Entity Relationship Models, In P. Loucopolous (Ed.) *Proceedings Of The Thirteenth International Conference On The Entity Relationship Approach,* Manchester, December 14-17.

[19] Gibbons, M., C. Limoges, H. Nowotny, S. Schwartzman, P. Scott, and M. Trow (1994): *The New Production of Knowledge: The Dynamics of Science and Research in Contemporary Societies*: Sage Publications.

[20] Gilberg, R.F. (1986): A Schema Methodology For Large Entity-Relationship Diagrams, In P.P. Chen (Ed.) *Proceedings Of Fourth International Conference On The Entity Relationship Approach*,

[21] Hardgrave, B.C. and N.P. Dalal (1995): Comparing Object Oriented and Extended Entity Relationship Models, *Journal Of Database Management*, **6** (3).

[22] Hu, P.J. and P.Y.K. Chau (1999): Examining the Technology Acceptance Model Using Physician Acceptance of Telemedicine Technology, *Journal of Management Information Systems*, **16** (2), Fall: p. 91-113.

[23] Lee, H. and B.G. Choi (1998): A Comparative Study of Conceptual Data Modelling Techniques, *Journal Of Database Management*, **9** (2), Spring: p. 26-35.

[24] Maier, R. (1996): Benefits And Quality Of Data Modelling-Results Of An Empirical Analysis, In B. Thalheim (Ed.) *Proceedings Of The Fifteenth International Conference On The Entity Relationship Approach,* Cottbus, Germany: Elselvier, October 7-9.

[25] Martin, J. and C.L. McClure (1985): *Diagramming Techniques for Analysts and Programmers*, Englewood Cliffs, N.J.: Prentice-Hall, xvi, 396.

[26] Mathieson, K. (1991): Predicting User Intention: Comparing the Technology Acceptance Model with the Theory of Planned Behaviour, *Information Systems Research*, **2** (3), September: p. 173-191.

[27] Moody, D.L. (1997): A Multi-Level Architecture For Representing Enterprise Data Models, In D.W. Embley and R.C. Goldstein (Eds.), *Proceedings Of The Sixteenth International Conference On Conceptual Modelling (ER'97),* Los Angeles, November 1-3.

[28] Moody, D.L. (2001): *Dealing with Complexity: A Practical Method for Representing Large Entity Relationship Models (PhD Thesis)*, Melbourne, Australia: Department Of Information Systems, University of Melbourne.

[29] Moody, D.L. (2002): Validation of a Method for Representing Large Entity Relationship Models: An Action Research Study, In *Proceedings of the Tenth European Conference on Information Systems (ECIS'2002),* Gdansk, Poland, June 6-8.

[30] Moody, D.L. (2002): Complexity Effects On End User Understanding Of Data Models: An Experimental Comparison Of Large Data Model Representation Methods, In *Proceedings of the Tenth European Conference on Information Systems (ECIS'2002),* Gdansk, Poland, June 6-8.

[31] Nordbotten, J.C. and M.E. Crosby (1999): The Effect of Graphic Style on Data Model Interpretation, *Information Systems Journal*, **9**.

[32] Nunally, J. (1978): *Psychometric Theory (2nd edition)*, New York: McGraw Hill.

[33] Nunamaker, J., M. Chen, and T.D.M. Purdin (1991): Systems Development in Information Systems Research, *Journal of Management Information Systems*, **7** (3), Winter: p. 89-106.

[34] Rescher, N. (1977): Methodological Pragmatism: Systems-Theoretic Approach to the Theory of Knowledge, Oxford: Basil Blackwell.

[35] Shanks, G., A. Rouse, and D. Arnott (1993): A Review of Approaches to Research and Scholarship in Information Systems, In *Proceedings of the 4th Australian Conference on Information Systems,* Brisbane,

[36] Shanks, G.G. (1996): *Building And Using Corporate Data Models (PhD Thesis)*, Melbourne, Australia: Department Of Information Systems, Monash University.

[37] Sheppard, B.H., J. Harwick, and P.R. Warshaw (1988): The Theory of Reasoned Action: A Meta-Analysis of Past Research with Recommendation for Modification and Future Research, *Journal of Consumer Research*, **15** (3), December: p. 325-343.

[38] Shoval, P. and M. Even-Chaime (1987): Database Schema Design: An Experimental Comparison Between Normalisation and Information Analysis, *Database*, **18** (3), Spring: p. 30-39.

[39] Shoval, P. and S. Shiran (1997): Entity-Relationship and Object-Oriented Data Modeling: An Experimental Comparison of Design Quality, *Data & Knowledge Engineering*, **21**: p. 297-315.

[40] Simsion, G.C. (1989): A Structured Approach To Data Modelling, *The Australian Computer Journal*, August.

[41] Teory, T.J., G. Wei, D.L. Bolton, and J.A. Koenig (1989): ER Model Clustering As An Aid For User Communication And Documentation In Database Design, *Communications Of The ACM*, August.

[42] Thalheim, B. (1999): The Strength of ER Modelling, In *Current Issues in Conceptual Modelling*, P.P. Chen, et al. (Eds.), Springer-Verlag (Lecture Notes in Computer Science).

[43] Wand, Y. and R.A. Weber (1993): On the Ontological Expressiveness of Information Systems Analysis and Design Grammars, *Journal of Information Systems*, October.

[44] Weber, R.A. (1996): Are Attributes Entities? A Study Of Database Designers' Memory Structures*, Information Systems Research*, June.
[45] Weber, R.A. (1997): *Ontological Foundations Of Information Systems*, Melbourne, Australia: Coopers And Lybrand Accounting Research Methodology Monograph No. 4, Coopers And Lybrand.
[46] Wynekoop, J.L. and N.L. Russo (1997): Studying Systems Development Methodologies: An Examination Of Research Methods*, Information Systems Journal*, **7** (1), January.

Extracting Conceptual Relationships from Specialized Documents

Bowen Hui[1] and Eric Yu[2]

[1] Department of Computer Science, University of Toronto bowen@cs.utoronto.ca
[2] Faculty of Information Studies, University of Toronto yu@fis.utoronto.ca

Abstract. Conceptual modeling has been fundamental to the management of structured data. However, its value is increasingly being recognized for knowledge management in general. In trying to develop suitable conceptual models for unstructured information, issues such as the level of representation and complexity of processing techniques arise. Here, we investigate the use of a conceptual model that is simple enough to allow efficient automatic extraction from documents. Our model focused on the problem-solution relationship that is central to the analysis of scientific papers. It also consists of supporting relationships such as benefits and drawbacks, assumptions, methods, extensions, and claims. Our study considered two kinds of documents – scientific research papers and patents. We evaluated the utility of the approach by building a prototype system and our user evaluation shows promising results.

1 Introduction

Conceptual modeling has been fundamental to the management of structured data. However, its value is increasingly being recognized for knowledge management in general. Much of human knowledge is expressed in the form of textual documents. There are many initiatives, such as the Semantic Web [1], to structure or mark up text into forms that will allow them to be interpreted and processed according to some semantic model. In trying to develop suitable conceptual models for unstructured information, one faces a number of issues and trade-offs. Ideally, one would like to have detailed conceptual distinctions and relationships that would enable specific inferences and conclusions to be drawn. However, extracting knowledge from text according to fine-grained conceptual models can be very labour-intensive and error-prone. It is also difficult to come up with models that can be widely agreed upon. Pragmatically, one would like to be able to adopt conceptual models that are sufficiently detailed to be useful for a given task, and that the cost of applying that conceptual model to the textual source is justified by the benefits. In this paper, we investigate the use of a conceptual model that is simple enough to allow efficient automatic extraction from documents. We evaluate the utility of the approach by building a prototype system to perform the extraction, then conducted empirical studies with users.

We focus on specialized classes of documents which have a dominant semantic relationship of interest. In scientific and technological research, a crucial relationship is that which relates a problem to a solution. Many documents identify a problem of current interest, then offer a solution to address the problem.

The ability to extract the problem-solution relationship quickly from this kind of document can aid in the cursory comprehension of the document, and to place this piece of means-ends knowledge within a network of such relationships. Supporting relationships include benefits and drawbacks, assumptions, methods, extensions, and claims.

Our study considered two kinds of documents which we believed to be amenable to this extraction approach – scientific research papers and patent documents. Information extraction techniques are used to automatically analyze documents according to the conceptual model. The extracted information is presented to users for evaluating the extent to which they find the tool useful and usable. The results are compared to those typically obtained in the information extraction (IE) and text summarization (TS).

First, we describe our proposed conceptual model in Section 2 and the environment under which we envision it will be used. Section 3 describes the components and algorithms of an implemented system that automatically extracts text into our conceptual model. In order to measure system performance, we use accuracy metrics, such as recall and precision, commonly used in information extraction and information retrieval (IR) literature. In addition, we use a well-known usability evaluation method from human-computer interaction (HCI) to measure the usability of our model. An empirical study is reported in Section 4. Finally, Sections 5 and 6 discuss the results, present the contributions, and identify future work.

2 Conceptual Model

Libraries have developed standard ways of organizing books and articles. Typical organization schemes include the use of the author's name, document title, or subject area, which are usually based on keywords [2,3]. All of these are explicit *tags* of literal strings in the preamble of the document. These schemes are useful as search terms, but they play a very small role in addressing the task of a user who is reading the document. Since these literal tags do not reflect the semantic content written in the paper, they cannot be used to address user tasks. In particular, unless it is written explicitly in the title, these tags do not help in identifying the softgoals of a problem or the weaknesses of a proposed solution. Therefore, we need a way to organize document concepts and relate them to the goals of different user tasks. To achieve this, a conceptual model for documents needs to be developed. Here, we discuss the model for patents and scientific papers that propose original solutions.

Both scientific articles and patents offer unique solutions, using assumptions and stating claims, to various domain goals. By examining user tasks operated around these elements, such as analyzing and comparing different techniques, we create a set of core concepts based on these common goal-oriented elements. By structuring the concepts into relations based on how the concepts are used, the model shown in Figure 1 is formed.

Figure 1 shows the ER model representing the structure of a document. A document identifies a problem and conveys a unique solution. Each problem may

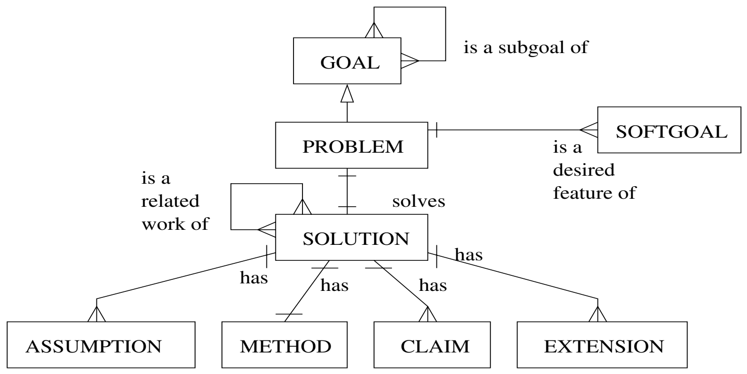


Fig. 1. The Conceptual Model Based On User Goals

be a subproblem of one or more larger goals. Note that a problem is just a small goal that is immediate to particular solutions. A problem also has softgoals, which particular attempted solutions may or may not have addressed. The solution is contrasted with other related works, and is composed of assumptions or beliefs, methods taken to achieve the solution, claims and limitations of the solution, and steps for future extensions. A document may have other kinds of information, such as background material or text structure. These kinds of information are not used in our model because they do not contribute to the goals of the users. More formally, our model has the following concepts: Goal, Problem, Softgoal, Solution, Assumption, Method, Claim, and Extension. The relationships and cardinalities among the concepts are shown in Figure 1.

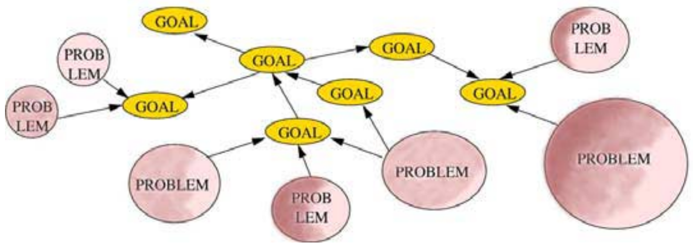


Fig. 2. Multiple Documents Connected By Our Ontology

The intention is to relate multiple documents together using an ontology of research problems consisting of Problem and Goal instance nodes, as illustrated in Figure 2. Each document can relate to multiple problems and goals and many documents can share the same problem. Goal nodes may be linked to each other if one is a subgoal of another. The size of the Problem node indicates the number of solutions associated with it and the colour density indicates the recency of publications.

Examining the landscape of ideas as in Figure 2 allows analysts to detect current patterns in research methodologies. Large PROBLEM nodes represent a well-tackled issue while small PROBLEM nodes represent widely opened questions. Colour is used as an indicator for chronology of the research. For example, a dark colour represents works in the recent decade, a neutral colour represents works from two decades before that, and a light colour represents all other previous works. Densely coloured nodes on the map represent hot topics while lightly coloured nodes represent less active research.

To address user tasks, we describe how an analyst might use a visionary system with our model and ontology. By implementing search and inference functions into the system, one could select a GOAL node and identify which companies or laboratories have done similar work. By comparing different METHOD nodes of the same PROBLEM, one could identify which methods have not yet been tried. Detecting a “collision of ideas” is also possible and can prevent intellectual property conflicts. Detecting “holes” in the ontology allows analysts to attempt novel ideas and set new trends in research and development. The ontology in Figure 2 serves as an overview of problems to tackle in our visionary system. Upon clicking on a particular node, the system would zoom into a document conceptual model that attempted to tackle the selected node. By clicking on a node in the document model, details would appear on demand. In this manner, the ontology provides an overview on one part of the screen while the concept structure of the paper is displayed beside it and detailed text beneath it. A mock-up of this application is illustrated in Figure 3. Details of the document model on the right side of the mock-up will be explained in Section 3. Coupled with this ontology, our model provides support for information researchers and addresses user goals effectively.

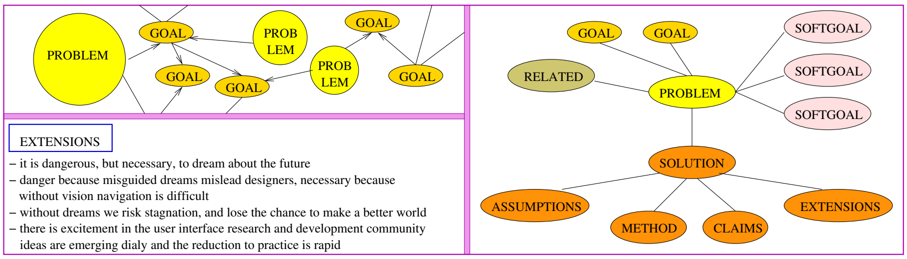

PROBLEM
GOAL
PROB LEM
GOAL
PROB LEM
GOAL
GOAL
GOAL
EXTENSIONS
– it is dangerous, but necessary, to dream about the future
– danger because misguided dreams mislead designers, necessary because without vision navigation is difficult
– without dreams we risk stagnation, and lose the chance to make a better world
– there is excitement in the user interface research and development community ideas are emerging dialy and the reduction to practice is rapid
GOAL
GOAL
SOFTGOAL
RELATED
PROBLEM
SOFTGOAL
SOFTGOAL
SOLUTION
ASSUMPTIONS
METHOD
CLAIMS
EXTENSIONS

Fig. 3. Our Visionary System

3 System Implementation

The purpose of the system is to evaluate the effectiveness of our model. We implemented a system that creates the conceptual model for given documents. An automatic extraction algorithm is used to analyze the free text and information is displayed in a graphical layout. Major components of this system are preprocessing, segmentation, merging, and interface generation.

Scientific articles and patents are entered into the system and relevant sections are extracted in the preprocessing stage. Extracted sections are passed

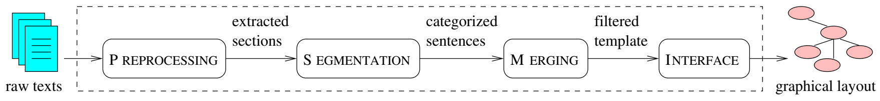


Fig. 4. Document Visualization System

into the segmentation module, where each sentence is assigned a category based on pre-defined hand-coded patterns. Categorized sentences are passed into the merging module, where similar sentences are merged and common phrases are removed. The output is a set of point-form phrases or sentences in a template format. Finally, the interface component takes the textual template and transform it into a graphical layout. Figure 4 shows the interaction of these components.

Slot Name	Abbrev.	Description
GOAL	G	high-level goals, domain goals
PROBLEM	P	immediate goals
RELATED	R	cited or discussed related works
SOLUTION	S	overall proposed approach
SOFTGOAL	F	desired benefits of the area
ASSUMPTIONS	A	assumptions, definitions, beliefs, trends
METHOD	M	particular algorithms, steps, devices
CLAIMS	C	results, claims, advantages, limitations
EXTENSIONS	Z	extensions, future directions
OTHER	X	other information: e.g. background, evidence

Table 1. Template Slots

All of these modules are based on a template structure which we have created and shown in Table 1. This canonical template is derived from the ER model in Figure 1 and is used as the target output so that information extracted from documents can fill up the slots. Because a text template represents the information of one document, some slots are different from the entities used in the ER model. For instance, in the ER model, a SOLUTION entity is related to other SOLUTION entities. In the text template, the proposed solution of the document is represented as SOLUTION and other works are put into RELATED. Also, rather than having multiple ASSUMPTION, CLAIM, and EXTENSION entities, we have created one slot for each of these that may store multiple extracted facts. Table 1 also includes a "dummy" slot, OTHER, which we have inserted to temporarily store information that is irrelevant for our system. This category and its information do not appear in the final interface to the users. A

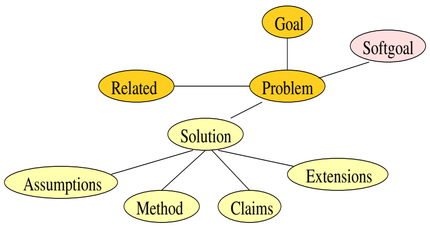


Fig. 5. Graphical Layout

graphical layout derived from the canonical template is shown in Figure 5. Each node corresponds to a slot in the template, with OTHER left out. This view of concepts is presented to the users.

3.1 Preprocessing

A document is converted into ASCII and excerpts are extracted and used as the basis of analysis. In particular, the abstract, introduction, and conclusion sections for scientific articles and the abstract and summary sections for patents are used. We believe that these sections are representative of the main ideas of the entire paper, and hence, sufficient in fulfilling our document analysis task. Documents that cannot be generated as text or that do not have the required sections are discarded.

3.2 Segmentation

The segmentation phase is responsible for categorizing sentences into template slots. An initial set of hand-coded patterns that use discourse cues were created from a small training sample. All patterns in the system are hand-coded because extensive results show that hand-coded patterns outperform machine-learned patterns [4]. The segmentation first tries to identify patterns that match sentences (for most of the categories) and then match patterns at the word level (for GOAL and SOFTGOAL only). If a sentence does not have an assigned category, the system tries to copy the category of the previous sentence. If by the end there are sentences still left unassigned then they get the category OTHER. These steps are illustrated in Algorithm 1.

```
for each document excerpt
{
    1. classify each sentence into a category
    2. if a sentence does not have a pattern after step 1, then
          if its previous sentence has an assigned category, then
             inherit that category
    3. identify patterns for phrases matching goals and softgoals
    4. classify all unassigned sentences as OTHER
}
```

Algorithm 1: The Segmentation Algorithm

Patterns are defined at the word level, matching sequences of words in the text. About 20 documents (8 patents,12 scientific articles – henceforth denoted as: 8 pat/12 sci) were used to develop these patterns. The current prototype has a total of 316 rules[3]. Examples of patterns and the number of patterns defined for each template slot are given in Table 2.

[3] Our patterns are implemented using regular expressions in Perl. When we count the number of patterns in this component, we are giving the number of *lines* of regular expressions we used. On average, patterns matching GOAL and SOFTGOAL were as short as 2 words per regular expression. The other patterns vary greatly; one regular expression may consist of just one phrase, while another may have a phrase that has a disjunction of 10 nouns and 10 verbs.

Unlike most IE systems, discourse heuristics are not used. However, these heuristics can be embedded into the module between step 2 and step 3.

Slots	Sample Patterns	Num
PROBLEM	"Our objective in ... is the study of"	38
RELATED	"Another successful approach"	26
SOLUTION	"We propose"	8
ASSUMPTIONS	"A long standing objection"	24
METHOD	"to achieve ... is used"	50
CLAIMS	"significant improvement"	54
EXTENSIONS	"yet to be investigated"	10
OTHER	"In this section, we ..."	46
GOAL	"technology in education"	17
SOFTGOAL	"immediate feedback"	43

Table 2. Sample Patterns

Based on these patterns, sentences from the document excerpts that are deemed relevant are categorized into one of the slots. These categorized sentences are the output of the segmentation phase.

3.3 Merging

The algorithm for the merging phase is shown in Algorithm 2.

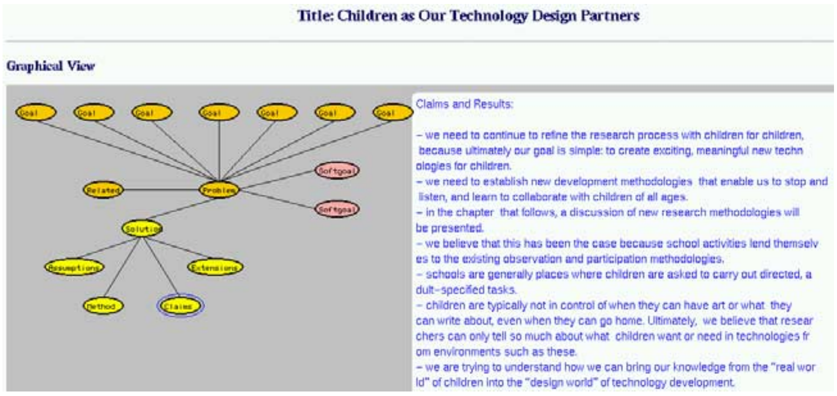
```
for each categorized sentence,
{
    1. order each sentence into the appropriate slot in a template
    for each slot in the template
    {
       2. for each sentence in this slot
          - remove common phrases and transition phrases
       3. for every pair of sentences in this slot
          - stem the sentences
          - if the stemmed pair is similar, then
               keep the more concise one
    }
}
```

Algorithm 2: The Merging Algorithm

The merging algorithm puts the categorized sentences into the appropriate slot (step 1) and makes the output more condensed (steps 2-3). In step 2, sentences are shortened by removing common cue phrases. In addition, topicalized phrases fewer than five words (such as "In addition" and "To our knowledge") are removed. Step 3 compares pairs of sentences within each slot and merges them if they are similar. Similarity occurs when a sentence is a substring of the other or when the two sentences have more than 80% identical words after stemming the sentences using the Porter algorithm [5]. To merge a pair of sentences, the system chooses the shorter one because it considers it to be more concise.

3.4 Interface

The interface produces the graphical model of the template and shows textual details on the side when a node is clicked on. If a node does not have any associated text – either because the algorithm did not detect relevant sentences or that the original excerpts did not have any information – then the interface displays "Text not available from excerpts." Figure 6 shows a snapshot of the system prototype, with the title of the paper placed at the top of the screen.



Title: Children as Our Technology Design Partners

Claims and Results:

– we need to continue to refine the research process with children for children, because ultimately our goal is simple: to create exciting, meaningful new techn ologies for children.
– we need to establish new development methodologies that enable us to stop and listen, and learn to collaborate with children of all ages.
– in the chapter that follows, a discussion of new research methodologies will be presented.
– we believe that this has been the case because school activities lend themselv es to the existing observation and participation methodologies.
– schools are generally places where children are asked to carry out directed, a dult–specified tasks.
– children are typically not in control of when they can have art or what they can write about, even when they can go home. Ultimately, we believe that resear chers can only tell so much about what children want or need in technologies fr om environments such as these.
– we are trying to understand how we can bring our knowledge from the "real wor ld" of children into the "design world" of technology development.

Fig. 6. Sample System Use

This figure shows that the Claims node is highlighted to indicate that it has been clicked on by the user. The details of the node appears in point-form on the right-hand side of the screen. The interface maintains the structural view of the paper while displaying details on demand.

4 An Empirical Study

An evaluation of system performance is measured via standard accuracy tests and an assessment of our document conceptual model is measured via standard usability tests. In order to assess accuracy and usability, the two major components – segmentation and merging – were evaluated separately. This section briefly describes the evaluation techniques used. For further details, please refer to cited references. The procedure and results of the two evaluations are reported.

4.1 Accuracy Evaluation

Participants. To achieve reliability, we performed an intercoder reliability study on 8 documents (4 pat/4 sci). The three coders were $\alpha = 43\%$ reliable above chance when considering all the categories and $\alpha = 53\%$ reliable above chance when considering only sentence level categories (i.e., without Goal and Softgoal). Then, a third-party coder continued and evaluated the system on 93 more documents (47 pat/46 sci).

Stimuli. A random set of patents were retrieved from the web site at the United States Patent and Trademark Office[4] by using search keys such as "chinese" and "fonts", or "multiplication" and "learning". A random set of scientific articles were retrieved from many sources over the Internet. Topics of articles used in training included psychology, linguistics, human-computer interaction, computer graphics, statistics, and economics. The specific topics and the number of documents used for testing are listed as follows:

- **Patents:** colour toy (11), education mathematics (12), design blouse (12), modern chair (12)
- **Scientific Articles:** women and language (10), children-related HCI (12), interface-related HCI (12), computational linguistics (12)

In sum, 51 patents and 50 scientific articles were used in this study.
Procedure. The coding system used were a set of letters to identify which sentence exhibit relevant words or phrases for particular slots in a template. Hence, we used the slot names as classification labels (cf. Section 3).
Analysis. Standard recall and precision measures from IR and IE were used:

$$recall = \frac{items\ gathered}{items\ in\ standard} \qquad precision = \frac{items\ correctly\ gathered}{items\ in\ standard}$$

where an *item* is typically a syntactic unit or a semantic concept. Combining precision and recall produces a single metric, *F-score*:

$$F - score = \frac{(\beta^2 + 1)P * R}{(\beta^2)P + R} \qquad when\ \beta = 1, F - score = \frac{2PR}{P + R}$$

This kind of evaluation measures fine-grained accuracy against a "gold standard". In our evaluation, the gold standard used was the manual validation of a third-party human evaluator.
Results. Segmentation results are shown from two perspectives – performance by domain and performance by category. The former is presented in Tables 3 and 4 and the latter in Table 5. (Note: N_c is the number of $\underline{c}$orrect items, N_g is the number of items $\underline{g}$athered by the system, and N_s is the number of $\underline{s}$tandard items (i.e., items marked by the evaluator).)

Domain	**N$_c$**	**N$_g$**	**N$_s$**	**Precision**	**Recall**
Pat 1: colour toy	236	319	324	73.981	72.839
Pat 2: education mathematics	349	429	430	81.351	81.162
Pat 3: design blouse	254	386	400	65.803	63.500
Pat 4: modern chair	219	326	311	67.177	70.418
Average	264	365	366	72.078	71.979

Table 3. Segmentation Results for Patents by Domain

From Tables 3 and 4, we see that accuracy results were generally low for scientific articles, about 48%, and significantly higher for patents, about 72%.

[4] The web site address is `http://www.uspto.gov`.

We believe that these results are credited to the consistent writing style that patents exhibit. In Table 5, RELATED, ASSUMPTIONS, and EXTENSIONS have less than 10% recall. This means that much of the information belonging to these categories were not extracted. RELATED and ASSUMPTIONS also have low precision scores of 35% and 33%. This means that much of the information that was classified in these slots were incorrect. The remaining categories received decent or good accuracies, as they are comparable to extraction systems that compete in Message Understanding Conferences (MUC) whose average F-score on the overall IE task is approximately 60 [4].

Domain	**N**$_c$	**N**$_g$	**N**$_s$	**Precision**	**Recall**
Sci 1: women and language	202	534	527	37.827	38.330
Sci 2: children-related HCI	428	769	791	55.656	55.656
Sci 3: interface-related HCI	260	507	502	51.282	51.792
Sci 4: computational linguistics	314	636	640	49.371	49.062
Average	301	612	615	48.534	48.710

Table 4. Segmentation Results for Scientific Articles by Domain

Category	**N**$_c$	**N**$_g$	**N**$_s$	**Precision**	**Recall**
GOAL	232	242	252	95.867	92.063
PROBLEM	133	216	261	61.574	50.957
RELATED	6	17	113	35.294	05.309
SOLUTION	95	110	119	86.363	79.831
SOFTGOAL	202	329	300	61.398	67.333
ASSUMPTIONS	9	27	109	33.333	08.256
METHOD	782	1516	1031	51.583	75.848
CLAIMS	387	656	826	58.993	46.852
EXTENSIONS	3	4	42	75.000	07.142
OTHER	413	789	872	52.344	47.362
Total	2262	3906	3925	57.911	57.631

Table 5. Segmentation Results By Category

4.2 Usability Evaluation

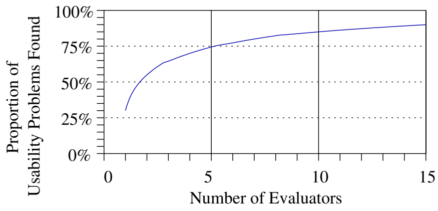


Fig. 7. Usability problems found by heuristic evaluation as a function of evaluators (taken from [6]).

Heuristic evaluation is an inspection method for assessing system usability. The evaluator's job is to inspect the interface by carrying out purposeful tasks that allows him/her to navigate through different parts of the interface. Evaluators are asked to comment on the interface with respect to a set of design principles. The result of a heuristic evaluation session is a compilation of problems with the system. Since different evaluators tend to identify different problems, Nielsen [6] empirically shows that 50% to 75% of problems can be identified with just 3 to 5 evaluators (see Figure 7). Hence, this method is very cost-effective.

Because our system extracts relevant information and puts it into a more concise representation, our system is in essence performing a basic summarization task. Therefore, we use text summarization metrics to evaluate the output of our system. First, we amalgamated facts on "summary worthiness" [7–10] and derived an initial set of heuristics, or *principles*, for the evaluation, as follows:

1. **Conciseness:** Components should not contain information that is irrelevant or redundant. Every extra unit of information competes with the relevant units of information and diminishes their relative visibility. All information should appear in a natural and logical order.
2. **Retention:** Information retained in the system output should be representative of the key concepts and main points made in the original document. Are the major objectives of the paper captured in the summary? What about the major steps in the proposed solution and the results?
3. **Coherence:** All information should be coherent within each component as well as the overall summary. Sentences need not be perfectly grammatical, but each point should make sense in its context.
4. **Consistency:** Each component should be expressed clearly in words, phrases, and concepts consistent with those in the original document. Users should not have to wonder whether different words, situations, or actions mean the same thing.
5. **Informativeness:** Information should be presented in a useful and easily accessible way. Some interface issues may be influential here as well. Irrelevant information should be omitted and words should not clutter the display of the information.
6. **Comprehensibility:** Each point of information should be easy to understand. Users should not have to look up related information in another part of the system in order to understand a particular component.
7. **Fit For Audience:** The information and the style of presentation fits for the intended audience. Audience may vary in the experience of the domain knowledge. Access to different kinds of information should be easy and clear. The ability to show, modify, and hide information should be made obvious to the users.
8. **Fit For Purpose:** The information and the style of presentation fits for the intended task and purpose. These two points should be taken into consideration.

To let evaluators assess the usefulness of the model, we used a question-answering task modeled after a conference reviewer's task. This way, evaluators acted as reviewers using only the system output. To keep the workload of the evaluator to a minimum, we limited the task to three questions:

1. What is the problem addressed by this work? Does it describe why the problem is significant?
2. Does the work present the approach taken to solve the problem targeted? Is the design or implementation of a system described in terms of key ideas of the approach?

3. What are the contributions of this work? Are the benefits and limitations clear? Are the results positive or negative?

The rest of this section describes the details of our user study.

Participants. Four University of Toronto graduate students with extensive experience in writing summaries and abstracts for scientific papers participated in this study. Each of them evaluated 4 system generated output shown on the system's graphical interface. They are henceforth referred to as evaluators.

Stimuli. Evaluators were provided with the system prototype on a web browser. On different pages, the browser displayed the instructions, two document models, the question-answering task description, and the summarization principles. When commenting on the principles, evaluators were allowed to read to the original documents. Of the 93 documents used in the accuracy experiment (cf. Section 4.1), four were chosen for this study based on these criteria:

– patent that scored the lowest on recall and precision
– patent that scored the highest on recall and precision
– scientific article that scored the lowest on recall and precision
– scientific article that scored the highest on recall and precision

Because some document excerpts were extremely short, an additional criteria in our selection is that the document must have more than 25 sentences. Finally, the documents used in this study had the accuracies shown in Table 6.

Label	**Document Type**	**Precision**	**Recall**
P1	Patent	32.0	25.806
P2	Patent	100.0	100.0
S1	Scientific article	29.730	30.556
S2	Scientific article	82.222	78.723

Table 6. Accuracy of Documents Used in Usability Evaluation

Procedure.[5] An overview of the system and an explanation of the evaluation procedure and the summarization principles were provided for the evaluators. Each evaluator evaluated 4 document outputs independently. Evaluators were asked to type in their comments in an email and summarize their results on a 5-point scale. Principles that could not be assessed in the session were assigned as non-applicable, which corresponds to a score of 0.

Analysis. To generate a summary of the collected data, the scores from each evaluator were tallied for each principle and document. These values gave a sense of how usable the system was with respect to each principle. Since there were four evaluators and four documents, each principle may have a maximum score of 80 points (i.e., if all judgments score a 5) and a minimum of 0 points (i.e., if all judgments were non-applicable).

Results. Each evaluator spent 1.5 to 2 hours. Combining the scores of all the evaluators together, the quantitative evaluation for each document, for each document type, and for all the documents, are presented in Table 7.

[5] The evaluation was presented and performed online. The entire procedure is available at `http://www.cs.utoronto.ca/~bowen/masters/realeval/heuristic-eval.html`.

Principle	P1	P2	Sum	Avg	S1	S2	Sum	Avg	Total	Average
Conciseness	11	11	22	2.7	12	16	28	3.5	50	3.1
Retention	11	09	20	2.5	17	14	31	3.9	51	3.2
Coherence	10	09	19	2.4	14	17	31	3.9	50	3.1
Consistency	16	11	27	3.4	17	18	35	4.4	52	3.3
Informativeness	11	09	20	2.5	15	14	29	3.6	49	3.1
Comprehensibility	04	08	12	1.5	17	15	32	4.0	44	2.8
Fit for Audience	13	12	25	3.1	16	16	32	4.0	57	3.6
Fit for Purpose	15	13	28	3.5	17	17	34	4.3	62	3.9

Table 7. Summary of Usability Results by Principles

Contrary to the accuracy evaluation in Section 4.1, the reverse results were found in our user study between the two types of documents – scientific articles were much better received than patents. The average usability for patents was 2.7 out of 5.0 and for scientific articles was 3.9 out of 5.0. Despite a large accuracy range between the two patents (28.6% and 100%), all the evaluators found them to be hard to understand. This problem may be due to the unfamiliar writing style of patents in general, as one of the evaluators found the title of the first patent hard to parse and asked "Is this supposed to mean something to me?" Moreover, the segmentation inaccuracy dampened user satisfaction. Comments from evaluators were consistent with these results: "Sometimes the info[rmation] was dead on and other times I couldn't decipher it", "The summary really lack in comprehensibility", and "If the correct information [were] pulled out consistently, this system would be very useful". We hypothesize that the usability scores would be higher if given better segmentation accuracy. Still, the current state of matters shows that users found our model helpful and useful, as an evaluator commented: "I really liked the graphical representation of the parts of the document and the fact that a user can click and see the summarized data on the right".

5 Discussion

The need for modeling unstructured data increases as more information becomes abundant in these formats. Increasing efforts have been placed on developing methods for information dissemination, extraction, summarization, and visualization [11, 12, 4, 13]. Related works in these areas have focused on developing or discovering elaborate ontologies for detailed information with applications in databases [14–17]. Issues to consider in developing suitable models and methods include how fine-grained the model should be and what level of processing is needed, both computationally and cognitively. Thus, the difficulty with elaborate models is filling in the details correctly; it is very difficult to develop automated extraction techniques with high accuracy and generalizability, and it is extremely labour-intensive and error-prone to manually annotate an elaborate model. Furthermore, if more "intelligent" language processing techniques were used, then the processing task becomes too intensive and are often intractable [18].

In contrast, we motivated a simple conceptual model (see Figure 1) for two kinds of specialized documents – patents and scientific articles that propose original solutions. The concepts in our model are based on user-centered goals, such as problem-solution, benefits and drawbacks, assumptions, methods, extensions, and claims. These concepts capture the high-level argumentative structure depicted in the documents. We have demonstrated our techniques on two document classes while achieving reasonable accuracy. To generalize our approach over other document types, one could group the patterns in the segmentation component into modules that correspond to particular document types. We believe this technique can also be used in the current system to improve the accuracy of patents and scientific articles. Based on our user feedback, we hypothesize that an increase in accuracy will also raise user acceptance.

In sum, our tests show that the users are getting value from the fairly simple modeling and processing of our approach.

6 Conclusions

We have argued for the need of modeling unstructured data and methods for processing it. In this paper, we have explored the use of a conceptual model that is *simple* enough to allow efficient automatic extraction from documents and *representative* enough so that users got value from using it. Our focus was on the problem-solution relationship – a particular kind of relationship that is crucial to research in science and technology. The model also provides supporting relationships, such as benefits and drawbacks, assumptions, methods, extensions, and claims. All of these concepts are designed based on user-centered goals in order to facilitate the processing of scientific documents by humans. The overall framework of this model is intended to be used in the context of an ontology of research problems, as illustrated in Figure 2, so that users may see how different problems relate to each other. Our ultimate goal in information management is to provide researchers with a map of on-going activities and tool support to do task specific searches.

Because we are modeling unstructured data, the validation of our model relies on user acceptance. We conducted two evaluations – an accuracy evaluation that measures extraction performance on our segmentation task and a usability evaluation that measures the acceptance of our document conceptual model. Our users found our model useful, although the displayed text was not always perfect. In the future, we hope to iterate on more elaborate conceptual models and improve the accuracy of the extraction techniques for more in-depth user studies. We expect domain novices (and potentially experts as well) will find that simpler models are more usable than elaborate ones. We hope to improve the system and make it available for real use.

Acknowledgments

Bowen Hui is grateful for many fruitful discussions with Eric Yu. She would also like to thank Fanis Tsandilas for his support throughout this project, the evaluators' efforts, and the valuable feedback from the anonymous reviewers. This project was in part funded by NSERC.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American (2001)
2. Weibel, S., Godby, J., Miller, E., R. Daniel, J.: OCLC/NCSA metadata workshop report. (1995)
3. MARC: US MARC Standards. (2001)
4. MUC-7: Proceedings of the 7th Message Understanding Conference (MUC-7). (1998)
5. Porter, M.: An algorithm for suffix stripping. Program **14(3)** (1980) 130–137
6. Nielsen, J.: Usability Engineering. Academic Press, Inc. (1993)
7. Sparck-Jones, K., Galliers, J.: Evaluating natural language processing systems: an analysis and review. New York: Springer (1996)
8. Hovy, E., Marcu, D.: Automated Text Summarization: Tutorial Notes. COLING-ACL'98, University of Montreal, Montreal, Quebec, Canada. (1998)
9. Teufel, S., Moens, M.: Discourse-level argumentation in scientific articles: human and automatic annotation. In: ACL Workshop on Towards Standards and Tools for Discourse Tagging. (1999)
10. Jing, H., Barzilay, R., McKeown, K., Elhadad, M.: Summarization Evaluation Methods: Experiments and Analysis. In: AAAI Intelligent Text Summarization Workshop. (1998) 60–68
11. Overmyer, S., Lavoie, B., Rambow, O.: Conceptual Modeling through Linguistic Analysis Using LIDA. In: Proceedings of the 23rd International Conference on Software Engineering (ICSE 2001). Volume Toronto, Canada. (2001)
12. Maybury, M.: Tools for the Knowledge Analyst: An Information Superiority Visionary Demonstration. IEEE COMPSAC 98. 22nd Annual International Computer Software and Applications Conference, Vienna, Austria. (1998)
13. TREC-10: Proceedings of the 10th Text REtrieval Conference (TREC-10). (2001)
14. Embley, D., Campbell, D., Jiang, Y., Liddle, S., Lonsdale, D., Ng, Y., Smith, R.: Conceptual-Model-Based Data Extraction from Multiple-Record Web Pages. Data & Knowledge Engineering **31** (1999) 227–251
15. Loh, S., Wives, L., de Oliveira, J.: Concept-Based Knowledge Discovery in Texts Extracted from the Web. SIGKDD Explorations **2:1** (2000)
16. Aussenac-Gilles, N., Biébow, B., Szulman, S.: Corpus analysis for conceptual modelling. In: CEUR Workshop Proceedings. Volume 51. (2000)
17. Shum, S.B., Motta, E., Dominique, J.: ScholOnto: An Ontology-Based Digital Library Server for Research Documents and Discourse. International Journal on Digital Libraries **3(3)** (2000) 237–248
18. Rounds, W.: The relevance of computational complexity theory to natural language processing. MIT Press (1991)

A Transactional Model for Data Warehouse Maintenance[⋆]

Jun Chen, Songting Chen, and Elke A. Rundensteiner

Department of Computer Science, Worcester Polytechnic Institute
Worcester, MA, USA 01609-2280
{junchen,chenst,rundenst}@cs.wpi.edu

Abstract. A Data Warehouse Management System (DWMS) incrementally maintains materialized views by issuing maintenance queries to the data sources. To address erroneous query results caused by concurrent source updates, state-of-the-art maintenance strategies typically apply compensations to resolve the conflicts. For this, they assume however that the source schema are not updated and remain stable over time. However, if schema changes occur in any of the sources, then an anomaly may arise, namely, the maintenance or the compensation queries may be broken. We now tackle this open problem by modeling the complete maintenance process as a special transaction, called a DWMS_Transaction. The anomaly problem can now be rephrased as the serializability of DWMS_Transactions. This allows us to apply well-established transaction theory to address this new anomaly problem. To achieve such serializability, we propose a multiversion concurrency control technique appropriate for loosely-coupled environments with autonomous sources. TxnWrap is complementary to maintenance algorithms from the literature by removing concurrency issues from their consideration. The experimental results confirm that TxnWrap achieves predictable steady performance even under a varying rate of concurrency.

1 Introduction

1.1 Data Warehouse Environments

Data warehouses (DW) [6, 1] are built by gathering data from data sources and integrating it into one repository customized to users' needs. One important task of a Data Warehouse Management System (DWMS) is to maintain the materialized view upon source changes, since frequent updates are common for most data sources. In addition, beyond data updates, we find that schema changes are also fairly common for modern applications. A schema change could occur for numerous reasons during the software life-cycle, including design errors, schema redesign during the early stages of database deployment, the addition of new

⋆ This work was supported in part by several grants from NSF, namely, the NSF NYI grant #IRI 97–96264, the NSF CISE Instrumentation grant #IRIS 97–29878, and the NSF grant #IIS 9988776.

functionalities and even new developments in the application domain, such as new tax laws or Y2K problems. Even in fairly standard business applications, rapid schema changes have been observed. In [10], significant changes (about 59% of attributes on the average) were reported for seven different applications ranging from project tracking, sales management to government administration.

Furthermore, in distributed environments, the data sources are typically owned by different providers and function independently from one another [19]. This implies that they will update their data and schemas autonomously without any concern for how this may affect the materialized views defined upon them [8], causing so called *DW maintenance anomalies*. These are exactly the problems we address in this work.

1.2 Motivating Example of the DW Maintenance Anomaly Problems

We will now illustrate the anomaly problems via motivating examples. We distinguish between three DW maintenance tasks, namely, View Maintenance (VM), View Synchronization (VS) and View Adaptation (VA). VM [19, 1, 16] maintains the DW view extent under source data updates. In contrast to VM, VS [8, 12] aims at rewriting the DW view definition when the source schema has been changed. Thereafter, View Adaptation (VA) [13, 7] incrementally adapts the view extent to again match the newly changed view definition. If there is no concurrency among source updates, namely, the current DW maintenance completes before the next source update occurs, then VM incorporates each source *data update* (DU) while VS and VA together incorporate the source *schema change* (SC) into the DW. However, the autonomous sources may undergo changes concurrently, causing the maintenance anomaly illustrated below.

Example 1. Assume we have four data sources (DSs) as shown in Figure 1.

DS[1]: Customer(Name, Address, Phone): Customer Info.
DS[2]: Tour(TourID, TourName, Age, Type, NoDays): Tour. Info.
DS[3]: Participant(Participant, TourID, StartDate, Loc): Participant. Info.
DS[4]: FlightRes(Name, Age, FlightNo, Dest): Reservation. Info.

Fig. 1. Description of Data Sources

The view Asia-Customer of the data warehouse is defined by the SQL query in Equation (1). Assume the data update " ΔC = insert ('Ben','MA',123456) into the Customer relation at DS[1]". In order to determine the delta effect on the DW extent, this now requires us to send the incremental view **maintenance query** Q [19] defined in Equation (2) down to the FlightRes relation at DS[4].

```
CREATE VIEW Asia − Customer AS
SELECT      C.Name, F.Age,
            F.FlightNo, F.Dest
FROM        Customer C, FlightRes F
WHERE       C.Name = F.Name
            AND F.Dest = 'Asia'
    (1)
```

```
SELECT 'Ben' as Name, F.Age,
       F.FlightNo, F.Dest
FROM   FlightRes F
WHERE  F.Name = 'Ben'
       AND F.Dest = 'Asia'
                        (2)
```

We distinguish between two kinds of anomaly problems that may arise:

– **Duplication Anomaly**: If during the transfer time of the query Q to the relation FlightRes in the DS[4], FlightRes has committed a new **data update** ΔF = insert ('Ben', 18,'AA3456','Asia'). This new tuple would also be included in the join result of Q and the extra tuple ('Ben', 18, 'AA3456','Asia') would be inserted into the view. However, later when the DW processes ΔF, the same tuple would be inserted into the view again. A duplication anomaly appears, as also observed by [19].
– **Broken Query Anomaly**: If during the transfer time of the query Q to DS[4], the FlightRes relation in DS[4] has a **schema change**, e.g., the attribute FlightRes.Age is dropped, then the query Q faces a schema conflict. In this case, the attribute Age required by Q is no longer available. Hence the query Q cannot be processed and an error message is returned to the DWMS. We then say that the query Q is broken, i.e., a broken query anomaly appears.

As illustrated by the example above, autonomous sources updates may conflict with the DW maintenance process. A concurrent data update may result in an **incorrect query result**, while a concurrent schema change may result in a **broken query**, i.e., an error message is returned.

1.3 Contributions of this Work

Our contributions in this work are:

1. We illustrate that the anomaly problems in Section 1.2 can be reformulated as a transaction problem by modeling the complete maintenance process as a transaction model, *DWMS_Transaction*. We demonstrate that the maintenance anomaly can be mapped to the serializability of *DWMS_Transaction*s.
2. We introduce a multiversion concurrency control algorithm, called TxnWrap, to achieve such serializability within our data warehouse context of autonomous data sources.
3. To the best of our knowledge, *TxnWrap* is the first work to put the existing view maintenance algorithms into the context of a sound theory, i.e., serializability theory, to remove concurrency concerns from them.
4. We have implemented the *TxnWrap* solution and integrated it into a data warehouse prototype system. Experiments comparing maintenance performance with and without *TxnWrap* enabled confirm that *TxnWrap* avoids the maintenance anomaly hence achieves a more stable performance.

The outline of the rest of the paper is as follows. The transaction model called *DWMS_Transaction* is introduced in Section 2. A multiversion concurrency control algorithm is proposed in Section 3. Section 4 presents experimental results. In Section 5 we review the related work while Section 6 concludes the paper.

2 DWMS_Transaction: Transactional Modeling of DW Maintenance

We first analyze the DW maintenance anomaly problems. Typically, the DW maintenance process involves reads from individual sources to calculate the delta changes and ultimately refresh the DW extent. Meanwhile, the autonomous sources may continue to commit various updates. Consequently, concurrency between these two processes may occur. This maintenance anomaly is similar to the anomaly in a traditional DBMS: assume a join is defined upon two source relations. Without any appropriate concurrency control, concurrent data updates or schema changes on these two relations during the computation of the join prcess would cause a concurrency problem. This is similar to the DW maintenance anomaly. A traditional DBMS deals with this problem (1) by applying a transaction model to encapsulate all operations that need to be executed atomically into one transaction, and (2) by using concurrency control strategies to guarantee the ACID properties of each transaction [3]. The ACID properties assure a consistent view of data inside each transaction hence solve the problem.

Similarly, in our DW context, the DW maintenance process can be viewed as a series of read operations over sources that should be executed atomically, while the source updates are independent write operations. There are read and write conflicts since the source writes are autonomous. The above discussion clearly leads us to the idea of applying a transactional model to solve the anomaly problem, as described in detail below.

2.1 Transactions in a Data Warehouse Environment

Each *DW maintenance process* is composed of the following two transactions:

1. A *source update transaction* at some data source DS_i is committed, denoted as "T_{DS_i}=w(DS_i)c(DS_i)" where w(DS_i) means the update on DS_i and i is the index of the data source, c(DS_i) denotes the commit.
2. A *DW maintenance transaction* computes the delta effects caused by source updates in order to refresh the DW. During this period, it may need to send maintenance queries to probe different data sources. At the end, the DWMS refreshes its view extent. We denote this as "$T_{DW} = r(DS_1)r(DS_2)...r(DS_n)$ $w(DW)c(DW)$", where $r(DS_i)$ denotes reading from DS_i, $w(DW)$ denotes writing or refreshing the DW extent and c(DW) denotes the commit.

However, these two types of transactions are not completely independent. As suggested above, the relationship is that each *source update transaction* would trigger a corresponding *DW maintenance transaction*. Furthermore, this *DW maintenance transaction* is required to see a consistent state of all sources, so not to conflict with other *source update transactions*. Otherwise the anomaly as shown in Section 1.2 may occur.

These observations give us the hint that a high-level transaction that integrates both sub-transactions may be needed. The ACID properties of this high-level transaction could resolve the conflicts between its sub-transactions, namely *DW maintenance transactions* and *source update transactions*.

2.2 A Transactional Model: DWMS_Transaction

We model a high-level transaction that integrates both *source update transaction* and its corresponding *DW maintenance transaction*.

Definition 1. *We model the complete DW maintenance process described in Section 2.1 as a **DWMS_Transaction**. Each DWMS_Transaction T starts after a local successfully committed* source update transaction T_{DS_i}*, and commits when the data warehouse database has been successfully refreshed, i.e., the commit of the* DW maintenance transaction T_{DW}*. Hence, we denote it as "*$T = T_{DS_i}T_{DW} = w(DS_i)c(DS_i)r(DS_1)\ r(DS_2)...r(DS_n)w(DW)c(DW)$*".*

Previous research has already hinted at this notion of a "global view of data warehouse transactions" [20]. However, [20] states that a global transactional approach is not feasible since the sources may not tolerate such a wait time before being allowed to commit their local updates. We observe that the *source update transaction* can in fact commit autonomously as a sub-transaction within the complete *DWMS_Transaction* by our proposed techniques. Thus the sources never have to wait.

With the concept of *DWMS_Transaction*, the DW maintenance anomaly can be rephrased as the well-known "read dirty data" transaction problem. With an appropriate concurrency control solution to achieve the ACID properties of such a *DWMS_Transaction*, the conflicts between the *DWMS_Transaction*s, namely, the *DW maintenance transactions* and *source update transactions*, can naturally be solved. Finally, this formal transaction-based solution lays a solid foundation for the future treatment of as of now unsolved problems including recovery or parallel processing of DW maintenance.

Note that a *DWMS_Transaction* is started when a source update has been committed. After that, we must have this source update incorporated into the DW to keep the view consistent. The reason is that the source update transaction is out of the control of the DWMS, thus the rollback of this update cannot be achieved. Hence a DWMS_Transaction would never be aborted. Whenever an error occurs, we instead have to redo the maintenance again until success.

2.3 DWMS_Transaction Scheduling

Two types of DWMS_transaction schedulers are possible, namely, sequential and parallel ones. A sequential scheduler allows no other *DW maintenance transac-*

tions to be processed until the current one has been completed. Most DW maintenance algorithms in the literature [19, 1] employ such a sequential scheduling for DW maintenance even if they do not utilize the transaction terminology. Parallel scheduling is more challenging, since the out-of-order DW maintenance processing introduces additional conflicts. In this paper, we will introduce our solution assuming a sequential scheduler to keep the description simple. We note that our transactional model can be easily extended to support such a parallel scheduler as we have done in [9]. This is in effect one advantage of adopting a transactional model for data warehouse maintenance.

2.4 Conflicts between DWMS_Transactions

We now examine the operations encapsulated within a *DWMS_Transaction*s to identify potential conflicts between them. From Definition 1, we know that there are three basic read and write operations within a *DWMS_Transaction*, namely, $w(DS_i)$, $r(DS_i)$ and $w(DW)$. There are three types of conflicts between these operations. First, some write operation $w_1(DS_i)$ in *DWMS_Transaction* T_1 conflicts with $w_2(DS_i)$ in another *DWMS_Transaction* T_2. Second, some write operation $w_1(DW)$ in *DWMS_Transaction* T_1 conflicts with $w_2(DW)$ in another *DWMS_Transaction* T_2. Obviously such write-write conflicts would naturally be taken care of by the local DBMS of the sources and DW, respectively. The third conflict is the read-write conflict, namely, one read operation $r_1(DS_i)$ of *DWMS_Transaction* T_1 may read some inconsistent query result written by $w_2(DS_i)$ of another *DWMS_Transaction* T_2.

Lemma 1. *The DW maintenance anomaly corresponds to the read-write conflicts of DWMS_Transactions.*

The proof of this lemma can be found in [4]. With Lemma 1, we rephase the DW maintenance anomaly problem in terms of the read-write conflicts of DWMS_Transactions. Next, to solve the DW maintenance anomaly problem, we should resolve such read-write conflicts.

2.5 Serializability of DWMS_Transactions

Lemma 1 leads us to a solution approach to resolve the conflicts, namely, we have to guarantee the serializability of DWMS_Transactions.

Theorem 1. *A history of DWMS_Transactions S is serializable iff it is equivalent to some serial schedule S' of the same DWMS_Transactions.*

Note that a *serial* schedule of DWMS_Transactions S' assumes the source update and maintenance process occur one after another. Obviously in this case, the source update and the maintenance process do not conflict with each other, hence no DW maintenance anomaly would occur. Consequently, by Theorem 1, no anomaly would happen in its equivalent history S either. In the next section, we will propose a concurrency control strategy to achieve such serializability for *DWMS_Transaction*s.

3 Multiversion Concurrency Control Algorithm for DWMS_Transactions

To achieve the serializability for traditional transactions, generally, two basic kinds of concurrency control algorithms [3], namely, **lock-based** or **multiversion-based**, can be used. The traditional **lock-based** concurrency control algorithms [3] work as follows: For every write, an exclusive write lock is obtained and released at the end of transaction; and for every read, a read lock is obtained and released after the read operation completes. This is however too restrictive for the schedule of *DWMS_Transactions*. By Definition 1, we know that DWMS_Transaction will read over all sources hence a read lock over all sources is required. However, the data sources are autonomous in our DW environment and would not be willing to accept locks from external agents as stated in Section 1.1. Hence locking algorithms are not suitable in the DW environment.

In the traditional **multiversion** concurrency control algorithms [3], the reader will read an older version than the version that the writer is currently writing to, thus avoiding the conflicts. In particular, we further differentiate between two categories of multiversion algorithms, i.e., finite version [15, 11] and unrestricted version [2] algorithms. Finite version algorithms pre-set a fixed number of maximal versions (some constant n). There is a so called "version expiration" [15, 11] phase when the number of versions exceeds the allowed maximum. In that case, we have to switch the reader to a newer existing version to allow for the writers to start making a new version. To achieve this, we require that either the writer synchronizes with all readers on the old version or aborts those readers. In comparison, in unrestrictive multiversion algorithms, the writers can always create a new version and the readers can find appropriate versions they need at any time. No blocking between the read/write would ever occur.

Based on above observations, we find that an unrestrictive multiversion algorithm is the appropriate design choice for DW maintenance. First, we require that the version that the reader (maintenance transaction) needs should never expire before the maintenance process has been completed. Otherwise the DW view would be inconsistent. Second, the autonomous writer (source update) is unlikely to synchronize with the reader (DW maintenance process). For this reason, we favor the choice of an unrestricted multiversion algorithm. Also note that, since the *DWMS_Transaction*s will read a monotonically increasing version of data instead of some random old ones, the version cleanup can be efficiently carried out.

We are now ready to introduce our unrestricted multiversion algorithm, called *TxnWrap*. The main idea is to extend the wrapper functionality beyond just communicating between sources and DW [1]. That is, the wrapper is responsible for storing the versions generated due to the source updates and answering any

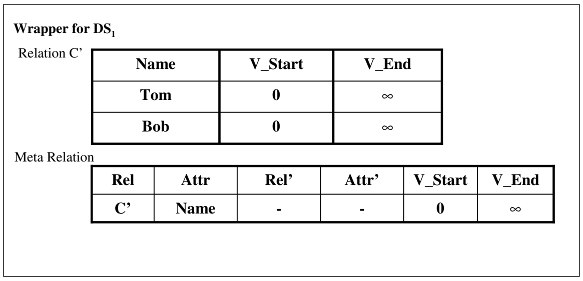
[1] Note that this is just one of the three architectural choices for where to store the versioned data, namely at the sources, the wrappers or the DW. Without loss of generality, here we select one of them to describe the main idea. We omit the

DW maintenance query using the versioned information. Below, we describe *TxnWrap*, in particular, first the version initialialization at the wrappers and then version management upon update notifications and query requests.

3.1 Wrapper Materialization

We propose to extend the wrapper of each source to contain version relations of source data, called *wrapper relations*. A simple full replication of the source data for the wrapper relations would guarantee that the wrapper can answer a maintenance query without any more source access. However we can further improve upon this by utilizing the view definitions of the DW to filter out unnecessary data. For example, we can "push down" the Select or Project operations into the DW view definitions to filter out unnecessary tuples or attributes [4] or utilize some foreign key constraints [14].

We further propose to add two more attributes V_Start and V_End into the wrapper relation to denote the life time of each tuple. V_Start denotes the start version of the tuple (by insertion), and V_End denotes the version when the tuple ends (by deletion). We initialize the tuples in the wrapper relation by setting their V_Start to 0 and V_End to ∞. This way we guarantee that all initial tuples are visible to all DWMS_transactions. We also build a *meta relation* at the wrapper for versioning of the source meta data. It contains six attributes, namely, Rel, Attr, *Rel'*, *Attr'*, V_Start and V_End, respectively. The pair of attributes "Rel" and "Attr" describes the source meta data. The pair of attributes "*Rel'*" and "*Attr'*" indicates the new name of a particular attribute or of a relation. Here at the meta data relation, "V_Start" and "V_End" denote the life span of the attributes, i.e., the version number when it's been created, renamed or dropped. Two source wrappers related to the view in Example 1 are shown in Figures 2 and 3, respectively.


Wrapper for DS_1

Relation C'

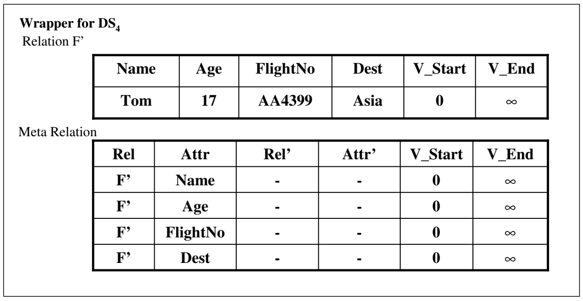
Name	V_Start	V_End
Tom	0	∞
Bob	0	∞

Meta Relation

Rel	Attr	Rel'	Attr'	V_Start	V_End
C'	Name	-	-	0	∞

Fig. 2. Wrapper Initialization of DS_1

space/performance trade-off discussions of these choices due to limited space, please refer to [4] for details.



Wrapper for DS_4

Relation F'

Name	Age	FlightNo	Dest	V_Start	V_End
Tom	17	AA4399	Asia	0	∞

Meta Relation

Rel	Attr	Rel'	Attr'	V_Start	V_End
F'	Name	-	-	0	∞
F'	Age	-	-	0	∞
F'	FlightNo	-	-	0	∞
F'	Dest	-	-	0	∞

Fig. 3. Wrapper Initialization of DS_4

3.2 Version Management

Basically, there are three classes of version operations the wrapper must perform. First, the wrapper has to create versions whenever an update is reported by a source. Second, the wrapper should read appropriate versions to answer each maintenance query. Third, the wrapper needs to clean up any versions that are no longer required by any *DWMS_Transaction*.

Version Creation in the Wrapper. The updates reported by a source can be categorized as data updates, which are *insert*, *delete* and *update*, or schema changes, which are *add*, *drop* and *rename* an attribute or *create*, *drop* and *rename* a table. To create versions, each *DWMS_Transaction* will be assigned a version number for creation of a new version based on its corresponding source update transaction. The corresponding version creations are shown below:

1. Delete a tuple t: update *t[V_End]* to be "i".
2. Insert a tuple t: insert this tuple, set *t[V_Start]* as "i" and set *t[V_End]* ∞.
3. Update a tuple t: treat it as a delete followed by an insert.
4. Drop an attribute: update the attribute's V_End to be "i" in the meta-relation.
5. Add an attribute: insert it into the meta relation with V_Start to be "i" and V_End to be ∞.
6. Rename an attribute: In meta relation, set "Rel'" and "$Attr'$" of the old attribute to its new names, update V_End to "i". Then add the renamed attribute to meta table.
7. Drop a table: Treat it as drop of all its attributes.
8. Add a table: Add all its attributes to the meta-relation.
9. Rename a table: Method similar to rename an attribute, but operate on all attributes.

For example, assume there are three source updates as shown in Figure 4 on the relations depicted in Figure 1. After the version creation step shown above, the wrapper relation and meta relation of DS_1 and DS_4 of Figure 2 and 3 are shown in Figures 5 and 6.

DS Transaction1 (DWMS_Transaction Version id = 1)

1. INSERT INTO Customer('Ben', 'CA', '2137461928');

DS Transaction2 (DWMS_Transaction Version id = 2)

1. INSERT INTO FlightRes('Ben', 18, 'AA3456', 'Asia');

DS Transaction3 (DWMS_Transaction Version id = 3)

1. Rename Customer.Name to Customer.First-Last;

Fig. 4. Three Source Updates

Wrapper for DS_1

Relation C'

Name	V_Start	V_End
Tom	0	∞
Bob	0	∞
Ben	1	∞

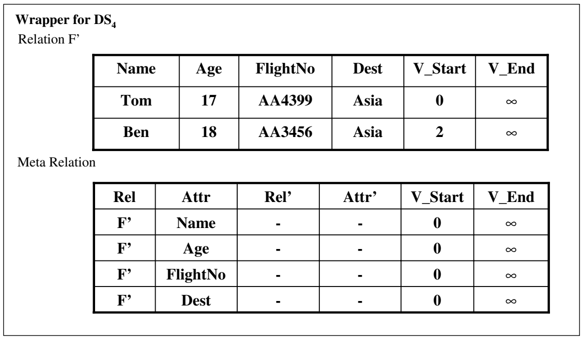
Meta Relation

Rel	Attr	Rel'	Attr'	V_Start	V_End
C'	Name	C'	First-Last	0	3
C'	First-Last	-	-	3	∞

Fig. 5. New Wrapper Content for DS_1

Version Reads in the Wrapper. The read version operation happens when a DWMS_Transaction wants to query a source. Rather than being processed by the source, the queries are actually rewritten by the wrapper and executed by reading appropriate versions of the wrapper schema and relations. Notice that the DWMS_Transaction requires to see a consistent state of all sources. Hence the versions created afterwards should be invisible to this DWMS_Transaction as suggested by Section 1.2. To achieve this, the original maintenance query is rewritten by augmenting it with the version conditions *"AND V_Start <= i AND i < V_End"*. This new query will be evaluated on the *wrapper relation* and not on the actual source relation. Here i is the version number of this DWMS_Transaction. For example, Figure 7 depicts the rewritten versioned queries for the first two source updates from Figure 4.

Version Cleanup in the Wrapper. Finally, we need to clean up versions that are no longer needed by any maintenance transactions. For a sequential scheduler, the cleanup is straightforward. That is, after the completion of a DWMS_Transaction with version number "i", all the data with V_End less than "i" will be deleted since they are no longer visible nor required by any later maintenance transaction. To achieve this, in the meta relation, we execute the update as *"delete * from meta-table where V_End <= i"* and perform cor-



Wrapper for DS$_4$

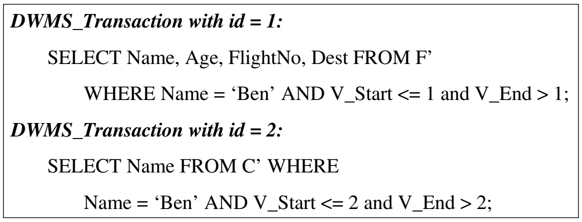
Relation F'

Name	Age	FlightNo	Dest	V_Start	V_End
Tom	17	AA4399	Asia	0	∞
Ben	18	AA3456	Asia	2	∞

Meta Relation

Rel	Attr	Rel'	Attr'	V_Start	V_End
F'	Name	-	-	0	∞
F'	Age	-	-	0	∞
F'	FlightNo	-	-	0	∞
F'	Dest	-	-	0	∞

Fig. 6. New Wrapper Content for DS_4



DWMS_Transaction with id = 1:

```
SELECT Name, Age, FlightNo, Dest FROM F'
    WHERE Name = 'Ben' AND V_Start <= 1 and V_End > 1;
```

DWMS_Transaction with id = 2:

```
SELECT Name FROM C' WHERE
    Name = 'Ben' AND V_Start <= 2 and V_End > 2;
```

Fig. 7. Version Queries at Wrapper

responding operations on these attributes in the wrapper relation. Then we also delete the tuples in the wrapper relations with $V\_End <= i$.

3.3 TxnWrap: A Multiversion Algorithm for DW Maintenance

We are now ready to introduce our concurrency control algorithm for DW maintenance.

1. First, the wrapper initializes its schema and relations as in Section 3.1.
2. Once a source commits a local update transaction and reports its updates to the wrapper, the wrapper gets a unique global *DWMS_Transaction* version id from the DWMS and uses it to create versions. Thereafter it notifies the DWMS about the updates.
3. When the wrapper receives a maintenance query from the DWMS, it rewrites the maintenance query according to the corresponding DWMS_Transaction id and executes it upon the wrapper relation.
4. Once a DWMS_Transaction is committed in the DW, version cleanup starts. The cleanup action can be done by a background daemon process.

Theorem 2. *A history of DWMS_Transactions scheduled by TxnWrap is serializable.*

Due to space limitations, please refer to [4] for the proof of Theorem 2.

4 Experimental Evaluation

4.1 Experimental Testbed and Setup

Since *TxnWrap* by itself is not a stand-alone system, we have plugged it into the data warehouse prototype (DyDa) system developed at WPI [5]. The basic DyDa system uses SWEEP [1], a compensation-based algorithm to handle concurrent data updates. It also has some enhanced view adaption strategies to detect and correct a mixture of concurrent data updates and schema changes [5]. By disabling all concurrency logic and instead plugging *TxnWrap* into the DyDa system, we now have a transactional-based maintenance solution. The java code to handle concurrencies in the DyDa system is more than 5000 lines, while in contrast the *TxnWrap* contains less than 900 lines of code. We have experimentally evaluated *TxnWrap* by comparing these two systems.

In our experimental study, we conducted our experiments on four Pentium III PCs with 256M memory, running Windows NT and Oracle8i. We employ six data sources with one relation each evenly distributed over three PC machines. Each relation has four attributes and 100000 tuples. There is one materialized join view defined upon these six source relations residing on a separate machine. Currently, we assume each *DWMS_Transaction* contains only one update. If we were to merge updates in a batch fashion, the performance should be even better since less maintenance queries would be required.

4.2 Data Update Processing

In this experiment, we evaluate the overhead of *TxnWrap* and compare it against the SWEEP [1] for data update processing under a varying rate of concurrency. SWEEP applies local compensation to remove the *duplication anomaly* as shown in Section 1.2. Figure 8 shows the average maintenance cost of one data update (on the y-axis) under a number of concurrent data updates for both systems.

From Figure 8, we can see that initially the maintenance cost of TxnWrap is larger than that of SWEEP when the number of concurrent data update is 0 (i.e., there is no concurrent data update). This is mainly caused by the additional version conditions to be evaluated. The maintenance cost of SWEEP increases under a number of concurrent data updates, because it has to compensate for concurrent data updates. In comparison, *TxnWrap* exhibits an almost fixed cost for handling of data updates independent of the number of concurrent data updates, and thus achieves more steady performance even under a heavy load of concurrent updates.

4.3 Schema Change Processing

In this experiment, we examine the performance of *TxnWrap* for schema change processing, which includes both view synchronization and view adaptation. We measure the maintenance cost by employing twenty schema changes but varying the time intervals between them. As described in Section 1.2, if there are

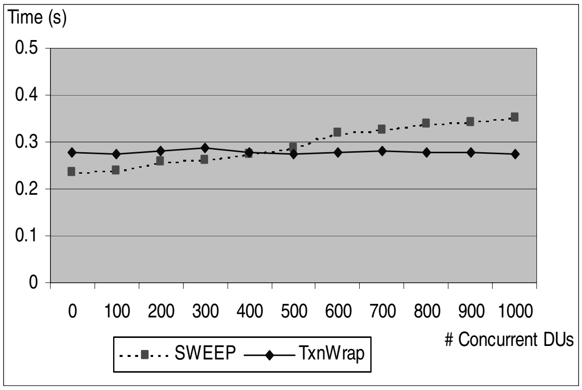


Fig. 8. Current Data Update Processing

some concurrent schema changes, the query would be broken and the current maintenance process may be aborted in the DyDa system. This is a significant cost. Figure 9 presents the maintenance as well as the abort cost (as part of the maintenance cost) of both systems (on the y-axis). From the figure, we see that the DyDa system has a varying abort cost under different system loads. Eventually when the time interval is larger than the maintenance time, the updates won't conflict with each other thus both systems have the same maintenance cost. In comparison, *TxnWrap*, employing a multiversion algorithm, avoids any abort and hence again achieves a steady performance.

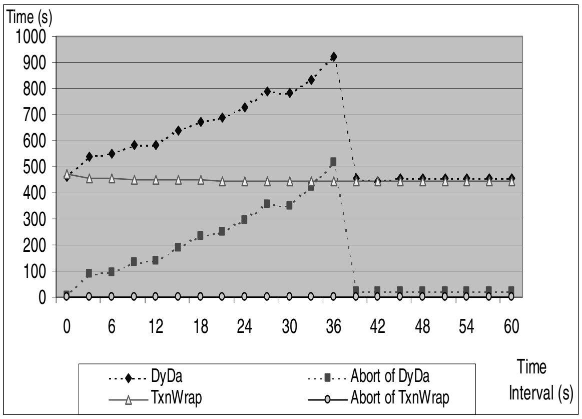


Fig. 9. Concurrent Schema Change Processing

5 Related Work

Maintaining DW materialized views under source updates is an important issue [19, 6]. Besides incremental view maintenance under source data updates [6], there is some work on view rewriting upon source schema changes [8], and on adapting the view extent after the view has been redefined [7, 13]. In a loosely-coupled environment, anomaly problems can arise due to autonomous source updates. [19] introduces a compensation-based algorithm to handle concurrent data updates restricted to one single source. SWEEP [1] applies local compensation over distributed sources. [16] proposes to materialize delta changes of both sources and views with timestamps, thus being able to asynchronously refresh the view. These efforts are all compensation-based and none of them can handle concurrent source schema changes.

Our TxnWrap approach employs a multiversion solution removing concurrency concerns from DW maintenance algorithms thus no longer requiring any compensation strategies. Our concurrently ongoing effort on the DyDa project [17, 18] is the first attempt to address mixed workloads of concurrent data updates and schema changes. In DyDa, we found that all maintenance modules (VM,VA,VS) would have to be explored to detect and correct various types of concurrency problems. This introduces much complexity and a formal basis is missing. In comparison, TxnWrap lays a solid basis to handle the DW maintenance concurrency problems.

[15] proposes to utilize a two-version algorithm to resolve the concurrency between the DW maintenance and DW read-only sessions. Our work instead focuses on the other end of the spectrum, i.e., the concurrency between the source updates and the associated DW maintenance. In the former work, the DW read-only sessions could be scheduled for any consistent DW version. While in our work, the DW maintenance reads will require a particular version of source data that matches the state in which the update had occurred. Hence the desired version should always be available to keep the view consistent.

6 Conclusions

While previous work focuses on specific compensation algorithms for DW maintenance [19, 1, 16], in this paper, we instead propose a transactional model that solves concurrency problems. In particular, we encapsulate both each source update and its DW maintenance process into one *DWMS_Transaction*. We then design a multiversion algorithm *TxnWrap* to achieve the serializability of *DWMS_Transaction*s. This way *TxnWrap* addresses the maintenance anomaly problem not only under concurrent data updates but also under concurrent schema changes. As an added benefit, we found *TxnWrap* very easy to implement. The experimental results reveal that our solution achieves a stable maintenance behavior and even outperforms previous solutions under highly concurrent work loads. To further enhance our data warehousing technology, we plan to study parallel maintenance [9], DW maintenance recovery and batch DW maintenance as our future work.

References

[1] D. Agrawal, A. E. Abbadi, A. Singh, and T. Yurek. Efficient View Maintenance at Data Warehouses. In *Proceedings of SIGMOD*, pages 417–427, 1997. 247, 248, 252, 258, 260

[2] D. Agrawal and S. Sengupta. Modular Synchronization in Multiversion Databases. In *Proceedings of SIGMOD*, pages 408–417, 1989. 253

[3] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database System*. Addison-Wesley Pub., 1987. 250, 253

[4] J. Chen, S. Chen, and E. A. Rundensteiner. TxnWrap: A Transaction Model for Data Warehouse Maintenance. Technical Report WPI-CS-TR-00-26, WPI, 2000. 252, 254, 257

[5] J. Chen, X. Zhang, S. Chen, K. Andreas, and E. A. Rundensteiner. DyDa: Data Warehouse Maintenance under Fully Concurrent Environments. In *Proceedings of SIGMOD Demo Session*, page 619, 2001. 258

[6] A. Gupta and I. Mumick. Maintenance of Materialized Views: Problems, Techniques, and Applications. *IEEE Data Engineering Bulletin*, 18(2):3–19, 1995. 247, 260

[7] A. Gupta, I. Mumick, and K. Ross. Adapting Materialized Views after Redefinition. In *Proceedings of SIGMOD*, pages 211–222, 1995. 248, 260

[8] A. M. Lee, A. Nica, and E. A. Rundensteiner. The EVE Approach: View Synchronization in Dynamic Distributed Environments. *IEEE TKDE*, 2002. to appear. 248, 260

[9] B. Liu, S. Chen, and E. A. Rundensteiner. A Transactional Approach for Parallel Data Warehouse Maintenance. In *Proceedings of DaWaK*, 2002, to appear. 252, 260

[10] S. Marche. Measuring the Stability of Data Models. *European Journal of Information Systems*, 2(1):37–47, 1993. 248

[11] C. Mohan, H.Pirahesh, and R. Lorie. Efficient and Flexible Methods for Transient Versioning of Records to Avoid Locking by Read-only Transactions. In *Proceedings of SIGMOD*, pages 124–133, 1992. 253

[12] A. Nica, A. J. Lee, and E. A. Rundensteiner. The CVS Algorithm for View Synchronization in Evolvable Large-Scale Information Systems. In *EDBT*, pages 359–373, 1998. 248

[13] A. Nica and E. A. Rundensteiner. View Maintenance after View Synchronization. In *International Database Engineering and Applications*, pages 213–215, 1999. 248, 260

[14] D. Quass, A. Gupta, I. S. Mumick, and J. Widom. Making Views Self-Maintainable for Data Warehousing. In *Conference on Parallel and Distributed Information Systems*, pages 158–169, 1996. 254

[15] D. Quass and J. Widom. On-Line Warehouse View Maintenance. In *Proceedings of SIGMOD*, pages 393–400, 1997. 253, 260

[16] K. Salem, K. S. Beyer, R. Cochrane, and B. G. Lindsay. How To Roll a Join: Asynchronous Incremental View Maintenance. In *SIGMOD*, pages 129–140, 2000. 248, 260

[17] X. Zhang and E. A. Rundensteiner. DyDa: Dynamic Data Warehouse Maintenance in a Fully Concurrent Environment. In *Proceedings of DaWaK*, pages 94–103, 2000. 260

[18] X. Zhang and E. A. Rundensteiner. Integrating the Maintenance and Synchronization of Data Warehouses Using a Cooperative Framework. In *Information Systems*, volume 27, pages 219–243, 2002. 260

[19] Y. Zhuge, H. García-Molina, J. Hammer, and J. Widom. View Maintenance in a Warehousing Environment. In *Proceedings of SIGMOD*, pages 316–327, May 1995. 248, 249, 252, 260

[20] Y. Zhuge, H. García-Molina, and J. L. Wiener. The Strobe Algorithms for Multi-Source Warehouse Consistency. In *Parallel and Distributed Information Systems*, pages 146–157, 1996. 251

The Account Data Model

Andrew Pletch[1], Chih-yang Tsai[2], and Charles Matula[3]

[1] Dept of Computer Science
[2] School of Business, State University of New York at New Paltz
New Paltz, NY 12561, USA
{pletcha,tsaic}@newpaltz.edu
[3] IBM, Poughkeepsie, NY 12601, USA
matula@us.ibm.com

Abstract. A new data model is proposed using a traditional accounting approach to model an organization's transactional data and extending this model to include not only financial information but also internal business process, learning and growth, and customer perspectives. It provides an organization with a single data model that supports both transactional and data warehousing needs. The model has a very small footprint and uses replication tables for data mining purposes. The model is generic so that changes in information requirements of the application do not imply the need to change the model itself. Implementations of the model can provide a uniform report generation environment for standard reports as well as OLAP and data mining activities. Implementations can include data used in query optimization strategies for applications such as association rule mining.

Keywords: Conceptual Model, Account Data Model, Transactional Data Warehouse

1 Introduction

Database development focus has recently shifted from processing transactions to knowledge discovery [7]. Competitive market forces require organizations to understand more about their external environments such as customers, competitors, and suppliers along with their internal resources such as inventory and process controls. The Entity-Relationship Model [5] has traditionally been used to design transactional databases because of the ease of mapping from the real world to the model. However, the resulting model is often too complicated to easily generate queries and efficiently obtain reports. As a result, data warehouses using a dimensional model [10] were developed and separated from the transactional model to support the need for knowledge discovery efforts such as OnLine Analytical Processing (OLAP) and data mining. In this study, we propose a new data model, the Account Data Model (ADM), which supports both on-line transaction processing and knowledge discovery endeavors. In this model, data is modelled as a set of hierarchically organized accounts similar to the double-entry system used in a standard Chart of Accounts hierarchy (see [12]) while data-changing events are modelled as transactions consisting of a collection of debit and credit components against these accounts and that offset one another. Moreover, any update to a real-world transaction is modelled in the ADM by creating new transactions to offset previous transactions. Thus, the whole history of a real-world transaction is preserved for process control purposes as a sequence of related ADM transactions. The design of ADM implies that each time the ADM is put to use a complex domain analysis process must be undertaken in order to produce the appropriate Chart of Accounts. There is a certain amount of domain-dependence in any given account hierarchy design but industry groups continue to develop accounting hierarchies specialized to various industries (A Google search on ”Chart of Accounts” will reveal a large number of organizations and governmental agencies that publish domain-specific charts of accounts). Although the design of the ADM for the most part is domain dependent, some general guidelines and domain-independent features are discussed in Sections 2 and 3.

An ER-based database holds meta-data structurally through a relationship chart that links key attributes between tables. Hence an application trying to identify the sales region generating the most orders needs to know the links from a customer ID in the order table to a customer table and from the state name in the customer table to a sales region table. On the other hand, data in a data warehouse is modelled in a format more conducive to producing reports. Like the ER Model, the typical star schema configuration for a data warehouse also holds meta-data structurally - a single fact table is linked to many dimension tables, one for each dimension, instead of deeply linked and highly normalized tables as in an ER model.

The Account Data Model (ADM) evolved from a production database that supported the sales and fulfilment requirements as well as the call-center operations for a medium-sized catalog sales company. The implementation footprint of this model consists of only four tables - two for accounts and two for transactions - in a relational database. All accounts and related transactions, regardless of the business activities they document, are modelled in the same set of four tables. In addition, the ADM can be extended to include the dimensional data typical of a data warehouse. Thus, the major strength of the ADM is that it can support both transactional and warehousing functionality from a single platform.

Another strength of the ADM model is its flexibility. Business processes may change from time to time or new reports may be needed to cover new dimensions of the business environment. In a traditional ER-model or star-schema data warehouse model, this involves redesign of the underlying tables, for example, adding new columns to tables which usually requires revisions in other tables and queries. In the ADM, the job of adding new attributes reduces to adding new accounts and transactions (ie, adding new rows to existing tables) without any structural changes to the original design.

The next section provides a more detailed description of the ADM model followed by Section 3 which addresses the data warehousing side of the ADM. Section 4 discusses some implementational issues as well as the potential usage of the model for OLAP and data mining projects followed by a brief conclusion of this study.

2 The Account Data Model

In this section, we introduce the ADM with a focus on its structure and transaction support functionality. Since ADM uses the same structure to support data warehousing functionality, we address only the design issues for data warehousing in the next section.

The ADM uses a hierarchy of accounts to model the information used in an organization's transactional processing. Most of this hierarchy follows standard accounting practices, i.e. a set of accounts organized in sub-trees rooted in five major accounting categories, namely ASSET, LIABILITY, EQUITY, REVENUE, and EXPENSE.

The leaf nodes in Figure 1 represented by boxes below the solid line are called sub-accounts. They are the only accounts which actually participate in transactions. Boxes above the solid line (referred to as controlling accounts) form a kind of type hierarchy and are used as summary accounts, summarizing the activity of the accounts below them in the hierarchy. Storing data in the context of a type hierarchy to which it belongs makes automated reasoning about this data much easier to achieve.

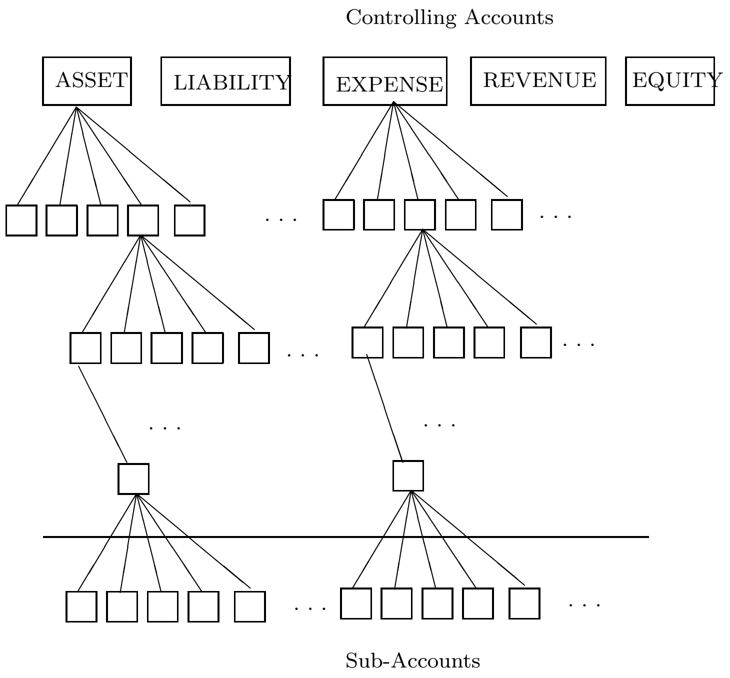


Fig. 1. An Account Hierarchy

The job of designing the hierarchical account structure falls naturally on people from the business side of the organization because, unlike the ER model, the mapping of

$$\boxed{\text{real-world activity}} \rightarrow \boxed{\text{ADM transaction}}$$

in the ADM is a non-trivial mapping whose design requires knowledge not only in predicting the types of activities the structure is intended to support but also on a deep understanding of the interdependency among business functions. The relatively high cost of designing the account hierarchy and the domain-dependent nature of this hierarchy are ameliorated by three facts:

– many industry groups have already developed standard charts of accounts for their specific industries,
– data quality is much improved since data is stored following the principals of double-entry bookkeeping and
– all reports based on any of an organization's transactional activity are single-sourced and so many, otherwise *ad-hoc* reports become available from a standard report generation interface.

Following accounting practices, a *transaction* is a collection of components that document either debit or credit operations against accounts. Transactions should be *balanced* in that every crediting component is offset by a corresponding debiting component. Ensuring that transactions are balanced before they are accepted pushes some of the data cleansing activity typical of the migration of data from an ER-based database to a data warehouse into the data entry phase.

Part of the justification of our claim that an ADM implementation can be used as a data warehouse is our confidence in the quality of the data stored in the model implementation due to the balanced-transaction nature of the model. Transactions and transaction components along with accounts and controlling accounts eventually come to reside in a standard four-table structure as shown dashed-box of Figure 2. We refer to this table structure as the *Quad*.

Each transaction involves a row in the transaction table and several rows in the transaction component table. A typical transaction can be better described using an example.

Example 1. A customer, XYZ, orders four units of widget at the unit price of \$15 and unit cost of \$10 on July 5, 2001.

The event in Example 1 generates the following accounting entries.

$$\begin{cases} \text{(a) Account Receivable–XYZ Co.} & \$60 & \\ \qquad \text{(b) Sales–XYZ Co.} & & \$60 \\ \text{(c) Cost of good Sold–widget} & \$40 & \\ \qquad \text{(d) Inventory–widget} & & \$40 \end{cases} \tag{1}$$

In addition, two inventory entries are also recorded for this event, one debited four units of widgets into customer XYZ's account and the other credited four

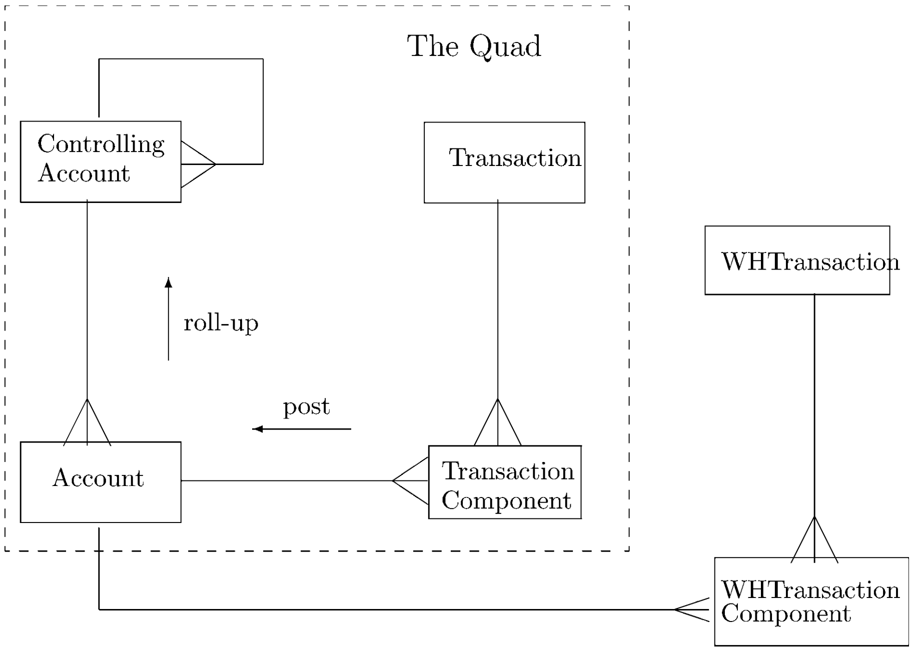


Fig. 2. The Quad with Data Warehouse Extension

units from the inventory account of widgets as shown in the following entries.

$$\begin{cases} \text{(e) Units sold –XYZ Co./widget} & 4 & \\ \qquad \text{(f) Inventory–widget} & & 4 \end{cases} \tag{2}$$

One row is added to the Transaction table in Figure 2 to represent this transaction. The six accounting entries correspond to six transaction components; half of them debiting accounts and half crediting. Six rows in the Transaction Component table of Figure 2 are also added.

At some point after a transaction has been entered it undergoes a process called *posting* which updates the sub-accounts in the account table that affected by the new transaction. These modifications are further propagated to the controlling accounts (rows in the Controlling Account table) higher up the accounting hierarchy from the affected sub-accounts by a process called *roll-up*. Both these activities are indicated diagramatically in Figure 2. When the Example 1 components are posted, we see the changes in Figure 3 to the balances of the six affected sub-accounts. Note that the negative values under inventory and inventory SKU (Stock Keeping Units) in the figure come from debiting a credit account (or crediting a debit account). Both posting and roll-up become part of the model through automated means - stored procedures, triggers and the like.

The ADM shows its true strength in its ability to add new information about any transactional application without the need for table redesign. This

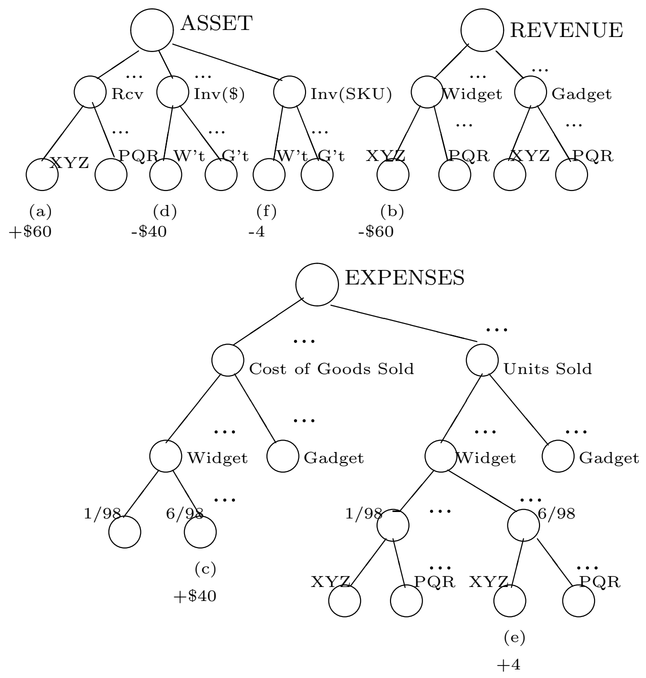


Fig. 3. The Components in Accounting Trees

is achieved by adding new accounts and sub-accounts for the new attributes and entities and then, for each type of affected transaction, additional transaction components are generated to support debiting or crediting the new accounts. In both cases we add more rows to existing tables; never more columns nor more tables. Indeed in the current implementation, transactions saved under an old design schema, which is then redesigned to affect different sub-accounts, will automatically reflect the new design simply by being brought into the user-interface and saved again without change. This is particularly useful if you make a mistake in your initial account design and need to restructure your account hierarchy.

There is no limitation on the kinds of transactions that can be stored in a single ADM implementation. What makes the ADM a viable data warehousing model is the fact that all transactional activity is stored in the same table structures. Adding new types of transactions or even modifying the design of existing transactions becomes a data-entry activity rather than a design endeavor.

The ADM does not support modifying or deleting an existing Quad transaction directly. When a business activity, a sales order for example, is later modified the ADM implements this modification by adding a totally new transaction. The effect of combining the components in the new transaction and the original transaction is to produce new account balances. Thus the original transaction and a corresponding modifying transaction provide a history of the ordering process. This historical record is not an addendum to the model, rather just another use of transactions. From a process point of view, every possible change to an account must be kept for the purpose of process control. For example, we might find from this history that a customer consistently places orders based on the optimistic side of its demand forecast and requests cancelling of its orders or reducing order quantity later. Typically the ADM is part of an ER-based database containing the remaining non-transactional information about the organization. For example, while most aspects of a shipping transaction fit into the ADM, properties such as Shipping Instructions and other *ad hoc* text information must be modelled elsewhere in the database.

3 Modelling Dimensions for Data Warehousing

As we have emphasized mainly the financial side of an organization's operations until now, we now seek to add to the controlling account hierarchy accounts which represent the typical dimensions and attributes associated with a data warehouse. Such accounts would not appear in a standard chart of accounts. For example, while having a customer address maybe be enough to support shipping and billing it is the additional dimensions associated with an address such as community type or size that might characterize the life style of a customer and so be useful in building customer profiles. One of the weaknesses in a traditional accounting system cited in [6] is its inability to integrate financial and non-financial data across the organization. Those non-financial accounts (dimensions) together with the original accounts in the *Quad* provide the essential information which best characterizes the forward and backward looking indicators of Kaplan's Balanced Scorecard [9] and thus must be present in a data warehouse if it is to be an effective tool for decision support. In the ADM, all factors and dimensions necessary to monitor the Balanced Scorecard [9] of an organization's short and long term objectives, financial and non-financial performance, and internal/external perspectives can be built into the model as dimension accounts. These additional dimensions, when found in a data warehouse, are usually associated with performance indicators like Sales in Dollars, Sales in SKU units, Transaction Cost, Degree of Customer Satisfaction, etc. Just as a data warehouse designer must decide what performance indicators are to be maintained in the warehouse, the ADM designer must also make the same decision. As a result, the performance of each dimension, the causal relationship between factors, and the organization's ability to support its strategies can be measured constantly with little additional data processing effort.

In the ADM we create separate dimension accounts for each performance indicator being modelled. Dimension accounts are modelled in a hierarchy separate from the the controlling accounts based on the five accounting categories. The root node on this hierarchy is called DIMENSIONS. Immediately under this node there is a node for each performance indicator being modelled. Under the performance indicator nodes, we place the dimension subtrees themselves. The simplest dimension subtree consists of a single root node labelled by the dimension name and one child node of this root node for each instance of the

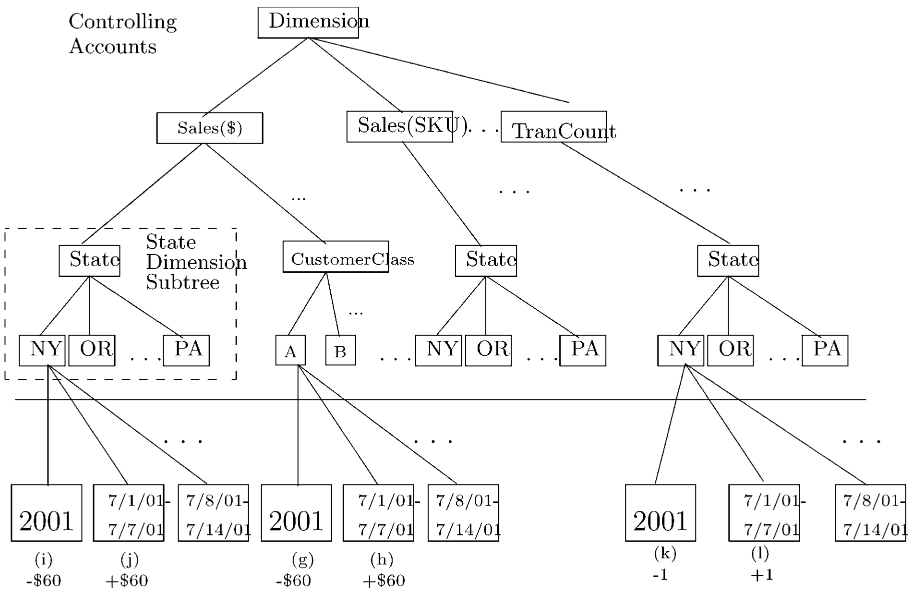


Fig. 4. Dimension Subaccounts

dimension (see the dashed-box in Figure 4). This subtree is repeated for each performance indicator subtree of the dimension hierarchy (see Figure 4).

Approaches such as the star-schema used in data warehousing are attractive precisely because they provide a uniform data structure for all kinds of different reports. The ADM also uses a fixed implementation data structure, which is completely independent of the application. Hence a simple, uniform query interface, which relies mostly on the fixed data structure of the implementation, can be used for a broad range of seemingly unrelated reports. Support for OLAP visualization tools is possible using this implementation by producing data cubes suitably small enough for export to client software involving any of the dimensions and performance indicators being modelled.

When the performance indictors are structured according to the four perspectives of the balanced scorecard [8], namely financial, customer, internal business process, and learning and growth, the double-entry system leads an ADM account designer naturally to consider more than one perspective for each transaction. With further ontological analysis, a *domain independent* structure at higher levels of the controlling account hierarchy can be developed to identify the *events*, *objects*, *properties*, *spaces*, and *times* that might contribute to the change of the *scale* (see [3], [4] for details) in the performance measures.

3.1 Dimension Accounts

The balances for the dimension accounts under a specific performance indicator account correspond to the appropriate performance indicator value associated with that dimension. For example, in Figure 4 the balance of the New York State account under the Sales (in dollars) performance indicator is the total sales (in dollars) for New York State during the periods covered by its sub-accounts.

3.2 Modelling Dimension Hierarchies

Some dimensions are themselves hierarchies. The State, Region, Country dimension is an example. We find that we gain more flexibility by modelling these as individual dimensions rather than building the pre-existing hierarchy into our model. Hence, the performance indicator subtrees usually do not involve many levels.

3.3 Modelling Dimension Time

Time is a special dimension. Not only does it have a built-in hierarchy, it is ubiquitous. In almost every report extracted from a data warehouse Time is a qualified parameter. We have chosen therefore to build the Time dimension into each of the other dimensions rather than show Time as a separate dimension. We do this by breaking the information recording cycle into time intervals - quarters, months, weeks or days, for example, and creating a separate sub-account for each dimension instance for each time interval over the life of that instance. In Figure 4 we can see that each State controlling account is linked to a sub-account for each week of an accounting year.

3.4 Modelling Dimension Transaction Components

Once the dimension hierarchy is in place we are able to add dimension components to all transactions, both existing and future. Adding these components at the time the transaction is initially entered may slow down the OLTP (OnLine Transaction Processing) process. Since, again from an accounting point of view, we use the accounting practice of “posting” (see Figure 2) to officially record the effect of a transaction on a sub-account, this is an appropriate point at which to add performance measure-related components to the transaction being posted and post these component amounts as well.

Since adding the dimension components to the same Transaction Component table used for OLTP might later result in a performance degradation, we use a replicated copy of the Transaction and Transaction Component table and add performance measure components only to the latter table (see Figure 2). There is very little difference between the table structures of the transact component and warehouse transaction (WHTransactComponent) component tables modelled in Figure 2. The two tables contain different information about the same transactions and are used for different purposes however.

As an example, when the transaction in Example 1 from customer XYZ, a New York based company, is posted to the account table, we create four additional transaction components to be posted to the warehouse transaction component table for customer relationship management (CRM) and sales analysis purposes each from a different performance measure. In (3), the CRM account, sales in dollars to Class-A customer in the week of 7/1/01-7/7/01 (h) is balanced by a cumulative account of sales in dollars to class-A customer for the year of 2001 (g) (see Figure 4 for posted amounts). This design not only allows us to find summary sales efficiently but also to ease the design effort in balancing each transaction. The sales analysis transaction components in (3) are designed in a similar fashion.

$$\left\{\begin{array}{lll} \text{(g) Class-A Customer (\$): 2001} & 60 & \\ \quad\text{(h) Class-A Customer (\$): 7/1/01-7/7/01} & & 60 \\ \text{(i) Sales (\$) from NY State: 2001} & 60 & \\ \quad\text{(j) Sales (\$) from NY State: 7/1/01-7/7/01} & & 60 \\ \text{(k) TranCount (\#) from NY State: 2001} & 1 & \\ \quad\text{(l) TranCount (\#) from NY State: 7/1/01-7/7/01} & & 1 \end{array}\right. \tag{3}$$

4 Implementation of the ADM

The transactional part ADM model has been implemented for a medium-size company using the Quad structure described earlier. Using the ADM as a data warehouse has been a development effort to this point. Since most dimensions are associated with lookup or reference tables in the database that are used to support pull-down menus in the application, the easiest way to populate the dimension subtrees has been through an automated process using the lookup tables as input. We use script-generated ad hoc SQL and triggers to populate the Dimension controlling account/account hierarchy whenever new dimensions are introduced or new values added to existing dimensions. As new performance indicators and/or dimensions are envisaged after the initial model is complete then it is simply a matter of adding the necessary accounts in order for the ADM to support warehousing this new information.

The sizes of the Quad tables in this application are described in the Table 1 below.

Support for OLAP applications comes in the form of the production of reduced row-size fact tables similar to Table 2 which are suitable for export to a client application. These tables are produced by *ad hoc* SQL which is generated by analyzing a text configuration file produced as output from a query

Table 1. Size of the Production Database

Table Name	Table Size(number of rows)
Controlling Account	10,000
Account	4,000,000
WHTransaction	2,000,000
WHTransaction Component	40,000,000

Table 2. OLAP Dimensions

D_1	D_2	D_3	$\cdots$	D_n	PI_1	PI_2	$\cdots$	PI_m

interface. The number of rows in the table will vary from request to request but should be small enough to be exported to a client PC. The table consists of a column for each dimension (D_i) and a column for each performance indicator (PI_i) selected for analysis.

The strategy behind producing Table 2 is to accept as input a list of dimensions (D_i) of interest along with a list of dimension instances (d_{ij}). The query must also specify the date range and the performance indicators of interest.

We show the input information abstractly as sets but these sets are easy to produce from an appropriate user interface. Currently we are specifying these sets in a text format file.

$$\begin{cases} D_1 = \{d_{11},\, d_{12},\, d_{13},\, \ldots\,,\, d_{1p_1}\} \\ D_2 = \{d_{21},\, d_{22},\, d_{23},\, \ldots\,,\, d_{2p_2}\} \\ D_3 = \{d_{21},\, d_{22},\, d_{23},\, \ldots\,,\, d_{2p_2}\} \\ \qquad \ldots \\ D_n = \{d_{n1},\, d_{n2},\, d_{n3},\, \ldots\,,\, d_{np_n}\} \\ \\ \{\text{Start\_Date, Stop\_Date}\} \\ \\ \text{PI} = \{\ PI_1, PI_2,\, \ldots\,,\, PI_m\ \} \end{cases} \tag{4}$$

From the input data in (4) a list of relevant sub-accounts is produced using ad hoc SQL; the sub-accounts (a_{ij}) associated with the specified dimension instances over the specified date range.

$$\begin{cases} A_1 = \{a_{11},\, a_{12},\, a_{13},\, \ldots\,,\, a_{1r_1}\} \\ A_2 = \{a_{21},\, a_{22},\, a_{23},\, \ldots\,,\, a_{2r_2}\} \\ A_3 = \{a_{21},\, a_{22},\, a_{23},\, \ldots\,,\, a_{2r_2}\} \\ \qquad \ldots \\ A_n = \{a_{n1},\, a_{n2},\, a_{n3},\, \ldots\,,\, a_{nr_n}\} \end{cases} \tag{5}$$

The SQL that produces the sub-account sets in (5) works mainly off the Controlling Account and Account tables mentioned in Table 1. Since the sets in (4) are relatively small, having been chosen because they show promise in yielding useful information from further analysis, the effort required to produce the sets in (5) is not great.

Finally, from each list of accounts in (5) we produce, using ad hoc SQL again, a list of transactions (t_{ij}) that contain components affecting the balances of these accounts. The queries that produce the transaction sets in (6) are made against the Table WHTransaction Component in Table 1. By placing an index on this table using the two columns *(Account_ID, Transact_ID)* the query is actually covered by this index ([11]). Hence the queries are executed without actual reference to the WHTransaction Component table but only to the *(Account_ID, Transact_ID)* index.

$$\begin{cases} T_1 = \{t_{11}, t_{12}, t_{13}, \ldots, t_{1s_1}\} \\ T_2 = \{t_{21}, t_{22}, t_{23}, \ldots, t_{2s_2}\} \\ T_3 = \{t_{21}, t_{22}, t_{23}, \ldots, t_{2s_2}\} \\ \quad\ldots \\ T_n = \{t_{n1}, t_{n2}, t_{n3}, \ldots, t_{ns_n}\} \end{cases} \tag{6}$$

The final set of transactions is

$$T = \bigcap_{i=1}^{n} T_i \tag{7}$$

Using an aggregate SQL query against the transactions in the final set, T, of transactions will produce Table 2. For an example involving twelve dimensions and two performance indicators, executing the SQL which includes building the sets described in (4) through (7) from the tables described in Table 1 and producing a final table described in Table 2 that contains 12000 rows takes on the order of two minutes running on a Sparc 20 with dual 70MHz CPUs.

4.1 ADM and *A-priori* Optimization:

We briefly describe the potential benefits of using the ADM to construct a data cube for OLAP and to run the *a-priori* test for mining association rules (see [1] and [2]). *A-priori* optimization relies on having prior knowledge about the extent to which items participate individually in the result set of a query before trying to consider their participation as a group in the result set of the same query. The hope is to reduce the size of the group to be considered by excluding individuals that do not meet the query criteria individually. The Quad can be implemented with one of the performance indicators being Transaction Count -the number of transactions a particular account participates in. The balance of each state dimension controlling account under the *TranCount* node in the Figure 4 is the number of transactions in the Quad which have been posted and rolled up to this account. Having this information available for every dimension at all times simply by querying the controlling account hierarchy provides precisely the kind of support information that *a-priori* relies upon. Indeed, we use this information to optimize the SQL used in the production of the set T of Equation (7). This is because one of the sets, $T_{smallest}$, in Equation (6) contains the fewest transactions based upon the *TranCount* balances for the different dimensions, D_i, in the query. Constructing the set T of Equation (7) is done by always looking for a subset of $T_{smallest}$.

5 Conclusion

The Account Data Model developed in this study integrates transactional and data warehouse information and activity in a single data model. The strengths of the ADM include its ability to support various types of data processing and decision making from operational to strategic in a single application-independent platform and to adjust to new business needs without the need to redesign the database. Activities such as the Virtuous Cycle of Data Mining [13] cease to be a redesign activity of table redesign and rebuild and become a data-entry activity.

The success of the model as a decision support tool relies heavily on the design of the account hierarchy. Thus, although the design of the hierarchy can be time-consuming compared to that of an ER-model, the reward of a good design can be a better decision support system which can encompass all aspects of Kaplan's balanced scorecard [8]. Further studies in ontology are needed to provide more structured, domain-dependent guidelines to facilitate the model design effort.

References

[1] Agrawal, R., Imielinski, T., and Swami, A.: Mining Association Rules between Sets of Items in Large Databases. In: Proc. ACM SIGMOD Washington DC Conference, May 1993, 207-216. 274

[2] Agrawal, R. and Srikant, R.: Fast Algorithms for Mining Association Rules. In: Proc. 20th Int'l Conf. on Very Large Data Bases, September 1994, 487-499. 274

[3] Bunge, M.: Semantics I: Sense and Reference. D. Reidel Publishing Company, Boston, 1974. 270

[4] Bunge, M.: Ontology I: The Furniture of the World. D. Reidel Publishing Company, Boston, 1977. 270

[5] Chen, P. P.: The Entity-Relationship Model: toward a Unified view of data. ACM Trans. on Database Systems, Vol. 1, 1, 1976, 9-36. 263

[6] Denna, E. L., Cherrington, J. O., Andros, D. P., Hollander, A. S.: Event-Driven Business Solution. Business One Irwin, Homewood, IL, 1993. 269

[7] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: Knowledge Discovery and Data Mining: Towards a Unifying Framework. In: Proceedings of Second International Conference on Knowledge Discovery and Data Mining, AAAI Press, 1996. 263

[8] Kaplan, Robert S., Norton, David P.: The Balanced Scorecard. Harvard University Press, Boston, 1996. 270, 275

[9] Kaplan, Robert S., Norton, David P.: The Strategy-Focused Organization: How balanced Scorecard Companies Thrive in the New Business Environment. Harvard University Press, Boston, 2000. 269

[10] Kimball, Ralph: A Dimensional Modeling Manifesto. DBMS, vol. 10, no. 9, 1997, 58-72. 263

[11] Page, Christopher: Configuring Database Systems. In: Proceedings of the Twelfth Systems Administration Conference. Boston, 1998. 274

[12] Horngren, Charles T., Harrison Jr., Walter T., Smith Bamber, Linda, Robinson, Michael A.: Accounting. 5/e. Prentice Hall, New York, 2002. 263

[13] Berry, M. J. A., Linoff, G. S.: Mastering Data Mining. Wiley, New York, 2000. 275

A Semantic Model for Hypertext Data Caching

Kai Cheng[1,2] and Yahiko Kambayashi[1]

[1] Department of Social Informatics, Graduate School of Informatics
Kyoto University, Sakyo Kyoto 606-8501, Japan
{chengk,yahiko}@db.soc.i.kyoto-u.ac.jp
[2] School of Computer
Wuhan University, Wuhan 430072, China

Abstract. In this paper, we propose a semantic model to capture the semantic locality in hypertext access for client–side caching. To characterize hypertext data from the perspective of clients, we define a *semantic region* as a cluster of semantically related *logical documents*. A logical document is defined as a sequence of subsequently visited inter–connected *documents* which in turn are composed of a container file and (optionally) a set of component files. This model makes it easy to deal with temporal locality, spatial locality and semantic locality in hypertext access. To verify the proposed model, we use an experimental hypertext system, called HyperDB. We generate a set of workloads and assess the performance of a set of caching algorithms using the synthetic workloads and the experimental hypertext system.

1 Introduction

Hypertext and hypermedia data has recently gained importance due to the success of the world-wide web and hypertext transfer protocol (HTTP). A salient feature of a hypertext system is that users search for desired data primarily by navigation or following links from one document to another, while query facilities, if any, are just used for pruning the navigation space [7]. The interactive nature of hypertext navigation poses challenges in system performance, especially in a client/server architecture where downloading a document from remote servers is much more time-consuming than reading from a local disk.

Caching has been widely used to alleviate performance bottlenecks in computer architecture, network and database systems[18]. Particularly, caching that utilizes computational and storage resources of the client machines has been a key solution to achieve high performance and scalability in client/server database systems [9]. Semantic caching is an advanced form of client caching suited for utilizing client resources, which maintains a semantic description for cached data, called *semantic region*, to capture the *semantic locality* of data usage [8]. Semantic caching gives higher priority to data relevant to the frequently used data as it believes those semantically related data are more likely to be accessed again in the near future.

In a query–retrieval based database system, semantic description of cached data can be straightly obtained from the formulas of query constraint. Taking as an example from [8], suppose we want to find all employees whose salary exceeds 50,00 and whose age are at most 30 years old. We can issue a query using this constraint formula $Q_1 = (Salary > 50,000 \wedge Age \leq 30)$. For caching of data with explicit semantic description, one can easily make use of query results to serve the subsequent query requests according to their semantic containment: only the remainder of a query should be issued to get complimentary data.

In a navigation–based hypertext system, however, there is no explicit semantic description altogether, instead the user expresses his information needs by interacting with the system, i.e. repeatedly choosing new data while looking at current data items [7]. As a result, caching of hypertext data cannot utilize the explicit semantic description to capture semantic locality. Another problem for caching hypertext data is that hypertext documents are inter–connected by hyperlinks and each document is also composed by a set of small components of other media. To take advantage of the features, a suitable model is essential.

Modeling hypertext data has been well researched in hypertext community [2, 5, 10, 11, 13, 22]. Dexter model [11] identifies the relevant abstractions found in a wide range of existing hypertext systems, providing a common vocabulary and its meaning in order to talk about hypertext systems. The Fundamental Open Hypertext Model (FOHM) [13] expands the Open Hypermedia Protocol (OHP) data model to describe a broader set of hypermedia "domains", such as navigational domain, spatial domain and taxonomic domain, to meet the requirements of interoperability between hypertext systems. The OHP protocol was always more concerned with navigational hypertext, whereas FOHM is capable of expressing all three domains.

Hypertext abstract machine (HAM) [6] is an architectural description of a general-purpose, transaction-based, multi-user server for a hypertext storage system. Furuta and Stotts' Trellis model [20] is a formal specification of hypertext based on petri nets. Hypertext has also been formalized as graphs [21] or automata as in [14, 17], where the authors studied the dynamic properties of hypertext in terms of "reader's experience", formalizing what readers see when they interact with a hypertext system. *Web machine* [1, 12] or *web automata* [19] are computation models for querying the web, which paid much attention to the navigational nature of the web.

The afore–mentioned models, however, either aimed at facilitating authoring activity and system design, or attempted to investigate computation mechanism for designing suitable query languages. To capture temporal locality, spatial locality as well as semantic locality, we need a model that can handle structural organization, browsing semantics as well as content relevance from the perspective of the user. As a client cache can only see a restricted fraction of the whole hypertext system and as there is tradeoff between model complexity and associated overhead, current work is not suitable for the caching purpose.

In this paper, we propose a new semantic model to capture the semantic locality of hypertext data caching. We define a *semantic region* in a cache as a cluster of semantically related *logical documents*. A logical document is defined as a sequence of subsequently visited interlinked *documents*, which in turn are composed of a container file and (optionally) a set of component files. We verify the proposed model by implementing an experimental hypertext database, called HyperDB. We then generate a set of workloads and assess the performance of a set of caching algorithms using the synthesized workloads and the simulated hypertext system.

The rest of paper is organized as follows. Section 2 proposes the semantic model. Section 3 describes a virtual hypertext database based on the proposed model, called HyperDB as well as workloads with various kinds of locality of reference for HyperDB. Section 4 gives a set of caching replacement algorithms that take into account various forms of reference locality, describing simulation results obtained under the synthetic workloads and the simulated hypertext system. Section 5 concludes the paper and describes some future directions.

2 Modeling Cached Hypertext Data

Hypertext data is a collection of documents (or "nodes") containing cross-references or "links" which, with the aid of an interactive *browser* program, allow the reader to move easily from one document to another. The extension of hypertext to include other media – sound, graphics, and video – has been termed "hypermedia", but is usually just called "hypertext", especially since the advent of the World-Wide Web and HTML.

In this section, we develop a semantic model by taking into account both the structural as well as the semantic features of hypertext data. Particularly, we define the concept of semantic region for hypertext data caches. This model characterizes a collection of data in a hypertext cache from three abstraction levels: physical documents, logical documents, and semantic regions based on physical structure, logical structure and semantic structure respectively.

2.1 Physical Structure of Hypertext Data

To capture the spatial locality of hypertext access, a cache should first understand physical structure of hypertext documents, the basic elements in a hypertext system. We describe this feature mainly following the Dexter model [11].

First, *document* is a basic element of a hypertext system. We define a (hypertext) document as a composition of a container (file) and (optionally) a set of media component files that represent media other than text such as image, audio and video (Fig. **??**). A container (file) consists of (1) textual content, a sequence of terms, sentences, paragraphs; (2) anchors and (3) hold places for media components.

An *anchor* is a point in a document representing a start point for a link. An anchor also specifies a valid range or *anchor text*, indicating what part of a document belongs to the anchor. Anchor texts often describe the linked document, used as a navigation guide to the information the user is seeking for. The anchor text of a is denoted by $text(a)$. Anchoring provides a mechanism for addressing locations within the content of a document.

Link is another basic element in a hypertext system. A *link* represents relations between documents. There are two kinds of links. A link from one anchor to another anchor is called span-to-span link, while a link from one anchor to a document is called as span-to-node link (Fig. **??**). In the following, we only consider span-to-node link, and represent a link as a triplet $< d_1, a, d_2 >$, where a is an source anchor in document d_1, d_2 is a destination document of this link.

Documents can be evaluated in terms of size, recency and frequency of reference. To measure the relevance of a document to some interested topics (later we call "semantic regions"), textual content (terms or sentences) will be evaluated on the basis of techniques in information retrieval (IR), such as vector space model (VSM) and TF–IDF scoring scheme. The content of a document d can be expressed as

$$contnentof(d) =< title, body >$$

where *title* is a sentence that describes the content of the document, and *body* is a sequence of terms in the document. Media component (files) are embedded in the hold places of container files. A media component file can be shared by one or more documents, thus whether a component file can be deleted by a garbage collector is determined by not only how often it has been used as in most caching schemes, but also determined by whether there is no more used by existing cached documents.

A hypertext database is often modeled as a directed hypergraph, with documents as nodes and links as edges. This model however is not suitable for client caches because a client cache does not see the whole structure of the potential hypergraph, instead what it can see are paths followed by the user in that hypergraph. To predict how the user uses the hypertext database for caching decision making, we should model the paths that the user often traverses, instead of the whole hypergraph.

2.2 Logical Documents: Logical Structure of Hypertext Data

As links created by hypertext authors do not always reflect what readers think, a cache often sees a subset of documents and a small fraction of paths (sequences of document-links) are often traversed. In a navigational access environment, users are apt to travel data items back and forth in accordance with paths. Thus, data items might be visited just because of it location, rather than its content. We define a path frequently traversed by some users as a *logical document.*

A logical document is a representation of user's perspective of the hypertext data. In other words, how authors created a hypertext database is one thing, while what the client would be interested is another. This distinguishes our model from any other hypertext models created from the point of view of hypertext authors or system designers.

Fig. **??** depicts two logical documents in a hypertext database: one is "A–B–E", the other is "A–D–G". In "A—D–G", starting from document **A**, the user often (13 times) chooses to follow a link to **D**, then **G**. It is reasonable to think that, for the user of the cache, "A–D–G" is a logical unit that contains specific information he needs. The first document in the path of a logical document is called an "entry document", while the last document traversed in the path is called a "terminal document".

Logical documents can be measured in terms of size, recency, frequency of reference. The size of a logical document is the length of path, which is indeed the number of documents contained in the path. A reference to a logical document is defined as a successful traversal starting from the entry document, walking through a link to the second document on the path within a limited time interval, and so on, until reaching the terminal document.

Logical documents represent the readers' viewpoint of hypertext data. That is, different paths leading to a same document imply different perspectives of the user, To deal with this difference, we define the content of a logical document to be $< title, body >$ with *title* being the union of anchor texts contained in the path and the title of the terminal document.

As shown in Fig. **??**, suppose we have a logical document $l =< [d_1, a_1], [d_2, a_2], [d_3] >$ where d_i $(i = 1, 2, 3)$ are documents in the repeating traversal path, a_i $(i = 1, 2)$ are anchors leading to a subsequent document. That is, the user first follows a link from anchor a_1 in d_1 to d_2, where he follows another link from anchor a_2 to d_3. Let $text(a_i)$ be the anchor text of a_i, $tile(d_i)$ and $body(d_i)$ be the title and body of a document respectively. Then we can define the content of logical document L to be

$$contnentof(l_i) < text(a_1) + text(a_2) + title(d_3), \quad body(d_3) >$$

Here "+" is string concatenation operation as in a typical programming language. For example, if the anchor texts on the path of a logical document are "Travel in Kyoto", "List of bus stations" and "Kyoto station", and the title of the terminal document is "Access to the Sinkansen superexpress", the the logical document will have a logical title "Travel to Kyoto, List of bus stations, Kyoto station, Access to the Sinkansen superexpress".

Note that logical documents can be of different sizes depending on the configuration of implementation and the actual usage status. A special case is when the size is 1, which means there is only one document included in the logical document. Thus, each visited document can a logical document.

2.3 Semantic Region: Semantic Structure of Hypertext Data

Semantic regions is a concrete description about user interests, which play an important role in identifying preference of users. We denote a semantic region as $R = (\sigma, \lambda)$, where σ is the semantic centroid (cluster center, or median). λ is the radius of the semantic region. A semantic region is a cluster of logical documents with a semantic centroid such that each logical document belongs to exactly one most suitable cluster, that is, it is closer to centroid of this cluster than any others. The centroid of a cluster is represented using a feature vector based on vector space model (VSM) and TF–IDF scoring scheme. For example, (30, 34, 120, 10) is a feature vector presentation with respect to (bread, butter, salt, knife).

As new documents come continuously, determining semantic regions for hypertext caching requires efficient single–pass clustering algorithms that consume a small amount of memory. Fortunately, there exist a number of streaming–data algorithms that can achieve high quality clustering [23, 4, 15]. In general, a clustering problem cane be described as follows: given the number k of clusters, a clustering algorithm will try to find k centroids so that each data point is assigned to the cluster defined by the centroid nearest to it. This is also known as "k-Median" problem.

Suppose the quality of clustering is measured by the sum of square distance of data points from their centroid, the randomized algorithm *LSEARCH*[15] can usually find a near–optimum solution in $O(nm + nk \log k)$ of time, where n is proportional to the number of data points, m is a small number. In this work, we will not evaluate various clustering algorithms, instead we assume we already know a suitable near–optimum algorithm that can always cluster new logical documents received. We will concentrate on exploring whether higher–level semantic information can help determine potential usage of hypertext data.

As content of a logical document has two parts: title and body, we need a method to combine them together. As terms in a title are generally more important than those in a body, we show stress more on title than on body. Suppose $\mathcal{L}$ is the set of logical documents for a hypertext cache. l_i is a logical document with content $< title, body >$. Let v_i^{title} and v_i^{body} be the TF–IDF based feature vectors for title and body of l_i respectively. The comprehensive feature vector of l_i can be calculated as a weighted sum of both, that is,

$$v_i = \omega \cdot v_i^{title} + v_i^{body}$$

Here ω is a parameter larger than 1. The combination of feature vectors for title part and body part enable to distinguish two logical documents even when they have the same terminal document. Again we consider the example about "how to access to Sinkansen superexpress" in "Kyoto station". Another reader may reference the same document after following a list of "NTT Western Japan", "Kyoto Office", "Location" , and then the terminal document. The first logical document is likely for general travelers, while the second logical document is much more suitable for business travelers.

3 HyperDB, Benchmarking Hypertext Semantic Caching

To verify the proposed model and to evaluate caching schemes with consideration of temporal, spatial, and semantic locality, we set up an experimental hypertext database system as a testbed. However, as mentioned before, currently we concentrate on evaluate semantic information derived according to our semantic model can be a help in better cache management.

3.1 HyperDB, A Virtual Hypertext Database

We need not actually generate all these elements and not need to employ a hypertext database management system to maintain it. We simplify the proposed model because we need only to maintain a set of features that provide necessary data for calculating logical documents, semantic regions. First, we can put aside the real content of documents or logical documents, because dealing with this feature we have to do much engineering work that has done by most IR researchers. Then all calculation involved in content of documents can be omitted for the time being.

We begin with generating a set of component files, each of which has a different size and media type. We then generate a set of container files, which has a size and a corresponding set of component files. In future, we will also consider terms and their TF-IDF scores with respect to a container file, but for the time being, we need not such a complicated implementation.

1. a set of components *Components*, each element is a triplet of *id*, media *type* and *size*.
2. a set of documents *Documents*, each of them has an *id*, a set of component id's and a set of terms with their corresponding TF-IDF *socres*;
3. a set of logical documents *LogicalDocuments*, with a *id*, a sequence of linked documents id's;
4. a set of semantic regions *SemanticRegions*, with a semantic centroid (a vector of weighted terms), a threshold. ;

A logical document is generated by choosing a sequence of documents (container files indeed). Each sequence has a length of 1, 2, 3 etc, indicating there is/are one or more documents.

3.2 Generating Accesses to HyperDB

For the hypertext database, we synthesize workloads. We use the following method to guarantee *temporal locality* in request stream. To generate a new request, we pick a random number and take different actions depending on the random number. If the number is higher than a certain constant μ, a new request is issued. If the number is lower than μ, we re-issue a request issued before. Thus, μ is the inherent hit ratio in the request stream.

If we need to re-issue an old request, we choose the request issued t requests ago with probability proportional to $1/t$. To determine this t, we maintain a sum S of $1/t$ for t from 1 to the number n of requests issued, that is $S = \sum_{t=1}^{t=n} 1/t$. Every time it is necessary to issue an old request, we pick a random number from 0 to 1 (call it r), calculating $r \cdot S$, and chooses t where,

$$\sum_{i=1}^{i=t-1} 1/i < r \cdot S < \sum_{i=1}^{i=1} 1/i$$

In essence, t is chosen with probability $1/(S \cdot t)$ and the recently issued requests are more likely to be re–issued. Temporal locality combined with hypertext structures and semantic regions, derives other kinds of reference locality, i.e. *spatial locality* and *semantic locality*, where spatially related or semantically related data items are more likely to be re–requested in a short period of time.

In the following, we apply this method in maintaining temporal locality at component–level, document–level, path-level and semantics-level, obtaining four groups of request streams. Table **??** lists the primary profiles of these workloads.

The workload A is generated on the basis of reference locality for basic objects without considering structural or semantic locality. Workload B is based on reference locality for documents, thus media components are accessed only when their container files are accessed for the first hand. Workload C is further based on logical documents. The recently accessed logical document will be more likely be re-accessed in the near future due to the temporal locality for logical documents. When a logical document is accessed, all member documents will with very high probability be accessed and all component files of those documents will be accessed consequently

Finally, workload D is based on reference locality of semantic regions. When a semantic region has recently been referenced (one of its member logical document was referred), then similar logical documents in the same semantic region are more likely to be reused in the near future. Within each semantic region, the recently used logical documents are more likely to be reused. We call the temporal locality for semantic regions as well as logical documents as "multiple temporal locality".

4 Semantic Caching for HyperDB

Based on the semantic model developed so far, we can now design algorithms for cache management. The baseline algorithm is LRU-K, proposed by E. J. O'Neil et al in [16]. The basic idea of LRU–K is to keep track of ΔT_K, i.e. time since last K'th reference (or infinitely large if there are no more than K references), using this information to estimate the popularity of a data item. The $K/\Delta T_K$ is usually called "dynamic access frequency" of a data item. A cache algorithm will try to find the least frequently used data (with smaller access frequency) and replace it with more popular data (with larger access frequency).

4.1 Size-Adjusted LRU-K (LRU-K-S)

As basic objects in hypertext databases are not identical in sizes, we should extend LRU-K to deal with the heterogeneous sizes of hypertext objects. The idea is to normalize miss penalty by data size, $K/(\Delta T_K \cdot Size)$, then use the normalized cost function to measure the potential of a data item. Consequently, we obtain a Size-Adjusted LRU-K (call it LRU-K-S).

4.2 Structure-Aware Caching (LRU-K-T)

Hypertext data has some structure where component data depend on container: when a container data item was accessed, all its component data will automatically be accessed. Conversely, when a container data item is replaced from a cache, the related components should also be deleted except some are shared by other containers.

So far, this kind of structure–based dependency has not been well addressed. We extend LRU-K to incorporate this property, assuming that each component file maintains a *reference counter*, indicating how many documents are currently sharing this component. Before the counter becomes zero, cache priority of this component depends on its container. When all its containers are removed (counter becomes zero), the component file will be deleted, similar to the idea of "garbage collection" in memory management. In this way, we obtain a Structure-Aware LRU-K (call it LRU-K-T).

4.3 Popular Path-Aware Caching (LRU-K-P)

Another extension to the standard LRU-K is making use of frequently traversed paths, namely, logical document. Recall that logical document is a larger granularity for cache management. As we only care about links that has been used at least once, we should not maintain a large collection of links. From the user's perspective, a never–used link may be of little interest and will less likely to be visited in the near future.

The time of last K traversals are recorded so that the "dynamic access frequency" about this logical document can be calculated. Next time when the user begin to visit a document at this path, cache manager will give higher priority to those documents on the paths or logical documents that have been "most recently and most frequently used". This is a Popular Path-Aware LRU-K, called LRU-K-P.

4.4 Content-Sensitive Caching (LRU-K-C)

Finally, we reach the point to develop the semantic caching scheme. The semantic region for navigational access is built upon document clustering techniques. First let us define some functions for managing priori ty queues on semantic regions, logical documents, documents and basic objects (single physical objects/files including both containers and components).

- $getEmbedders(d)$ returns a all documents that embed the object d
- $getEmbedded(p)$ returns a all objects embedded in the document p
- $getSemanticRegions(p)$ returns all semantic regions that a document p belongs to
- $getDocuments(t)$ returns all documents in semantic region t
- $getMostSemanticRegion(T)$ is a function for selecting a semantic region with highest priority in a set T of semantic regions

– *getLeastSemanticRegions*(T) is a function for selecting a semantic region with lowest priority in a set T of semantic regions
– *getLeastDocument*(L) returns a document with LEAST priority in set P
– *getLeastObject*(D), returns a object with LEAST priority in set D.

4.5 Experimental Evaluation

We assess the performance of the proposed algorithms using synthetic workloads obtained in Section 3. Although most studies on caching algorithms use trace–driven experiments or event–driven experiments [3], we choose in this work to use event–driven approach for two reasons. First, to the best of our knowledge there are no suitable benchmarks available for our purpose which include both a collection of hypertext data and the workloads with respective to those data. Second, our algorithms, especially content–sensitive caching, rely on techniques in other research fields, such as usage mining, text clustering etc. a thorough evaluation of all specific techniques is difficult and we will address in future work.

Fig. **??** shows results (hit ratios) of experiments under workload A, where no spatial or semantic locality is considered. The plots show LRU-K-S is better while others including LRU-K-C performed not so good. The reason is that the structure–aware or semantic-aware algorithms try to bias towards documents closer spatially or semantically to recently used ones, however, workload A lacks of such characteristics. This can also explain the results in Fig. **??** under workload D, where LRU-K-C performed better than others, since in workload D, logical documents within a same semantic region tends to be referenced at the same time.

5 Conclusion and Future Work

Semantic locality is the most important feature of data access, which has been exploited in semantic caching schemes for traditional query–retrieval based database systems. However, for a navigation based hypertext database system, such as the web, the advanced locality of reference has not yet been incorporated as there is no explicit description of semantic regions in the form of access. In this paper, we have proposed a semantic model that took into account both structural feature as well as semantic feature of hypertext data. Our model is especially suitable for navigational access to seeking desired data in a hyperlinked information space.

We verified the proposed model by creating an experimental hypertext database called HyperDB, then generated workloads with different locality of reference. These workloads can be used for analysis of semantic caching schemes for hypertext data. Preliminary experiments have done to evaluate some semantic caching schemes.

Our future work is to include textual content and test for efficiency under a verity of clustering schemes and parameterizing the process for constructing semantic regions.

Acknowledgments

The authors would like to thank Mukesh Mohania and Yanchun Zhang for many interesting discussions.

References

[1] Serge Abiteboul and Victor Vianu. Queries and Computation on the Web. In *Proceedings of 6th International Conference on Database Theory (ICDT'97)*, pages 262–275, January 8-10, Delphi, Greece, 1997. 277

[2] Foto N. Afrati and Constantinos D. Koutras. A Hypertext Model Supporting Query Mechanisms. In *Proceedings of European Conference on Hypertext*, pages 52–66, 1990. 277

[3] Charu Aggarwal, Joel L. Wolf, and Philip S. Yu. Caching on the World Wide Web. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):95–106, 1999. 285

[4] Paul S. Bradley, Usama M. Fayyad, and Cory Reina. Scaling Clustering Algorithms to Large Databases. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 9–15, New York City, New York, USA, August 1998. AAAI Press. 281

[5] Andrea Caloini. Matching Hypertext Models to Hypertext Systems: A Compilative Approach. In *Proceedings of European Conference on Hypertext*, pages 91–101, 1992. 277

[6] Brad Campbell and Joseph M. Goodman. HAM: A General Purpose Hypertext Abstraction Machine. *Communications of the ACM*, 31(7):856–867, July 1988. 277

[7] Chris Clifton and Hector Garcia-Molina. Indexing in a Hypertext Database. In *Proceedings of the 16th International Conference on Very Large Data Bases (VLDB)*, pages 36–49, Brisbane, Queensland, Australia, August 1990. Morgan Kaufmann. 276, 277

[8] Shaul Dar, Michael Franklin, Bjorn Jonsson, Divesh Srivastava, and Michael Tan. Semantic Data Caching and Replacement. In *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB)*, Bombay, India, September 1996.
`http://www.cs.umd.edu/projects/dimsum/papers/semanticcaching.ps.gz`. 276, 277

[9] Michael J. Franklin. *Client Data Caching*. Kluwer Academic Press, Boston, 1996. 276

[10] Richard Furuta and P. David Stotts. A Functional Meta-Structure for Hypertext Models and Systems. *Electronic Publishing*, 3(4):179–205, 1990. 277

[11] Frank G. Halasz and Mayer D. Schwartz. The Dexter Hypertext Reference Model. *Communications of the ACM*, 37(2):30–39, 1994. 277, 278

[12] Alberto O. Mendelzon and Tova Milo. Formal Models of Web Queries. In *Proceedings of ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems(PODS)*, pages 134–143, Tucson, Arizona, 1997. 277

[13] David E. Millard, Luc Moreau, Hugh C. Davis, and Siegfried Reich. FOHM: a Fundamental Open Hypertext Model for Investigating Interoperability between Hypertext Domains. In *Hypertext*, pages 93–102, 2000. 277

[14] Luc Moreau and Wendy Hall. On the Expressiveness of Links in Hypertext Systems. *The Computer Journal*, 41(7):459–473, 1998. 277

[15] Liadan O'Callaghan, Nina Mishra, Adam Meyerson, Sudipto Guha, and Rajeev Motwani. Streaming-Data Algorithms For High-Quality Clustering. In *International Conference on Data Engineering (ICDE)*, 2002. 281

[16] Elizabeth J. O'Neil, Patrick E. O'Neil, and Gerhard Weikum. The LRU-K Page Replacement Algorithm for Database Disk Buffering. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 297–306, New York, 1993. 283

[17] Seongbin Park. Structural Properties of Hypertext. In *UK Conference on Hypertext*, pages 180–187, 1998. 277

[18] Curt Schimmel. *Unix Systems for Modern Architectures.* Addison-Wesley, 1994. 276

[19] Marc Spielmann, Jerzy Tyszkiewicz, and Jan Van den Bussche. Distributed Computation of Web Queries Using Automata. In *Proceedings of ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems(PODS)*, pages 97–108, Madison, Wisconsin, USA, 2002. 277

[20] P. David Stotts and Richard Furuta. Programmable Browsing Semantics in Trellis. In *Hypertext*, pages 27–42, New York, 1989. 277

[21] Frank WM. Tompa. A Data Model for Fexible Hypertext Database Systems. *ACM Transations of Information Systems*, 7(1):85–100, 1989. 277

[22] Marcelo Augusto Santos Turine, Maria Cristina Ferreira de Oliveira, and Paulo Cesar Masiero. A Navigation-Oriented Hypertext Model Based on Statecharts. In *Hypertext*, pages 102–111, 1997. 277

[23] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *SIGMOD Conference*, pages 103–114, 1996. 281

Understanding and Simulating Narratives in the Context of Information Systems

Angelo E. M. Ciarlini[*] and Antonio L. Furtado

Departamento de Informática - Pontifícia Universidade Católica do R.J.
R. Marquês de S. Vicente, 225 - 22.453-900 Rio de Janeiro, Brazil
{angelo,furtado}@inf.puc-rio.br

Abstract. A modelling tool is described, which integrates simulation, plan recognition and knowledge discovery modules to help the specification and implementation of complex information systems. The tool is based on the concept of *plots of narratives*, defined as partially ordered sets of events leading to the achievement of various (possibly interconnected) goals. Based on a formal description of the behaviour of the participating agents, a planning algorithm is used to simulate valid plots of narratives. Plan recognition and knowledge discovery techniques are used to detect typical behaviours in observed plots. The framework also contains modules for presenting plots as texts, graphs and 3D animations, so as to provide informative and easy-to-handle user interfaces.

1 Introduction

Complex information systems usually deal with multiple different agents, including programs, humans and corporations. The programs involved often span thousands of lines, trying to fulfil mutually conflicting and/or collaborating goals. The amount of time and money spent on the specification and implementation of such systems tends to be huge. To reduce development costs, Software Engineering techniques have been introduced to promote the reuse of components. However, the choice of the right components and the way they should be combined and parameterized is far from trivial. The problem is even harder in actual practice, because users are seldom in a position to know what they exactly need, so they are unable to specify all requirements beforehand. Such difficulty is quite natural, since the users' characteristics are mutable and their goals often change. Information systems have then to be continuously adapted and restructured, demanding, again, a considerable amount of resources.

Simulation tools can be very useful to verify whether or not the behaviour of the various agents of the system under development corresponds to what is expected. Methods and tools for simulation, plan recognition and knowledge discovery (KD) may give important hints about how components can be selected and combined to ful-

[*] The work of the first author was supported by CNPq – Brazil

fil a goal of some agent. The system we have been developing integrates modules that perform these tasks, based on the concept of *plots of narratives*. Plots of narratives can be seen as sets of events leading to the achievement of various (possibly interconnected) goals. The system also contains modules to present plots as texts and animations, which is crucial, since the development process is basically interactive and the plots have to be fully understood by the users.

Section 2 briefly describes the logical framework upon which our approach is based. In section 3, we present the overall architecture of our system. Section 4 shows how plan recognition and the automatic construction of libraries of typical plans are used to understand the context of an information system. Section 5 describes the simulation of plots of narratives. Section 6 presents the different ways whereby plots can be visualized. Section 7 contains concluding remarks. To illustrate the concepts, we use as example a simplified information system comprising the business relationships between a company called Alpha and its clients and employees.

2 Logical Framework

Our logical framework can be compared to the *Event Calculus* [17], especially to one of its variants: *Event Calculus with Preconditions* [4], in that it is adequate for the representation of a course of actual actions about which we may have only partial information. If we want to reason about hypothetical situations, it is necessary to also take into account the question of if and when an event can occur. The description of narratives with the *Situation Calculus* [22] enables us to reason better about the hypothetical situations that result from hypothetical actions. The difference in our approach is that we focus on the use of simulation for the creation of coherent narratives, so we needed a formalism that could be directly used by planning and goal-inference algorithms. We assume the specification of information systems in three levels: the static level, the dynamic level and the behavioural level. The *static level* specifies the kinds of facts represented and the *dynamic level* the operations that bring about state transitions in the information systems. The *behavioural level* is intended to describe *why* events occur (i.e. the goals leading to the execution of operations) and *how* operations are usually combined in typical plans. Before giving details about the three specification levels, we shall introduce some basic concepts of *plot logic*, i.e. the logic we use to specify them. For space reasons, we limit ourselves to a brief overview, referring to [3] for the detailed formal treatment.

2.1 Syntax

We assume that the relevant facts of the information system under development are stored in a database. Our logic uses then a syntactic structure (a *plot*) for specifying and reasoning about the possible sequences of database states that are (or can be) created from an initial state by the execution of a (partially ordered) set of operations. Plot logic is based on a many-sorted database first-order logic [2] with a special sort of timestamps *Tmstmp* (representing instants of time), having two special constants, t_{In} and t_{Fin}, denoting the beginning (associated with the initial database state) and the end, respectively, of the narrative described by a plot.

The language is partitioned into database and constraint languages. Terms, formulae and sentences are as usual. *Database literals*, i.e. those in the database language, are used to represent the existence of an entity or the value of an attribute. The literal *level(john,X)*[1], for instance, can be used to say that *John has salary level X*. *Database facts* are represented by *positive ground database literals*. A *constraint* is an atomic formula of the constraint language, specifying restrictions on the values of variables (e.g. if variable *X* refers to the salary level of *John*, *X>2* could indicate that the value of this attribute has to be higher than 2).

An *event ev* is a quadruple *<sign(ev),tsp(ev),pre(ev),post(ev)>*, where *sign(ev)* is its signature containing a name and a list of parameteres, *tsp(ev)* is a timestamp, used to represent its occurrence time and *pre(ev)* and *post(ev)* are sets of database literals specifying, respectively, the pre- and post-conditions of the event. The event

<hire(john,1), t1, {person(john),¬employee(john)}, {employee(john) ,level(john,1)}>

can be used to represent that John was hired at time t1 with salary level 1.

A plot *PLT* is a quadruple *<S_0(PLT),Evs(PLT),Cntrs(PLT),Pord(PLT)>* where *S_0(PLT)* is a set of database facts (the initial database state), *Evs(PLT)* is a set of events, *Cntrs(PLT)* is a set of constraints on the variables that appear in *Evs(PLT)* and *Pord(PLT)* is a partial order of the timestamps of the events in *Evs(PLT)*.

We use metapredicates, constructed with modalities, to talk about events (executions of operations) and data along the plot. An *atomic modal formula* is one of the following metapredicates, to be interpreted as indicated: *o(t,en)* (event with signature *en* occurs at constant or variable time *t*); *e(t,dblt)* (database literal *dblt* is established[2] at time *t*); *h(t,dblt)* (database literal *dblt* necessarily holds at time *t*); *h(cnstr)* (constraint *cnstr* necessarily holds); *p(t,dblt)* (database literal *dblt* possibly holds at time *t*); and *p(cnstr)* (constraint *cnstr* possibly holds). The *modal formulae* are built from the metapredicates with the use of connectives and quantifiers as usual.

2.2 Semantics

The meaning of a *terminal event* (i.e. an event without free variables) is given by a partial function that transforms database states. A terminal event maps every database state in which its pre-conditions hold into a database state where all its post-conditions hold. Terminal events satisfy the frame requirement, so that facts that are neither established nor negated by the post-conditions hold at the target state iff they hold at the source state. The meaning of a non-terminal event is given by the set of all terminal events that can be obtained by means of variable substitutions.

A *trace* is a sequence of database states generated by consecutive terminal events. A plot is used to represent the set of all traces that can be obtained by performing variable substitutions (compatible with the constraints of the plot) and establishing a total order of its events (compatible with the partial order of the plot).

The satisfaction of our atomic formulae by a plot *PLT* is defined as follows:

[1] In all examples throughout this paper variables are going to be represented by strings starting by uppercase letters while constants start by lowercase letters.

[2] This metapredicate was introduced because a condition is valid only *after* (but not *at*) its establishment.

- *o(t,en)* is satisfied iff, for all possible variable substitutions that are consistent with the constraints of *PLT,* there is an event with signature *en* occurring at time *t*.
- *e(t,dbl)* is satisfied iff, for all possible variable substitutions that are consistent with the constraints of *PLT,* there is an event occurring at time *t* that has *dbl* among its post-conditions (or $t=t_{In}$ and *dbl* holds in the initial database state).
- *h(t,dbl)* is satisfied iff, for all possible variable substitutions that are consistent with the constraints of *PLT* and for all traces consistent with the partial order of the plot, there is a database state, associated with time *t*, in which *dbl* holds.
- *p(t,dbl)* is satisfied iff there is a variable substitution (consistent with the constraints of *PLT*) and a total order for the events in the plot (consistent with the partial order of the events in *PLT*) generating a trace in which the database literal *dbl* necessarily holds at the state corresponding to time *t* (i.e. *h(t,dbl)*).
- *h(cnstr)* is satisfied iff constraint *cnstr* holds for every variable substitution (consistent with the constraints of *PLT*), i.e. it is entailed by the constraints of *PLT.*
- *p(cnstr)* is satisfied iff there is at least one variable substitution such that *cnstr* holds, i.e. it is consistent with the constraints of the plot.

The satisfaction of non-atomic formulae is defined as usual.

A computational method can be applied for establishing whether an atomic temporal formula is satisfied by a plot. Its temporal reasoning criterion is an adaptation of the *Modal Truth Criterion (MTC)* [6] to a database context conforming to the *Closed World Assumption (CWA)* [24]. The method relies on searching for events in the plot and on checking whether its initial database satisfies a database literal. It uses the auxiliary concepts of *establisher, user and clobberer* events with respect to a database literal. A *user* is an event that has the database literal among its pre-conditions. An *establisher* is an event that occurs before the *user* and contains the database literal among its post-conditions. A *clobberer* is an event that occurs between the establisher and the user and has the negation of the database literal among its post-conditions. When an event can be transformed into a clobberer, by introducing additional order requirements, and/or by means of a valid variable substitution (i.e. one compatible with the set of constraints of the plot), it is said to be a possible *clobberer*.

Metapredicates *o(t,en)* (event with signature *en* occurred at time *t*) and *e(t,dbl)* (database literal *dbtl* was established at time *t*) are trivially verified by examining the events associated with time *t* (or the initial database state if *t* corresponds to the initial time). We evaluate the satisfaction of a metapredicate *h(dbl,t)* (*dbl* necessarily holds at time *t*) by checking whether *dbl* is established at a time *t'* (necessarily before *t*) and there is no possible clobberer between *t* an *t'*. A special case is *dbl* being established at the initial database state. Metapredicates *p(dbl,t)* (*dbl* possibly holds at time *t*) are evaluated by checking if it is possible to add further order requirements and/or a valid variable substitution, such that *dbl* necessarily holds at time *t*. Metapredicates *h(cnstr)* and *p(cnstr)* are verified by checking entailment and consistency, respectively.

In the next subsection, we describe the kinds of formulae that are used in our three-level modelling approach. It is on the basis of such formulae that simulations and KD tasks are performed by specific modules of our system.

2.3 Three-Level Conceptual Modelling

At the static level, *facts* are classified according to the Entity-Relationship model. Thus, a fact may refer either to the existence of an entity instance, or to the values of its attributes, or to its relationships with other entity instances. In the schema of Company Alpha database we could have, for instance, database literals such as *person(mary)*, specifying an entity instance (i.e. *Mary is a person*), *level(mary,1)*, specifying the value of an attribute (i.e. *Mary's salary level is 1*), and *serving(mary,beta)*, specifying a relationship instance (i.e. *Mary is serving client Beta*).

The dynamic level covers the specification of how database states may change. We specify a set of operations, the executions of which correspond to the possible events that may happen. Operations are special formulae of our plot logic. They are introduced as conditionals, defined as follows:

$$o(t,en) \rightarrow h(t,pre_1) \wedge \ldots \wedge h(t,pre_k) \wedge h(cnstr_1) \wedge \ldots \wedge h(cnstr_m) \wedge \\ e(t,post_1) \wedge \ldots \wedge e(t,post_n)$$

This formula specifies that if an operation generating events with signature *en* occurs at time *t*, then every pre-condition pre_i $(1 \leq i \leq k)$ necessarily holds at time *t*, every constraint $cnstr_i$ $(1 \leq i \leq m)$ necessarily holds and every post-condition $post_i$ $(1 \leq i \leq n)$ is established at time *t*. Variables are assumed as globally universally quantified. An example of an operation over Company Alpha database is:

$$o(t,hire(PERS,LVL)) \rightarrow h(t,person(PERS)) \wedge h(t,\neg employee(PERS)) \wedge h(LVL>0) \wedge \\ e(t, employee(PERS)) \wedge e(t, level(PERS,LVL))$$

In this formula, PERS and LVL are variables. It says that if PERS is hired with salary level LVL, then the salary level is higher than 0, PERS is a person but not an employee and, as a result of the operation, PERS becomes an employee with salary level LVL.

As the only time instant mentioned in an operation formula is its occurrence time, the time parameter can be removed from its definition and we can separate the atomic formulae in groups: signature, constraints, pre-conditions and post-conditions. Some of the pre-conditions are merely specifications of the domains of the parameters of the operation, so it is useful to distinguish them from the others. Borrowing from Fillmore's case grammars [7], a major contribution from the field of Linguistics, we also specify the semantic role of each parameter in order to facilitate the automatic generation of texts from plots.

Operation *hire*, for instance, is specified by the following Prolog clause:

```
oper(hire(PERS,LVL),          /* name and parameters */
    [person/object, integer /with_salary_level],  /* domains and semantic roles */
    [LVL>0],                              /* constraints */
    [¬employee(PERS)],                    /* pre-conditions */
    [ employee(PERS)) , level(PERS,LVL)])  /* post-conditions */
```

This gives a precise meaning to events based on the operation. The event *hire(mary,2)* is then read as *Company Alpha hired person Mary with salary level 2*.

Carefully designed application-oriented operations enable the various agents to handle the database in a consistent way. The question remains of whether they will coexist well with a system supporting such operations, and, if so, what actual usage patterns will emerge. Ideally, the designers of an information system should try to predict how agents will behave within the scope of the system, so as to ensure that the specification at the two preceding levels is adequate from a *pragmatic* viewpoint. The ability to make predictions about behaviour is also crucial for decision-making based on simulations of future events. In order to deal with this problem we need to model the goals that motivate the actions, the situations that bring them about and the usual patterns adopted by the agents to reach their goals.

At the behavioural level, we specify goal-inference rules connecting situations to goals. Goal-inference rules are conditionals in our plot logic. They have, as antecedent, some *situation* (expressed as a conjunction of metapredicates) which, if observed in the plot, will arouse in a given agent the impulse to act in order to reach some *goals* (expressed in the consequent as a conjunction of metapredicates of type *h(t,dbl)* and *h(cnstr)*). Variables appearing in the antecedent are assumed as globally universally quantified while those appearing only in the consequent are assumed as existentially quantified over the consequent. An example of a goal-inference rule in the context of Company Alpha database could be:

e(T1,account_status(C, inactive)) → h(T2, ¬account_status(C, inactive)) ∧ h(T2>T1)

This rule expresses the idea that *whenever the account status of a client becomes inactive, Company Alpha will want to turn it active again*. In this case, it is a goal-inference rule associated with Company Alpha itself. Other rules generating other goals for the company, or for its clients, or for its employees, could be analogously defined. The identification of the motivated agent is attached to each rule, enabling the use of this information by the modules of the system.

The specification of behaviour is complemented by a *Library of Typical Plans*. A *typical plan* is a description of how an agent (or class of agents) usually proceeds towards some goal. It consists of either a set of partially ordered operations or plans, or of a set of specialized alternative plans able to achieve the goal. Plans of both kinds are expressed in the *Library* as complex operations. Let us call the operations introduced in the previous subsection *basic operations*. Then, a *complex operation* can be defined from the repertoire of basic operations (or from other complex operations, recursively) by either composition (part-of hierarchy), giving origin to *composite* operations, or by generalization (is-a hierarchy), yielding *generic* operations. In case of composition, we define the component operations and the ordering requirements, if any (recalling that we allow plans to be *partially-ordered*). Complex operations are specified by the Prolog predicate *op_complex*, with the following syntax:

op_ complex(Operation_Signature, Domains_Semantic_Roles,
Tagged_Components, Order_of_Tags, Goals)

The tags mentioned above are used to specify the partial order of the components. The information about generic operations is complemented by predicate *is_a*:

is_a(Alternative_Operation_Name, Generic_Operation_Name)

3 The Overall Architecture

Fig. 1 shows the overall architecture of the system we have been developing. The arrows represent the flow of data. The kernel of the system is a module that semi-automatically generates plots of narratives. This module, named *Interactive Plot Generator (IPG)* [9], operates on a context that comprises the following items: a database specifying the initial state of the plots to be generated; a library of typical (basic and complex) operations that can be executed; and a set of logical rules, to infer goals usually pursued by the various agents, as certain situations arise in the course of plots.

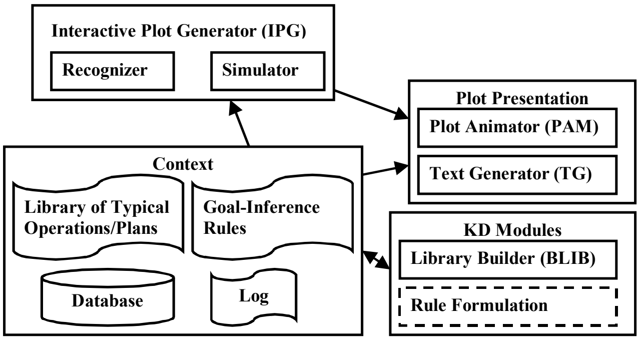


Fig. 1. Overall Architecture

IPG can be used in three different forms, roughly distinguished by the relative emphasis placed on its two sub-modules: the Recognizer and the Simulator. The first form is centred on the Recognizer. The user enters *observations*, which are taken by the system as hints about the plot to be generated (some of the operations to be executed, a few assertions about the situation of agents at intermediate states and goals to be finally reached, etc.), and the system looks for a typical plan pattern (or set of patterns) that may fit the observations supplied. The system fills in what is missing in the characterization of agents, taking into consideration the observations and the pertinent facts found in the initial state of the database. The second form, in contrast, is based exclusively on the Simulator. The user enters observations about the situation he wishes to simulate; then, the rules specifying the behaviour of agents are evaluated over these observations and the initial state of the database, in an attempt to infer new goals. From this point on, alternate phases occur of plan generation (to achieve such goals) and again of inference of goals (at each state reachable by the plans), until no further goals result. The third form involves an integrated use, in which a recognized plan is used as the initial input of a simulation run.

In order to help the user to specify the context, the system incorporates knowledge discovery modules. One such module, the *Library Builder (BLIB)* tries to identify typical plans from past events recorded in the database log. BLIB uses goal-inference rules to try to interpret the occurrence of groups of combined events. Such tentatively identified plans are then presented to the user, who decides whether each plan should be introduced in the library as a new complex operation. The implementation of a module to help the formulation of goal-inference rules is still under investigation.

Once a plot has been generated by IPG or extracted from the log, there are three different forms they can be presented to users. The first one corresponds to the direct presentation of a graph of events. This kind of presentation, however, is rather unappealing and does not convey to the user all information implicit in a plot. The second form is the generation of a text in natural language, which is the task performed by the *Text Generator (TG)* module. The third form, which is currently under implementation, is the animation of the plot using the *Plot Animator Module (PAM)*. It uses graphical information associated with the objects and agents of the context, together with scripts associated with the basic events, to create animations.

The prototype modules are implemented in Prolog using constraint programming techniques. Part of PAM has also been implemented in Java. In the rest of the paper, the basic features of the modules are described.

4 Understanding Narratives

Agents of information systems combine multiple actions to perform useful, often complex, tasks. Understanding the way such actions are combined is an important part of the implementation of any system. If a library of typical plans (adopted in the domain of the information system) is available, plan recognition algorithms can detect plans commonly being executed and their associated goals. Using this information, the designer can automate good plans, study alternatives, foresee the interference between plans, and even take measures to prevent detrimental plans. At run-time, the availability of such a library enables the system to co-operate with users more effectively, since their goals can be automatically identified. However, the preliminary construction of the library is not trivial. It demands a lot of knowledge about the domain, and can benefit from the use of an interactive module already accumulating some of this knowledge. Besides providing a library to be used in the future, the construction process itself enhances the understanding about the information system under development. In this section, we describe the Recognizer and the Library Construction modules, showing how they are used towards the interpretation of the narratives recorded in an information system.

4.1 Plan Recognition

The Recognizer uses a plan-recognition algorithm adapted from the algorithm defined by Kautz [14]. The input of the algorithm is a special kind of formula of our plot logic, named *observation*. An observation has the following form:

$$o(t_1, en_1) \wedge \ldots \wedge o(t_n, en_n) \wedge h(t_{n+1}, dblt_1) \wedge \ldots \wedge h(t_{n+k}, dblt_k) \wedge h(cnstr_1) \ldots \wedge h(cnstr_m)$$

Metapredicates of type *o(t,en)* indicate the occurrence of events that are part of the plan to be recognized. Metapredicates of type *h(t,dblt)* and *h(cnstr)* specify intermediate states and constraints that should be preserved when the plan is executed. An example of an observation is: *o(T1,hire(leonard,1))*$\wedge$ *o(T2,fire(PERS))*$\wedge$ *h(T1,serving(PERS,beta)), h(T2>T1)*. This observation says that *Leonard is hired with salary level 1 and then an employee who is serving Beta is fired.*

The algorithm examines the library and builds an *explanation graph* for each event in the observation, showing the plans the event might be part of. From event *hire(Leonard,1)*, the system creates an explanation graph showing that the corresponding plans might be: *engage Leonard as a stable employee*, *obtain a new client (who will be served by Leonard) or renovate the assistance provided to a certain client (which is one of the ways of improving the service)*. Given the event *fire(PERS)*, another graph is generated showing that the corresponding plans might be *renovate the assistance* or *reduce costs.*

After generating an explanation graph for each event, the graphs are matched, yielding a graph explaining how all the events can be regarded as part of a single plan. In the process, most of the variables are instantiated so as to allow the matching. By consulting the graph, plans can be identified. The system checks, for each possibly identified plan, whether the plan is *applicable* (i.e. it is compatible with the intermediate states and constraints specified in the observation, and its pre-conditions hold at the initial database state). During this check, the remaining free variables are usually instantiated. Both applicable and non-applicable detected plans are informed to the user together with their associated goals. In our example, the only resulting plan corresponds to an instance of complex operation *renovate_assistance*. If *Mary* is assigned to *Beta* at the initial state, the recognized plan is *renovate_assistance(beta, leonard,mary)*, whose goal is *turn the account status of Beta active*.

The user can start a simulation either by using a detected plan as a starting point or by asking the Simulator to try to fulfil its recognized goal in a different way.

4.2 Construction of Libraries of Typical Plans

Behavioural level specifications involve: (a) goal-inference rules, and (b) typical plans. In some cases, it is possible to at least formulate (a) to a reasonable extent, by investigating the intentions and vague objectives or *soft goals* of the prospective agents [23,16], whereas (b) cannot be determined, since the people consulted feel unable to predict the use of basic operations specified by the designers. It is then convenient to interpose a trial phase, wherein agents are given access to a prototypical version of the system, with a *log* of executed operations being recorded for later analysis. The design effort proceeds by extracting from the log a series of *plots*, corresponding to a subsequence of the log, circumscribed to a given time interval, and filtered so as to only retain events directly or indirectly related to certain objects.

Having isolated a number of plots, the Library Builder(BLIB) analyses them by applying goal-inference rules previously obtained. Such rules are, for the time being, restricted to mention only two times: one when a situation causing a goal is verified and the other when the goal is achieved. Given a plot *PL* and a goal-inference rule saying that an agent *A*, confronted with situation *S*, will have the desire to achieve a goal *G*, BLIB tries to identify an event or sequence of events *O* in *PL* that transforms a state at which *S* holds in a state at which *G* holds. Plan *P* is then obtained from *O* by a second more refined filtering process, which only keeps the events whose post-conditions contribute to *G*, plus, proceeding backwards, recursively, those that achieve pre-conditions of events already included in *P*; the strictly necessary ordering requirements between events are also verified. Using this information, BLIB suggests the inclusion in the library of a *composite operation* incorporating plan *P*, as one way to perform the transformation indicated by the goal-inference rule.

To illustrate the process, assume that the following plot was extracted from the log:

PL = [s0, complain(beta, mary), hire(leonard), hire(john), replace(mary,leonard,beta), fire(mary)]

where s0 denotes the preceding database state. Now, consider the rule below, associated with *Company Alpha* to be tentatively applied to *PL*:

e(T1,account_status(C, inactive)) $\rightarrow$ *h(T2,* $\neg$ *account_status(C, inactive))* $\wedge$ *h(T2>T1)*

expressing that whenever the account status of a client is inactive, Company Alpha will want to turn it active again.

BLIB is then able to identify that *complain(beta,mary)* creates a situation in which the status of *Beta* is inactive. It also determines that the sequence of events *[hire(leonard),replace(mary,leonard,beta),fire(mary)],* in which the irrelevant *hire(john)* is eliminated, turns the status of Beta active again. The following complex operation is then a candidate to be included in the library.

```
op_complex( renovate_assistance(beta,leonard,mary),   /* signature */
   [client/ to, person/ with, employee/ 'in the position of']),  /* domains and sematic roles */
   [f1: hire(leonard), f2: replace(mary,leonard,beta), f3: fire(mary)],  /* tagged components */
   [f1<f2, f2<f3], /* order of components */
   [ ¬ account_status(beta,inactive)])  /* goals */
```

The semantic roles of the parameters are provided by the user when he agrees in including the operation. All operations included are organized in an index that associates triples of the form <*Agent,Situation,Goal*>, taken from each rule, with the set of corresponding identified plans.

We say that two plans *Pi* and *Pj* are *similar* if, even with different parameter values and executed in a different order, they coincide with respect to: (1) number and type of events, (2) order dependencies, (3) co-designation/ non-co-designation schemes of variables. When a new identified plan *Pi* is similar to a plan *Pj* already present at the associated entry of the library, BLIB checks if the *most specific generalization* [8] *P** between *Pi* and *Pj* is more general (contains a larger number of variables) than *Pj*; if so, *Pj* is replaced by *P**, otherwise no change occurs. This is the way variables are introduced in the definitions of the complex operations. On the other hand, if *Pi* is not similar to any *Pj* at the entry, then this new *Pi* is included and, in addition, BLIB suggests the definition of a *generic operation* to be also included in the entry.

The *first phase* of BLIB creates no more than one-level composition and generalization hierarchies, the only slightly more involved possibility being the presence of composite operations as alternatives of a generic operation. Due to our concern to avoid complicated and time-consuming cases of propagation, we decided to keep this simple structure as long as new plots continue to be submitted as input. However, at any time after a batch of plots has been processed, BLIB can be called to execute a reorganization run over the index, so as to produce certain improved multi-level hierarchies. This *second phase* of BLIB allows three kinds of restructuring, which are attempted in the order indicated: (1) multi-level generalizations; (2) generic operations as components of composite operations; (3) multi-level compositions. More details about the library construction process can be found in [11].

5 Simulating Narratives

The initial input for the Simulator is an observation, with the same format specified in 4.1. An initial plot is created using the definitions in this formula. A metapredicate *h(cnstr)* adds *cnstr* to the set of constraints of the plot. A metapredicate *o(t,ev_name)* adds a new event, with *t* as timestamp and containing the pre-conditions and post-conditions defined in the operation formula associated with the *ev_name* parameter. Constraints defined in this operation formula are also added to the set of constraints of the plot. A set of metapredicates $h(t,lit_1),...,h(t,lit_n)$ creates a fictitious event with timestamp *t*, without post-conditions, and with $\{lit_1,...,lit_n\}$ as pre-conditions. If a timestamp is a variable, a new timestamp constant is automatically introduced.

After this initial phase, the plot is usually no longer coherent, i.e. it has events containing unfulfilled pre-conditions. We then use a planning algorithm to explore the alternatives for achieving these pre-conditions at the corresponding times. Such alternatives involve the addition of new events and constraints. Our planner is an adaptation of the non-linear planner Abtweak [28], which operates as required by MTC.

When a coherent plot is generated, the plot is presented to the user. If the user decides to continue the simulation of this plot, the inference of new goals takes place. Goals are inferred by taking into account the problems and opportunities emerging at possible states of a partial plot. The goal-inference rules are exhaustively evaluated: whenever the antecedent holds but not the consequent, the consequent is taken as an observation to be added to the plot before the next planning stage. Notice that, at the start of the simulation, both the observations informed by the user and those that can be inferred from an initial plot, containing only the initial database state, are added.

The treatment of plot constraints is performed by constraint solvers of each data sort during the planning task and the inference of new goals. During the planning task, the constraint solvers run in parallel with the search. They discard a plot with inconsistent constraints as soon as their presence is detected.

Our plots are not restricted to incorporating only successful plans. In trying to provide adequate means for handling negative interactions along plots [27], we realized that the solution of conflicts and competitions sometimes requires the inclusion of totally or partially *failed* plans, which conventional plan generators reject. When a goal is abandoned, events occurring prior to the moment of abandonment must be kept as part of the narrative, and thus influence its continuation. We use two main mechanisms to handle goal abandonment and competitive plan execution: *conditional goals* and *limited goals*. A conditional goal has attached to it a *survival* condition, which the planner must check to determine whether the goal should still be pursued. Limited goals are those that have an associated *limit* (expressed as a natural number). The limit restricts the number of events that can be inserted to achieve the goal.

In order to exemplify the simulation process, imagine that a set of goal-inference rules were defined for the information system of company Alpha. The rules say that

1. *Whenever a new client has no employee to serve it, and there are people still unassigned to any client, such people will compete for the position until one of them achieves the goal.*

2. *Whenever an employee is not content with his current salary, he will try, in accordance with his persistence, to achieve the status corresponding to the next salary level.*
3. *Whenever an employee has a status higher than his current salary indicates, he will be promoted to the next salary level.*
4. *Whenever the salary of an employee E goes beyond the budget of client C, whom E is serving, E will not be permitted to continue serving C.*
5. *Whenever an employee E is dissociated from his/her client and there are neither clients without employee nor potential clients to conquer, E will cease to be Alpha's employee.*

Using the plot logic, rule (1), for example, would be specified in the following way:

$e(T1,client(C)) \wedge \neg e(T1,serving(E2,C)) \wedge h(T1,person(E)) \wedge \neg h(T1,serving(E,C2))$
$h(T2>T1) \wedge h(T2,serving(E,C)).$ *Survival Condition:* $\neg serving(E3,C)$

Imagine also that the initial database has three employees (*David*, *Leonard* and *John*), one unemployed person (*Mary*) and three clients (*Beta*, *Epsilon* and *Lambda*). *David* is serving *Epsilon* and *Leonard* is serving *Lambda*. All employees have salary level 1, and the budgets of the client companies cannot afford salary levels above 2. Both *David* and *Leonard* have the ambition of reaching salary level 3, but *David's* persistence is stronger than *Leonard's*. One of the possible narratives generated by the system after iterating across 6 stages, without providing any initial observation, is:

David and Leonard work hard and their status is raised to 2. John and Mary compete for client Beta. John wins and Mary's actions towards this objective are interrupted. David and Leonard have their salary raised. David and Leonard try to increase their status again. But now this requires that two training courses be taken. David takes them, whereas Leonard abandons his attempts to reach a higher status. David's salary is raised again. Since David's salary now exceeds Epsilon's budget, Mary is hired and replaces David at the service of Epsilon. Since David is no longer serving any client, and there is no current client to which he might be appointed, nor even potential clients to bring in, David is fired.

6 Presenting Narratives

6.1 Generating Texts

Although text generation [13,21] is by itself a complex process, a fairly informative *Text-Generator* module (henceforward, TG), described in more detail in [10], can be constructed if two requirements are met: (1) a comprehensive database schema is specified and made accessible, including, among other features, the definition of application-oriented *operations*; (2) a sequence of events based on the operations is provided. The sequence can be obtained either from a log registering actual executions of operations or as a result of a simulation run. In the first case, a preliminary step is filtering only application-oriented operations performed during a certain time interval and associated with the object of interest. In the second case, it is necessary (at least in the current version of TG) to assume a total order of the events compatible with the partial order of the generated plot.

Processing for text generation is generally divided into two stages [26,21]. The first (producing the strategic component) determines “what to say”, i.e. the contents and structure of the discourse, whereas the second (tactical component) finds out “how to say”, expanding in natural language the message produced by the strategic component. Until now, we have mainly concentrated on the strategic component, taking very little from the powerful instrumentality of Computational Linguistics [12,21,18].

Distinct events in a plot (denoted by executions of operations) are the syntactical units of the narrative for TG. By inspecting, in the schema, the signature of the operation involved, the semantic roles of the parameters can be expressed unambiguously. We express the connection between pre-conditions and effects (post-conditions) using the following scheme: *As <pre-conditions>, <operation>, so <effects>.*

The *textual markers* [25] “as” and “so” were chosen as being relatively neutral to express various kinds of enablement and consequence, so as to accommodate as many cases as possible with minimum strain. All operations have, at least, the trivial precondition that the actual arguments be of the types indicated for their parameters. Thus, given *hire(mary)*, the system will check if *Mary is a person*. Such obvious preconditions are not spelled out in the text; thus no pre-condition will be listed for some events. If two or more pre-conditions are present, they are separated by “and”.

TG classifies the effects into: 1. creation of entity instance, 2. deletion of entity instance, 3. assignment of value to attribute, 4. removal of value of attribute, 5. creation of relationship instance, and 6. deletion of relationship instance. Cases (1) and (2) include gaining or losing membership in a specialized class, e.g. a person starts or ceases to be an employee. Coupled occurrences of the pairs (4)-(3) or (6)-(5) are recognized and treated as *modifications*, rather than independent deletions-creations.

So far, we have only shown how TG is guided by information taken from the static and dynamic levels of the schema. Let us now turn to the behavioural level. If, on examining an event E and the part of the plot coming before its occurrence, TG decides that a declared goal-inference rule is applicable, it uses for E the expanded scheme: *Since <situation> and as <pre-conditions>, then <event>, so that <goal> and, in addition, <effects>.*

The consideration of goals results in a refinement of the characterization of events. A fact F that is both a pre-condition and part of a motivating situation can be said to both *enable* and *motivate* the operation. Since the latter relation is more significant, F is shown only once under the stronger “since” textual marker. (Notice that a fact can be part of <situation> without being a pre-condition: e.g. an employee can be led to strive for level 2 if some other employee was able to reach it, but this is exclusively a motivation, not a required condition for his actions). For analogous reasons, effects that are also part of a goal are separately introduced by “so that”. Also, if TG conjectures that a typical plan was used to achieve the goal, it introduces a parenthetical comment, following the scheme, where one or the other kind of complex operation may be missing: *(thus <complex operation by composition>, and, in this way, <complex operation by generalization>).* An example follows. Although informative, the resulting text is highly redundant and clearly needs stylistic improvement.

Plot: [s0,complain(beta,mary),hire(leonard),replace(mary,leonard,beta),fire(mary)]

Narrative: As employee Mary was serving client Beta, client Beta complained about employee Mary, so the account of client with denomination Beta became inactive. Company Alpha hired person Leonard, so person with name Leonard became employee and the level of employee with name Leonard became 1. Since the account of client Beta was inactive, and as employee Mary was serving client Beta and employee Leonard was not serving any client, then company Alpha replaced employee Mary by employee Leonard for client Beta, so that the account of client with denomination Beta ceased to be inactive, and, in addition, employee with name Leonard, instead of Mary, started to be serving client with denomination Beta. Since employee Mary was not serving any client, then company Alpha fired employee Mary, so that person with name Mary ceased to be employee, and, in addition, the level of person with name Mary ceased to be 1 (thus, company Alpha renovated assistance to client Beta with employee Leonard in the position of employee Mary, and, in this way, company Alpha improved service for client Beta).

6.2 Generating 3D Animations

The Plot Animator Module (PAM) has been implemented to provide a more appealing visual medium for the representation of plots. PAM uses 3D game engines that provide a number of services for real-time animation. This strategy enabled us to focus on the dynamic creation of narratives.

To represent objects relevant to the animation, 3D models that can be parameterized are stored in the database. An object can be associated with a set of models, from which the representation is chosen depending on the situation. We have extended the definition of the basic operations to allow the specification of how a script representing the corresponding events can be automatically generated.

The choice of the linear order of the events is relevant to the animation, not only because of the order according to which the events are animated, but also because the previous occurrence of an event may change the way another event is to be shown. The user chooses a linear order of events by connecting nodes representing the events in a graph; only connections satisfying the partial order requirements are permitted.

Once a plot has a linear order, an animation script is generated. The initial state of the plot and the definition of the events are retrieved from the context. The script for the representation of each event is specified according to the definitions in the corresponding operation. How objects are shown depends on the values of their attributes, which determine the 3D models to be used and how they should be parameterized. As soon as the script for the representation of each event is completed, the engine is activated to generate the animation.

7 Concluding Remarks

Most of the system is operational, but there is still some integration work to be done, in particular with respect to the TG and BLIB modules, which are currently using a simplified notation. Also, the present plot logic formalism assumes that events are instantaneous and does not cover the possibility of effects caused by combined events. We plan to overcome these limitations, possibly by adapting the line proposed in [5].

The ongoing research on the KD part of the system is focused on goal analysis and on methods and tools for the discovery [15,20] of goal-inference rules. Some of our decisions concerning the implementation of BLIB were only tentative, and need to be revised as we work directly with databases of a realistic size. The introduction of new transformations and additional criteria, based especially on statistic measures, are possibilities to be considered in the construction and restructuring of libraries of typical plans. In the future extensions of TG, we intend to fully explore and refine the characterization of the connections among events, pre-conditions, effects and goals. Detailed work remains to be done on the *tactical component* of the text generation process, corresponding to rhetorical and stylistic enhancements [1]. PAM will be considerably expanded, a central issue being its integration with TG, in order to combine animation and text in a single interface, as complementary forms of telling a story.

References

[1] Brown, G., G. Yule: Discourse Analysis. Cambridge University Press (1983)

[2] Casanova, M.A., Bernstein, P.: A Formal System for Reasoning about Programs Accessing a Relational Database. ACM Transactions on Programming Languages and Systems, 2(3) (1980) 386-414

[3] Ciarlini, A., Veloso, P., Furtado, A.: A Formal Framework for Modelling at the Behavioural Level. In: Proc. The Tenth European-Japanese Conference on Information Modelling and Knowledge Bases, Saariselk□, Finland (2000)

[4] Cervesato, I., Franceschet, M., Montanari, A.: Modal Event Calculi with Preconditions. In: Proc. of the 4th. International Workshop on Temporal Representation and Reasoning, Daytona Beach, FL, USA (1997) 38-45

[5] Cervesato, I., Montanari, A.: A Calculus of Macro-Events: Progress Report. In: Proc. of the Seventh International Workshop on Temporal Representation and Reasoning □TIME□00, Nova Scotia, Canada (2000)

[6] Chapman, D.: Planning for conjunctive goals. Artificial Intelligence, 32. (1987) 333-377

[7] Fillmore, C.:The Case for Case. In: Bach, E., Harms, R. (eds.): Universals in Linguistic Theory. Holt, Rinehart and Winston (1968)

[8] Furtado, A.: Analogy by Generalization and the Quest of the Grail. ACM/SIGPLAN Notices, 27, 1 (1992)

[9] Furtado, A., Ciarlini, A.: Operational Characterization of Genre in Literary and Real-life Domains. In: Proc. ER□99 Conceptual Modelling Conference, Paris, France (1999)

[10] Furtado, A., Ciarlini, A.: Generating Narratives from Plots Using Schema Information. In: Proc. NLDB□00 Applications of Natural Language to Information Systems, Versailles, France (2000)

[11] Furtado, A., Ciarlini, A.: Constructing Libraries of Typical Plans. In: Proc. CaiSE□01, The Thirteenth International Conference on Computer Advanced Information System Engineering, Interlaken, Switzerland (2001)

[12] Grice, H. P.: Logic and Conversation. In: Cole, P., Morgan, J. (eds.): Syntax and Semantics, vol. 3. Academic Press (1975)

[13] Grosz, B. J., Jones, K. S., Webber, B. L. (eds.): Readings in Natural Language Processing. Morgan Kaufmann, San Mateo (1986)
[14] Kautz, H. A.: A Formal Theory of Plan Recognition and its Implementation. In: Allen, J. F. et al (eds.): Reasoning about Plans. Morgan Kaufmann, San Mateo (1991)
[15] Kolodner, J. L: Case-Based Reasoning. Morgan Kaufmann, San Mateo (1993)
[16] Kowalski, R., Sadri, F.: From Logic Programming towards Multi-Agent Systems. In: Annals of Mathematics and Artificial Intelligence, 25, 1-2 (1999) 391-419
[17] Kowalski, R., Sergot, M.: A Logic-Based Calculus of Events. New Generation Computing, 4. Ohmsha Ltd and Springer-Verlag (1986) 67-95
[18] Mann, W. C., Thompson, S. A. Rhetorical Structure Theory; Toward a Functional Theory of Text Organization. Text, 8, 3 (1988)
[19] Marriot, K., Stuckey, P.J: Programming with Constraints. MIT Press (1998)
[20] Matheus, C.J., Chan, P.K., Piatesky-Shapiro, G.: Systems for Knowledge Discovery in Databases. IEEE Transactions on Knowledge and Data Engineering, 5, 6 (1993)
[21] McKeown, K.R.: Text Generation: Using Dscourse Strategies and Focus Constraints to Generate Natural Language Text. Cambridge University Press, Cambridge (1992)
[22] Miller, R., Shanahan, M.: Narratives in the Situation Calculus. Journal of Logic & Computation, Vol. 4, Number 5 (1994)
[23] Mylopoulos, J., Chung, L., Yu, E.: From Object-Oriented to Goal-Oriented Requirements Analysis. Communications of the ACM, 42, 1 (1999) 31-37
[24] Reiter, R.: On Closed World Databases. In: Gallaire, H., Minker, J. (eds.): Logic and Databases. Plenum Press (1978) 55-76
[25] Scott, D.R., Souza, C.S.: Getting the Message across in RST-Based Text Generation. In: Dale, R., Mellish, C., Zonk, M. (eds.): Current Research in Natural Language Generation. Academic Press (1990)
[26] Thompson, H.: Strategy and Tactics: a Model for Language Production. In: Papers 13th Regional Meeting Chicago Linguistic Society (1977)
[27] Wilensky, R.: Points: a Theory of the Structure of Stories in Memory. In: Grosz, B. J., Jones, K. S., Webber, B. L. (eds.): Readings in Natural Language Processing. Morgan Kaufmann, San Mateo (1986)
[28] Yang, Q., Tenenberg, J., Woods, S.: On the Implementation and Evaluation of Abtweak. In: Computational Intelligence Journal, Vol. 12, Number 2, Blackwell Publishers (1996) 295-318

Global Schema Generation Using Formal Ontologies

Farshad Hakimpour[1]
farshad@geo.unizh.ch
Department of Geography
University of Zurich

Andreas Geppert[2]
geppert@acm.org
Credit Suisse Financial Services
Zurich, Switzerland

Abstract. This paper deals with the problem of handling semantic heterogeneity during schema integration. Semantics refer to the meaning of data in contrast to syntax, which solely defines the structure of schema elements. We focus on the part of semantics related to the meanings of terms used to name schema elements. Our approach does not rely on the names of the schema elements or the structure of the schema. Instead, we present an approach based on formal ontologies presented in a logical language for integrating schemas to generate a global schema. Semantic similarity relations between definitions in formal ontologies are defined and used for merging ontologies. We show how similarity relations are discovered by a reasoning system using a higher-level ontology. The result of the merging process is used for schema integration. Schema integration is used to obtain the global schema of a tightly-coupled federated database system. Afterwards, we illustrate how the produced global schema can help for the mapping of data elements.

1 Introduction

Global schema generation is a critical task performed during information system integration. A major obstacle is *semantic heterogeneity*. Semantics refer to the study of the relation between symbols and what they represent—also called *interpretation* of symbols. Communication is generally the main purpose of the symbols and any misinterpretation of symbols representing data during communication between information systems is called semantic heterogeneity. In our work, the terms used for naming schema elements are the symbols. Therefore, our goal is to reduce the number of misinterpretations of such terms.

Explicit and formal definitions of the semantics of the terms guided researchers to apply *formal ontologies* [7] as a potential solution to semantic heterogeneity. A formal ontology consists of *logical axioms* that convey the meaning of terms for a particular *community* [2]. A set of logical axioms defining one term is called *intensional definition*[3] and there is *only* one intensional definition per term for each community. These definitions use minimal assumptions from the application domain and explicitly state the specification of a conceptualization [7]. Consensus on ontological definitions

1. The work of Farshad Hakimpour is funded by Swiss National Science Foundation (Project Number: 2100-053995).
2. Work done while with the University of Zurich

among members of a community is an important difference between ontologies and conceptual schemas [3, 12]. Conceptual schemas are application-dependent and concerned with challenges of computer representation and independence from implementational issues. In contrast, ontologies are based solely on the understanding of the members of a community and help to reduce ambiguity in communication. Note that formal ontologies carry more knowledge than schema definitions in databases. Schemas are mainly concerned with organizing data in databases according to the specification of application requirements. However, schema definitions are not independent from the ontological definitions and convey part of the knowledge about the ontology of a community.

In this paper, we show how formal ontologies can be used to derive global schemas from local schemas during database integration. The approach relies on formal ontologies being available for the local schemas. These ontologies are merged, giving rise to similarity relations (such as equality or specialization). The knowledge gained about these relations is then used for global schema definitions in such a way that semantic conflicts (at the schema level) can be detected and resolved. The result of merging ontologies is also used for defining data mappings, i.e. the mapping of local database entities into data elements on the global level.

This paper is structured as following. Section 2 discusses related work. In section 3, an overview of the approach is presented. In sections 4, 5 and 6, we introduce the steps of our approach. Section 4 describes similarity relations and illustrates the process of merging ontologies by an example. In section 5, we discuss the generation of global schemas based on a merged-ontology. Section 6 presents tasks during data mapping. The last section concludes the paper and discusses future work.

2 Related Work

Two pioneer projects using ontologies for information integration and interoperability are KRAFT and COIN. KRAFT uses shared ontologies [18] as a basis for mapping between ontology definitions and communication between agents. It detects a set of ontology mismatches and establishes the mapping between a shared ontology and local ontologies. The COIN [6] project presents a suitable architecture for semantic interoperability. The role of the *Domain Model* in the COIN-architecture can be compared to that of an ontology. However, the Domain Model is close to a conceptual schema rather than an ontology.

The work of Larson *et. al.* in [11] is very close to our work. We address some challenges already addressed in their work. However, an important difference is that we distinguish and emphasize the difference between the ontological characteristics of attributes and their representational and implementational characteristics. For example, characteristics such as domain, uniqueness and cardinality are present in the relation

3. Intensional definitions are estimating every *intensional relation* (defined in [7]). For instance, "Faculty" is an intensional relation and its estimation by an intensional definition is: ι[Faculty(x)]= Employee(x) $\wedge$ ($\exists$y: Course(y) $\wedge$ teaches(x,y))

definitions in an ontology while security constraints or scale are representational characteristics.

The work of Kim *et. al.* [10] presents a comprehensive study for the classification of schema heterogeneities. Also, solutions for several types of schema heterogeneity in RDBs and OODBs are presented. Another comprehensive work in this area is presented in [5]. Both authors address problems of schema heterogeneity, but do not distinguish between the schematic and semantic issues while we focus on the semantic problems here.

Miller *et. al.* in [16] distinguish schema integration from schema mapping. They focus on the mapping of data, taking the integrated schema for granted. Likewise, we consider phases of schema integration and data mapping. However, data mapping in our work is not yet as complete as the one in [16] and will require further investigations.

Bergamaschi *et. al.* [1], Palopoli *et. al.* [16] and Medhavan *et. al.* [14] propose approaches using thesauri. Bergamaschi *et. al.* introduced a semiautomatic approach assisting domain experts in extracting relations in thesauri from schema structure by help of domain experts. Based on the extracted relations they introduce an algorithm to integrate schema definitions into a global homogeneous schema. Palopoli *et. al.* rely on domain-specific knowledge and enhance that with knowledge extracted from the schema definition. Cupid [14] proposes an approach for schema matching. The approach takes both the similarity of the terms in the schema definitions (language similarity) and the structure of the schema into account (structural similarity). Cupid improves thesauri with a coefficient for every entry in the thesauri–similar to [16]. It also categorizes schema elements into clusters—similar to [1]. For communication between different application-domains, coefficients of the thesauri entries must be set by experts in both application-domains. In contrast, we try to establish the similarity relations based on formal ontologies (defined in section 4), while by using a thesauri synonym and hyponym relations are provided by domain experts.

The Inter-Ontology Relationships Manager module in OBSERVER [15] maintains the same relations between ontological definitions as presented in [1]. By means of inter-ontology relations, OBSERVER replaces terms in user queries with suitable terms in target ontologies.

Projects such as SHOE [9] and On2Broker [4] use formal ontologies as the basis for search engines on the Internet. They use ontologies for translating queries. In such cases, data sources are numerous and their schemas (such as DTDs and XML Schemas) are subject to frequent changes—they may not even have any schema. The drawback of this approach for structured databases is the high processing cost, because for every query, ontologies must be processed to derive required mappings. On the other hand, human supervision to validate translations is not possible, due to the need for immediate action. Lack of human supervision makes this approach less reliable for precise and strict applications.

In addition to the closely related work we would like to point out three further major areas of research, namely building, formalising and processing ontologies. *Building* refers to the extraction of specifications from a community's conceptualization. *Formalization* defines the way how we express ontologies instead of using natural languages.

A variety of formalisms can be used such as logical languages, RDF, ER-diagrams, UML diagrams and Conceptual Graphs which can be distinguished with respect to their degree of expressiveness. *Processing* refers to reasoning with the formalized definitions. This includes finding relations between definitions, unidentified inconsistencies among the definitions, classifying instance data according to the definitions, etc. Each of the topics are the subject of further work which is beyond the scope of this paper.

3 Overview

Formal ontologies are used in this work to generate a global schema, which then defines the organization of data in a federated database system. In this approach, class and attribute names in database schema definitions must be based on the terms defined in the formal ontology of a community. As illustrated in Fig. 1, terms used in schema p1 and p2 are based on the definitions in formal ontology p. This is done by linking class and attribute names in the schema definitions to terms in the formal ontology. In order to find out whether elements from different schemas are related, we define *similarity relations*. Detection of similarity is based on intensional definitions of terms in the formal ontologies. In the first phase, a reasoning system is used to *merge* formal ontologies. The result of merging is used by a schema integrator to build a global schema from local schemas. In the last phase, we find the possible meaningful mappings in the generated global schema and by that establish the mapping of data between the (global and local) databases. This approach is semiautomatic and needs human supervision during schema integration.

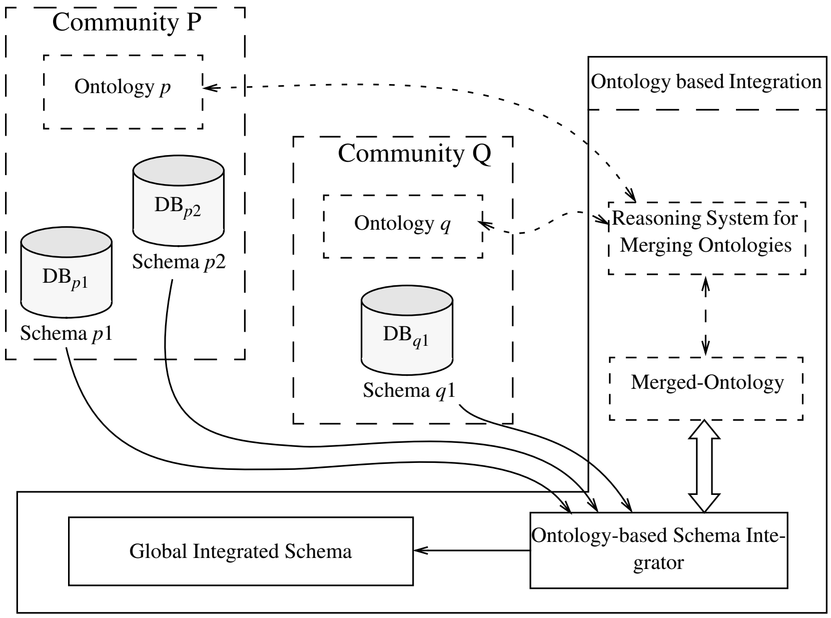


Fig. 1. Global schema generation based on a common ontology produced by integration of ontologies.

4 Merging Ontologies

An important step during database integration is relating schema elements from different local databases. Valid integration relies on a sound understanding of the meaning of the schema elements—this is, it must be found out if schema elements from different schemas refer to the same real world entities, or whether they are different (and if yes, to which degree they differ).

To that end, we rely on ontologies available for local schemas. Schema integration is then based on the merger of these ontologies. Thus, the main task in merging ontologies is to match the intensional definitions from different formal ontologies. The matching is done based on *similarity relations*. By *similarity*, we refer to the particular relations *equality*, *specialization*, *overlapping* and *disjoint* between intensional definitions in two different formal ontologies.

The similarity relations between two terms ${}^pT_i$ and ${}^qT_j$ defined in formal ontologies p and q are defined as following (ι maps a term to its intensional definition) [8]:

- ${}^pT_i$ is *Equal* to (or synonym of) ${}^qT_j$ if and only if both intensional definitions are the same:

$$({}^pT_i = {}^qT_j) \Leftrightarrow (\iota[{}^pT_i] \equiv \iota[{}^qT_j]) . \tag{1}$$

- ${}^pT_i$ is a *Specialization* (or hyponym) of ${}^qT_j$ if and only if the conjunction of the two definitions is the same as the definition of ${}^pT_i$ (then ${}^qT_j$ is a *Generalization* or hypernym of ${}^pT_i$):

$$({}^pT_i \le {}^qT_j) \Leftrightarrow ((\iota[{}^pT_i] \wedge \iota[{}^qT_j]) \equiv \iota[{}^pT_i]) . \tag{2}$$

- ${}^pT_i$ is *Overlapping with* ${}^qT_j$ if and only if the conjunction of the two definition is not false for all possible states of the world:

$$({}^qT_j \sim {}^pT_i) \Leftrightarrow (((\iota[{}^pT_i] \wedge \iota[{}^qT_j]) \equiv \iota[T_k]) \wedge \neg(\iota[T_k] \equiv False))\ ^4. \tag{3}$$

T_k is called *conjunction* concept or *conjunction* relation here.

The matching in this phase requires both ontologies to commit to a specific ontology called *higher-level ontology* [17]. A reasoning system will use the higher-level ontology as a common reference in two ontologies for matching.

Note that this approach is not aiming at detecting or resolving mismatches between definitions in ontologies. Consequently, we do not use the term *integrating ontologies* to avoid the false impression that any of the communities should agree with or commit to the result of ontology merging. This is because members of one community should not necessarily agree with the terms defined in the ontology of the other communities.

The following example shows parts of the ontologies which are used for a sample merging process. The reasoning system finds the specialization relation between concepts described in Tables 1-3, as shown in Fig. 2[5].

4. If T_k can be proven to be false in all possible worlds then the two intensional definitions are *disjoint*.

Table 1: Higher level ontology for both ontologies p and q

```
(defconcept Person (?p)
:documentation "An individual is a Person if and only if they
are identified by a Social_Security_Number."
:<<=>>   (exists (?x Social_Security_Number)
                              (is-identified-by ?p ?x)))
(deffunction identifies ((?x Social_Security_Number))
                          :-> (?p Person))
(deffunction is-identified-by ((?p Person))
                          :-> (?x Social_Security_Number)
:<<=>>   (identifies ?x ?p))
(defrelation makes ((?p Person)(?q Quantity)))
(defconcept Quantity (?q)
:<<=>>   (and (exists (?u Unit) (= (measured-in ?q) ?u))
              (exists (?v Number) (= (value-of ?q) ?v)))
```

Table 2: Part of ontology p

```
(defconcept Employee(?x)
:documentation "An Employee is hired by the University of
Zurich; and if an individual is an Employee then:
- they are identified by a Social_Security_Number;
- they earn a salary; and
- their age are greater than 14."
:<<=>>   (is-hired-by ?x university_of_zurich)
:=>      (and     (exists (?y Social_Security_Number)
                          (= (is-identified-by ?x) ?y))
                  (exists (?z Salary) (= (earns ?x) ?z))
                  (and   (> (value-of (age ?x)) 14)
                         (= (measured-in (age ?x)) year))))
(deffunction earns ((?x Employee)) :-> (?y Quantity)
:=>      (makes ?x ?y))
(defconcept Salary (?z Quantity)
:documentation "Salary is a type of Quantity. A Quantity is
a Salary if and only if a Employee is paid by the Quantity."
:<<=>>   (exists (?x Employee) (earns ?x ?z)))
(defconcept Faculty (?x Employee)
:documentation "Faculty is an Employee who teaches at least
one Course."
:<<=>>   (exists (?y Course) (teaches ?x ?y)))
(defconcept Professor (?x Faculty))
(defconcept Lecturer (?x Faculty))
(assert (forall (?x Professor) (not (Lecturer ?x))))
```

Table 3: Part of ontology q

```
(defconcept Citizen (?c Person))
```

5. Note that Fig. 2, 3 or 4 are only simplified illustrations of ontologies as a means for further discussion. Moreover, even the presented logical axioms are only build to show the capability of this approach.

```
(defconcept Alien (?a Person))
(assert (forall (?x Citizen) (not (Alien ?x))))
(defrelation is-paid ((?q Quantity)(?p Person))
:<<=>>  (makes ?p ?q))
(defconcept Pay (?w Quantity)
:documentation "Pay is a type of Quantity. If and only if a
Quantity is paid to a Person then it is a Pay."
:<<=>>  (exists (?x Person) (is-paid ?w ?x))
```

Detection of the similarity relations is a major task of a reasoning system. When given the above intensional definitions, a reasoning system (this example is implemented using PowerLoom [13]) can detect the following similarities (also shown in Fig. 4):

- "Employee" defined in ontology q is a specialization of "Person";
- "Salary" is a specialization of "Pay"; and
- "earns" is a specialization of "makes".

5 Global Schema Generation

We now show how two schemas (S_{p1} and S_{q1}) based on two different ontologies (p and q) can be integrated into a global schema (S_G). Schema integration is done in two main phases: global class derivation and global attribute derivation. In the first phase the class hierarchies are generated, and in the second phase classes are enhanced by attributes.

5.1 Class Integration

All the classes in the local schemas must be based on concept definitions in the community's formal ontology. In other words, the names of all schema elements used in schema definitions are uniquely referring to definition in the formal ontology of the community[6]. As an example, class "Resident" in schema S_{q1} (Fig. 3) is based on the term "Person" defined in a formal ontology p. We show this link to a term in ontology p with τ_p — e.g.,

$$\tau_q\text{: } (S_{q1}.\text{Resident}) \rightarrow \text{“Person” or } \tau_p\text{: } (S_{p1}.\text{Prof}) \rightarrow \text{“Professor”}. \quad (4)$$

"τ" returns only one term in the respective ontology. If the database designer does not link a schema element to a term in the ontology the integration process will not be able to relate it to other schema elements in other schemas.

The global class derivation is performed as follows:

- For every class in the local schema we create a class in the global schema. This guarantees that every class in the local schema is represented by a class in the global schema, which is important for the data mapping phase. The creation of classes is done top-down by starting from superclasses in the class hierarchy of the lo-

6. Whether the term definition already exists in the formal ontology or it is added during database design; or if the links are established just before the integration process or earlier are not in the scope of this paper. We take them for granted here and focus on the integration process.

cal schemas. The subclass relation with the existing superclasses in the global schema is established while a subclass is added.

- During the insertion of the classes, if an *equal* concept has already been represented by another class in the global schema, a new class is not added. That is, to add a new class such as "Resident" to the global schema the following should hold:

$$\forall\, c \in S_G : \neg[\tau_q(S_{q1}.\text{Resident}) = \tau_p(S_G.c)]. \tag{5}$$

For example, the class "Resident" should not be added if a class based on the term "Person" is already present in the global schema. In this case, only an alias name "Resident" is stored for the existing class (this is needed during both global attribute generation and the data mapping phases).

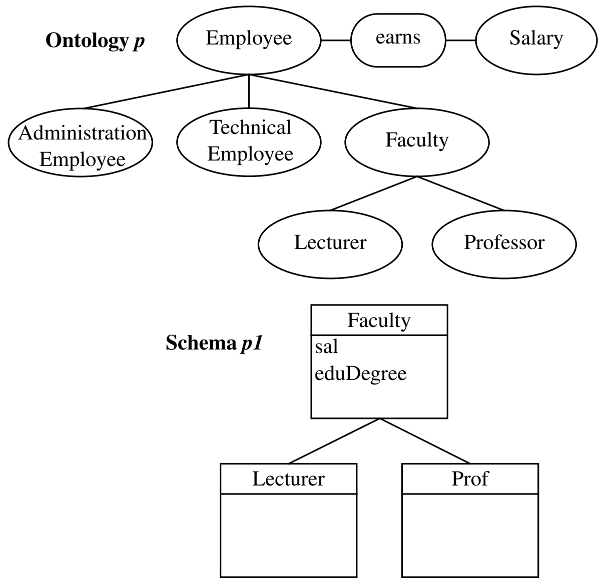


Fig. 2. Schema *p1* and ontology *p*.

- The specialization relation of the concepts in the merged ontology is also reflected as a subclass relation in the global schema. Therefore, we establish a subclass relation with every class based on a generalized concept of the current class—with the condition that the subclass relation cannot be inferred from the class hierarchy in S_G. As an example, the class "$S_G.S_{p1}$.Faculty" is defined as a subclass of "$S_G.S_{p1}$.Resident" considering that the following holds according to the merged-ontology:

$$\tau_p(S_G.\text{Faculty}) \leq \tau_q(S_G.\text{Resident}). \tag{6}$$

- New classes based on the generated conjunction concepts (during merging ontologies) may be added to the global schema. As an example, if two classes in the global schema are based on two conjunction concepts, such as:

$$\tau_p(S_G.\text{Lecturer}) \sim \tau_p(S_G.\text{Student}), \tag{7}$$

then a class based on conjunction concept may be added to the global schema. In such a case a class based on the conjunction concept (e.g., teaching assistant) can be added to the global schema. The subclass relation must be established with both classes (multiple inheritance). However, such cases of the conjunction similarity relation happen very often during the merging process, although many of them are not relevant to applications — e.g., foreign lecturer, native lecturer, foreign professor, etc. Therefore, there is a need for supervision at this point to revise these cases and decide about the necessity of creating such classes. (This task can be done also during merging ontologies, if affecting the merged-ontology by the application view is not an issue).

- If two classes refer to two overlapping or disjoint concepts, while the corresponding concepts have a common superconcept, a class based on the common superconcept is defined in the global schema. As an example, if the class "S_{q1}.Resident" did not exist in schema p1 in Fig. 3 (the schema contained only two classes foreigner and citizen), a class based on the concept "Person" will be added to the global schema.

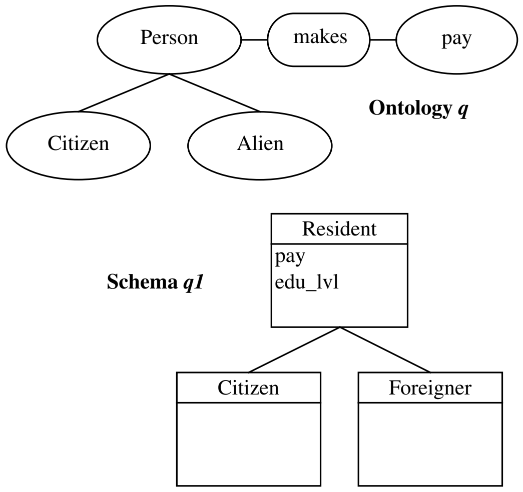


Fig. 3. Schema *q1* and ontology *q*

5.2 Filling Classes with Attributes

All attributes in the schemas represent binary relations (either by pointing to another instance of a class or by taking a value of a primitive type such as string or integer value). Since we consider all classes as being based on concepts, attributes must be based on binary relation definitions in the respective ontology — e.g.,

$$\tau_p\text{: } (S_{p1}.\text{Faculty.sal}) \rightarrow \text{“earns” or } \tau_q\text{: } (S_{q1}.\text{Faculty.pay}) \rightarrow \text{“makes”}. \quad (8)$$

This means that an attribute name in a local schema is linked to a binary relation definition in the ontology. Note that this does not imply every binary relation should be represented in the schema definition—e.g., social security number is not linked to any attribute in any of the classes S_{q1}.Resident or S_{p1}.Employee in schema definitions. The class of the attribute should be based on a concept definition which is not disjoint from the concept in the domain of its binary relation. As an example, the domain of $\tau_p(S_{p1}$.Faculty.sal) must not be disjoint from $\tau_p(S_{p1}$.Faculty). This constraint ensures that the τ_q mapping and schema definitions comply (or commit) to the community's ontology.

During the derivation of attributes for each class in a local schema, we define an attribute in the respective class in the global schema. This confirms that each attribute in the local schemas has a counterpart in the global schema. However, an *equal* relation might have already been represented by another attribute in the same global class. For example, before we add a new attribute "pay" to the class "Resident" in the global schema, we check the following:

$$\forall\, a \in S_G.\text{Resident}: \neg[\tau_q(S_{q1}.\text{Resident.pay}) = \tau_p(S_G.\text{Resident}.a)]. \tag{9}$$

If this constraint yields false, we consider the two attributes as semantically equal and keep an equality link between them for the data mapping phase. Unlike the case of classes where we only keep an alias name, here we maintain both attribute definitions in the class. The main reason is that the equality link does not indicate the similarity in the data type, unit, structure, etc. of the value of the attribute. We will deal with this matter in the data mapping phase.

The same is done in the case attributes are based on relations in specialization similarity. This case also should only be detected in this phase and will be used during data mapping. As an example, while defining attribute "sal" for class "Employee", the following similarity was detected and will be kept for the data mapping phase:

$$\tau_p(S_G.\text{Employee.sal}) \le \tau_q(S_G.\text{Employee.pay}). \tag{10}$$

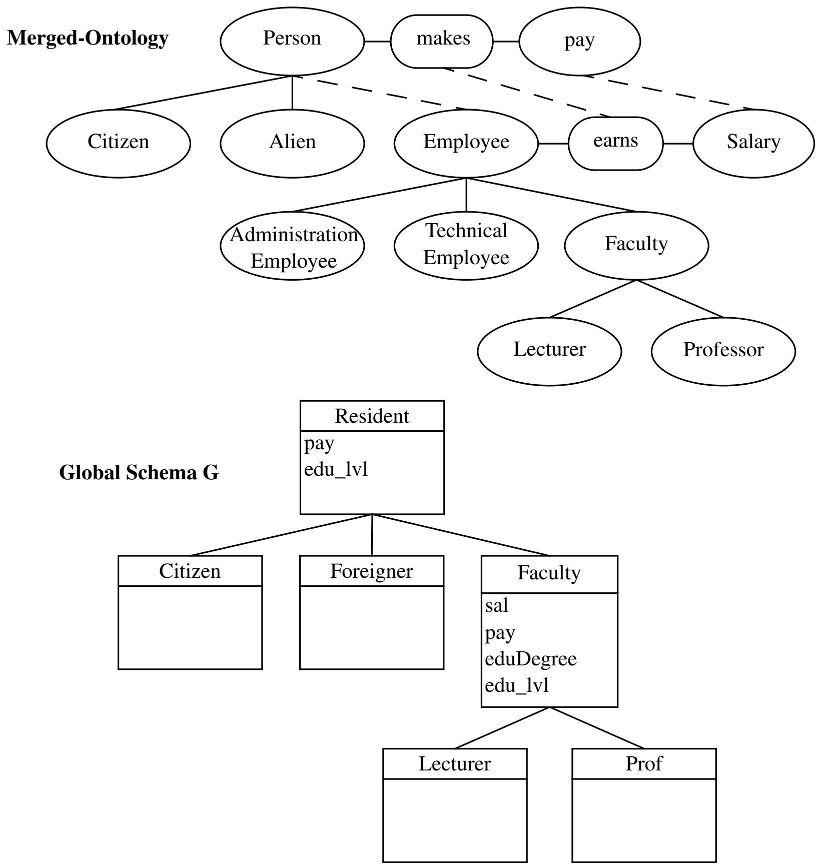


Fig. 4. Result of merging parts of formal ontologies by finding a specialization similarity and creation of class hierarchy based on merged ontology.

We did not find the case of attributes based on conjunction relations relevant to our work. The detection of such cases is not problematic, though. (As an example one can imagine two attributes S_{q1}.Resident.father and S_{p1}.Employee.colleague.)

In general, relation definitions in ontologies can not only relate instances of two concepts, they additionally may relate two concepts; or instances of a concept with subconcepts of another. An example of this case is shown in Fig. 5. In such a case an attribute based on the relation is not filled in with a value (e.g., a number or a character string) or an instance of a concept, but with a code that represents a set of individuals or a range of values (e.g., "M.Sc. degree" in Fig. 5 or "tall" for an attribute height).

6 Data Mapping

The generated global schema can be used for the integration of two databases instantiating the schemas S_{p1} and S_{q1}. The instances of classes in the local databases are mapped to those of the global schema and vice versa. This mapping of instances is straightforward and relies on the information kept during the integration process. Afterwards, a set of operations performs the mapping of the data at the global schema**.**

Classes in two schemas referring to the same definition are mapped to the same class by means of alias names. In case classes are in a specialization relation in the global schema, all instances of the subclass can be mapped to the superclass, but not the other direction. For the mapping from a superclass to its subclass, we need a *classification criterion.* As an example, instances of the subclass "S_G.Faculty" are mapped to instances of the superclass "S_G.Resident". However, in order to map instances of the superclass "S_G.Resident" to the subclass "S_G.Faculty", these instances should satisfy a classification criterion. By referring to the definitions in section 4, one can see that if an instance of "S_G.Resident" is an "Employee" and "teaches" a "Course" it can be mapped to class "S_G.Faculty". A reasoning system finds such criteria by referring to the intensional definitions. The reasoning system can classify the instances during the data mapping in the global schema.

A problem occurs during data mapping when two instances (from two databases) are classified under one class in the global schema, while they represent the same individual in the domain. To deal with this problem, we need an *identification criterion* to recognize if two objects in the underlying databases represent different individuals. This criterion must be present in both local class definitions — such as social_security_number in our example. The identification criterion may not necessarily be the primary key in one or both systems, though.

Attributes of a class in a local schema are mapped directly to their counterpart in the global schema. A set of rules map attributes in the global schema. In case two attributes are linked by equal similarity, the attribute values will be mapped mutually. In case two attributes are linked by a specialization similarity, the value of the specialized attribute can be mapped to the generalized one, but not the other way round. By looking at the definitions in section 4, one can see that every "earns" relation is a "makes" relation but not the other way round. Yet, one can argue that in this example (Fig. 3), all values of attribute "S_G.Resident.pay" can be mapped to "S_G.Faculty.sal" and that this is what happens during data mapping. However, in general there are cases in which at-

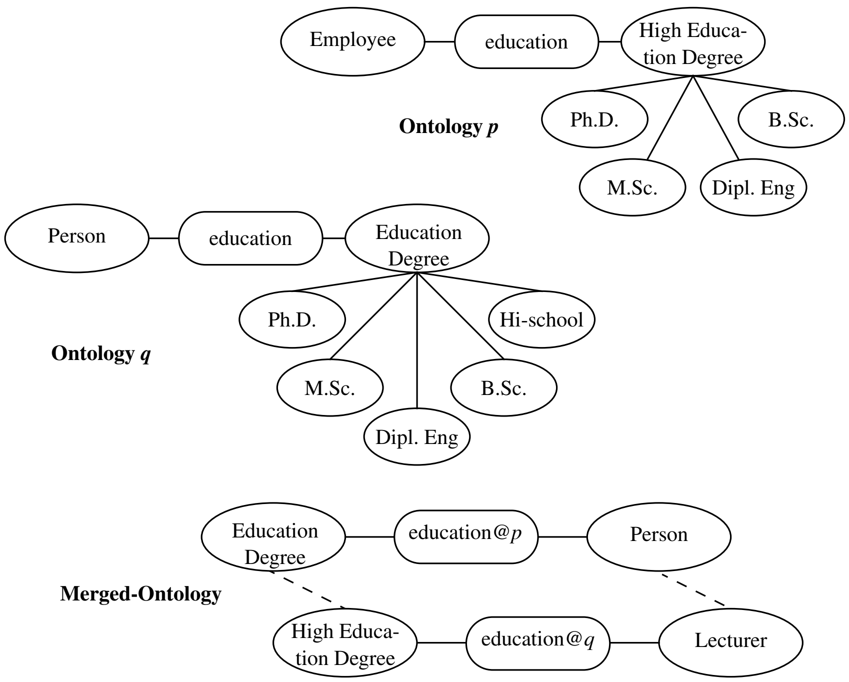


Fig. 5. Example of relations between instances of a concept and subconcepts of another.

tribute values should be mapped by considering a classification criterion. (Attributes referring to sibling and sister relations are simple examples of this case.)

An attribute mapping often requires a data conversion process (e.g., integer to real). This is because during the integration we do not employ any knowledge about the data types. There is often need for further processes for data mapping—a simple example is concersion of units. If the changes are not due to the structural differences, a detailed ontology can eventually help in some cases (such as unit conversion).

Furthermore, we only consider mapping of one element of the schema to only one other element. More complicated data mappings in which a global class is an aggregation of classes in the local schemas or an attribute may be the result of calculation over many attributes from local schemas (as presented in [16]) are not treated here. However, the focus of this work has been to solve semantic problems in terms of finding similarities between classes and attributes and finding mappings between them, but not correspondence of data values.

7 Conclusion

In this paper, we present an approach for generating global schemas. This solution uses formal ontologies as a basis for the integration and for resolving heterogeneity problems during the integration of local schemas. We only use the intensional definitions in formal ontologies, that is, we do not use the matching of the terms used to name schema elements. This approach also does not rely on or consider the structures expressed by the schema.

The quality of the formal ontologies plays an important role here. There are two important quality measures for the success of this approach:

- accordance of an ontology with the *conceptualization* (defined in [7]) of the community and
- details of explicit specifications of implicit assumptions in the community are important during building ontologies.

Another factor that plays an important role in the success of this approach is the commitment of the schema definitions to the community's formal ontology. There are constraints that should apply to the schema definitions. One such constraint is discussed in the beginning of section 5.2. Such constraints should be clearly specified.

Building a higher level ontology that many communities agree upon is a difficult task. This makes ontologies expensive to build. However, it is a price worth paying to avoid semantic conflicts that can be even more expensive. Furthermore, formal ontologies are long term assets that will remain independent of the application systems. As ontologies are becoming more popular they can be used for other purposes other than database integration.

Although we considered the integration of classes and attributes, methods are not discussed in this paper. Methods are often considered as parametric attributes based on the relation definitions of arity higher than two in a formal ontology. However, we can only handle those methods that represent an *action* to be based on a definition in the formal ontology (actions change the states of the world). That is, methods that are results of implementation (such as triggers) may not have a counterpart in the communities' ontology. This is a general discussion that may apply rarely to attributes which are only used for implementational reasons (such as object id) and do not have a counterpart in the ontology.

Acknowledgments

We would like to thank, Ruxandra Domenig for her comments on the early draft paper, Sabine Timpf for the fruitful discussions and Ross Purves for his help to improve the presentation of the paper.

References

1. S. Bergamaschi, S. Castano, S. De Capitani di Vimercati, S. Montanari, and M. Vincini. An intelligent approach to information integration. In Nicola Guarino, editor, *Formal Ontology in Information Systems*, pages 253–267. IOS Press, 1998.
2. Y. A. Bishr, H. Pundt, W. Kuhn, and M. Radwan. Probing the concept of information communities—a first step toward semantic interoperability. In M. Goodchild, Max Egenhofer, R. Fegeas, and C. Kottman, editors, *Interoperating Geographic Information Systems*, pages 55–69. Kluwer Academic, 1999.
3. Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems*. Addison-Wesley, third edition, 2000.

4. Dieter Fensel, J¸rgen Angele, Stefan Decker, Michael Erdmann, Hans-Peter Schnurr, Steffen Staab, Rudi Studer, and Andreas Witt. On2broker: Semantic-based access to information sources at the www. In *Proceedings of the World Conference on the WWW and Internet (WebNet 99)*, pages 366–371, October 1999. ftp://ftp.aifb.uni-karlsruhe.de/pub/mike/dfe/paper/webnet.pdf.
5. Manuel Garcia-Solaco, Felix Saltor, and Malu Castellanos. Semantic heterogeneity in multidatabase systems. In Omran A. Bukhres and Ahmed K. Elmagarmid, editors, *Object-oriented Multidatabase Systems: A Solution for Advanced Applications*, chapter 5, pages 129–202. Printice-Hall, 1996.
6. Cheng Hian Goh, Stephane Bressan, Stuart Madnick, and Michael Siegel. Context interchange: New features and formalisms for the intelligent integration of information. *ACM Transaction on Information Systems*, 17(3):270–290, 1999.
7. Nicola Guarino. Formal ontology and information systems. In Nicola Guarino, editor, *Formal Ontology in Information Systems, Proceedings of FOIS'98*, pages 3–17, Trento, Italy, June 1998. IOS Press, Amsterdam.
8. Farshad Hakimpour and Andreas Geppert. Resolving semantic heterogeneity in schema integration: An ontology base approach. In Chris Welty and Barry Smith, editors, *Proceedings of International conference on Formal Ontologies in Information Systems FOIS'01*. ACM Press, October 2001.
9. Jeff Heflin and James Hendler. Semantic interoperability on the web. In *Extreme Markup Languages 2000*, 2000. http://www.cs.umd.edu/projects/plus/SHOE/pubs/extreme2000.pdf.
10. Won Kim, Injun Choi, Sunit Gala, and Mark Scheevel. On resolving schematic heterogeneity in multidatabase systems. *Distributed and Parallel Databases*, 1(3):251–277, July 1993.
11. James A. Larson, Shamkant B. Navathe, and Ramez Elmasri. A theory of attribute equivalence in database with application to schema integration. *IEEE Transactions on Software Engineering*, 15(4):449–463, April 1989.
12. Pericles Loucopoulos. Conceptual modelling. In Pericles Loucopoulos and Robert Zicari, editors, *Conceptual Modelling, Databases, and CASE: an Integrated View of Information system development*, chapter Introduction, pages 1–26. John Wiley & Sons, Inc., 1992.
13. Robert M. MacGregor, Hans Chalupsky, and Eric R. Melz. *PowerLoom Manual*. University of Southern California, http://www.isi.edu/isd/LOOM/PowerLoom/documentation/manual.pdf, November 1997.
14. Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with cupid. In Peter M. G. Apers, Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Kotagiri Ramamohanarao, and Richard T. Snodgrass, editors, *VLDB 2001, Proceedings of 27th International Conference on Very Large Databases, September 11-14, 2001, Roma, Italy*, pages 49–58. Morgan Kaufmann, September 2001.
15. E. Mena, V. Kashyap, A. Illarramendi, and A. Sheth. Domain specific ontologies for semantic information brokering on the global information infrastructure. In Nicola Guarino, editor, *Formal Ontology in Information Systems*. IOS press, 1998.
16. Renée J. Miller, Laura M. Haas, and Mauricio A. Hernández. Schema mapping as query discovery. In Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang, editors, *VLDB 2000, Proceedings of 26th International Conference on Large Data Bases, September 10-14, 2000, Cairo, EgyptVery*, pages 77–88. Morgan Kaufmann, 2000.

17. Pepijn R. S. Visser and Zhan Cui. Heterogeneous ontology structures for distributed architectures. In *ECAI-98 Workshop on Applications of Ontologies and Problem-solving Methods*, pages 112–119, 1998.
18. Pepijn R.S. Visser, Dean M. Jones, M.D. Beer, T.J.M. Bench-Capon, B.M. Diaz, and M.J.R. Shave. Resolving ontological heterogeneity in the kraft project. In *10th International Conference and Workshop on Database and Expert Systems Applications DEXA'99*. University of Florence, Italy, August 1999.

Automatically Extracting Ontologically Specified Data from HTML Tables of Unknown Structure

David W. Embley[1], Cui Tao[1], and Stephen W. Liddle[2]

[1] Department of Computer Science
{embley,ctao}@cs.byu.edu
[2] School of Accountancy and Information Systems
Brigham Young University, Provo, Utah 84602, U.S.A.
liddle@byu.edu

Abstract. Data on the Web in HTML tables is mostly structured, but we usually do not know the structure in advance. Thus, we cannot directly query for data of interest. We propose a solution to this problem based on document-independent extraction ontologies. The solution entails elements of table understanding, data integration, and wrapper creation. Table understanding allows us to recognize attributes and values, pair attributes with values, and form records. Data-integration techniques allow us to match source records with a target schema. Ontologically specified wrappers allow us to extract data from source records into a target schema. Experimental results show that we can successfully map data of interest from source HTML tables with unknown structure to a given target database schema. We can thus "directly" query source data with unknown structure through a known target schema.

1 Introduction

The schema-mapping problem for heterogeneous data integration is hard and is worthy of study in its own right [MBR01]. The problem is to find a *semantic correspondence* between one or more *source schemas* and a *target schema* [DDH01]. In its simplest form the semantic correspondence is a set of *mapping elements*, each of which binds an attribute in a source schema to an attribute in a target schema or binds a relationship among attributes in a source schema to a relationship among attributes in a target schema. Such simplicity, however, is rarely sufficient, and researchers thus use queries over source schemas to form attributes and relationships among attributes to bind with target attributes and attribute relationships [MHH00, BE02]. Furthermore, as we shall see in this paper, we may also need queries beyond those normally defined for database systems. Thus, we more generally define the *semantic correspondence for a target attribute* as any named or unnamed set of values that is constructed from source elements, and we define the *semantic correspondence for a target n-ary relationship* among attributes as any named or unnamed set of n-tuples over constructed value sets. The sets of values for target attributes may be constructed in any way, e.g. directly taken from source values, computed over source values, or manufactured from source attribute names or from strings in table headers or footers.

Car	*Year*	*Make*	*Model*	*Mileage*	*Price*	*PhoneNr*
0001	1999	Ford	Mustang	42,130	$10,988	405-936-8666
0002	1998	Ford	Taurus	63,168	$7,988	405-936-8666
				...		
0011	1992	ACURA	legend		$9500	
0012	2000	AUDI	A4		$34,500	
0013	1985	BMW	325e		$2700.00	
				...		

Car	*Feature*
0001	Yellow
0001	Power Steering
	...
0002	Black
0002	Power Brakes
	...
0011	grey
0011	Auto
	...

Fig. 1. Sample Tables for Target Schema

A considerable amount of data is available on the Web in the form of HTML tables, and it is often the case that we wish to integrate sources encoded in various HTML table formats. Thus we limit our current investigation to sources that are HTML tables found on the Web. Our target schema is an augmented conceptual-model instance (described in Section 2).

As a running example, we use car advertisements, which are plentiful on the Web and which often present their information in tables. Suppose, for example, that we are interested in viewing and querying Web car ads through the target database in Figure 1, whose schema is

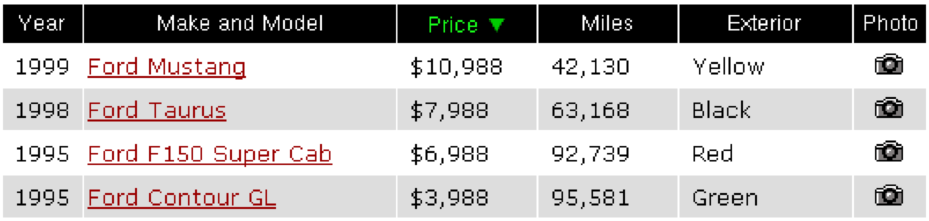
$$\{Car,\ Year,\ Make,\ Model,\ Mileage,\ Price,\ PhoneNr\},\ \{Car,\ Feature\}.$$

Figures 2 [Bob02], 3 [Bob02], and 4 [Aut01] show some potential source tables. The data in the tables in Figure 1 is a small part of the data that can be extracted from Figures 2, 3, and 4.

1.1 Matching Problems

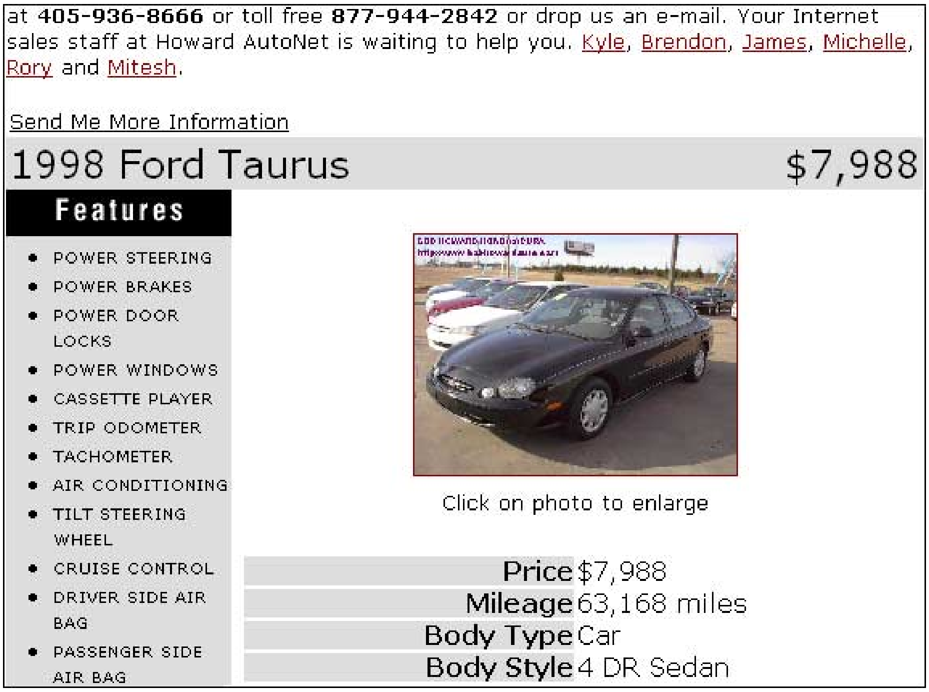
It is easy to see that *Year* in the source table of Figure 2 and *Year* in the table of Figure 4 map to *Year* in the target table of Figure 1. But it is harder to see that both *Exterior* in Figure 2 and *Colour* in Figure 4 map to *Feature* in Figure 1 and even harder to see that the attributes *Auto*, *Air Cond.*, *AM/FM*, and *CD* in Figure 4 should map as values for *Feature*, but only for "Yes" values.

There are many *matching problems* that arise when trying to match source HTML tables with a target schema. For example, *Make* and *Model* are separate attributes in Figure 1 but are merged as one attribute in Figure 2 (*merged attributes*). *Exterior* in Figure 2 and *Colour* in Figure 4 contain colors; but colors in the target are a special kind of *Feature* and thus the set of colors in Figures 2 and 4 is a subset of the feature values we want for Figure 1 (*subsets*). *Mileage* in Figure 1 and *Miles* in Figure 2 have the same meaning, but the attribute names are not the same (*synonyms*).



Year	Make and Model	Price ▼	Miles	Exterior	Photo
1999	Ford Mustang	$10,988	42,130	Yellow	📷
1998	Ford Taurus	$7,988	63,168	Black	📷
1995	Ford F150 Super Cab	$6,988	92,739	Red	📷
1995	Ford Contour GL	$3,988	95,581	Green	📷

Fig. 2. Excerpt of Table from `www.bobhowardhonda.com`



at **405-936-8666** or toll free **877-944-2842** or drop us an e-mail. Your Internet sales staff at Howard AutoNet is waiting to help you. Kyle, Brendon, James, Michelle, Rory and Mitesh.

Send Me More Information

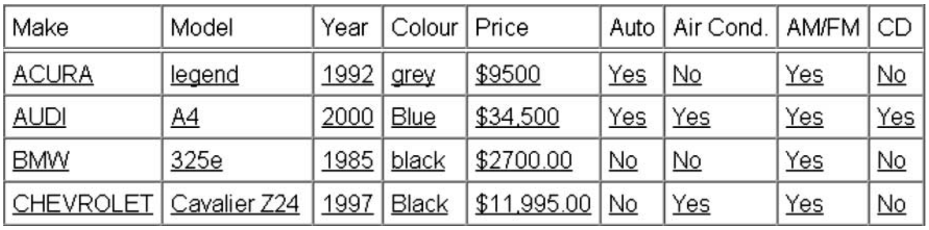
1998 Ford Taurus $7,988

Features

- POWER STEERING
- POWER BRAKES
- POWER DOOR LOCKS
- POWER WINDOWS
- CASSETTE PLAYER
- TRIP ODOMETER
- TACHOMETER
- AIR CONDITIONING
- TILT STEERING WHEEL
- CRUISE CONTROL
- DRIVER SIDE AIR BAG
- PASSENGER SIDE AIR BAG

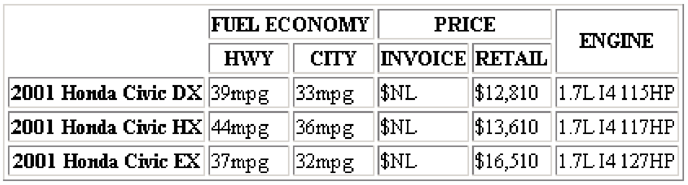
Price	$7,988
Mileage	63,168 miles
Body Type	Car
Body Style	4 DR Sedan

Fig. 3. Excerpt of Subtable Linked from `www.bobhowardhonda.com`



Make	Model	Year	Colour	Price	Auto	Air Cond.	AM/FM	CD
ACURA	legend	1992	grey	$9500	Yes	No	Yes	No
AUDI	A4	2000	Blue	$34,500	Yes	Yes	Yes	Yes
BMW	325e	1985	black	$2700.00	No	No	Yes	No
CHEVROLET	Cavalier Z24	1997	Black	$11,995.00	No	Yes	Yes	No

Fig. 4. Excerpt of Table from `autoscanada.com`



	FUEL ECONOMY		PRICE		ENGINE
	HWY	CITY	INVOICE	RETAIL	
2001 Honda Civic DX	39mpg	33mpg	$NL	$12,810	1.7L I4 115HP
2001 Honda Civic HX	44mpg	36mpg	$NL	$13,610	1.7L I4 117HP
2001 Honda Civic EX	37mpg	32mpg	$NL	$16,510	1.7L I4 127HP

Fig. 5. Table with a More Complex Attribute Structure

There are more problems. Information might be hidden behind a link (clicking on *Ford Taurus* in Figure 2 yields the information in Figure 3). HTML lists (like the feature list in Figure 3) are conceptually similar to one-dimensional tables and should be treated accordingly. A significant challenge for our study is that many HTML tables are used for layout rather than data structuring purposes. Often HTML tables are nested. For example, the table beginning with *Price* then *Mileage* in Figure 3 is nested within the larger layout table. Furthermore, the nested table in Figure 3 has its attributes in the left column, rather than in the top row, which is more typical. The table in Figure 5 has compond attributes (e.g. *CITY* and *HWY* are subdivisions of *FUEL ECONOMY*). Moreover, attributes may appear both in the top row and in the left column and may, at the same time, be compound. Sometimes it is hard to tell: are the *Honda Civic* strings in Figure 5 attributes or values with implicit attributes?

Other matching problems include: missing information (e.g. there are no phone numbers in Figure 2); extra information present (e.g. photographs in Figure 2); factored information (e.g. the make *Ford* could have been factored out of the table in Figure 2); multiple occurrences of the same attribute-value pair (e.g. the same price appears three times for the Ford Taurus in Figures 2 and 3); multiple values where one is expected (e.g. the tables of Figure 1 expect one contact phone number but Figure 3 shows two different phone numbers); and attributes appearing as values (e.g. in Figure 4 the features *Auto*, *Air Cond.*, *AM/FM*, and *CD* are all attributes rather than values). This list is not exhaustive. It certainly illustrates, however, that there are many problems to solve.

1.2 Matching Solutions—Our Contribution

Rather than directly try to find mappings from source schemas to target schemas as suggested in [MHH00, DDH01, MBR01, BE02], our contribution in this paper is to argue for a different approach, show that it works, and explain why it may be superior. The approach requires table understanding [LN99b] and extraction ontologies [ECJ$^+$99] and results in establishing a semantic correspondence between a source schema and a target schema. Our approach includes four steps.

1. *Form Attribute-Value Pairs.* Using table understanding techniques, we determine, for example, that ⟨*Year*: 1999⟩ and ⟨*Exterior*: *Yellow*⟩ are two of the attribute-value pairs for the first record in Figure 2.

2. *Adjust Attribute-Value Pairs.* We convert, for example, the recognized attribute-value pair $\langle$*CD*: *Yes*$\rangle$ in row two of Figure 4 to *CD*, meaning that this car has a CD player, and the the pair $\langle$*CD*: *No*$\rangle$ in row three to a null string, meaning that this car has no CD player.
3. *Perform Extraction.* The extraction ontology recognizes, for example, that *42,130* in the first row of Figure 2 should be extracted as the *Mileage* for the first car in Figure 1 and that the first part of *Ford Mustang* should be extracted as the *Make* while the second part should be the *Model*.
4. *Infer Mappings.* Given the recognized extraction (which, by the way, need not be 100%), the system can infer the general mapping from source to target. Based on the extraction examples above, the system would know, for example, that the *Miles* values in Figure 2 map to *Mileage* in the target (Figure 1), that the first part of the *Make and Model* strings map to *Make*, and that the remaining characters in the strings map to *Model*.

We present the details of our contribution in the remainder of the paper as follows. Section 2 describes extraction ontologies. Section 3 then provides the details for each of the four steps of our approach. In Section 4, we report the results of an experiment we conducted in which we derived source-target mappings for several dozen HTML tables for car advertisements, which we found on the Web. We summarize and point to future research work in Section 5.

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 ads. The conceptual-model instance includes objects, relationships, constraints, and descriptions of strings for lexical objects. When we apply an extraction ontology to a Web page, the ontology identifies objects and relationships and associates them with named object sets and relationship sets in the ontology's conceptual-model instance and thus wraps the page so that it is understandable in terms of the schema implicitly specified in the conceptual-model instance. The hard part of writing a wrapper for extraction is to make it robust so that it works for all sites, including sites not in existence at the time the wrapper is written and sites that change their layout and content after the wrapper is written. Wrappers based on extraction ontologies are robust. Robust wrappers are critical to our approach: without them, we may have to create (by hand or at best semiautomatically) a wrapper for every new table encountered; with them, the approach can be fully automatic.

Figure 6 shows part of our car-ads application ontology, including object and relationship sets and cardinality constraints (Lines 1-4) and a few lines of the *data frames* (Lines 5-11) that describe lexical objects (*constants*) and *keywords* that signal the presence of a particular object or relationship. We use Perl syntax for regular expressions. When we apply an extraction ontology to extract data, we first find record boundaries and divide a document into chunks of information, one for each record. We then apply regular expressions to each record, one at a time, to extract a list of constants and keywords. To this list, we

```
Car [-> object];                Car [0:1] has Year [1:*];
Car [0:1] has Make [1:*];       Car [0:1] has Model [1:*];
Car [0:1] has Mileage [1:*];    Car [0:*] has Feature [1:*];
Car [0:1] has Price [1:*];      PhoneNr [1:*] is for Car [0:1];
Year matches [4]
     constant {extract "\d{2}";   context "\b'[4-9]\d\b";
               substitute "^" -> "19"; },   ...
Mileage matches [8]
     keyword "\bmiles\b", "\bmi\.", "\bmi\b",
             "\bmileage\b", "\bodometer\b";
...
```

Fig. 6. Car-Ads Extraction Ontology (Partial)

apply heuristics that form structured records according to the constraints given in the extraction ontology. See [ECJ+99] for full details.

3 Derivation of Source-to-Target Schema Mappings

We assume that each tuple in the top-level table corresponds to a primary object of interest. One consequence of this assumption is that we can simply generate an object identifier for each of these objects. Indeed, this is how we obtain the values under the attribute *Car* in Figure 1. Another consequence of this assumption is that we can easily group source information into record chunks, one for each object of interest. The information chunk for the *1998 Ford Taurus* in Figure 2, for example, is the second tuple in the table plus all the information in Figure 3. Having record chunks allows us to more easily build the atomic relationships between an object of interest and its associated data once we find the semantic correspondence for each target attribute. Hence, we are able to reduce the problem of finding a semantic correspondence to just finding the correspondence for each target attribute A, which we defined earlier as the problem of finding the set of values constructed from source elements that corresponds to A.

We accomplish this objective using a "back-door" approach. Instead of directly searching for a mapping that associates each target attribute A with a value set in a source, we use our extraction ontology to search for values in the source that are likely to be found in the value set for A. Then, from the pattern of values we find, we infer what the mapping must be. This approach more easily allows us to recognize some of the unusual indirect mappings we are likely to encounter such as merged attributes, attributes encoded as values, or factored information, as discussed and illustrated in the introduction. The following four subsections correspond to the four steps of our proposed approach.

3.1 Form Attribute-Value Pairs

The table understanding problem takes as input a table (for our work here, an HTML table) and produces standard records as output. Each record produced is a set of attribute-value pairs. A successful table-understanding system, for example, would produce the first record of the table in Figure 4 as

$$\{\langle \textit{Make}\text{: } \textit{ACURA}\rangle, \langle \textit{Model}\text{: } \textit{legend}\rangle, \langle \textit{Year}\text{: } 1992\rangle, \langle \textit{Colour}\text{: } \textit{grey}\rangle,$$
$$\langle \textit{Price}\text{: } \$9500\rangle, \langle \textit{Auto}\text{: } \textit{Yes}\rangle, \langle \textit{Air Cond.}\text{: } \textit{No}\rangle, \langle \textit{AM/FM}\text{: } \textit{Yes}\rangle, \langle \textit{CD}\text{: } \textit{No}\rangle\}.$$

The hard part of table understanding is recognizing which cells contain attributes and which contain values and then recognizing which attributes go with which values. If we could depend on the attributes always being in column headers with one attribute for every column, attribute/value identification would be much simpler; but this is not always the case even though it is common. Moreover, since HTML table creators do not always use the tags $\langle th\rangle$ and $\langle td\rangle$ as they were intended, we cannot use HTML tags to solve this problem. [LN99a] has a solution for a common subclass of HTML tables, but not for all HTML tables.

Once we have identified attributes, we can immediately associate each cell in the grid layout of a table with its attribute. If a cell is not empty, we also immediately have a value for the attribute and thus an attribute-value pair. If a cell is empty, however, we must infer whether the table has a value based on internal factoring or whether there is no value. Figure 7a shows an example of internal factoring. The empty *Year* cell for the *Contour GL* indicates *1995*, whereas the *Price* for the *Honda* is simply missing. We recognize internal factoring in a two-step process: (1) we detect potential factoring by observing a pattern of empty cells in a column, preferrably a leftmost column or a near-leftmost column; (2) we check to see whether adding in the value above the empty cell helps complete a record by adding a value that would otherwise be missing.

Once we recognize the factoring in a table, we can rewrite the schema as a nested schema that reflects the factoring and then unnest the table to distribute factored values and to return the schema for the nested table to an unnested schema. We denote a nested component of a schema by $(A_i, ..., A_n)^*$ where the A_i's are attribute names. In general, nested components may appear in-

Year	*Make*	*Model*	*Price*
1995	Ford	F150 Super Cab	$6,988
		Contour GL	$3,988
	ACURA	INTEGRA LS	$14,500
	Honda	Civic EX	
1994	Ford	F150	$4,488

(a)

Year	*Make*	*Model*	*Price*
1995	Ford	F150 Super Cab	$6,988
1995	Ford	Contour GL	$3,988
1995	ACURA	INTEGRA LS	$14,500
1995	Honda	Civic EX	
1994	Ford	F150	$4,488

(b)

Fig. 7. Internal Factoring in Tables

side and alongside other nested components. The nested schema that defines the internal factoring for the table in Figure 7a is *Year*, (*Make*, (*Model*, *Price*)*)*.

To unnest a table with a nested schema, we use a μ operator whose definition is based on the unnest operator in [KS91]. We write $\mu_N t$ to unnest the nested component N of nested table t. To unnest table T_a in Figure 7a, for example, we can apply $\mu_{(Make,\ Model,\ Price)^*}\mu_{(Model,\ Price)^*}T_a$, which yields the table in Figure 7b. Here, we start with *Year*, (*Make*, (*Model*, *Price*)*)*. After the first operation $\mu_{(Model,\ Price)^*}$, the schema is *Year*, (*Make*, *Model*, *Price*)* and the *Make*s have been distributed to the *Model*s, i.e. *Ford* appears in the empty cell in the *Make* column in Figure 7a in our example. Then after the second operation $\mu_{(Make,\ Model,\ Price)^*}$, which distributes the *Year*s to each empty cell in the *Year* column, we have the table in Figure 7b. Alternatively, we could have achieved the same result by applying $\mu_{(Model,\ Price)^*}\mu_{(Make,\ (Model,\ Price)^*)^*}T_a$, which first distributes the years to each make and then distributes *Year*-*Make* pairs to each *Model*. In either case, once we have resolved internal factoring with the μ operator, we immediately have the table's attribute-value pairs.

3.2 Adjust Attribute-Value Pairs

After discovering attribute-value pairs, we make some adjustments to prepare each record for recognition by the extraction ontology. We format attribute-value pairs for easy recognition by the extraction ontology. We add in linked sub-information for each record (i.e. we add the information in Figure 3 to the ordered pairs for the second record in Figure 2). If we wish to process nontext items such as icons, we could replace them with text; for example, we could replace a color-swatch icon with the name of the color. Finally, we process Boolean indicators, such as "*Yes/No*" in Figure 4, by replacing them with attribute-name values.

For our running example, the adjusted attribute-value pairs in the first record of Figure 4 become {*Make*: *ACURA*; *Model*: *legend*; *Year*: 1992; *Colour*: *grey*; *Price*: \$9500; *Auto*; *AM/FM*}. Here, the Boolean-valued attribute-value pair ⟨*Auto*: *Yes*⟩ has become *Auto*, meaning the car has an automatic transmission, and the pair ⟨*AM/FM*: *Yes*⟩ has become *AM/FM*, meaning the car has an AM/FM radio. Further, the attributes for the *No* values, *Air Cond.* and *CD* have disappeared, meaning the car has neither air conditioning nor a CD player.

When attribute names are the values we want and the values are some sort of Boolean indicator (e.g. Yes/No, True/False, 1/0, cell checked or empty), we transform the Boolean indicators into attribute-name values with the help of a β operator which we introduce here. Syntactically we write $\beta^A_{T,F}r$ where A is an attribute of relation r and T and F are respectively the Boolean indicators for the *True* value and the *False* value given as A values in r. The result of the β operator is r with the *True* values of the A column replaced by the string A and the *False* values of A replaced by the null string. As an example, consider $\beta^{Auto}_{Yes,No}\beta^{Air\ Cond.}_{Yes,No}\beta^{AM/FM}_{Yes,No}\beta^{CD}_{Yes,No}T$ which transforms the table T in Figure 4 to the table in Figure 8.

Make	*Model*	*Year*	*Colour*	*Price*	*Auto*	*Air Cond.*	*AM/FM*	*CD*
ACURA	legend	1992	grey	\$9500	Auto		AM/FM	
AUDI	A4	2000	Blue	\$34,500	Auto	Air Cond.	AM/FM	CD
BMW	325e	1985	black	\$2700.00			AM/FM	
CHEVROLET	Cavalier Z24	1997	Black	\$11,995.00		Air Cond.	AM/FM	

Fig. 8. Table in Figure 4 Transformed by the β Operator

3.3 Perform Extraction

Once we have adjusted attribute-value pairs as just discussed, we apply our extraction ontology. For our running example, the extraction for the first record in Figure 4 yields

$$\{\langle \textit{Car}: 0011\rangle, \langle \textit{Year}: 1992\rangle, \langle \textit{Make}: \textit{ACURA}\rangle, \langle \textit{Model}: \textit{legend}\rangle, \langle \textit{Mileage}: \ \rangle, \langle \textit{Price}: \$9500\rangle, \langle \textit{PhoneNr}: \ \rangle\},$$
$$\{\langle \textit{Car}: 0011\rangle, \langle \textit{Feature}: \textit{grey}\rangle\},$$
$$\{\langle \textit{Car}: 0011\rangle, \langle \textit{Feature}: \textit{Auto}\rangle\},$$
$$\{\langle \textit{Car}: 0011\rangle, \langle \textit{Feature}: \textit{AM/FM}\rangle\}.$$

We emphasize that our extraction ontology is capable of extracting from unstructured text as well as from structured text. Indeed, we can directly extract the phone numbers and features from the text, list, and horizontal table in Figure 3.

For direct extraction we introduce the ϵ operator, which is based on a given extraction ontology. We define $\epsilon_S t$ as an operator that extracts a value, or values, from unstructured or semistructured text t for object set S in the given extraction ontology O according to the extraction expression for S in O. The ϵ operator extracts a single value if S functionally depends on the object of interest x in O, and it extracts multiple values if S does not functionally depend on x. As an example, $\epsilon_{PhoneNr} P$ extracts *405-936-8666* from the unstructured text in page P in in $Figure$ 3 and returns it as the single-attribute, single-tuple, constant relation $\{\langle \textit{PhoneNr}: \textit{405-936-8666}\rangle\}$.

We can use the ϵ operator in conjunction with a natural join to add a column of constant values to a table. For example, assuming the phone number *405-936-8666* appears in page P with the table in Figure 2, which indeed it does, we could apply $\epsilon_{PhoneNr} P \bowtie T$ to add a column for *PhoneNr* to table T in Figure 2.

At this point we could take a data-warehousing approach and directly insert this extracted information into a global database as Figure 1 implies. Alternatively, instead of populating the global database, we can use this information to infer a mapping from the source to the target and extract information from sources whenever a query is posed against the global database schema.

3.4 Infer Mappings

We record the sequence of transformations produced when we form attribute-value pairs and when we adjust attribute-value pairs in preparation for extraction, and we observe the correspondence patterns obtained when we extract tuples with respect to a given target ontology. Based on this sequence of transformations and these correspondence patterns, we can produce a mapping of source information to a target ontology. As a simple example, consider mapping the table T_a in Figure 7a to the target schema for the tables in Figure 1. We first apply the μ operator to do the unnesting and obtain table T_b in Figure 7b. We then observe that objects extracted for the *Year* object set in the target come from the *Year* column in T_b. Similarly, for *Make*, *Model*, and *Price*, we also observe a direct correspondence. Hence, we can record the semantic correspondence of T_a and the target schema as the mapping $Year = \pi_{Year}\mu_{(Make,\ Model,\ Price)^*}\mu_{(Model,\ Price)^*}T_a$, $Make = \pi_{Make}\mu_{(Model,\ Price)^*}T_a$, $Model = \pi_{Model}T_a$, and $Price = \pi_{Price}T_a$.

Creating an inferred mapping has two important advantages. (1) The global view can be virtual. Since we have a formal mapping, we can translate any query applied to the global view to a query on the source, optimize it, execute it, and return the results from the source for the global query. (2) We can obtain additional values not recognized by the ontology, but which are nevertheless valid values in the source. For example, *Super Cab* may not be technically part of the model for the *1995 Ford F150* in Figure 2, and thus the ontology might not recognize it as part of the model. Nevertheless, someone declared *Super Cab* to be part of the model, and we should therefore extract it as such. Using the mapping, we extract full strings under *Model* in Figure 7a and thus we obtain *F150 Super Cab* as the model for the *1995 Ford* even though the extraction ontology may only pick up *F150* as the model. As another example, the mapping approach would obtain *all* the *Features* in the list in Figure 3 even though the ontology may not recognize all of them as features. When we use the mapping, we generalize over the structure and infer additional information not specifically recognized by the ontology.

To complete our task, we now define a few more operators. These operators, together with the ones we defined earlier, provide the complete set of operators we need for mapping all the HTML tables we have encountered.

For merged attributes we need to split values. We can divide values into smaller components with a δ operator which we introduce here. We define $\delta^{A}_{B_1,...,B_n}r$ to mean that each value v for attribute A of relation r is split into $v_1, ..., v_n$, one for each new attribute $B_1, ..., B_n$ respectively. Associated with each B_i is a procedure p_i that defines which part of v becomes v_i. In this paper we specify each procedure p_i by regular expressions similar to those defined for extraction ontologies shown in Figure 6. The result of the δ operator is r with n new attributes, B_1, ..., B_n, where the B_i value on row k is the string that results from applying p_i to the string v on row k for attribute A. As an example, consider $\delta^{Make\ and\ Model}_{Make,Model}T$, where T is the table in Figure 2, the expression associated with *Make* is `extract "\S+" context "\S+\s"` which extracts the characters of the string value up to the first space, and the expression associated with *Model* is `extract "\S.*" context "\s.+"` which extracts all the remaing

Make	*Model*
Ford	Mustang
Ford	Taurus
Ford	F150 Super Cab
Ford	Contour GL

Fig. 9. Columns Added to the Table in Figure 2 by the δ Operator

characters in the string after the first space. This operation adds the two columns in Figure 9 to the table in Figure 2.

For split attributes we need to merge values. We can gather values together and merge them with a γ operator which we introduce here. Syntactically, we write $\gamma_{B \leftarrow A_1 + \ldots + A_n} r$ where B is a new attribute of the relation r and each A_i is either an attribute of r or is a string. The result of the γ operator is r with an additional attribute B, where the B value on row k is a sequential concatenation of the row-k values for the attributes along with any given strings. As an example, consider $\gamma_{Model\ with\ Trim \leftarrow Model + \texttt{" "} + Trim} T_1$ which converts Table T_1 in Figure 10 to Table T_2.

We can use standard set operators to help sort out subsets, supersets, and overlaps of value sets. We can, for example, take a union of the exterior colors in Figure 2 and features in Figure 3 to form part of the set for *Feature* in Figure 1. After adding needed projection and renaming operations, this union is $\rho_{Exterior \leftarrow Feature} \pi_{Exterior} T \cup \rho_{Features \leftarrow Feature} T/Make$ *and Model/Features*, where T is the table in Figure 2 and T/*Make and Model/Features* is a path expression that follows the link under *Make and Model* in table T to the *Features* list in Figure 3. When we need subsets of a set, we can extend the standard selection operator $\sigma_C r$ to allow C to be a regular expression that identifies the subset of values we wish to include. With set-difference operations, we can also resolve overlapping sets by operator combinations.

Now that we have the operators we need, we can give examples. Figure 11 gives the mapping from the source table in Figure 4 to the target table in Figure 1. Observe that we have transformed all the Boolean values into attribute-name values and that we have gathered together all the features as *Feature* values. Figure 12 gives the mapping for the car ads from the site for Figures 2 and 3. For purposes of illustration, we assume for the mapping in Figure 12 that we have recognized the list and the horizontal table in Figure 3 as tables.

T_1

Make	*Model*	*Trim*
Ford	Contour	GL
Ford	Taurus	LX
Honda	Civic	EX

T_2

Make	*Model*	*Trim*	*Model with Trim*
Ford	Contour	GL	Contour GL
Ford	Taurus	LX	Taurus LX
Honda	Civic	EX	Civic EX

Fig. 10. Application of the γ Operator to Table T_1 Yielding Table T_2

Target Attribute	Source Derivation Expression for Value Sets
Year	$\pi_{Year}T$
Make	$\pi_{Make}T$
Model	$\pi_{Model}T$
Price	$\pi_{Price}T$
Feature	$\rho_{Colour \leftarrow Feature}\pi_{Colour}T$ $\cup\ \rho_{Auto \leftarrow Feature}\pi_{Auto}\beta^{Auto}_{Yes,\ No}T$ $\cup\ \rho_{Air\ Cond. \leftarrow Feature}\pi_{Air\ Cond.}\beta^{Air\ Cond.}_{Yes,\ No}T$ $\cup\ \rho_{AM/FM \leftarrow Feature}\pi_{AM/FM}\beta^{AM/FM}_{Yes,\ No}T$ $\cup\ \rho_{CD \leftarrow Feature}\pi_{CD}\beta^{CD}_{Yes,\ No}T$

Fig. 11. Inferred Mapping from Source Table T in Figure 4 to Target Table in Figure 1

Target Attribute	Source Derivation Expression for Value Sets
Year	$\pi_{Year}T$
Make	$\pi_{Make}\delta^{Make\ and\ Model}_{Make,\ Model}T$
Model	$\pi_{Model}\delta^{Make\ and\ Model}_{Make,\ Model}T$
Mileage	$\rho_{Miles \leftarrow Mileage}\pi_{Model}T$
Price	$\pi_{Price}T$
PhoneNr	$\epsilon_{PhoneNr}T/Make\ and\ Model$
Feature	$\rho_{Exterior \leftarrow Feature}\pi_{Exterior}T$ $\cup\ \rho_{Features \leftarrow Feature}T/Make\ and\ Model/Features$ $\cup\ \rho_{Body\ Type \leftarrow Feature}T/Make\ and\ Model/Body\ Type$ $\cup\ \rho_{Transmission \leftarrow Feature}T/Make\ and\ Model/Transmission$ $\cup\ \rho_{Engine \leftarrow Feature}T/Make\ and\ Model/Engine$

Fig. 12. Inferred Mapping from the Source Tables T in Figure 2 and *T/Make and Model* in Figure 3 to the Target Table in Figure 1

Observe that we have split the makes and models as required, matched the synonyms *Miles* and *Mileage*, extracted the *PhoneNr* from the free text, and gathered together all the various features as *Feature* values.

4 Experimental Results and Discussion

We gathered tables of car advertisements from many more than a hundred different English-language sites (several dozen were from non-U.S. sites). Because of human resource limitations, however, we analyzed only 60. In gathering tables, we encountered very few our implemented system could not process. Our system does not (yet) (1) convert color-swatch icons to color names, (2) recognize check marks rendered from images, (3) read legends for abbreviated attribute names in column headers, and (4) handle embedded links to subpages describing more than one car. Every car-ads table we encountered had simple attributes in the top row; thus we discarded no tables for structural reasons.

Of the 60 car-ads tables we analyzed, 40 included links to other pages containing additional information about an advertised car (Figures 2 and 3 show a typical example). For all 60 tables, we first applied our system to identify and list attribute-value pairs for tuples of top-level tables, and then for the 40 tables with links, we appropriately associated linked information (without alteration) with each tuple. We then applied our extraction step and looked for mapping patterns.

Since our objective was to obtain mappings (rather than data), it was not necessary for us to process every tuple in every table. Hence, from every table, we processed only the first 10 car ads. As a threshold, we required six or more occurrences of a pattern to declare a mapping. A human expert judged the correctness of each mapping. We considered a mapping declaration for a target attribute to be completely correct if the pattern recognized led to exactly the same mapping as the human expert declared, partially correct if the pattern led to a unioned (or intersected) component of the mapping, and incorrect otherwise. For data outside of tables, the system mapped an individual value to either the right place or the wrong place or did not map a value it should have mapped. Because of differences in granularity, we separate table mappings from individual-value mappings in reporting our results.

4.1 Results

We divided the 60 car-ads tables into two groups: 10 "training" tables and 50 "test" tables. We used the 10 "training" tables to adjust our car-ads extraction ontology to recognize ordered pairs derived from our table-understanding procedure and also to fine-tune our ontology and update it with the latest makes, models, and features. Adjusting for recognizing ordered pairs was straightforward—we simply added the various attribute names we found in the training set as keywords and sometimes as context identifiers for values (particularly for *Mileage* and *Price*, which both need something other than standard numbers to correctly identify them). Updating our ontology with the latest makes, models, and features was also straightforward, although it was a bit tedious especially for features, which tend to be more prolific in dealer sites on the Web than in classified ads posted by individuals.

For the 10 training tables, we were able to identify 100% of the 57 mappings while declaring no false mappings. Furthermore, we correctly found 94.6% of the values in the linked data, while incorrectly declaring only 5.4%. These numbers decreased for the 50 test tables. For these 50 test tables, we were able to completely identify 95.3% of the 300 mappings and partially identify 1.3%, while declaring no false mappings and failing to declare 3.3% of the mappings. Based on a sample of nearly 3,000 values found in unstructured secondary pages, we found that the precision and recall ratios for the 50 test tables were approximately 86% and 97% respectively. This corresponds well with our previous car-ads experiments [ECJ+99].

Of the 357 mappings we discovered, 172 were direct, in the sense that the attributes in the source and target schemas were identical. Of the 185 indirect matches, 29 used synonyms and thus required only renaming with a ρ operator, 5 had Boolean values and thus required a β operator, 68 included features scattered under various attributes and in raw text and thus required $\cup$ and ϵ operators, 19 provided only factored telephone numbers and thus required ϵ and $\bowtie$ operators, 89 needed to be split and thus required a δ operator, and some required combinations of these operators (e.g. synonyms and union). The values we needed to split came in a variety of different combinations and under a variety of different names. We found, for example, *Description* as an attribute for the combination *Year+Make+Model+Feature*, *Model Color* as an attribute for *Make+Model+Color*, and *Model* as an attribute for *Year+Make+Model*.

4.2 Discussion

As mentioned in our earlier discussion, discovering correct mappings can lead to an increase in values extracted compared to values that would have been extracted by the extraction ontology alone and can also therefore lead to the acquisition of additional knowledge for the extraction ontology. In our experiments, we required a 60% or greater match to declare a mapping match. Overall, we actually achieved roughly 90-95%, a much higher percentage. This, however, leaves about 5-10% of the approximately 3,000 values encountered in tables as being unrecognized by the extraction ontology (and potentially many more since we processed only 10 car ads per site). Examples include non-U.S. models such as the Toyota Starlet or Nissan Presea; elaborately described features such as "telescoping steering wheel"; abbreviations not encountered previously such as "leath int" for "leather interior"; and features simply not encountered in previous experiments, such as "trip computer".

We missed 10 mappings and partially identified 4 mappings. Our system missed 6 mappings of car model because the extraction ontology was targeted to U.S. car-ads, and so non-U.S. car-ads introduced models that our system did not recognize. 2 more mappings of car model were missed because the extraction ontology was not sufficiently robust (for Jaguar and SAAB models). The system missed 1 price mapping and 1 mileage mapping because the extraction ontology was overly restrictive. All of these problems can be corrected by adjusting the extraction ontology. The 4 partial mappings were for car features. Most of these cases could also be corrected by adjusting the extraction ontology. One interesting case was a table that included a "Description" column that contained an unstructured paragraph of text that would be more appropriately treated as if it were a linked page (where we expected to find unstructured text). Another interesting case was a table that included listings for trailers mixed in with car ads (e.g. "1999 Load Rite Trailer").

5 Conclusion

In this paper, we suggested a different approach to the problem of schema matching, one which may work better for the heterogeneous HTML tables encountered on the Web. In essence, we transformed the matching problem to an extraction problem over which we could infer the semantic correspondence between a source table and a target schema. We then showed how to discover the appropriate queries for source-to-target mapping rules. We gave experimental evidence to show that our approach can be successful. In particular, we correctly inferred 94% of the appropriate mappings to our target car-ads ontology from 60 HTML car-ads Web tables with a precision of 98%.

As a next step in our work on extraction from HTML tables, we intend to implement the ideas we have on forming attribute-value pairs for tables in linked information, for nested tables, and for less common cases—where attributes for multiple records appear on the left, where attributes appear both on top and on the left, and where attributes are nested and compound. Once we have attribute-value pairs, we can directly apply the mapping techniques discussed here.

Many tables are behind forms, in the so-called "hidden Web" [RGM01], and we are currently working on extracting data from the hidden Web [LYE01]. Once extracted, if the result is a table, we can use the techniques presented here to extract the data into a target view. If the result is not a table, we use previous techniques we have developed [ECJ+99] to extract the data. Further, we also plan to piece together all the components we have developed in our data-extraction work [DEG] into a comprehensive extraction tool.

Acknowledgements

This material is based upon work supported by the National Science Foundation under grant No. IIS-0083127.

References

[Aut01] autoscanada.com, Summer 2001. 323

[BE02] J. Biskup and D. W. Embley. Extracting information from heterogeneous information sources using ontologically specified target views. *Information Systems*, 2002. (to appear). 322, 325

[Bob02] www.bobhowardhonda.com, January 2002. 323

[DDH01] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data (SIGMOD 2001)*, pages 509–520, Santa Barbara, California, May 2001. 322, 325

[DEG] Homepage for BYU data extraction research group. URL: http://osm7.cs.byu.edu/deg/index.html. 336

[ECJ+99] 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. 325, 327, 334, 336

[KS91] H. F. Korth and A. Silberschatz. *Database System Concepts*. McGraw-Hill, Inc., New York, New York, second edition, 1991. 329

[LN99a] S. Lim and Y. Ng. An automated approach for retrieving heirarchical data from HTML tables. In *Proceedings of the Eighth International Conference on Informaiton and Knowledge management (CIKM'99)*, pages 466–474, Kansas City, Missouri, November 1999. 328

[LN99b] D. Lopresti and G. Nagy. Automated table processing: An (opinionated) survey. In *Proceedings of the Third IAPR Workshop on Graphics Recognition*, pages 109–134, Jaipur, India, September 1999. 325

[LYE01] S. W. Liddle, S. H. Yau, and D. W. Embley. On the automatic extraction of data from the hidden web. In *Proceedings of the International Workshop on Data Semantics in Web Information Systems (DASWIS-2001)*, pages 106–119, Yokohama, Japan, November 2001. 336

[MBR01] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with Cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*, pages 49–58, Rome, Italy, September 2001. 322, 325

[MHH00] R. Miller, L. Haas, and M. A. Hernandez. Schema mapping as query discovery. In *Proceedings of the 26th International Conference on Very Large Databases (VLDB'00)*, pages 77–88, Cairo, Egypt, September 2000. 322, 325

[RGM01] S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*, Rome, Italy, September 2001. 336

On the Expressive Power of Data Integration Systems

Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza", Via Salaria 113, I-00198 Roma, Italy
{cali,calvanese,degiacomo,lenzerini}@dis.uniroma1.it
http://www.dis.uniroma1.it/~[cali,calvanese,degiacomo,lenzerini]

Abstract. There are basically two approaches for designing a data integration system. In the global-as-view (GAV) approach, one maps the concepts in the global schema to views over the sources, whereas in the local-as-view (LAV) approach, one maps the sources into views over the global schema. The goal of this paper is to relate the two approaches with respect to their expressive power. The analysis is carried out in a relational database setting, where both the queries on the global schema, and the views in the mapping are conjunctive queries. We introduce the notion of query-preserving transformation, and query-reducibility between data integration systems, and we show that, when no integrity constraints are allowed in global schema, the LAV and the GAV approaches are incomparable. We then consider the addition of integrity constraints in the global schema, and present techniques for query-preserving transformations in both directions. Finally, we show that our results imply that we can always transform any system following the GLAV approach (a generalization of both LAV and GAV) into a query-preserving GAV system.

1 Introduction

Data integration is the problem of combining the data residing at different sources, and providing the user with a unified view of these data, called global (or, mediated) schema [9]. The global schema is therefore the interface by which users issue their queries to the system. The system answers the queries by accessing the appropriate sources, thus freeing the user from the knowledge on where data are, and how data are structured at the sources.

The interest in this kind of systems has been continuously growing in the last years. Many organizations face the problem of integrating data residing in several sources. Companies that build a Data Warehouse, a Data Mining, or an Enterprise Resource Planning system must address this problem. Also, integrating data in the World Wide Web is the subject of several investigations and projects nowadays. Finally, applications requiring accessing or re-engineering legacy systems must deal with the problem of integrating data stored in pre-existing sources.

The design of a data integration system is a very complex task, which comprises several different issues [10]. One of the most important aspect is the specification of the mapping between the global schema and the sources, and the use of such a specification for carrying out query processing.

Two basic approaches have been used to specify the mapping between the sources and the global schema [9, 11, 12]. The first approach, called *global-as-view* (or simply GAV), requires that the global schema is expressed in terms of the data sources. More precisely, to every element of the global schema, a view over the data sources is associated, so that its meaning is specified in terms of the data residing at the sources. The second approach, called *local-as-view* (LAV), requires the global schema to be specified independently from the sources. In turn, the sources are defined as views over the global schema. The relationships between the global schema and the sources are thus established by specifying the information content of every source in terms of a view over the global schema.

Intuitively, the GAV approach provides a method for specifying the data integration system with a more procedural flavor with respect to the LAV approach. Indeed, whereas in LAV the designer of the data integration system may concentrate on specifying the content of the source in terms of the global schema, in the GAV approach, one is forced to specify how to get the data of the global schema by queries over the sources.

A comparison of the LAV and the GAV approaches is reported in [16]. It is known that the former approach ensures an easier extensibility of the integration system, and provides a more appropriate setting for its maintenance. For example, adding a new source to the system requires only to provide the definition of the source, and does not necessarily involve changes in the global view. On the contrary, in the GAV approach, adding a new source may in principle require changing the definition of the concepts in the global schema.

It is also well known that processing queries in the LAV approach is a difficult task [15, 16, 8, 1, 7, 3, 4]. Indeed, in this approach, the only knowledge we have about the data in the global schema is through the views representing the sources, and such views provide only partial information about the data. Since the mapping associates to each source a view over the global schema, it is not immediate to infer how to use the sources in order to answer queries expressed over the global schema. Thus, extracting information from the data integration system is similar to query answering with incomplete information, which is a complex task [17]. On the other hand, query processing looks much easier in the GAV approach, where we can take advantage that the mapping directly specifies which source queries corresponds to the elements of the global schema. Indeed, in most GAV systems, query answering is based on a simple unfolding strategy.

Besides the above intuitive considerations, a deep analysis of the differences/similarities of the two approaches is still missing. The goal of this paper is to investigate on the relative expressive power of the LAV and the GAV approaches. In particular, we address the problem of checking whether a LAV system can be transformed into a GAV one, and vice-versa. Obviously, we are interested in transformations that are equivalent with respect to query answering, in the sense that we want that every query posed to the original system has the same answers when posed to the new system. To this end, we introduce the notion of query-preserving transformation, and the notion of query-reducibility between classes of data integration systems. Results on query reducibility from LAV to GAV systems may be useful, for example, to derive a procedural specification from a declarative one. Conversely, results on query reducibility from GAV to LAV may be useful to derive a declarative characterization of the content of the sources starting from a procedural specification.

We study the problem in a setting where the global schema is expressed in the relational model, and the queries used in the integration systems (both the queries on the global schema, and the queries in the mapping) are expressed in the language of conjunctive queries. We show that in such a setting none of the two transformations is possible. On the contrary, we show that the presence of integrity constraints in the global schema allows reducibility in both directions. In particular, inclusion dependencies and a simple form of equality-generating dependencies suffice for a query-preserving transformation from a LAV system into a GAV one, whereas single head full dependencies are sufficient for the other direction. Finally, we introduce the GLAV approach, where both LAV and GAV assertions are allowed in the mapping, and illustrate how to adapt the technique from LAV to GAV to devise a query-preserving transformation from GLAV to GAV.

Also, the results presented in the paper shows that techniques for answering queries under integrity constraints are relevant in data integration. In particular, several approaches to answering queries under different forms of dependencies have been proposed in the last years (see for example [14]). Our results imply that these approaches can be directly applied to query answering in LAV, GAV, and GLAV systems with inclusion dependencies. Data integration is thus a good candidate as an application for experimenting these techniques in real world settings.

The paper is organized as follows. In Section 2 we describe the formal framework we use for data integration, and we introduce the notions of query-preserving transformation, and of query-reducibility between classes of data integration systems. In Section 3 we show that in the relational model without integrity constraints, the classes of LAV and GAV systems are not mutually query-reducible. In Section 4 we present the results on query-reducibility in the case where integrity constraints are allowed in the global schema. Finally, Section 5 concludes the paper with a discussion on the GLAV approach.

2 Framework for Data Integration

We set up a formal framework for data integration in the relational setting. We assume that the databases involved in our framework are defined over a fixed (infinite) set Δ of objects. A database $\mathcal{DB}$ for a relational schema $\mathcal{R}$ is a relational structure $(\Delta^{\mathcal{DB}}, \cdot^{\mathcal{DB}})$ over $\mathcal{R}$ with $\Delta^{\mathcal{DB}} \subseteq \Delta$. When needed, we denote a relation r of arity n by r/n. Given a query q over $\mathcal{DB}$, we denote by $q^{\mathcal{DB}}$ the set of tuples of objects in $\Delta^{\mathcal{DB}}$ obtained by evaluating q over $\mathcal{DB}$, i.e., the set of *answers* to q over $\mathcal{DB}$. In particular, we focus on *conjunctive queries* (CQs) with equality atoms and constants. We denote a CQ of arity n over a relational schema $\mathcal{R}$ as

$$\{ \langle X_1, \ldots, X_n \rangle \mid \varphi(X_1, \ldots, X_n, Y_1, \ldots, Y_m) \}$$

where $X_1, \ldots, X_n$ are the *distinguished variables* (not necessarily pairwise distinct), $Y_1, \ldots, Y_m$ are the existentially quantified *non-distinguished variables*, and $\varphi(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ is a conjunction of atoms over predicate symbols in $\mathcal{R}$, involving constants, and the variables $X_1, \ldots, X_n, Y_1, \ldots, Y_m$. For a relation r/n, we write the CQ $\{\langle X_1, \ldots, X_n \rangle \mid r(X_1, \ldots, X_n)\}$ simply as r.

We consider also constraints over a relational schema. In particular, we consider inclusion dependencies, simple equality-generating dependencies, and single head full dependencies [2]. Given a relation r and a tuple $\mathbf{A}$ of distinct attributes of r, we denote the projection of r over $\mathbf{A}$ by $r[\mathbf{A}]$. Similarly, given a tuple t of r, we denote the projection of t over $\mathbf{A}$ by $t[\mathbf{A}]$. An *inclusion dependency* is a dependency of the form $r[\mathbf{A}] \subseteq r'[\mathbf{A}']$, where r and r' are two relations of a relational schema $\mathcal{R}$ and $\mathbf{A}$ and $\mathbf{A}'$ are two sequences of distinct attributes of the same arity, belonging to r and r' respectively. A database $\mathcal{DB}$ satisfies $r[\mathbf{A}] \subseteq r'[\mathbf{A}']$ if $r[\mathbf{A}]^{\mathcal{DB}} \subseteq r'[\mathbf{A}']^{\mathcal{DB}}$. A *simple equality-generating dependency* has the form $r \rightarrow A = A'$, where r is a relation of a relational schema $\mathcal{R}$, and A and A' are two distinct attributes of r. A database $\mathcal{DB}$ satisfies $r \rightarrow A = A'$ if for every tuple $t \in r^{\mathcal{DB}}$, it holds that $t[A] = t[A']$. A *single head full dependency* has the form $q \subseteq r$, where r is a relation of a relational schema $\mathcal{R}$ and q is a conjunctive query over $\mathcal{R}$ of the same arity as r. A database $\mathcal{DB}$ satisfies $q \subseteq r$ if $q^{\mathcal{DB}} \subseteq r^{\mathcal{DB}}$.

A *data integration system* $\mathcal{I}$ is a triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where

– $\mathcal{G}$ is the *global schema*, expressed in the relational model, possibly with constraints.
– $\mathcal{S}$ is the *source schema*, also expressed in the relational model.
– $\mathcal{M}$ is the *mapping* between $\mathcal{G}$ and $\mathcal{S}$, constituted by a set of *assertions* of the form

$$q_{\mathcal{S}} \subseteq q_{\mathcal{G}}$$

where $q_{\mathcal{S}}$ and $q_{\mathcal{G}}$ are two queries of the same arity, respectively over the source schema $\mathcal{S}$ and over the global schema $\mathcal{G}$.

Intuitively, the source schema describes the schema of the data sources, which contain data, while the global schema provides a reconciled, integrated, view of the underlying sources. The assertions in the mapping establish the connection between the relations of the global schema and those of the source schema. As typical in data integration, we consider here mappings that are *sound*, i.e., the data provided by the queries over the sources satisfy the queries over the global schema, but do not necessarily characterize completely the answer of the queries over the global schema [16, 9, 7]. User queries are posed over the global schema and are answered by retrieving data from the sources, making use of the mapping.

Two basic approaches for specifying the mapping have been proposed in the literature: *global-as-view* (GAV) and *local-as-view* (LAV) [16, 9]. In the GAV approach, the mapping $\mathcal{M}$ associates to each relation g in $\mathcal{G}$ a query $\varrho_{\mathcal{S}}(g)$ over $\mathcal{S}$, i.e., a GAV mapping is a set of assertions, one for each relation g of $\mathcal{G}$, of the form

$$\varrho_{\mathcal{S}}(g) \subseteq g$$

In the LAV approach, the mapping $\mathcal{M}$ associates to each relation s in $\mathcal{S}$ a query $\varrho_{\mathcal{G}}(s)$ over $\mathcal{G}$, i.e., a LAV mapping is a set of assertions, one for each relation s of $\mathcal{S}$, of the form

$$s \subseteq \varrho_{\mathcal{G}}(s)$$

Observe that in both cases we associate to a relation (either global or local) a single query. We call *GAV (with constraints)* the class of integration systems (with constraints) with a GAV mapping. Similarly for *LAV (with constraints)*.

Given an integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, we call *source database* (for $\mathcal{I}$), a database for the source schema $\mathcal{S}$, and *global database* (for $\mathcal{I}$) a database for $\mathcal{G}$ satisfying the constraints of $\mathcal{G}$. Let $\mathcal{D}$ be a source database. A global database $\mathcal{B}$ *satisfies an assertion* $q_{\mathcal{S}} \subseteq q_{\mathcal{G}}$ *in* $\mathcal{M}$ *with respect to* $\mathcal{D}$, if $q_{\mathcal{S}}^{\mathcal{D}} \subseteq q_{\mathcal{G}}^{\mathcal{B}}$. The global database $\mathcal{B}$ is said to be *legal for* $\mathcal{I}$ *with respect to* $\mathcal{D}$, if it satisfies all assertions in the mapping $\mathcal{M}$ with respect to $\mathcal{D}$. Observe that, in general, several global databases exist that are legal for $\mathcal{I}$ with respect to $\mathcal{D}$.

Queries posed to an integration system $\mathcal{I}$ are expressed in terms of the relations in the global schema of $\mathcal{I}$. Given a source database $\mathcal{D}$ for $\mathcal{I}$, the answer $q^{\mathcal{I},\mathcal{D}}$ to a query q to $\mathcal{I}$ with respect to $\mathcal{D}$, is the set of tuples t of objects in $\mathcal{D}$ such that $t \in q^{\mathcal{B}}$ for *every* global database $\mathcal{B}$ legal for $\mathcal{I}$ with respect to $\mathcal{D}$. The set $q^{\mathcal{I},\mathcal{D}}$ is called the set of *certain answers* of q to $\mathcal{I}$ with respect to $\mathcal{D}$.

Given two integration systems $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M}' \rangle$ over the same source schema $\mathcal{S}$ and such that all relations of $\mathcal{G}$ are also relations of $\mathcal{G}'$, we say that $\mathcal{I}'$ is *query-preserving* with respect to $\mathcal{I}$, if for every query q to $\mathcal{I}$ and for every source databases $\mathcal{D}$ for $\mathcal{S}$, we have that

$$q^{\mathcal{I},\mathcal{D}} = q^{\mathcal{I}',\mathcal{D}}$$

In other words, we say that $\mathcal{I}'$ is query-preserving with respect to $\mathcal{I}$ if, given a query over the global schema of $\mathcal{I}$, the certain answers we get for the query on the two integration systems are identical.

To compare classes of integration systems, we introduce the concept of query-reducibility. A class $\mathcal{C}_1$ of integration systems is *query-reducible* to a class $\mathcal{C}_2$ of integration systems if there exist a function $f : \mathcal{C}_1 \to \mathcal{C}_2$ such that, for each $\mathcal{I}_1 \in \mathcal{C}_1$ we have that $f(\mathcal{I}_1)$ is query-preserving with respect to $\mathcal{I}_1$.

3 Comparing LAV and GAV without Constraints

In this section we consider data integration systems without constraints in the global schema. We want to check whether any GAV system can be transformed into a LAV one which is query-preserving wrt it, and vice-versa. We show that both transformation are not feasible.

We begin with the transformation from LAV to GAV.

Theorem 1. *The class of LAV data integration systems is not query-reducible to the class of GAV systems.*

Proof. We prove the theorem by exhibiting a particular LAV system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, a source database $\mathcal{D}$ for $\mathcal{S}$, and a set of queries such that, for any GAV system $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M}' \rangle$, the certain answers of the queries wrt $\mathcal{D}$ differ in $\mathcal{I}$ and $\mathcal{I}'$.

The LAV system $\mathcal{I}$ is as follows. The global schema $\mathcal{G}$ is constituted by $g_1/2$ and $g_2/2$, while the source schema $\mathcal{S}$ is constituted by a single relation $s/2$. The mapping $\mathcal{M}$ is

$$\varrho_{\mathcal{G}}(s) \;=\; \{\langle X, Y \rangle \mid g_1(X, Z) \wedge g_2(Z, Y)\}$$

By contradiction, assume there is a GAV system $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M}' \rangle$ that is query-preserving with respect to $\mathcal{I}$. Observe that, since no constraints are allowed in the global schema, the introduction of a new relation in $\mathcal{G}'$ is useless if we want to construct a system that is query-preserving wrt $\mathcal{I}$; in fact, the newly introduced predicates could not be related to g_1 and g_2. Therefore, we can assume that $\mathcal{G}' = \mathcal{G}$. It follows that the mapping $\mathcal{M}'$ has the form

$$\varrho_{\mathcal{S}}(g_1) = \{\langle X, Y \rangle \mid \xi_1(X, Y, Z_1, \ldots, Z_{k_1}, c_1, \ldots, c_{h_1})\}$$
$$\varrho_{\mathcal{S}}(g_2) = \{\langle X, Y \rangle \mid \xi_2(X, Y, W_1, \ldots, W_{k_2}, d_1, \ldots, d_{h_2})\}$$

where ξ_1 and ξ_2 are conjunctions of atoms over the only relation s, $Z_1, \ldots, Z_{k_1}$ and $W_1, \ldots, W_{k_2}$ are existentially quantified variables, and $c_1, \ldots, c_{h_1}$ and $d_1, \ldots, d_{h_2}$ are constants of Δ.

We take the source database $\mathcal{D}$ to be such that $s^{\mathcal{D}} = \{\langle a, b \rangle\}$, where a and b are two constants, and we consider the following queries:

$$q_1(X, Y) = \{\langle X, Y \rangle \mid g_1(X, Z) \wedge g_2(Z, Y)\}$$
$$q_2(X, Y) = \{\langle X, Y \rangle \mid g_1(X, Y)\}$$
$$q_3(X, Y) = \{\langle X, Y \rangle \mid g_2(X, Y)\}$$

The certain answers of q_1, q_2, and q_3 to $\mathcal{I}$ wrt $\mathcal{D}$ are the following: $q_1^{\mathcal{I},\mathcal{D}} = \langle a, b \rangle$, $q_2^{\mathcal{I},\mathcal{D}} = \emptyset$, and $q_3^{\mathcal{I},\mathcal{D}} = \emptyset$.

If one of $\varrho_{\mathcal{S}}(g_1)^{\mathcal{D}}$ or $\varrho_{\mathcal{S}}(g_2)^{\mathcal{D}}$ is non-empty, we have that one of $q_2^{\mathcal{I}',D}$ or $q_3^{\mathcal{I}',D}$ is non-empty, and hence a contradiction. When both $\varrho_{\mathcal{S}}(g_1)^{\mathcal{D}}$ and $\varrho_{\mathcal{S}}(g_2)^{\mathcal{D}}$ are empty, we immediately obtain that $q_1^{\mathcal{I}',\mathcal{D}} = \emptyset$. Contradiction.

This result shows that the mechanism of query answering in LAV cannot be directly simulated by the corresponding mechanism in GAV, which is basically unfolding, i.e., the substitution in the user query of the global relations with their definition given by the mapping.

We now turn to the transformation from GAV to LAV.

Theorem 2. *The class of GAV data integration systems is not query-reducible to the class of LAV systems.*

Proof. We exhibit a particular GAV system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and a query such that, for any LAV system $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M}' \rangle$ we can construct a source database $\mathcal{D}$ for $\mathcal{S}$ such that the certain answers of the query to $\mathcal{I}$ and $\mathcal{I}'$ differ wrt $\mathcal{D}$.

Let $\mathcal{I}$ be as follows. The global schema $\mathcal{G}$ is constituted by a single relation $g/2$, while the source schema $\mathcal{S}$ is constituted by $s_1/2$ and $s_2/2$. The mapping $\mathcal{M}$ is

$$\varrho_{\mathcal{S}}(g) \ = \ \{\langle X, Y \rangle \mid s_1(X, Z) \wedge s_2(Z, Y)\}$$

As in the previous case, we observe that the introduction of new relations in $\mathcal{G}'$ is not significant if we want to construct a system that is query-preserving wrt $\mathcal{I}$. Hence we assume that $\mathcal{G}' = \mathcal{G}$, and the mapping $\mathcal{M}'$ has the form

$$\varrho_{\mathcal{G}}(s_1) = \{\langle X, Y \rangle \mid \eta_1(X, Y, Z_1, \ldots, Z_{k_1}, c_1, \ldots, c_{h_1})\}$$
$$\varrho_{\mathcal{G}}(s_2) = \{\langle X, Y \rangle \mid \eta_2(X, Y, W_1, \ldots, W_{k_2}, d_1, \ldots, d_{h_2})\}$$

where η_1 and η_2 are conjunctions of atoms over the only relation g, $Z_1, \ldots, Z_{k_1}$, $W_1, \ldots, W_{k_2}$ are existentially quantified variables, and $c_1, \ldots, c_{h_1}, d_1, \ldots, d_{h_2}$ are constants in Δ.

We define the source database $\mathcal{D}$ such that $s_1^{\mathcal{D}} = \{\langle a, b \rangle\}$ and $s_2^{\mathcal{D}} = \{\langle b, c \rangle\}$, where a, b, and c are constants, distinct from $c_1, \ldots, c_{h_1}, d_1, \ldots, d_{h_2}$. Consider the query

$$q(X, Y) \ = \ \{\langle X, Y \rangle \mid g(X, Y)\}$$

whose certain answers in $\mathcal{I}$ are $\{\langle a, c \rangle\}$.

Let $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M}' \rangle$ be a LAV system. We show that $\langle a, c \rangle \notin q^{\mathcal{I}', \mathcal{D}}$, by constructing a global database $\mathcal{B}'$ which satisfies $\mathcal{M}'$ wrt $\mathcal{D}$ and such that $\langle a, c \rangle \notin q^{\mathcal{B}'}$. We construct $g^{\mathcal{B}'}$ as follows. We associate to each variable or constant V appearing in the definition of $\varrho_{\mathcal{S}}(s_1)$ a distinct constant $\psi(V)$, such that $\psi(X) = a$, $\psi(Y) = b$, and $\psi(V) = V$ if V is a constant. Then, for each atom $g(V_1, V_2)$ appearing in $\varrho_{\mathcal{S}}(s_1)$, we add the tuple $\langle \psi(V_1), \psi(V_2) \rangle$ to $g^{\mathcal{B}'}$. We do the same for $\varrho_{\mathcal{S}}(s_2)$, with $\psi(X) = b$ and $\psi(Y) = c$. Such a construction of $g^{\mathcal{B}'}$ ensures that $\langle a, c \rangle \notin g^{\mathcal{B}'}$ (by construction) and that $\mathcal{B}'$ is legal for $\mathcal{I}'$ wrt $\mathcal{D}$, as $\langle a, b \rangle \in s_1^{\mathcal{B}'}$ and $\langle b, c \rangle \in s_2^{\mathcal{B}'}$. Therefore $\langle a, c \rangle \notin q^{\mathcal{I}', \mathcal{D}}$. This proves the claim.

This result shows that we are not able to deduce the information of a LAV mapping, which specifies the role of each source relation wrt the global schema, from the information contained in a corresponding GAV mapping, which gives direct information on how query answering may be performed.

4 Comparing LAV and GAV with Constraints

We address the question of query-reducibility in the case where integrity constraints are allowed in the global schema.

One direction is almost immediate: single head full dependencies suffice for query-reducibility from GAV systems to LAV systems. Indeed, if $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ is a GAV system, we define a corresponding LAV system $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M}' \rangle$ as follows. For every source relation s in $\mathcal{S}$, we define a corresponding new relation g_s in $\mathcal{G}'$, and we include in $\mathcal{M}'$ the assertion $s \subseteq \varrho_{\mathcal{G}}(g_s)$. Now, for every $\varrho_{\mathcal{S}}(g) \subseteq g$ in $\mathcal{M}$, we introduce in $\mathcal{G}'$ the single head full dependency $\rho'_{\mathcal{S}}(g) \subseteq g$, where $\rho'_{\mathcal{S}}(g)$ denotes the conjunction obtained from $\rho_{\mathcal{S}}(g)$ by substituting every atom $s(x_1, \ldots, x_n)$ with $g_s(x_1, \ldots, x_n)$. It is easy to see that the resulting data integration system $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M}' \rangle$ is a LAV system, and that the transformation is query-preserving. Observe also that the size of $\mathcal{I}'$ is linearly related to the size of $\mathcal{I}$.

We now turn to the question of reducing LAV systems to GAV systems. We show that, when inclusion and simple equality generating dependencies are allowed on the global schema, we can obtain from every LAV system a query-preserving GAV system. Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a LAV integration system. Without loss of generality, we can assume that no equality atoms appear in the conjunctive queries in the mapping $\mathcal{M}$. We define a corresponding GAV integration system $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M}' \rangle$ as follows. For technical reasons, we first rewrite all queries in the mapping $\mathcal{M}$ so that variables appear in each atom at most once, by adding suitable equalities to the body of the queries. For example, the query $\{\langle X \rangle \mid \mathsf{cites}(X, X)\}$ is rewritten as $\{\langle X \rangle \mid \mathsf{cites}(X, Y) \wedge Y = X\}$.

Then $\mathcal{I}$ is as follows:

- The set of sources $\mathcal{S}$ remains unchanged.
- The global schema $\mathcal{G}'$ is obtained from $\mathcal{G}$ by introducing:
  - a new relation $\mathit{image\_s}/n$ for each relation s/n in $\mathcal{S}$;
  - a new relation $\mathit{expand\_s}/(n+m)$ for each relation s/n in $\mathcal{S}$, where m is the number of non-distinguished variables of $\varrho_{\mathcal{G}}(s)$; we assume variables in $\varrho_{\mathcal{G}}(s)$ to be enumerated as $Z_1, \ldots, Z_{n+m}$, with $Z_1, \ldots, Z_n$ being the distinguished variables;

  and by adding the following dependencies:
  - for each relation s/n in $\mathcal{S}$ we add the inclusion dependency

    $$\mathit{image\_s}[1, \ldots, n] \;\subseteq\; \mathit{expand\_s}[1, \ldots, n]$$

  - for each relation s in $\mathcal{S}$ and for each atom $g(Z_{i_1}, \ldots, Z_{i_k})$ occurring in $\varrho_{\mathcal{G}}(s)$, we add the inclusion dependency

    $$\mathit{expand\_s}[i_1, \ldots, i_k] \;\subseteq\; g[1, \ldots, k]$$

  - for each relation s in $\mathcal{S}$ and for each atom $Z_i = Z_j$ occurring in $\varrho_{\mathcal{G}}(s)$, we add the simple equality generating dependency

    $$\mathit{expand\_s} \;\rightarrow\; i = j$$

- The GAV mapping $\mathcal{M}'$ associates to each global relation $\mathit{image\_s}$ the query

  $$\varrho_{\mathcal{S}}(\mathit{image\_s}) = s$$

  and to the remaining global relations the empty query.

It is immediate to verify the following theorem.

Theorem 3. *Let $\mathcal{I}$ be a LAV integration system, and $\mathcal{I}'$ the corresponding GAV integration system defined as above. Then $\mathcal{I}'$ can be constructed in time that is linear in the size of $\mathcal{I}$.*

We illustrate the transformation with an example.

Example 1. Consider a LAV integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ where:

– The global schema $\mathcal{G}$ is constituted by the relations $\mathsf{cites}/2$, expressing that a paper cites another paper, and $\mathsf{sameTopic}/2$, expressing that two papers are on the same topic.
– The source schema $\mathcal{S}$ is constituted by three relations: source_1, containing pairs of papers that mutually cite each other; source_2, containing pairs of papers on the same topic, each with at least one citation; and source_3, containing papers that cite themselves.
– The LAV mapping $\mathcal{M}$ between the source schema and the global schema is:

$$\begin{aligned}
&\varrho_{\mathcal{G}}(\mathsf{source}_1) = \{\langle X, Y\rangle \mid \mathsf{cites}(X,Y) \wedge \mathsf{cites}(Y,X)\}\\
&\varrho_{\mathcal{G}}(\mathsf{source}_2) = \{\langle X, Y\rangle \mid \mathsf{sameTopic}(X,Y) \wedge \mathsf{cites}(X,Z) \wedge \mathsf{cites}(Y,W)\}\\
&\varrho_{\mathcal{G}}(\mathsf{source}_3) = \{\langle X\rangle \mid \mathsf{cites}(X,Y) \wedge X = Y\}
\end{aligned}$$

Then the corresponding GAV integration system $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M}' \rangle$ is as follows:

– The source schema $\mathcal{S}$ remains unchanged.
– The global schema $\mathcal{G}'$ is constituted by the relations $\mathsf{cites}/2$, $\mathsf{sameTopic}/2$ as before, and the additional relations $\mathit{image\_}\mathsf{source}_1/2$, $\mathit{image\_}\mathsf{source}_2/2$, $\mathit{image\_}\mathsf{source}_3/1$, $\mathit{expand\_}\mathsf{source}_1/2$, $\mathit{expand\_}\mathsf{source}_2/4$, and $\mathit{expand\_}\mathsf{source}_3/2$. Moreover, $\mathcal{G}$ contains the following inclusion dependencies:

$$\begin{aligned}
\mathit{image\_}\mathsf{source}_1[1,2] &\subseteq \mathit{expand\_}\mathsf{source}_1[1,2]\\
\mathit{image\_}\mathsf{source}_2[1,2] &\subseteq \mathit{expand\_}\mathsf{source}_2[1,2]\\
\mathit{image\_}\mathsf{source}_3[1] &\subseteq \mathit{expand\_}\mathsf{source}_3[1]\\
\mathit{expand\_}\mathsf{source}_1[1,2] &\subseteq \mathsf{cites}[1,2]\\
\mathit{expand\_}\mathsf{source}_1[2,1] &\subseteq \mathsf{cites}[1,2]\\
\mathit{expand\_}\mathsf{source}_2[1,3] &\subseteq \mathsf{cites}[1,2]\\
\mathit{expand\_}\mathsf{source}_2[2,4] &\subseteq \mathsf{cites}[1,2]\\
\mathit{expand\_}\mathsf{source}_3[1,2] &\subseteq \mathsf{cites}[1,2]\\
\mathit{expand\_}\mathsf{source}_2[1,2] &\subseteq \mathsf{sameTopic}[1,2]\\
\mathit{expand\_}\mathsf{source}_3 &\rightarrow 1 = 2
\end{aligned}$$

– The GAV mapping $\mathcal{M}'$ is

$$\varrho_{\mathcal{S}}(\mathit{image\_}\mathsf{source}_i) = \mathsf{source}_i, \qquad i \in \{1,2,3\}$$

■

We now show that the LAV integration system $\mathcal{I}$ and the corresponding GAV integration system $\mathcal{I}'$ obtained as above are indeed query-equivalent. The proof is based on the observation that both integration systems $\mathcal{I}$ and $\mathcal{I}'$ can be captured by suitable *logic programs* (we refer to [13] for notions relative to logic programming).

We first concentrate on GAV systems. The logic program $\mathcal{P}_{\mathcal{I}'}$ associated to a GAV system $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M}' \rangle$ is defined as follows:

– For each inclusion dependency $g_1[\mathbf{A}] \subseteq g_2[\mathbf{B}]$ in $\mathcal{G}'$, where $\mathbf{A}$ and $\mathbf{B}$ are sets of attributes, we first introduce a "pseudo-rule" of the form (assuming for simplicity that the attributes in $\mathbf{A}$ and $\mathbf{B}$ are the first h ones in g_1 and g_2, respectively):

$$g_2(X_1, \ldots, X_h, X_{h+1}, \ldots, X_n) \leftarrow g_1(X_1, \ldots, X_h, Y_{h+1}, \ldots, Y_m)$$

Then, for each simple equality generating dependency in $\mathcal{G}$ of the form $g_2 \rightarrow i{=}j$, we substitute in the above pseudo-rule each occurrence of X_j with X_i. We skolemize the resulting pseudo-rule, obtaining a rule of the form

$$g_2(Z_1, \ldots, Z_k, f_{k+1}(Z_1, \ldots, Z_k), \ldots, f_n(Z_1, \ldots, Z_k)) \quad \leftarrow \quad g_1(Z_1, \ldots, Z_k, W_{k+1} \ldots, W_m)$$

where each f_i is a fresh Skolem function.

– For each assertion $\varrho_{\mathcal{S}}(g) \subseteq g$ in the mapping $\mathcal{M}'$, where $\varrho_{\mathcal{S}}(g) = \{\langle X_1, \ldots, X_n \rangle \mid \varphi(X_1, \ldots, X_n, Y_{n+1}, \ldots, Y_m)\}$, we have a rule of the form

$$g(X_1, \ldots, X_n) \leftarrow \varphi(X_1, \ldots, X_n, Y_{n+1}, \ldots, Y_m)$$

with the proviso that, if a simple equality generating dependency applies to g, then we have to equate the appropriate variables.

In addition, the relations in $\mathcal{S}$ can be seen as predicates that are given extensionally. That is, a source database $\mathcal{D}$ for $\mathcal{I}'$ can be seen as a finite set of ground facts in logic programming terms.

By applying results from logic programming theory [13], we can show the following lemma.

Lemma 1. *Let $\mathcal{I}'$ be a GAV integration system, $\mathcal{D}$ a source database for $\mathcal{I}'$, $\mathcal{P}_{\mathcal{I}'}$ the corresponding logic program as defined above, and M_{min} the minimal model of $\mathcal{P}_{\mathcal{I}'} \cup \mathcal{D}$. Then, given a query q over $\mathcal{G}'$, for every tuple $\langle c_1, \ldots, c_n \rangle$ of objects in $\mathcal{D}$ we have that*

$$\langle c_1, \ldots, c_n \rangle \in q^{\mathcal{I}, \mathcal{D}} \qquad \textit{if and only if} \qquad \langle c_1, \ldots, c_n \rangle \in q^{M_{min}}$$

Proof (sketch). By considering the semantics of constraints in $\mathcal{G}'$, and the corresponding translation in $\mathcal{P}_{\mathcal{I}'}$, it can be shown that the certain answers of q to $\mathcal{I}'$ wrt $\mathcal{D}$ are those that are correct answers to q for the logic program $\mathcal{P}_{\mathcal{I}'} \cup \mathcal{D}$. The claim follows from the classical result in logic programming that the correct answers to a logic program are those that are true in the minimal model. □

In other words, for GAV integration systems, the tuples of constants in the certain answer to a query q are equal to those that satisfy q in the minimal model of the corresponding logic program.

Let us turn to LAV integration systems. Without loss of generality, we can assume that equality generating dependencies have been folded into queries by suitably renaming variables. Given a LAV integration system $\mathcal{I}$, we can define an associated logic program $\mathcal{P}_{\mathcal{I}}$ by introducing rules for dependencies as before, and by treating queries in the mapping as done in [5]. In particular, given the query associated to source s (for simplicity of presentation, we assume s to be a unary relation and the relations in the query to be binary)

$$\varrho_{\mathcal{G}}(s) \;=\; \{\langle X\rangle \mid g_1(X,Y_1)\wedge\cdots\wedge g_k(X,Y_k)\}$$

by applying skolemization we get

$$\varrho_{\mathcal{G}}(s) \;=\; \{\langle X\rangle \mid g_1(X,f_1(X))\wedge\cdots\wedge g_k(X,f_k(X))\}.$$

Then, we can introduce in $\mathcal{P}_{\mathcal{I}}$ the following rules, derived from the skolemized query:

$$\begin{array}{c} g_1(X,f_1(X)) \leftarrow s(X) \\ \cdots \\ g_k(X,f_k(X)) \leftarrow s(X) \end{array}$$

Based on the results in [5], we can prove also for LAV integration systems a lemma analogous to Lemma 1.

Lemma 2. *Let $\mathcal{I}$ be a LAV integration system, $\mathcal{D}$ a source database for $\mathcal{I}$, $\mathcal{P}_{\mathcal{I}}$ the corresponding logic program as defined above, and M_{min} the minimal model of $\mathcal{P}_{\mathcal{I}}\cup\mathcal{D}$. Then, given a query q over $\mathcal{G}$, for every tuple $\langle c_1,\ldots,c_n\rangle$ of objects in $\mathcal{D}$ we have that*

$$\langle c_1,\ldots,c_n\rangle \in q^{\mathcal{I},\mathcal{D}} \qquad \textit{if and only if} \qquad \langle c_1,\ldots,c_n\rangle \in q^{M_{min}}$$

In other words, also for LAV integration systems, the tuples of constants in the certain answer to a query q are equal to those that satisfy q in the minimal model of the corresponding logic program.

With these lemmas in place we can prove our main result.

Theorem 4. *Let $\mathcal{I}$ be a LAV integration system, and $\mathcal{I}'$ the corresponding GAV integration system defined as above. Then $\mathcal{I}'$ is query-preserving wrt $\mathcal{I}$.*

Proof (sketch). Let $\mathcal{P}_{\mathcal{I}}$ be the logic program capturing $\mathcal{I}$ and $\mathcal{P}_{\mathcal{I}'}$ the logic program capturing $\mathcal{I}'$. Then it is possible to show that, for every source database $\mathcal{D}$ for $\mathcal{I}$ and every global relation g of the global schema $\mathcal{G}$ of $\mathcal{I}$, we have (modulo renaming of the Skolem functions) that

$$g^{M_{min}} = g^{M'_{min}}$$

where M_{min} and M'_{min} are the minimal model of $\mathcal{P}_{\mathcal{I}}\cup\mathcal{D}$ and of $\mathcal{P}_{\mathcal{I}'}\cup\mathcal{D}$, respectively. Hence, by considering Lemma 1 and Lemma 2, we get the claim. □

5 Discussion

In the previous sections we have studied the relative expressive power of the two main approaches to data integration, namely, LAV and GAV. We have shown that, in the case where integrity constraints are not allowed in the global schema, LAV and GAV systems are not mutually query-reducible. On the other hand, the presence of integrity constraints allows us to derive query-preserving transformations in both directions.

In particular, we have demonstrated that inclusion dependencies and a simple form of equality-generating dependencies in the global schema are sufficient for transforming any LAV systems into a query-preserving GAV system whose size is linearly related to the size of the original system. Interestingly, the technique can be easily extended for transforming any GLAV system into a GAV one.

In the GLAV approach to data integration, the relationships between the global schema and the sources are established by making use of both LAV and GAV assertions [6]. More precisely, in a GLAV system, we associate a conjunctive query $q_{\mathcal{G}}$ over the global schema to a conjunctive query $q_{\mathcal{S}}$ over the source schema. Therefore, GLAV generalizes both LAV and GAV.

By exploiting the technique presented in Section 4, it is not difficult to see that any GLAV system can be transformed into a query-preserving GAV one, with the same technique presented above. The key idea is that a GLAV assertion can be transformed into a GAV assertion plus an inclusion dependency. Indeed, for each assertion

$$q_{\mathcal{S}} \subseteq q_{\mathcal{G}}$$

in the GLAV system (where the arity of both queries is n), we introduce a new relation symbol r/n in the global schema of the resulting GAV system, and we associate to r the query

$$\varrho_{\mathcal{S}}(r) = q_{\mathcal{S}}$$

plus the inclusion

$$r \subseteq q_{\mathcal{S}}$$

Now, it is immediate to verify that the above inclusion can be treated exactly with the same technique introduced in the LAV to GAV transformation, and therefore, from the GLAV system we can obtain a query-preserving GAV system whose size is linearly related to the size of the original system.

References

[1] Serge Abiteboul and Oliver Duschka. Complexity of answering queries using materialized views. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 254–265, 1998. 339

[2] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases.* Addison Wesley Publ. Co., Reading, Massachussetts, 1995. 341

[3] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. Answering regular path queries using views. In *Proc. of the 16th IEEE Int. Conf. on Data Engineering (ICDE 2000)*, pages 389–398, 2000. 339

[4] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. View-based query processing and constraint satisfaction. In *Proc. of the 15th IEEE Symp. on Logic in Computer Science (LICS 2000)*, pages 361–371, 2000. 339

[5] Oliver M. Duschka and Michael R. Genesereth. Answering recursive queries using views. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'97)*, pages 109–116, 1997. 348

[6] Marc Friedman, Alon Levy, and Todd Millstein. Navigational plans for data integration. In *Proc. of the 16th Nat. Conf. on Artificial Intelligence (AAAI'99)*, pages 67–73. AAAI Press/The MIT Press, 1999. 349

[7] Gösta Grahne and Alberto O. Mendelzon. Tableau techniques for querying information sources through global schemas. In *Proc. of the 7th Int. Conf. on Database Theory (ICDT'99)*, volume 1540 of *Lecture Notes in Computer Science*, pages 332–347. Springer, 1999. 339, 341

[8] Jarek Gryz. Query folding with inclusion dependencies. In *Proc. of the 14th IEEE Int. Conf. on Data Engineering (ICDE'98)*, pages 126–133, 1998. 339

[9] Alon Y. Halevy. Answering queries using views: A survey. *Very Large Database J.*, 10(4):270–294, 2001. 338, 339, 341, 342

[10] Richard Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'97)*, 1997. 339

[11] Alon Y. Levy. Logic-based techniques in data integration. In Jack Minker, editor, *Logic Based Artificial Intelligence*. Kluwer Academic Publisher, 2000. 339

[12] Chen Li and Edward Chang. Query planning with limited source capabilities. In *Proc. of the 16th IEEE Int. Conf. on Data Engineering (ICDE 2000)*, pages 401–412, 2000. 339

[13] John W. Lloyd. *Foundations of Logic Programming (Second, Extended Edition)*. Springer, Berlin, Heidelberg, 1987. 347

[14] Lucian Popa, Alin Deutsch, Arnaud Sahuguet, and Val Tannen. A chase too far? In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 273–284, 2000. 340

[15] Xiaolei Qian. Query folding. In *Proc. of the 12th IEEE Int. Conf. on Data Engineering (ICDE'96)*, pages 48–55, 1996. 339

[16] Jeffrey D. Ullman. Information integration using logical views. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 1997. 339, 341, 342

[17] Ron van der Meyden. Logical approaches to incomplete information. In Jan Chomicki and Günter Saake, editors, *Logics for Databases and Information Systems*, pages 307–356. Kluwer Academic Publisher, 1998. 339

Property-Based Semantic Reconciliation of Heterogeneous Information Sources

Jeffrey Parsons[1] and Yair Wand[2]

[1]Faculty of Business Administration
Memorial University of Newfoundland, St. John's, NF, A1B 3X5, Canada
jeffreyp@mun.ca
[2]Faculty of Commerce and Business Administration
The University of British Columbia, Vancouver, BC, V6T 1Z2 Canada
yair.wand@ubc.ca

Abstract. Integrating information from diverse sources is of great importance in the database area. The main difficulty in information integration is reconciling data semantics. Common approaches to semantic reconciliation are based on first identifying similar entity types in various sources, and then reconciling entity type properties (attributes and relationships). Such approaches assume all instances to be reconciled belong to well-defined types. We suggest an alternative approach based on two fundamental principles. First, reconciliation does not require that instances be assigned to specific types. Instead, sources can be reconciled by analyzing similarities of properties. Second, properties that appear different may be manifestations of a higher-level property that has the same meaning across sources. We present the fundamental ideas underlying our approach, analyze its potential advantages, suggest how the approach can be formalized, demonstrate with examples the feasibility of using it for semantic reconciliation, and suggest directions for further research.

1 Introduction

The advent of database technology has led to a proliferation of independent information sources as organizations develop databases for specific applications. Often, information about the same (kinds of) entities is distributed among different sources. Consequently it is necessary to combine or integrate data from these sources to enable them to interoperate. Since different sources are developed independently, integration typically involves resolving the semantics of the data in the separate sources. Much of the semantics of data is carried in the conceptual schemas of the sources.

Several relatively recent phenomena have further increased the need for combining information from multiple independent sources. First, the Internet has generated a wealth of sources that can be accessed easily, but that were usually created completely independently and for different purposes. Often, these sources contain information in the form of semi-structured data, where the underlying schemas may only be partially or implicitly specified. Second, the growing tendency of organizations to integrate their business processes by enabling their information systems to communicate and exchange data has created the need to reconcile the semantics of data in the communicating systems.

The recognition of the importance of integrating and reconciling data from different sources has led to a great deal of work addressing this issue. However, there appears to have been limited success in this area. According to Smith and Obrst, while there has been progress in addressing "syntactical and structural aspects of integration… no similar infrastructure is in place to help address [the] challenge of semantic reconciliation" [22, p. 26].

We suggest that a root cause underlying the apparent lack of progress on semantic integration is the absence of a strong theoretical foundation for understanding what it means to combine/integrate/reconcile data from independent sources.

Common approaches to reconciliation rely on two fundamental assumptions. First, they assume that records in the sources include information about instances of entity types or classes (e.g., 'customer'). This means that class reconciliation is a precondition to reconciling data about instances. Second, these approaches assume that data elements in a source can be mapped into a well-defined semantic data model. Such models typically employ constructs such as entities, relationships, and attributes. Hence, prior to reconciling specific data elements (or groups of elements) one would need to map them into semantic constructs. However, it is well known that the same piece of information might be represented using different semantic constructs (e.g. an entity type or an attribute). Thus, the need to resolve construct mismatches adds a level of complexity to the reconciliation process, even before one can explore similarities in meanings among the various sources.

In this paper, we contend that a more fruitful approach to semantic reconciliation is to focus first on reconciling the meaning of attributes. Based on this idea, we suggest a theoretical foundation for reconciliation, describe an approach to attribute-based reconciliation, and indicate the potential usefulness of the approach by demonstrating how one can apply it to query data distributed over heterogeneous sources.

2 Semantic Interoperability in Heterogeneous Systems

Research on semantic interoperability addresses the practical need to combine information from several independent sources that pertain to a common underlying subject domain, but that differ in the way they represent the semantics of that domain. Different sources might differ on implementation technologies, underlying data models, the approaches used to map the subject domain into the data structures, or on their views of the subject domain.

As mentioned above, the challenges of semantic interoperability have received renewed attention with the growth and evolution of the Internet. In that environment, sources typically are independent of each other and tend to lack the degree of structure found in traditional databases.

In this section, we review research on semantic interoperability in both these domains.

2.1 Semantic Interoperability in Databases

Traditionally, organizations have maintained their data in highly structured repositories that provide logical structures to facilitate transaction processing and/or query operations. The hierarchical, network, and relational models of data reflect an evolution of logical data structures that offer increasing flexibility in accessing and using data resources.

The independent development of applications to serve different needs inevitably leads to semantic heterogeneity. There are three main sources of difficulties in reconciling the semantics of independent data sources. First, information sources may be based on different data models. For example, information from a data source based on the relational model may need to be combined with information from a source based on an object-oriented model. Second, different information sources may contain different semantics reflecting the views and needs of the users for whom the source was developed. Third, the data may lack a well-defined schema based on a data model. For example, a semi-structured XML document may include untagged textual information.

Most of the existing work on semantic reconciliation has been in the area of traditional structured databases. Early work addressed schema integration and focused on the development of a unified global schema [1], [20]. Such an approach is termed *tightly coupled*. The practical difficulty of developing global schemas and the need to preserve component database autonomy led to *loosely coupled* multidatabase approaches that map component databases onto a canonical data model [12]. In addition to approaches that focus on data model structure, additional work has focused on: identifying types of semantic conflict and suggesting mechanisms for reconciling differences [15], [17]; using the notion of *semantic values* to provide a context for reconciliation [21]; reconciling information represented as data or as schema in different sources [14]; using statistical and other approaches to measure affinity or similarity to resolve heterogeneity [4], [5], [8], [10]; using semantic networks to detect and resolve heterogeneities among component databases [11]; and separating heterogeneity in the semantics of data from heterogeneity in the representation of data [19]. One large-scale application was work done on US Air Force flight operations databases by the MITRE Corporation [7]. That work focused on attribute matching and used similarities between data element names and statistics about data element values to identify similarities in meaning.

2.2 Semantic Interoperability in Semi-structured Sources

Research to support semantic heterogeneity in semi-structured sources originated in digital libraries [16], addressing such issues as developing query languages to search documents containing some structured elements [6]. More recent work has largely focused on resolving heterogeneity by developing and using specific domain ontologies onto which document concepts can be mapped [9], [13], [23].

In the remainder of this paper, we focus on resolving semantic heterogeneity in structured sources; however, we believe the approach can also be applied to support interoperability in semi-structured sources.

3 The Root Problem and a Candidate Resolution

Semantic reconciliation involves mapping and matching the meaning of terms in different sources. Since various sources might employ different data models that use different representation constructs (e.g. the relational model versus some object-oriented models) the same terms might refer to different constructs. This can be termed the *data model commitment* problem, and its solution requires the resolution of *structural* heterogeneity prior to the consideration of semantic heterogeneity.

The need to do semantic reconciliation emerges because sources might impose different views of the domain being represented. In particular, sources might differ in the choice of classes or types into which entities or objects are organized. Such models incorporate a *class commitment*, in which information about individual entities (or objects) is tied to the entity types (or classes) to which the entities (or objects) belong.

Anchoring representation to classes, while widely practiced, creates numerous problems in information modeling and database operations within a single source database. A possible solution is to separate the way instances and properties are represented in the database from any specific classification [18]. Classes can then be defined as a second "layer" above the instances and their properties. Similarly, we claim that semantic integration can benefit by alleviating the problems resulting from binding instances to classes. Thus, in the following we propose how to extend the two-layer foundation to support reconciliation of data from multiple sources.

Our approach is based on three observations:

Observation 1. Irrespective of the problems of data model commitment and class commitment, what is common to databases is that they represent information about individual entities or things [18]. This information can reflect either the state of entities or changes to this state. In either case, information is stored as *data items* reflecting *attributes* of entities.

Observation 2. Although data models exhibit class commitment, designers use attributes to identify similarity of entities. Entity types are represented in terms of attributes shared by groups of entities.

Observation 3. Data attributes in different sources that appear to be different may represent the same concept at a more generic level. For example, in two independent sources, the attributes 'customer number' and 'telephone number' may be used to realize the more general concept of an 'entity identifier'.

We contend that these three observations provide a basis for resolving semantic heterogeneity in both structured databases and semi-structured information sources.

In particular, they point to the importance of attributes for semantic reconciliation. This is consistent with the observation made by Clifton et al: "Perhaps the most basic problem in database integration is determining attribute correspondences: what attributes (e.g., columns in a relational table) in two databases reflect the same kind of real-world information" [7, p. 1]. Furthermore, since virtually all data models include attributes as a fundamental construct, focusing on attributes can also serve to alleviate the data model commitment problem mentioned above.

4 Property Precedence as a Basis for Semantic Reconciliation

4.1 Foundations

In order to reconcile and integrate independent information sources, the semantics or meaning of the data in each source must be established. The meaning of data in a database can only be established in terms of what the data represent. We therefore begin by positing the following:

Premise. The contents of databases represent properties of individual things or changes to these properties.

Based on this premise, we turn to *ontology*, which deals with concepts for describing 'what is out there' – in particular, things and their properties. Since we want our approach to be general (independent of any specific domain), we use a generalized ontology of concepts (as opposed a specific domain ontology for some subject matter). We have chosen to adapt the Bunge-Wand-Weber (BWW) ontology, which provides such a generalized set of concepts [2], [3], [26], [28]. Moreover, the BWW ontology has been used to analyze and gain insights into a variety of information systems and database concepts [18], [24], [25], [26], [27], [28].

In this subsection we present the ontological basis for our approach following Parsons and Wand [18].[1] Since we are interested in reconciling information sources, we use our observations about the importance of attributes as a guide for choosing the relevant ontological concepts. Nevertheless, the concepts below are ontological and refer to "real world" domains, not to information sources.

Postulate*. The world is made of *things* that possess *properties*.

This postulate can be viewed as the underlying ontological justification for the observation (see Section 3) that, regardless of data model, databases contain information about the properties of things. We assume databases represent human perceptions of existing or possible worlds; hence things may have a real existence or may be perceived. For example, both an existing and a planned product are considered things.

Principle*. There are no things without properties and *properties are always attached to things* [2, pp. 36, 58, 62].

Based on the observation (see Section 3) that attributes provide the basis for determining the similarity of things in establishing entity types or classes, when

[1] Postulates and definitions adapted from Bunge's model are indicated with *.

organizing information there is a special interest in identifying which things possess which properties.

Definition*. The *scope* of a property, P, is the set of things possessing the property, denoted Scope(P).

Definition*. The *form* of a thing, x, is the set of properties possessed by the thing, denoted Form(x).

The key to our approach to semantic reconciliation is the observation (see Section 3) that attributes that appear to be different at one level can be regarded as the same at a more abstract level. This idea is formalized via the notion of *property precedence*.

Definition*. Let P_1 and P_2 designate two properties. P_1 will be said to *precede* P_2 iff every thing possessing P_2 also possesses P_1.

It follows from the definition that P_1 precedes P_2 if and only if the $\text{Scope}(P_2) \subseteq \text{Scope}(P_1)$.

For example, the property 'move on land' precedes the properties 'walk' and 'run' since every thing that can walk and every thing that can run, can move on land. The set of things that run is a subset of the set of things that can move on land.

Definition. Let $\boldsymbol{P}$ denote a set of properties, $\wp(\boldsymbol{P})$ the power set (set of all subsets) of $\boldsymbol{P}$. The *preceding properties* of a property P in $\boldsymbol{P}$ are all properties that must be possessed by every instance possessing P. That is, the function Preceding: $\boldsymbol{P} \rightarrow \wp(\boldsymbol{P})$, such that $\text{Preceding}(P)=\{Q \in \boldsymbol{P} \mid \text{Scope}(Q) \supseteq \text{Scope}(P)\}$.

Definition. The *preceded properties* of a property P are all properties for which P is a preceding property. That is, the function Preceded: $\boldsymbol{P} \rightarrow \wp(\boldsymbol{P})$ such that $\text{Preceded}(P)=\{Q \in \boldsymbol{P} \mid \text{Scope}(P) \supseteq \text{Scope}(Q)\}$.

Recall our observation that a class (or type) in an information source is represented in terms of a set of attributes. All instances of a class possess these properties. Precedence has special importance in the context of the relationship between properties and classes (or entity types). Often, classes are described by some *generic* properties, while the instances possess *specific* different properties implying the generic ones. We say that the specific property *manifests* the generic one.

There are two main types of manifestation. First, a given generic property can be manifested by a specific value (termed *value manifestation*). For example, having a given weight is preceded by the property 'has weight.' Each instance having a specific weight value can be a member of a class that includes in its definition 'has weight.' Second, the generic property can be specialized in different ways (termed *specialization manifestation*). For example 'moves on land' may be manifested as 'crawls,' 'walks,' 'hops,' or 'runs.' Animals propelling in one of these ways can be members of a class that includes in its definition the property 'moves on land.' Hence, we define:

Definition. *A manifestation* of P in x is a property of x preceded by P. *The manifestation* of P in x is all properties of x preceded by P: $\text{Manifest}(P,x) = \text{Preceded}(P) \cap \text{Form}(x)$.

As noted, classes are often defined by generic properties (e.g. 'having legs') while instances have specific properties (e.g. 'having two legs'). If a class includes in its definition a property P, usually its instances will possess properties preceded by P, each a manifestation of the property P for the instance. When all manifestations are taken from the same domain, the property of a given instance can be considered the *value* of the generic property P (used to define the class) for the instance. Thus, manifestation can be considered a *generalization of the notion of value* of a given (generic) property. This generalization enables a representation of the specific properties of an instance in terms of generic (preceding) properties of classes. For example, 'has four legs' can be represented as ('has legs', 4), and 'being able to propel by crawling' can be represented as ('propel on land', 'crawl').

4.2 Toward a Precedence Algebra

As discussed above, classes are often defined in terms of generic properties while instances possess manifestations of these properties. One source might contain properties that are more generalized than the other. For example, a marketing database might contain a field 'age group' while a demographic database might contain a field 'age' (likely represented as birth date).

The following two lemmas establish some relationships between the observations that can be made about general properties and the observations that can be made about their manifestations.

Definition. A property G is termed *fully manifested* by a set of preceded properties S iff every thing that possesses G possesses at least one of the properties in S.

For example, if there are only four ways for animals to propel on land, (i.e., S is the set {'crawl', 'walk', 'hop', 'run'}), then the property 'propel on land' (G) is fully manifested by S.

Lemma 1. Let $\mathbf{S_i}$ be a set of preceded properties of a property G_i. If G_2 is fully manifested by $\mathbf{S_2}$, and for each property in $\mathbf{S_2}$ there is a preceding property in $\mathbf{S_1}$, then G_1 precedes G_2.

This lemma enables the inference of precedence between general properties based on the relationships between their manifestations. For example, given a marketing database containing the field 'age group' and a demographic database containing the field 'age', we may be interested in knowing if there is a precedence relation between 'age group' and 'age.' If there is a mapping from every age (e.g, 3) to a specific age group (e.g., 'toddler'), then 'age group' precedes 'age'. Note that there generally will not be a mapping from 'age group' to 'age'.

Lemma 2. Let $\mathbf{S_i}$ be a set of preceded properties of a property G_i. Let G_1 be a property for which all instances have manifestation(s) (in $\mathbf{S_1}$). If G_2 is preceded by G_1, then each property in $\mathbf{S_2}$ is preceded by one or more properties in $\mathbf{S_1}$.

This lemma enables the inference of precedence between manifestations based on the relationships between their general properties. For example, suppose that 'participating in an academic program' (G_2) is preceded by 'being a student' (G_1). Suppose further that the manifestations of G_1 are 'undergraduate' and 'graduate' and that manifestations of G_2 are {'accounting', 'engineering', 'MBA', 'counseling'}. Lemma 2 enables us to infer that a student in one of the programs must be (at least) a graduate or undergraduate student. While this conclusion might seem trivial when dealing with a single source, it might not be so when the information comes from two sources (e.g. a database of all who study in a university, and a source listing just those students enrolled as graduates or undergraduates).

5 Using Precedence to Support Interoperability

5.1 Operational Similarity

The concept of precedence provides a useful foundation for supporting the joint use of information from several sources. Consider an application that needs to use data from several sources having different schemas. Combining data from the various sources will require the identification of data fields in the sources that have the same meaning. According to our assumptions, the data in each source either describe properties of things (instances), or changes to these properties. Thus, we focus our attention on identifying properties that are "similar" in some sense.

We propose that two properties, P_1 and P_2, will be considered *similar* if there exists a property G that precedes both. If P_1 and P_2 are similar in that sense, they can be represented in the joint application as: (G, value=P_1) and (G, value=P_2), or, simply (G, P_1) and (G, P_2). Recall, the notion of value we are using is generalized, as it does not imply all values are taken from the same domain. For example, a graduate student can be defined in terms of having an advisor or in terms of being enrolled in some program. Using this approach we can then consider G as the common property represented in the data, and P_1, P_2 its manifestations. In turn, P_i can be further manifested by a value (for example, the specific program in which the student is enrolled).

To see how this approach can be used to integrate data, assume there are two sources related to customer activity. In one source, customer activity is described in terms of value of goods purchased per year. In the other source customer activity is described in terms of number of purchase transactions per year. One can query the two databases by asking: "what is the activity of customer x?" To describe the possible answer, we use a notation whereby a more generic property appears to the left of the less generic property (i.e., the manifestation (or value) can be embedded). In the customer example, depending on the source, the answer for the above query can be given in two ways: (activity, (value-of-goods, v)) or (activity, (number-of-purchases, n)).

Furthermore, assume whether a customer is preferred or not depends on the level of the activity (where the level is defined according to the manifestation of activity in the source). If one wants to inquire about all active customers, then the clause "where customer-status = 'preferred'" can be parsed in one source to 'value >…' and in the other to 'number > …'. Moreover, if one wants to list the status for all customers, the status can be displayed with the complete list of manifestations, e.g. ('preferred', (value, v)). The list appearing to the right of a property can be termed the *evidence*, as it shows how a higher-level property is obtained via a precedence sequence.

5.2 A Precedence-Based Methodology for Supporting Interoperability

In order for the notion of precedence to serve as a practical basis for reconciling sources to support interoperability, the sources will need to be examined to identify precedence relationships. The following steps constitute a general method for semantic reconciliation:

1. Identify all intra-source precedences for each available source.
2. Identify all inter-source precedences (i.e., where properties in one source precede or are preceded by properties in another).
3. In cases where there are inter-source precedences, use the two lemmas (subsection 4.2) to infer additional precedences or to seek precedences that might exist.
    a. Lemma 1 can indicate the highest-level property that can be used for reconciliation. In cases where the conditions of Lemma 1 arise, identify G_1 for reconciliation (it is the more generic).
    b. Lemma 2 can be used to indicate that lower level properties in one source can be viewed as manifestations of lower level properties in the other.

5.3 Queries in a Precedence-Based Environment

In the following, we demonstrate what could be the results of a query applied to two sources. Let the data in source 1 represent values (manifestations) of the generic property G_1 and the data in source 2 represent manifestations of the generic property G_2. Assume G_1 precedes G_2.

Consider the query: “Find all G_1 values (for instances possessing G_1)”. Table 1 shows what the results might look like.

Table 1. Possible results of querying two sources

Source	Value and evidence*	Comments
1	$S_1(1)$	G_1 is the defining property. Need not be reported.
1	?	No manifestation of G_1 (although the instance possesses it)
2	$(S_1(2),(G_2,S_2(3))$	G_1 precedes G_2 and the manifestation $S_2(3)$ implies $S_1(2)$
2	$(G_2,?)$	Because G_1 precedes G_2 it is known that the instance possesses G_1, but the value of S_1 is not known because S_2 is not known.

* $S_i(k)$ designates the k-th value of the possible manifestations of property G_i

To illustrate, we return to the student example mentioned earlier. In source 1, all instances are students (G_1). Source 2 contains the program of studies of a person (G_2). It is known that only students can be assigned a program of studies. Let:

S_1 = {undergraduate, graduate}
S_2 = {‘accounting’, ‘engineering’, ‘MBA’, ‘counseling’}

Lemma 2 indicates that knowing the value of S_2 can, in principle, provide the value of S_1. Assume further analysis has found that 'accounting' and 'engineering' are undergraduate programs, and 'MBA' and 'counseling' are graduate programs. Now, consider the query: "list for each student whether they are graduate or undergraduate." Substituting in Table 1 the specific values of the two sources, we obtain:

Table 2. Possible results of querying two sources about students

Source	Value and evidence*	Comments
1	$S_1(1)$ undergraduate	Being a student is the defining property. The value for S_1 is in the record in source 1 for this student.
1	?	Whether the student is graduate or undergraduate is unknown for this student.
2	$(G_1,(S_1(2),G_2,S_2(3)))$ (graduate, (program,'MBA"))	Being in the MBA program implies being a graduate student (the possibility of the inference follows from Lemma 2).
2	(is in a program, ?)	This person attends a program, thus is a student, however, the program is not known.

*$S_i(k)$ designates the k-th value of the possible manifestations of property G_i

5.4 An Example

Consider the need to reconcile data from two sources. Assume source 1 includes records of people who made purchases, but that only some are considered active customers and have a customer number, implying having the property 'has an account.' Assume source 2 also includes records of people who have made purchases, but includes 'time of last purchase.' Further assume that in source 2, purchasers are considered active customers if time since last purchase is less than three months.

If one now lists the properties of all people who made purchases, one possible generic property will be 'being active.' There are two ways in which this property can be manifested: In source 1: ('active', 'has an account'), and in source 2: ('active', 'made purchase in the past three months'). Thus, this representation enables querying the interoperated application on the property 'active'.

Assume there are three other generic properties. First, source 1 uses the property 'customer number' to identify some instances, while source 2 uses the property 'telephone number' to identify some instances. These two types of unique identifiers can be reconciled using a generic property 'id' with two manifestations – in source 1 (id, 'customer number') and in source 2 (id, 'telephone number'). If one inquires about the identity of a purchaser, depending on the source, the answer can be: (id, ('customer number', '12345')) or (id, ('telephone number', '987-6543')).

The case of id is different than the previous one (of 'active' customer), as 'customer number' and 'telephone number' are not actual values, but a form of *intermediate* property (i.e., each has both preceding and preceded properties).

Second, assume source 1 uses the property 'age group' with values ('toddler', 'child', 'youth', 'young adult', 'adult', 'mature adult', 'senior'), while instances in source 2 have the property 'age' with a specific value. These properties can be reconciled by recognizing that a given value in 'age group' precedes (i.e., is a generic representation of) a group of values of 'age.' Thus, data in source 1 can be represented as ('age group', 'child'), ('age group', 'youth') and so on, while data in source 2 can be transformed and represented as (age group, ('toddler' (age, '1'))),. The sources can then be reconciled through the generic representation in source 1 ('age group', 'ag'), where 'ag' denotes the age group and the evidence in terms of actual age in source 2. In this example, the sources contain different levels of granularity (age groups versus specific ages), but they are still able to interoperate. Furthermore, there is access to the more detailed data from source 2.

Third, assume source 1 includes 'sales person id' while source 2 includes 'sales team number'. These can be reconciled as manifestation of a preceding property: 'sales contact'.

Based on the four properties, one can create a class:

Customer: (id, active, sales contact, age group)

Specific instances or records in source 1 can be represented as:

((id, (customer number, n)), (active, 'has customer id'), (sales contact (sales person id, p)), (age group, g)), where n, t, g, are specific values.

Specific instances or records in source 2 can be represented as:

((id, (telephone number, tn)), (active, (last purchase, 'less than 3 months')), (sales contact, (sales contact (team, t))), (age group,(g, (age, a))), where tn, p, and a are specific values.

This approach achieves what can be termed *manifestation independence*. That is, it supports using information from multiple sources independent of how generic properties are manifested in the specific sources.

5.5 Implementation Issues

The application of the property-precedence approach to semantic reconciliation requires a property precedence schema. Within a single data source, precedences can be identified in terms of subsumed scopes. The scopes can be computed by identifying, for each property, all instances possessing it (as suggested in the two-layer approach of Parsons and Wand [18]). We term the precedences within a source the *local precedence schema*. However, it might not be possible to infer all precedences from the data. The data may not be complete, leading to "spurious" precedences that will disappear when more instances are known. In addition, there might exist additional knowledge about property precedence that was not used originally to design the database. Finally, the inter-source precedences cannot be deduced by scanning multiple sources, as the data will need to be reconciled first, and this is the issue we are trying to solve.

An approach to overcome this potential problem is to define, for a given domain, a generic schema of accepted property precedences. Such a schema can be considered the *property ontology* of the domain. It captures an important aspect of the domain semantics. We term such a schema *global precedence schema*. A global precedence schema can be used to identify cases where a preceding property cannot be identified in the local precedence schema because it is not directly relevant in a specific database. Moreover, as more sources are interoperated, such a schema can evolve to include more generic properties.

In a practical situation, such a global precedence schema might be operated by an independent party and act as a *precedence broker*. A multi-source application can identify possible generic properties by querying the precedence broker about precedences within and across sources.

To demonstrate the use of a precedence broker, consider two sources about species of animals native to different countries. These sources keep information about physical characteristics such as limbs. In source A, each animal has a property indicating the number of limbs (e.g., 'four limbs'), which is preceded by the property 'has limbs.' Source B includes the generic properties 'has wings,' 'has arms,' and 'has legs'. These generic properties in turn precede manifestations such as 'two arms' or 'four legs.'

Consider a query to both sources about all species having a specified number of limbs. Processing the query will require reference to the precedence relations explicitly defined or implicit in each source. In source A, the precedence relation 'has four limbs' $\rightarrow$ 'has limbs' (where $\rightarrow$ means 'is preceded by') will be explicit, since both properties are defined. Source B, however does not contain a property meaning 'has limbs.' Instead, it only contains precedences such as 'has two wings' $\rightarrow$ 'has wings,' 'has four legs' $\rightarrow$ 'has legs,' 'has two arms' $\rightarrow$ 'has arms.' Thus, within the context of this source, a query specifying the number of limbs is meaningless.

To facilitate information sharing between these sources, it may be necessary to refer to a precedence broker containing generic precedence relations. For example, the broker may contain precedences such as 'has legs' $\rightarrow$ 'has limbs,' 'has wings' $\rightarrow$ 'has limbs,' and 'has arms' $\rightarrow$ 'has limbs.' These precedences will enable inferring that instances in source B that possess properties preceded by 'has limbs' should be retrieved in any query involving the property 'has limbs,' even though source B does not contain that property.

6 Conclusions and Further Research

We propose the concept of property precedence as a novel approach to support the reconciliation of semantics of properties across independent data sources. The approach can be used in both traditional structured databases and in emerging semi-structured independent information sources on the Internet.

Further research is necessary to formalize the model and to more fully understand its capabilities. The approach should be tested on a large practical example. As well, work is needed to explore ways to implement the approach, to develop domain specific property precedence schemas, and to evaluate their usefulness. As well, there are various additional kinds of precedence to be analyzed, in particular those related to different manifestations of the same property. As well, mutual properties (relating two instances) need to be explored. From an implementation point of view, XML-style tagging can be explored as a way to represent precedence orders within information sources. The precedence broker architecture needs to be designed and evaluated with respect to feasibility and practicality.

References

[1] Batini, C., M. Lenzerini, and S.B. Navathe, “A Comparative Analysis of Methodologies for Database Schema Integration,” *ACM Computing Surveys*, 18(4), 1986, 323-364.

[2] Bunge, M., Treatise on Basic Philosophy (Volume 3), Ontology I: The Furniture of the World, Boston: Reidel, 1977.

[3] Bunge, M., Treatise on Basic Philosophy (Volume 4), Ontology II: A World of Systems, Boston: Reidel, 1979.

[4] Castano, S., V. De Antonellis, M. Fugini, and B. Pernici (1999), “Conceptual Schema Analysis: Techniques and Applications,” *ACM Transactions on Database Systems*, 23(3), 286-333.

[5] Castano, S., V. De Antonellis, and S. De Capitani di Vimercati (2001), “Global Viewing of Heterogeneous Data Sources,” *IEEE Transactions on Knowledge and Data Engineering*, 13(2), 277-297.

[6] Clarke, C., G. Cormack, and F. Burkowski (1995), “Schema-Independent Retrieval from Heterogeneous Structured Text,” in *Fourth Annual Symposium on Document Analysis and Information Retrieval*, 279-289.

[7] Clifton, C., E. Housman, and A. Rosenthal (1997), “Experience with a Combined Approach to Attribute-Matching Across Heterogeneous Databases,” *Data Mining and Reverse Engineering. Proceedings of DS-7*, IFIP 1997.

[8] Cohen, W. (1998), “Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity,” *Proceedings of the Conference on Information and Knowledge Management (CIKM’98)*, Seattle, WA, 201-212.

[9] Goble, C., S. Bechhofer, L. Carr, D. De Roure, and W. Hall (2001), “Conceptual Open Hypermedia = The Semantic Web?” *Proceedings of the 2001 Semantic Web Workshop*, Hong Kong, China, 7 pages.

[10] Hulgeri, A., G. Bhalotia, C. Nakhe, and S. Chakrabarti (2001), “Keyword Search in Databases,” *IEEE Data Engineering Bulletin*, 24(3), 22-32.

[11] Lee, J.-O. and D.-K. Baik (1999), “SemQL: A Semantic Query Language for Multidatabase Systems,” *Proceedings of the Conference on Information and Knowledge Management (CIKM’99)*, Kansas City, MO, 259-266.

[12] Litwin, W., L. Mark, and N. Roussopoulos (1990), “Interoperability of Multiple Autonomous Databases,” *ACM Computing Surveys*, 22(3), 267-293.

[13] Maedche, A. and S. Staab (2001), “Learning Ontologies for the Semantic Web,” *Proceedings of the 2001 Semantic Web Workshop*, Hong Kong, China, 10 pages.

[14] Miller, R., “Using Schematically Heterogeneous Structures,” *Proceedings SIGMOD’98*, Seattle, WA, 189-200.

[15] Naiman, C. and A. Ouksel, “A Classification of Semantic Conflicts in Heterogeneous Information Systems,” *Journal of Organizational Computing*, 5(2), 1995, 167-193.

[16] Paepcke, A., C.-C. Chang, H. Garcia-Molina, and T. Winograd (1998), “Interoperability for Digital Libraries Worldwide,” *Communications of the ACM*, 41(4), 33-43.

[17] Parent, C., S. Spaccapietra (1998), "Database Integration: an Overview of Issues and Approaches," *Communications of the ACM*, vol. 41, no 5, pp. 166-178, May 1998.

[18] Parsons, J. and Y. Wand, (2000), "Emancipating Instances from the Tyranny of Classes in Information Modeling," *ACM Transactions on Database Systems*, 25(2), 228-268.

[19] Qian, X. (1993), "Semantic Interoperation Via Intelligent Mediation," Proceedings of the 1993 International Workshop on Research Issues in Data Engineering, 228-231.

[20] Reddy, M.P., B.E. Prasad, P.G. Reddy, and A. Gupta, "A Methodology for Integration of Heterogeneous Databases," *IEEE Transactions on Knowledge and Data Engineering*, 6(6), 1994, 920-933.

[21] Sciore, E., M. Siegel, and A. Rosenthal (1994), "Using Semantic Values to Facilitate Interoperability Among Heterogeneous Information Systems," *ACM Transactions on Database Systems*, 19(2), 254-290.

[22] Smith, K. and L. Obrst, "Unpacking the Semantics of Source and Usage to Perform Semantic Reconciliation in Large-Scale Information Systems," *ACM SIGMOD Record*, 28(1), 1999, 26-31.

[23] Toivonen, S. (2001), "Using RDF(S) to Provide Multiple Views into a Single Ontology," *Proceedings of the 2001 Semantic Web Workshop*, Hongkong, China, 6 pages.

[24] Wand, Y., "A Proposal for a Formal Model of Objects," in W. Kim (ed.) and F. Lochovsky (eds.), *Object-oriented Concepts, Databases and Applications*, Reading, MA: Addison-Wesley, 1989, 537-559.

[25] Wand, Y., V. Storey, and R. Weber, "An Ontological Analysis of the relationship Construct in Conceptual Modelling", *ACM Transactions on Database Systems*, Vol. 24, No. 4, December 1999, pp. 494-528.

[26] Wand, Y. and Weber, R., "An Ontological Model of an Information System," *IEEE Transactions on Software Engineering*, 16(11), November 1990, 1282-1292.

[27] Wand, Y. and Weber, R., "On the Ontological Expressiveness of Information Systems Analysis and Design Grammars," *Journal of Information Systems*, 1993, 217-237.

[28] Wand, Y. and Weber, R., "Towards a Theory of Deep Structure of Information Systems," *Journal of Information Systems*, 1995, 203-223.

Conceptual integration of multiple partial geometric models

Simone Santini[1] and Amarnath Gupta[2]

[1] Bio-Informatics Research Network, Department of Neuroscience, University of California San Diego 9500 Gilman Drive, La Jolla, CA 92093-0715, USA
ssantini@ncmir.ucsd.edu,
WWW home page: http://columbo.ucsd.edu

[2] San Diego Supercomputer Center, University of California San Diego 9500 Gilman Drive, La Jolla, CA 92093-0715, USA
gupta@sdsc.edu

Abstract. Many scientific databases need to manage complex 3D geometric data such as models of the cerebral cortex. Often the complexity of the data forces users to construct muliple, simpler representations, which cover the real, complete model only partially and approximately. In order to recover the original information one needs to integrate these partial and incomplete models. In this paper, we first develop a conceptual model for objects and relationships in 3D geometric data, as well as their partial and approximate representations. We then establish mapping relationships between different approximations of the same data. Finally, we present a geometric information integration technique that will perform the integration where possible, and determine, for some cases, when the integration cannot be performed.

1 Introduction

In this paper, we study a general version of the problem of answering queries using views ([8, 1, 2], see [6] for a recent survey) that arises in certain scientific application that deal with complex geometric data models. Location-sensitive data models were studied, for instance, in [7]. Our application context is that of brain volume rendering, like magnetic resonance imaging (MRI), which are used for studying abnormalities of the brain, planning surgery, and diagnosing. Often these studies involve comparing the structure of certain brain regions under different treatments, or for different patients. Informally expressed, a typical query on a brain scan database is the following: *Find the average thickness of region A for patients over age 60, having Alzheimer's disease, and showing no shrinkage within 0.5 cm around region B*

The cortex is a very complex geometric object, containing numerous folds, troughs and crevices, making the measurement of even seemingly simple properties like distance between two points very complicated and inefficient to compute. Consequently, different research groups have developed software to "simplify" the cortex geometry, by mapping the original cortex data to geometrically simpler surfaces such as the sphere or the plane. Different representations may be used to label, measure and visualize different properties of the cortex geometry. The convenience of using simplified representations, of course, comes at a price: The distortion introduced during simplification may allow us to retrieve the original value of a property only after some mathematical computation and, often, within a certain error. More importantly, every simplification *preserves* only certain properties of the original data, while others are lost. For example, turning a cortex into a sphere preserves distances but loses the local curvature of the cortical surface. Given an arbitrary query, it is necessary to first find a set of usable representations from which each property (i.e., attribute) referred to in the query can be faithfully retrieved. Thus, the problem of answering queries using multiple representations, is akin to the problem of answering queries using views, but with the important distinction that the version of the attribute seen in a representation, is not identical to the original attribute value but *a functional correlate* of it.

The problem of re-writing queries using view has attracted a lot of attention from researchers coming from two different perspectives: query optimization and data integration [6], the point of view presented in this paper being closer to the latter perspective. The main difference between the work presented here and the existing research can be best summarized by making reference to a sentence from [6]: talking informally of the conditions under which a given view can be useful in answering a query, Halevy writes: *a view can be useful for a query if the set of relations it mentions overlaps with that of the query, and it selectes some of the attributes selected by the query.*

It is our contention that, in some scientific (and, more generally, numerical) applications, this condition is too restrictive. In many cases of practical interest (including our scenario), some of the attributes requested in the query are not available from any of the representations, but can be computed starting from attributes available from one or more representations. Taking a functional view of representations allows us to include these cases.

In this paper, we reformulate the answering queries using views problem with two modifications. On one hand, we allow for partial representations which are not subsets of the conceptual model but which are related to it functionally. On the other hand, our model allows to include considerations of geometric invariance in the representation: it may be possible in some cases to compute a function both in the representation as in the conceptual model, but the two functions can't be related because the transformation between model and representation doesn't have the required invariance properties.

2 Preliminaries

Our domain model is a simple entity-relationship model, similar to that in [4], $M = (E, R, A)$, where $E = \{E_1, \ldots, E_n\}$ is the set of entity types, $R = \{R_1, \ldots, R_m\}$ is the set of relations, $A = \{a_{ik} : E_i \to T_{ik}\}$ is the set of entity attribute functions. For the sake of simplicity, we will not consider attributes on relations. An n-ary relation will be represented as a monic from an index set to the product space of the entity types it associates:

$$R_i : d \longrightarrow E_{i,1} \times \cdots \times E_{i,r_i} \tag{1}$$

In the expression of a query involving the relation R_i we will often eliminate the existential quantifier over the index d, and use the notation

$$R(e_1 : E_1, \ldots, e_n : E_n) \equiv \exists d : R(d) = (e_1, e_2, \ldots, e_n) \tag{2}$$

Finally, we will consider *conjunctive* queries of the type

$$\begin{aligned}(a_1, \ldots, a_a, \beta_1, \ldots \beta_b) \leftarrow R_1(e_{11} : E_{11}, \ldots, e_{1m} : E_{1m}), \ldots \\ R_n(e_{n1} : E_{n1}, \ldots, e_{nm} : E_{1m}), c_1, \ldots, c_p\end{aligned} \tag{3}$$

where the c_i's are conditions on the attributes of relations or entities. Note that we don't allow relation variables, but only entity variables.

The model we will use in the paper is a simple model of the cerebral cortex: in which the cortex is represented as a surface (immersed in the three-dimensional Euclidean space) divided into regions; selected points of each region have associated histological data, such as neuron density measurements, or data about the neurons found in that region.

3 Representations

A representation $\hat{M}^q$ of a model M is defined as $\hat{M}^q = (\hat{E}^q, \hat{R}^q, \hat{A}^q)$. $\hat{E}^q = \{\hat{E}_1^q, \ldots, \hat{E}_{n_q}^q\}$ is the set of entity representations, $\hat{R}^q = \{\hat{R}_1^q, \ldots, \hat{R}_{m_q}^q\}$ is the set of relation representations, and $\hat{A} = \{\hat{a}_{ik}^q : \hat{E}_i^q \rightarrow \hat{T}_{ik}^q\}$ is the set of attribute representations.

Definition 1. *A representation $\hat{M}^q$ is* legal *if there exists a morphism ϕ^q such that $\phi^q : E_i \rightarrow \hat{E}_i^q$, $\phi^q : R_i \rightarrow \hat{R}_i^q$, $\phi^q : a_{ik} \rightarrow \hat{a}_{ik}^q$, and such that*

– *For every $\hat{R}_i^q$ the following diagram commutes:*

$$\begin{tikzcd} d \arrow[r, "R_i"] \arrow[d, "id"'] & E_1 \times \cdots \times E_p \arrow[d, "\phi^q \times \cdots \times \phi^q"] \\ d \arrow[r, "\hat{R}_i^q"'] & \hat{E}_1^q \times \cdots \times \hat{E}_p^q \end{tikzcd} \tag{4}$$

– *For every a_{ik}^q there exist a function $f_{ik}^q : T_{ik} \rightarrow \hat{T}_{ik}^q$ such that the following diagram commutes:*

$$\begin{tikzcd} E_i \arrow[r, "a_{ik}"] \arrow[d, "\phi^q"'] & T_{ik} \arrow[d, "f_{ik}^q"] \\ \hat{E}_i^q \arrow[r, "\hat{a}_{ik}^q"'] & \hat{T}_{ik} \end{tikzcd} . \tag{5}$$

If ϕ^q is an isomorphism, then the model $\hat{M}^q$ is said to be *faithful*. Similarly to the morphism ϕ^q between the conceptual model and its representation, it is possible to define a morphism between two representations [3].

Definition 2. *ψ is a morphism between the representation $\hat{M}^q$ and the representation $\hat{M}^p$, represented as $\hat{M}^q \xrightarrow{\psi} \hat{M}^p$ if $\psi : \hat{E}^q_i \rightarrow \hat{E}^p_i$, $\psi : \hat{R}^q_i \rightarrow \hat{R}^p_i$, $\psi : \hat{a}^q_{ik} \rightarrow \hat{a}^p_{ik}$, for every $\hat{R}^q_i$ the following diagram commutes:*

$$\begin{tikzcd} d \arrow[r, "\hat{R}^q_i"] \arrow[d, "\psi"'] & \hat{E}^q_1 \times \cdots \times \hat{E}^q_n \arrow[d, "\psi\times\cdots\times\psi"] \\ d \arrow[r, "\psi\hat{R}^q_i"'] & \hat{E}^p_1 \times \cdots \times \hat{E}^p_n \end{tikzcd} , \qquad (6)$$

for every a^q_{ik} there exist a function $f^{qp}_{ik} : \hat{T}^q_{ik} \rightarrow \hat{T}^p_{ik}$ such that the following diagram commutes:

$$\begin{tikzcd} \hat{E}^q_i \arrow[r, "\hat{a}^q_{ik}"] \arrow[d, "\psi"'] & \hat{T}^q_{ik} \arrow[d, "f^{qp}_{ik}"] \\ \hat{E}^p_i \arrow[r, "\psi a^q_{ik}"'] & \hat{T}^p_{ik} \end{tikzcd} \qquad (7)$$

whenever ψa^q_{ik} is defined.

An ordering relation can be established between representations as follows:

Definition 3. *Given two representations $\hat{M}^q$ and $\hat{M}^p$, it is $\hat{M}^q \leq \hat{M}^p$ if, for every morphism $\hat{M}^q \xrightarrow{\psi} \hat{M}^p$, there is $\hat{M}^p \xrightarrow{\psi'} \hat{M}^q$ such that $\psi' \circ \psi = id$.*

This partial ordering between representations captures to a certain degree the notion of "structural representativity" of a representation.

In many cases, the representation of a relation $\hat{R}^q_i$ is only partial, in the sense that there exist tuples $e_1, \ldots, e_n$ for which $R_i(e_1, \ldots, e_n)$ hold, but such that the relation $\hat{R}^q_i$ does not hold for the corresponding tuple $(\hat{e}^q_1, \ldots, \hat{e}^q_n)$.

Example

> Consider a model of the cerebral cortex, and a representation $\hat{M}$ of the cortical surface limited to the occipital cortex. The relation adjacent(e_1, e_2) will only be represented if e_1 and e_2 are in the occipital cortex.

This kind of structural restriction of a relation is captured by the concept of *defining condition*:

Definition 4. *Given a relation R_i and a representation $\hat{R}_i^q$ of that relation, let P_i^q be a predicate on the entities that R_i relates, and let $id_{|P_i^q} : E_1 \times \cdots \times E_n \to E_1 \times \cdots \times E_n$ be the restriction of the identity of $E_1 \times \cdots \times E_n$ to the set of entities for which the predicate P_i^q is true. The predicate P_i^q is the* defining condition *for the representation R_i^q if it makes the following diagram commute:*

$$\begin{tikzcd} d \arrow[r, "R_i"] \arrow[d, "\phi^q"'] & E_1 \times \cdots \times E_n \arrow[r, "id_{|P_i^q}"] & E_1 \times \cdots \times E_n \arrow[d, "\phi^q \times \cdots \times \phi^q"] \\ \hat{d} \arrow[rr, "\hat{R}_i^q"'] & & \hat{E}_1^p \times \cdots \times \hat{E}_n^p \end{tikzcd} \tag{8}$$

In terms of sets, this means that the the co-domain of the relation representation $\hat{R}_i^q$ is

$$\mathrm{cod}(\hat{R}_i^q) = \{(\hat{e}_{i,1}, \ldots, \hat{e}_{i,n}) : \hat{e}_{i,j} = \phi^q(e_{i,j}) \wedge e_{i,j} \in \mathrm{cod}(R_i^q) \wedge P_i^q(e_{i,1}, \ldots, \hat{e}_{i,n})\} \tag{9}$$

Attributes that can't be computed directly from a representation can sometimes be computed indirectly through a representation morphism. Given a representation $\hat{M}^q$, the set of attributes directly computable from $\hat{M}^q$ is

$$A_0(\hat{M}^Q) = \{a_{ik} : \exists \hat{a}_{ik}^q = \phi^q(a_{ij})\} \tag{10}$$

In order to define the set of attributes that can be computed indirectly from the representation $\hat{M}^q$, it is first necessary to determine which relations can be reached from $\hat{M}^q$ in a given number of steps. The set of representations directly reachable from $\hat{M}^q$ in k steps (or *k-reachable* from $\hat{M}^q$) is:

$$R_k(\hat{M}^q) = \left\{ \hat{M}^r : \exists \psi_1, \ldots, \psi_k \, \hat{M}^q \overset{\psi_1 \circ \ldots \circ \psi_k}{\longrightarrow} \hat{M}^r \right\} \tag{11}$$

while the set of k-reachable attributes is

$$A_k(\hat{M}^q) = \left\{ a_{ik} : \exists \hat{M}^r \hat{a}_{ik}^r = \phi^r(a_{ij}) \wedge \hat{M}^r \in R_k(\hat{M}^q) \right\} \tag{12}$$

The set of representations and the set of attributes reachable from $\hat{M}^q$ will be indicated with $R_\infty(\hat{M}^q)$ and $A_\infty(\hat{M}^q)$ respectively.

From a functional point of view, an attribute a_{ik} is k-reachable from the representation $\hat{M}^{r_k}$ if there is a function h such that

$$\begin{tikzcd} & & & E_i \arrow[r, "a_{ik}"] \arrow[d, "\phi^{r_1}"'] & T_{ik} \arrow[d, "f"] \\ \hat{E}_i^{r_k} \arrow[r, "\psi^{k-1}"] \arrow[rrrr, bend right, "h"'] & \cdots \arrow[r, "\psi^2"] & \hat{E}_i^{r_2} \arrow[r, "\psi^1"] & \hat{E}_i^{r_1} \arrow[r, "\hat{a}_{ik}^{r_1}"] & \hat{T}_{ik}^{r_1} \end{tikzcd} \tag{13}$$

An attribute which is not in $A_\infty(\hat{M}^q)$ is said to be *unreachable* from $\hat{M}^q$, while an attribute which does not belong to $A_\infty(\hat{M}^q)$ for any available representation $\hat{M}^q$ is said to be *hopeless*. Any query involving hopeless attributes can't be answered given the available representations.

4 Query rewriting

In a typical database queries are given in terms of the conceptual model M. The database, on the other hand, contains but a series of representations $\{\hat{M}^1, \ldots, \hat{M}^r\}$ and a series of morphisms $\hat{M}^i \xrightarrow{\psi^{ij}} \hat{M}^j$ between the representations.

In a rather general interpretation, a query can be formally defined as a partial recursive function from databases to databases [5]. This definition assume that the conceptual database is always isomorphic to its representation. In our case, this is not true, and we must distinguish between an *conceptual query* and an *grounded query*. A conceptual query is a partial recursive function from the set of models to a *result schema* given by the signature of the set of attributes that the query requires If $S_r = (T_1, \ldots, T_r)$ is the result schema, a query is a function $Q : M \to S_r$. We will write a query as:

$$(a_{i_1 k_1}(\xi_1), \ldots, a_{i_u k_u}(\xi_u)) \leftarrow R_{j_1}(\zeta_{11}, \ldots, \zeta_{1n}), \ldots, R_{j_p}(\zeta_{p1}, \ldots, \zeta_{pn}), \\ c_1(\nu_{11}, \ldots, \nu_{1m}), \ldots, c_v(\nu_{v1}, \ldots, \nu_{vm}) \quad (14)$$

Note that, for the sake of simplicity, we have assumed that all the relations are n-ary, and that all the conditions depend on m variables. The variables ξ_i and ζ_{ij} take values in the set of entities, while the variables ν_{ij} take value in the set of attributes.

The *grounding* of a query over the set of representations $\{\hat{M}^1, \ldots, \hat{M}^m\}$ consists in a representation function $\phi^{i_1} \times \cdots \times \phi^{i_p}$ and a query $Q' : \hat{M}^{i_1} \times \cdots \times \hat{M}^{i_p} \to \hat{S}_r$ such that there is a function f for which the following diagram commutes:

$$\begin{tikzcd} M \arrow[r, "Q"] \arrow[d, "\phi^{i_1}\times\cdots\times\phi^{i_p}"'] & S_r \\ \hat{M}^{i_1} \times \cdots \times \hat{M}^{i_p} \arrow[r, "Q'"'] & \hat{S}_r \arrow[u, "f"] \end{tikzcd} \quad (15)$$

Query rewriting consists, given the model M and a query Q, in the determination of a suitable set of representations $\hat{M}^{i_1} \times \cdots \times \hat{M}^{i_p}$ and a query Q' which grounds the original query.

One way to transform the query Q into the query Q' in a way that maintains the commutativity of the diagram can be sketched as follows:

1. add formally a representation $\hat{M}^0$ isomorphic to M to the set of representation;
2. translate Q into a query Q'_0 expressed in terms of $\hat{M}^0$ (which is a grounding of Q because of isomorphism);

3. replace one relation of $\hat{M}^0$ with its representation taken from some $\hat{M}^i$; replace all the attributes in $\hat{S}_r$ that can be computed from $\hat{M}^i$ with the corresponding representations;
4. if no relations and no attributes are computed in terms of $\hat{M}^0$, stop, otherwise repeat the previous step.

The substitutions of step 2 come from a set of possible substitutions called *substitution opportunities*, defined as follows:

Definition 5. *A* substitution opportunity *is a query fragment including one relation an w constraints:*

$$R_k(\zeta_{k1},\ldots,\zeta_{kn}), c_1(\nu_{11},\ldots,\nu_{1m}),\ldots,c_w(\nu_{w1},\ldots,\nu_{wm}) \tag{16}$$

for which there is a representation $\hat{M}^q$ such that:

1. $\hat{R}^q_k = \phi^q(R_k)$ exists,
2. All the values of the variables that make $c_1,\ldots,c_w$ true also make the defining condition P^q_k true.

The substitution *of R_k with the representation $\hat{R}^i_k$ is indicated as $(R_k, c_1,\ldots,c_m) \rightarrowtail (\hat{R}^q_k, c_1,\ldots,c_m)$ or, if the conditions $c_1,\ldots,c_n$ can be omitted without causing confusion, as $R_k \rightarrowtail \hat{R}^q_k$.*

When we rewrite a query using a representation, it is important to guarantee that we don't lose the possibility of computing attributes. The following definition is related to the conditions under which this is guaranteed:

Definition 6. *A substitution*

$$(R_k(\zeta_1,\ldots,\zeta_n), c_1,\ldots,c_m) \rightarrowtail (\hat{R}^q_k(\zeta^q_1,\ldots,\zeta_n), c_1,\ldots,c_m)$$

is strictly non-disruptive *if the following condition is true:*

> *For each variable $\zeta_i : E_i$ that doesn't appear in any other relation but the relation R_k that is being substituted, all the attributes $a_{ij}(\zeta_i)$ that appear in the query can be computed from $\hat{R}^q_k$, that is, $a_{ij}(\zeta_i) = f(\hat{a}^q_{ij}(\zeta_i))$*

The substitution is weakly non-disruptive *if each attribute $a_{ij}(\zeta_i)$ can be computed indirectly from $\hat{R}^q_k$, that is, $a_{ij} \in A_\infty(\hat{M}^q)$*

The rationale of this definition is the following: if the only relation in which the entity type E_i appears is replaced with a representation which does not allow to compute the required attributes, then the attributes can no longer be computed. Non-disruptiveness guarantees that all the attributes that can only be computed from the replaced relation will be computable directly or indirectly from the representation that replaces it.

The following property is the key for the application of substitution operations:

Proposition 1. *Let $\hat{M}^0 \xrightarrow{Q} S_r$ be a query, and $\hat{M}^0 \times \hat{M}^1 \times \cdots \times \hat{M}^{n-1} \xrightarrow{Q'} \hat{S}_r$ be a grounding for it. Let $\hat{M}^0 \times \hat{M}^1 \times \cdots \times \hat{M}^{n-1} \times \hat{M}^n \xrightarrow{Q''} \hat{S}_r$ be a query obtained from Q' by a non-disruptive substitution $(R_k, c_1, \ldots, c_m) \rightarrowtail (\hat{R}_k, c_1, \ldots, c_m)$, then Q'' is a grounding of Q.*

Proof (sketch). Assume, for the sake of simplicity, that the relation R_k is binary: $R_k(\zeta_1, \zeta_2)$, with $\zeta_1 : E_1$, and $\zeta_2 : E_2$, and that the substitution contains a single condition c. Also, set $W = \hat{M}^1 \times \cdots \times \hat{M}^{n-1}$, so that the query Q' can be written as $\hat{M}^0 \times W \xrightarrow{Q'} \hat{S}_r$, and assume that the entity types E_1 and E_2 do not appear in any other relation.

The query Q can be written as the composition of two functions: $Q' = f \circ \sigma$, where σ is the selection fuction which selects the entities that satisfy the query condition, and f is the attribute computation function.

The function σ, in turn, can be decomposed as $\sigma = \sigma_0 \circ \sigma_k$. The function σ_k produces the set of variable assignments $\zeta_1 \leftarrow e_1 : E_1$, and $\zeta_2 \leftarrow e_2 : E_2$ that satisfy $(R_k, c_1, \ldots, c_m)$, with their variable bindings, while σ_0 will select those entities that satisfy all the other conditions and are compatible with the bindings of R_k. The function σ can be written as $\sigma : E_1 \times E_2 \times \cdots E_n \to E_1 \times E_2 \times \cdots E_n$.

After the substitution, the relation R_k is replaced by $\hat{R}_k$, which selects elements in $\hat{E}_1 \times \hat{E}_2$. Because of the properties of substitution, every pair in $E_1 \times E_2$ which make c true also make the defining condition of the representation true, therefore, for every pair (e_1, e_2) which would be selected by σ_k, there is a pair $(\hat{e}_1, \hat{e}_2)$ in $\hat{R}_k$. It is therefore possible to define a function $\hat{\sigma}_k : \hat{E}_1 \times \hat{E}_2 \to \hat{E}_1 \times \hat{E}_2$ which selects the pairs $(\hat{e}_1, \hat{e}_2)$ corresponding to the pairs (e_1, e_2) selected by σ_k, that is, a function $\hat{\sigma}_k$ such that

$$\begin{tikzcd}
E_1 \times E_2 \arrow[r, "\sigma_k"] \arrow[d, "\phi"'] & E_1 \times E_2 \arrow[d, "\phi\times\phi"] \\
\hat{E}_1 \times \hat{E}_2 \arrow[r, "\hat{\sigma}_k"'] & \hat{E}_1 \times \hat{E}_2
\end{tikzcd} \qquad (17)$$

Defining $\hat{\sigma} = \sigma_0 \circ \sigma_k$, one can show that $\sigma : \hat{E}_1 \times \hat{E}_2 \times \cdots E_n \to \hat{E}_1 \times \hat{E}_2 \times \cdots E_n$ and that the following diagram commutes

$$\begin{tikzcd}
E_1 \times \cdots \times E_n \arrow[r, "\sigma_k"] \arrow[d, "\phi"'] & E_1 \times \cdots \times E_n \arrow[d, "\phi\times\phi\times\mathrm{id}\times\cdots\times\mathrm{id}"] \\
\hat{E}_1 \times \hat{E}_2 \times E_3 \times \cdots \times E_n \arrow[r, "\hat{\sigma}_k"'] & \hat{E}_1 \times \hat{E}_2 \times E_3 \times \cdots \times E_n
\end{tikzcd} \qquad (18)$$

In other words, the selection function can be rewritten so that the output is composed of the same tuples of entity, short of a transformation ϕ.

Since the transformation is non-disruptive, it is possible to rewrite in the same way the attribute computing function f as $\hat{f}$ so that the following diagram commutes:

$$\begin{tikzcd}
E_1 \times \cdots \times E_n \arrow[r, "\sigma_k"] \arrow[d, "\phi"'] & E_1 \times \cdots \times E_n \arrow[r, "f"] \arrow[d, "\phi\times\phi\times\mathrm{id}\times\cdots\times\mathrm{id}"] & T_1 \times \cdots \times T_n \arrow[d, "\phi\times\phi\times\mathrm{id}\times\cdots\times\mathrm{id}"] \\
\hat{E}_1 \times \hat{E}_2 \times E_3 \times \cdots \times E_n \arrow[r, "\hat{\sigma}_k"'] & \hat{E}_1 \times \hat{E}_2 \times E_3 \times \cdots \times E_n \arrow[r, "\hat{f}"'] & \hat{T}_1 \times \hat{T}_2 \times E_3 \times \cdots \times E_n
\end{tikzcd} \tag{19}$$

In conclusion, the query rewriting problem outlined in this section can be cast in the following terms:

– Let M be a model, with representations $\hat{M}^1, \ldots, \hat{M}^r$; $M \xrightarrow{Q} S_r$ a query; $a = \{a_1, \ldots, a_k\}$ the set of attributes required by the query, $R = \{R_1, \ldots, R_n\}$ the set of query relations, and $\hat{R} = \{\hat{R}_1^1, \hat{R}_1^2 \ldots, \hat{R}_n^m\}$ the set of rewriting opportunity.
– find a subset $P \in \hat{R}$ such that: (1) for each $R_i \in R$ there is a ϕ^q such that $R_i^q = \phi^q R_i \in P$ and (2) for each $a_{ik} \in a$ there is $R_i^q \in \rho$ coming from a representation $\hat{M}^q$ such that $a_{ik} \in A_{k_i}(\hat{M})$ for some k_i.

5 Geometric Functions

Up to this point, we haven't quite considered the geometric nature of our data, but we have focussed on the structural properties of the representations, that is, on whether a given representation preserved the relations and the attributes of the conceptual model.

In particular, so far we have always considered that the conditions c_i were only on the value of attributes, and that the result of the query was also a set of tuples formed by atribute values. In geometric queries, however, one has often to compute functions that require the conservation of certain properties.

Example

In the cortex model above, each region has an attribute cent_i, of type Point representing the centroid of the region. A typical spatial query, then, might request all the regions whose centroid is within a certain distance from the centroid of a given region. That is, the query will contain a condition like

$$c_i(\nu_{i1}, \nu_{i2}) = d(\nu_{i1}\text{cent}, \nu_{i2}\text{cent}) \leq D \tag{20}$$

where D is a constant.

This example leads to a number of observations. First, the representation of the entities ν_{i1}, ν_{i2} requires not only that the representation contain the attribute cent, but also that it preserve the properties of the distance function.

By and large, every representation of regions will allow the computation of centroids, but there is a priori no guarantee that the distance between centroids in the representation will correspond to the distance between the centroids on the suface. Second, although the surface of the conceptual model is immersed in $\mathbb{R}^3$, for many applications (including brain modeling) the distance that is computed is not the $\mathbb{R}^3$ distance between the centroids of the regions, but the distance along the surface (more precisely: the lenght of the shortest surface geodesic that passes through the two points). The way in which this distance is computed, therefore, depends on the surface that is, on the representation. As a consequence, when the condition is rewritten, the distance function d will have to be replaced with a representation $d^q = \phi^q d$. The central concept for this type of replacement is that of *invariance.*

Definition 7. *Let* X, Y *be two spaces, and* $f : X^n \to Y$ *a function. Let* $\mathfrak{G} : X \to X$ *be a group of transformations on* X*. The function* f *is invariant with respect to the group* $\mathfrak{G}$ *if, for every* $g \in \mathfrak{G}$, $f \circ g^n = f$.

In particular, we are interested in invariance with respect to the following groups: the identity group (consisting only of the unit) $\mathfrak{I}$, the group of translations $\mathfrak{T}$, the group of direct isometries (rotation and translation) $\mathfrak{D}$, the group of homeomorphisms $\mathfrak{H}$, and the general permutation group $\mathfrak{P}$.

A function invariant to $\mathfrak{I}$ is the function that computes the coördinates of the center of a particulat region in a given reference system. A function invariant to $\mathfrak{T}$ is that which computes the orientation of a region with respect to a reference line. A typical example of a function invariant to $\mathfrak{D}$ is distance, while functions invariant to $\mathfrak{H}$ are, for instance, functions that determine whether two regions touch each other, or count the number of holes in a region.

When we go from the original model to a representation, the transform may fail to be invariant with respect to some of these groups. Consequently, functions that rely on the corresponding invariants can't be computed from the representation.

Example

A *flat map* is a projection of the cortical surface on a plane in a way that maintains, as well as possible, the area of certain anatomically relevant regions. Since the cortical surface is not topologically equivalent to a plane, when a flat map is created, *cuts* are introduced which may go across regions. The topological invariant *connectedness* and the isometric invariant *distance* are lost in this map: regions that are connected in the cortex may fail to be connected in the map, and it is impossible to recover cortical distances from the flat map. Therefore, every query containing predicates about the connectdeness of regions, or conditions on the distance between points can't be answered using the flat map.

The conditions under which this happen depend on the representation morphism ϕ^q, in particular, for an invariance involving the entity type E_i, on the function $\phi^q : E_i \to \hat{E}_i^q$. Consider, in the way of example, the case of distance computation. The situation is summarized by the following diagram:

$$\begin{tikzcd}
E_i \times E_i \arrow[r, "d"] \arrow[d, "g"'] \arrow[rr, bend left, "\phi^q"] & \mathbb{R} & \hat{E}_i^q \times \hat{E}_i^q \arrow[l, "\hat{d}"] \arrow[d, "g"] \arrow[dl, phantom] \\
E_i \times E_i \arrow[ur, "d"] \arrow[rr, "\phi^q"'] & & \hat{E}_i^q \times \hat{E}_i^q \arrow[ul, "\hat{d}"]
\end{tikzcd} \tag{21}$$

From the diagram it is clear that the representation of E_i will be distance-invariant if for all $g \in \mathfrak{G}$, $\phi^q = g^{-1} \circ \phi^q \circ g$, and $d = \hat{d} \circ \phi^q$. In general, if this is true for a group $\mathfrak{G}$, we will say that the representation is *$\mathfrak{G}$-covariant in the strong sense* (or *strongly $\mathfrak{G}$-covariant*).

In some cases, this condition is too restrictive: all we really need is that there be a way to compute the distance d starting from the distance $\hat{d}$, that is, that there exist a function u such that:

$$\begin{tikzcd}
E_i \times E_i \arrow[r, "d"] \arrow[d, "g"'] \arrow[rr, bend left, "\phi^q"] & \mathbb{R} & \hat{E}_i^q \times \hat{E}_i^q \arrow[dl, "\hat{d}"] \arrow[d, "g"] \\
E_i \times E_i \arrow[ur, "d"] \arrow[rr, bend right, "\phi^q"'] & \mathbb{R} \arrow[u, "u"] & \hat{E}_i^q \times \hat{E}_i^q \arrow[l, "\hat{d}"]
\end{tikzcd} \tag{22}$$

note that it follows from this diagram that if $d = \hat{d} \circ \phi^q$, then $u = \mathrm{id}$. In the general case, one requires that the function u be well defined and computable, that is, that it be expressible only as a function of $\hat{d}$ and ϕ^q. In general, if this is true for a group $\mathfrak{G}$, we will say that the representation is *$\mathfrak{G}$-covariant in the weak sense* (or *weakly $\mathfrak{G}$-covariant*).

Definition 8. *Let $c(\nu_1 : E_1, \ldots, \nu_n : E_n)$ be a condition in the query, which can depend on the computation of certain functions on the entity variables ν_i. Let $\mathfrak{g}(c)$ the group of transformations to which c is invariant. A rewriting $\{\hat{M}^1, \ldots, \hat{M}^m\}$ is $\mathfrak{g}$-*preserving *if the following conditions are true:*

1. *there is a representation $\hat{M}^p$ which contains a representation of all the entity types of c;*
2. *all the variables ν_i have been replaced with variables ν_i^p which take values in the representation $\hat{M}^p$;*
3. *the representation $\hat{M}^p$ is $\mathfrak{g}$-covariant, at least in the weak sense.*

Proposition 2. *Let Q be a query and Q' a rewriting so that for every condition c invariant with respect to a group $\mathfrak{G}$ the rewriting is $\mathfrak{G}$-preserving and such that for every attribute function a_{ik} invariant with respect to a group $\mathfrak{H}$, the rewriting is $\mathfrak{H}$-preserving. Then the query Q' is a grounding of Q.*

The proof is a repeated application of the invariance property, and is omitted.

6 The Query rewriting process

A query like that of (14) can be represented as in the following diagram

$$\begin{tikzcd}
 & & R_1 \arrow[dl] \arrow[d] \arrow[dr] & & R_2 \arrow[dl] \arrow[dr] & & R_3 \arrow[d] & \\
 & \zeta_1 \arrow[dl, "\mathfrak{e}"'] \arrow[d, "\mathfrak{G}_1"'] & \zeta_2 \arrow[d, "\mathfrak{e}"] & \zeta_3 \arrow[dl, "\mathfrak{e}"'] \arrow[d, "\mathfrak{e}"'] \arrow[dr, "\mathfrak{G}_2"] & & \zeta_4 \arrow[d, "\mathfrak{e}"] & \zeta_5 \arrow[dll, "\mathfrak{G}_2"'] \arrow[d, "\mathfrak{G}_3"] & \zeta_6 \arrow[dl, "\mathfrak{G}_3"] \arrow[d, "e"] \\
a_1 & c_1 & c_3 & a_2 & c_2 & a_3 & c_4 & a_4
\end{tikzcd} \tag{23}$$

where we have already identified all the common variables between the conditions, the relations, and the attribute computation inserting, if necessary, variable matching conditions of the form $c(\zeta_1, \zeta_2) = (\zeta_1 \equiv \zeta_2)$. The symbols $\mathfrak{G}_i$ on some of the arrows mean that the condition or attribute at the end of the arrow is invariant to the group $\mathfrak{G}_i$. The process of query rewriting consists of transforming this diagram, through a process of repeated substitutions of relations, into an equivalent diagram composed exclusively of representations. For the query (23), one such diagram is the following:

$$\begin{tikzcd}
 & & R_1^1 \arrow[dl] \arrow[d] \arrow[dr] & & R_2^1 \arrow[dl] \arrow[dr] & & R_3^1 \arrow[dl] \arrow[d] & & R_3^2 \arrow[dl] \arrow[d] \arrow[dr] & \\
 & \zeta_1^1 \arrow[dl] \arrow[d, "\mathfrak{G}_1"] & \zeta_2^1 \arrow[d] & \zeta_3^1 \arrow[dl] \arrow[d] \arrow[dr, "\mathfrak{G}_2"] & & \zeta_4^1 \arrow[d] \arrow[dr] & \zeta_5^1 \arrow[dll, "\mathfrak{G}_2"] \arrow[dr] & \zeta_4^2 \arrow[dl] & \zeta_5^2 \arrow[dl] \arrow[d, "\mathfrak{G}_3"] & \zeta_6^2 \arrow[dl, "\mathfrak{G}_3"] \arrow[d] \\
\hat{a}_1^1 & c_1^1 & c_3^1 & \hat{a}_2^1 & c_2^1 & \hat{a}_3^1 & \bowtie & \bowtie & c_4^2 & \hat{a}_4^2
\end{tikzcd} \tag{24}$$

In this diagram, the superscripts attached to conditions and attributes refer to the representation that is used to compute them. The diagram uses two representations: $\hat{M}^1$, which provides representations for the relations R_1, R_2, and R_3, and $\hat{M}^2$, which represents the relation R_3. Note that the relations R_3 is split between two representations: $\hat{R}_3^1$ is used to compute the condition c_2 and the attribute a_3, and the representation $\hat{R}_3^2$, which is used to compute the condition c_4 and the attribute a_4. While the entity type of the variable $\zeta_5 : E_5$ in the original relation is invariant to groups $\mathfrak{G}_2$ and $\mathfrak{G}_3$, in the derived diagram, the representation $\hat{E}_5^1$ is invariant only to $\mathfrak{G}_2$, and the representation $\hat{E}_5^2$ is invariant only to $\mathfrak{G}_3$. Since the relation is split, it is necessary to bind the variables that appear in both reprsentations (ζ_4 and ζ_5, in this case) so that they represent the same instance of the entity. This is done by introducing binding relations[3] $\bowtie$ between the variables ζ_4^1 and ζ_4^2 and between the variables ζ_5^1 and ζ_5^2. We assume that all the instances of all entities have a unique identifier so that the binding condition can be written as $\zeta_4^1.\mathrm{id} = \zeta_4^2.\mathrm{id}$.

The passage from one diagram to another is done through *legal substitutions*, changes that leave the semantics of the query unchanged, while reducing the presence of the conceptual model and introducing representations of the various relations. Legal substitutions belong to four groups, which we call α-substitutions, β-substitutions, γ-substitutions, and ϵ-substitutions, defined below. From the diagrams above it is clear that attributes and conditions play a similar rôle and, from the point of view of representation, they are interchangeable, therefore we will consider diagrams with conditions only, to avoid the multiplication of special cases.

ϵ-substitution ϵ-substitutions are the simplest: they remove from the diagram a relation that is not used to compute anything. There are two types of ϵ-substitution. The simplest is

$$R \rightarrowtail \emptyset \tag{25}$$

which states that a relation R disconnected from any variable can be eliminated from the diagram. The second ϵ-substitution is

$$\begin{tikzcd}
R \arrow[d] & & & \hat{R}^i \arrow[dl] \arrow[dr] & & \rightarrowtail & & \hat{R}^i \arrow[dl] \arrow[dr] & \\
\zeta_1 \arrow[dr, "\mathfrak{G}"] & & \zeta_1^i \arrow[dl] \arrow[d, "\mathfrak{G}"'] & \cdots & \zeta_n^i \arrow[d, "\mathfrak{G}'"] & & \zeta_1^i \arrow[d, "\mathfrak{G}"'] & \cdots & \zeta_n^i \arrow[d, "\mathfrak{G}'"] \\
& \bowtie & c_1^1 & & c_n^1 & & c_1^1 & & c_n^1
\end{tikzcd} \tag{26}$$

which states that a relation with variables that do not participate in the computation of any condition or attribute can be eliminated.

α-substitutions. α-substitutions deal with the replacement of a single relation, or part of a single relation, with a representation. A general form of an α-

[3] We use this symbol to represent the binding relation because, in most cases, the condition results in a join on the unique identifier of the entities between the different relations.

substitution is:

$$\begin{tikzcd}
 & & R_1 \arrow[dll] \arrow[d] \arrow[dr] & & \rightarrowtail & & & R_1 \arrow[dll] \arrow[d] & \hat{R}_1^k \arrow[d] \arrow[dr] & \\
\zeta_1 \arrow[d] & \cdots & \zeta_i \arrow[dl] \arrow[dr, "\mathfrak{G}"'] & \zeta_j \arrow[d, "\mathfrak{G}"] & & \zeta_1 \arrow[d] & \cdots & \zeta_i \arrow[d] \arrow[dr] & \zeta_i^k \arrow[d] \arrow[dr, "\mathfrak{G}"'] & \zeta_j^k \arrow[d, "\mathfrak{G}"] \\
c_1 & c_2 & & c_i & & c_1 & & c_2 & \bowtie & c_i
\end{tikzcd} \tag{27}$$

The conditions under which the substitution can be made are that the entity types E_i and E_j on which the variables ζ_i and ζ_j take values be represented in $\hat{R}_1^k$ and that they both be $\mathfrak{G}$-invariant.

β-substitutions β-substitutions intoduce representations for entity variables involved in two or more relations. A rather general example of β-substitutions is the following:

$$\begin{tikzcd}
 & R_1 \arrow[dl] \arrow[d] & R_2 \arrow[dl] \arrow[d] \arrow[dr] & & \rightarrowtail & & R_1 \arrow[dl] \arrow[d] & & \hat{R}_2^1 \arrow[dl] \arrow[d] & R_2 \arrow[d] \arrow[dr] & & \\
\zeta_1 \arrow[d] & \zeta_2 \arrow[d] \arrow[dr, "\mathfrak{G}"'] & \zeta_3 \arrow[d, "\mathfrak{G}"] \arrow[dr] & \zeta_4 \arrow[d] & & \zeta_1 \arrow[d] & \zeta_2 \arrow[d] \arrow[dr] & \zeta_2^1 \arrow[d] \arrow[dr, "\mathfrak{G}"'] & \zeta_3^1 \arrow[d, "\mathfrak{G}"] \arrow[dr] & \zeta_3 \arrow[d] \arrow[dr] & \zeta_4 \arrow[d] & \\
c_1 & c_2 & c_3 & c_4 & & c_1 & c_2 & \bowtie & c_3 & {} & \bowtie & c_4
\end{tikzcd} \tag{28}$$

Similarly to the previous case, the substitution can be done if all the entity types involved in the condition being represented are contained in the chosen representation, and if they are $\mathfrak{g}(c)$-invariant.

γ-substitutions γ-substitutions merge redundant representations that have been introduced while removing model relations using α- and β-substitutions. Their general form is the following:

$$\begin{tikzcd}
\hat{R}_1^i \arrow[d] \arrow[dr] & & & \hat{R}_1^j \arrow[dl] \arrow[d] & \rightarrowtail & & \hat{R}_1^k \arrow[dl] \arrow[d] \arrow[dr] & \\
\zeta_1^i \arrow[d, "\mathfrak{G}_1"'] & \zeta_2^i \arrow[d] & \zeta_2^j \arrow[dl] \arrow[dr, "\mathfrak{G}_2"'] & \zeta_3^j \arrow[d, "\mathfrak{G}_2"] & & \zeta_1^k \arrow[d, "\mathfrak{G}_1"'] & \zeta_2^k \arrow[dr, "\mathfrak{G}_2"'] & \zeta_3^k \arrow[d, "\mathfrak{G}_2"] \\
c_1 & \bowtie & & c_2 & & c_1 & & c_2
\end{tikzcd} \tag{29}$$

The problem of query rewriting can therefore be cast into the problem of finding the optimal path in an optimization tree: the nodes of the tree represent diagrams and the edges are marked with the substitutions that bring from a diagram to another. The detailed description of the optimization algorithms is beyond the scope of this paper.

The following proposition is an immediate consequence of the fact that all the subsitutions presented here are non-disruptive:

Proposition 3. *Let* $\hat{M}^0 \xrightarrow{Q} S_r$ *be a query, and* $\hat{M}^0 \times \hat{M}^1 \times \cdots \times \hat{M}^{n-1} \xrightarrow{Q'} \hat{S}_r$ *be a query obtained from* Q *through* α*-,* β*-,* γ*-, and* ϵ*-substitutions. Then* Q' *is a grounding for* Q*.*

In other words, the set of α-, β-, γ-, and ϵ-substitutions is sound. Its completeness is a more complicated issue. It is possible to show that, every query Q that can be represented by a given set of representations can be grounded in that set using only α-, β-, γ-, and ϵ-substitutions. We still don't know whether *all* groundings of Q can be found using only α-, β-, γ-, and ϵ-substitutions.

7 Future Directions

We are continuing the work presented in this paper in three directions: first, we are exploring the completeness properties of the set of substitutions. In particular, we are interested in whether *all* possible groundings of a query can be derived using ony the substitutions. Second, we are studying PTIME optimization algorithms to solve the problem posed by the query rewriting. Finally, we are trying to extend the model to other circumstances of practical interest, most notably the case in which the representations does not allow an exact computation of the attributes of the model, but itroduce an error.

References

1. Serge Abiteboul and Oliver M. Duschka. Complexity of answering queries using materialized views. In *Proceedings of PODS*, pages 254–263, 1998.
2. Foto Afrati, Che Li, and Jeffrey Ullman. Generating efficient plans for queries using views. In *Proceedings of SIGMOD*, pages 319–330, 2001.
3. Andrea Asperti and Giuseppe Longo. *Categories, Types, and Structures.* MIT Press, 1991.
4. Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Accessing data integration systems through conceptual schemas. In *Proceedings of ER2001*, pages 270–284, 2001.
5. A. Chandra and D. Harel. Computable queries for relational databases. *Journal of Computer and System Sciences*, 21(2):156–178, 1980.
6. Alon Halevy. Answering queries using views: a survey. *Very Large Data Bases Journal*, 2001.
7. Yoshiharu Ishikawa and Hiroyuki Kitagawa. Source description-based approach for the modeling of spatial information integration. In *Proceedings of ER2001*, pages 41–55, 2001.
8. Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. In *Proceedings of PODS*, pages 95–104, 1995.

Evaluating the Quality of Process Models: Empirical Testing of a Quality Framework

Daniel L. Moody[1,2], Guttorm Sindre[1], Terje Brasethvik[1], Arne Sølvberg[1]

[1]Department of Computer and Information Science
Norwegian University of Science and Technology (NTNU)
Trondheim, Norway N-7491
[2]School of Business Systems, Monash University
Melbourne, Australia 3800
{dmoody,guttors,brase,asolvber}@idi.ntnu.no

Abstract. This paper conducts an empirical analysis of a conceptual model quality framework for evaluating the quality of process models. 194 participants were trained in the concepts of the quality framework, and then used it to evaluate models represented in a workflow modelling language. A randomised, double-blind design was used, and the results evaluated using a combination of quantitative and qualitative techniques. An analysis was also conducted of the framework's likelihood of adoption in practice, which is an issue rarely addressed in IS design research. The study provides strong support for the validity of the framework and suggests that it is likely to be adopted in practice, but raises questions about its reliability. The research findings provide clear direction for further research to improve the framework.

1 Introduction

1.1 The Importance of Conceptual Model Quality

The traditional thrust of software quality assurance has been to use "brute force" testing at the end of the development process [52]. However empirical studies show that more than half the errors which occur during systems development are the result of inaccurate or incomplete requirements [24, 29]. Requirements errors are also the most common reason for failure of systems development projects [50, 51]. This suggests that more effort should be spent to quality assure conceptual models, in order to catch requirements defects as soon as they occur, or to prevent them from occurring altogether [59].

1.2 Previous Research

There have been a large number of empirical and conceptual studies published on the quality of data models [e.g. 3, 4, 5, 7, 8, 10, 17, 22, 25, 26, 28, 30, 31, 32, 33, 34, 36, 40, 46, 47, 48, 49, 53, 57]. However, the issue of process model quality has received far less attention in the literature. We were able to find only one previous empirical study on the quality of process models [45], and this was a single case study. This is surprising, as from the user's viewpoint, the most important consideration in whether the system meets their requirements is whether it supports the required functionality.

1.3 Conceptual Model Quality Framework

This paper conducts an empirical analysis of the conceptual model quality framework proposed by Lindland et al [27]. This represents the *a priori* theory being tested by this research. The framework is based on semiotic theory (the theory of signs), and has been developed for evaluating quality of conceptual models of all types. The framework defines three levels of quality, called *quality categories*:

- Syntactic quality: whether the model conforms to the grammar rules of the modelling language being used.
- Semantic quality: whether the model accurately represents user requirements.
- Pragmatic quality: whether the model can be understood by all relevant stakeholders.

These quality categories are based on the three levels of semiotic theory [38].

1.4 Objectives of this Paper

The broad research questions addressed by this paper are:

- Does the framework provide a reliable and valid basis for evaluating the quality of process models?
- Is the framework likely to be adopted in practice?

The second research question addresses an issue that is rarely addressed in IS design research, despite the fact that this is critical for research to have an impact on practice.

2 Research Methodology

2.1 Validation of IS Design Methods

The question of how to validate IS design methods has been a longstanding issue in the IS field [e.g. 15, 19, 42, 43, 44, 55, 58]. There are inherent problems evaluating any method or design technique since there is typically no theory, no hypotheses, no experimental design and no data analysis to which traditional evaluation criteria can be applied [55]. As a result, IS design research tends to emphasise the development of new design methods and frameworks, while addressing the evaluation of methods in only a limited fashion [9, 13, 15, 32, 56, 58]. The Method Evaluation Model [37] is a theoretical model for evaluating IS design methods, which incorporates two distinct aspects of method "success": actual efficacy and adoption in practice. The constructs of the model and relationships between them are summarised in Figure 1:

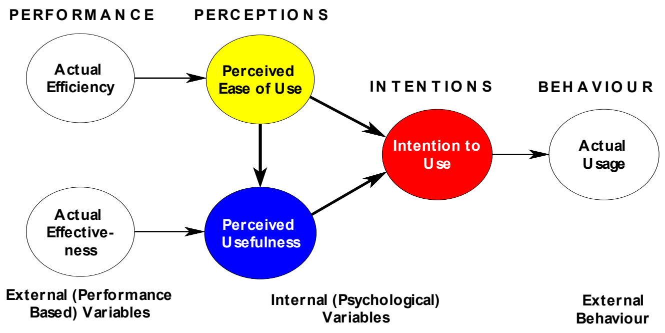


Fig. 1. The Method Evaluation Model

The definitions of the constructs are:

- Actual Efficiency: the extent to which the method reduces the effort required to perform the task;
- Actual Effectiveness: the extent to which the method improves the quality of the result;
- Perceived Ease of Use: the extent to which a person believes that using the method would be free of effort;
- Perceived Usefulness: the extent to which a person believes that the method would be useful;
- Intention to Use: the extent to which a person intends to use the method;
- Actual Usage: the extent to which the method is actually used.

Actual Efficiency and Actual Effectiveness measure whether the method actually improves task performance, while Perceived Ease of Use and Perceived Usefulness represent perceptions of the method's efficiency and effectiveness. Adoption in practice (corresponding to Intention to Use and Actual Usage) is determined by perceptions, which are in turn determined by performance.

2.2 Research Design

The research design is summarised in Figure 2. This shows the experimental treatment, experimental tasks, materials and dependent variables. A randomised, double-blind design was used, in which each participant reviewed multiple models and each model was evaluated by multiple reviewers.

2.3 Participants

There were 194 participants in this study, all of whom were third year students in the Department of Computer and Information Science at the Norwegian University of Science and Technology.

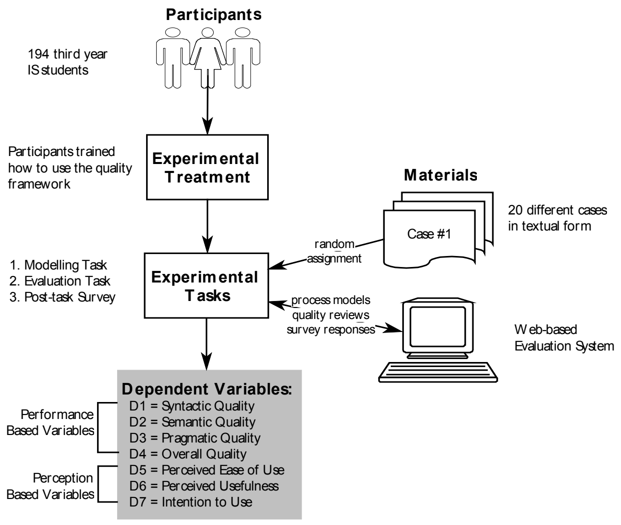


Fig. 2. Research Design

2.4 Participants

There were 194 participants in this study, all of whom were third year students in the Department of Computer and Information Science at the Norwegian University of Science and Technology.

2.5 Experimental Treatment

To prepare the participants for the evaluation task, they were given the original paper defining the framework, two 45 minute lectures explaining the concepts of the framework and a 45 minute lecture on the evaluation exercise itself.

2.6 Experimental Tasks

Process Modelling Task. A total of twenty different case descriptions were prepared for use in the experiment. This was done to minimise the chance of collaboration between participants. Each participant had two weeks to develop a model to meet the requirements of the case they were assigned, and submit it using a web-based system.

Evaluation Task. Following completion of the process modelling task, three reviewers were assigned to each model. Models were randomly assigned to reviewers, and reviews were double blind (reviewers did not know whose models they were evaluating, and modellers did not know who their reviewers were). Each participant was assigned cases which were different to each other and to the one they had modelled themselves. Participants had one week to complete the evaluation task, and record the results of their reviews using the web-based system.

Post Task Survey. Finally, participants were required to answer a post task survey, which measured their perceptions of the framework. The survey consisted of 16 closed questions and 3 open questions. Each item was measured on a 5-point Likert scale, using the opposing-statements format.

2.7 Dependent Variables

We distinguish between two types of dependent variables:

- Performance based measures: How well did subjects actually perform the evaluation task?
- Perception based measures: How effective did subjects perceive the framework to be in performing the evaluation task?

These types of measures represent the difference between actual efficacy and perceived efficacy as defined in the Method Evaluation Model.

Performance Based Variables. Four dependent variables were used to evaluate performance on the evaluation task: Syntactic Quality, Semantic Quality, Pragmatic Quality, and Overall Quality. A total of 564 ratings of each type were collected as part of the study (an average of three per model). These were used to assess the reliability and validity of the evaluation framework. All quality ratings were given on a 7 point Likert scale, from 1 (poor) to 7 (excellent). As well as rating the models, reviewers were required to identify all defects in the models, classified by quality category.

Perception Based Variables. Three dependent variables were used to measure participants' perceptions of the framework: Perceived Ease of Use, Perceived Usefulness, and Intention to Use. These are constructs of the Method Evaluation Model. Each of these constructs was measured using items on the post task survey:

- Perceived Ease of Use: Questions 1, 10, 11, 14 and 16.
- Perceived Usefulness: Questions 2, 5, 6, 7, 9, 12, 13 and 15.
- Intention to Use: Questions 3, 4 and 8.

The order of the items was randomised to avoid monotonous responses [18].

2.8 Theoretical Model

The relationship between all dependent variables and their empirical indicators are shown in Figure 3. In the diagram, circles indicate latent variables, rectangles indicate observed variables (in this case individual survey items), dotted lines indicate measurement relationships, while solid lines indicate causal relationships. The empirical indicators associated with the performance based variables represent the different reviewer ratings.

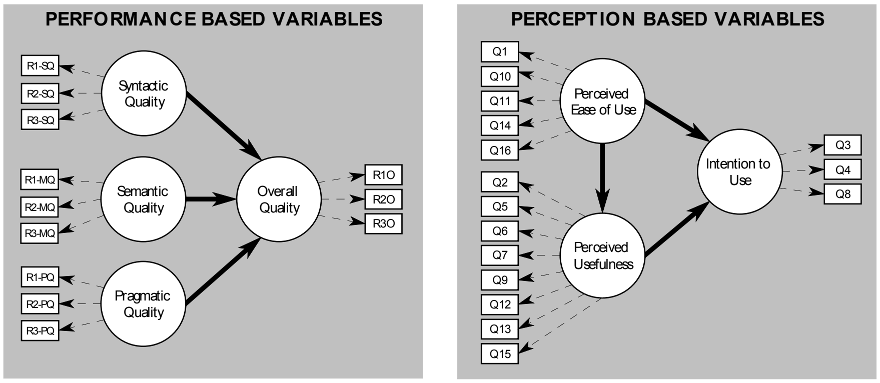


Fig. 3. Theoretical Model

3 Results and Discussion

3.1 Summary of Analyses

The results were evaluated using a combination of quantitative and qualitative analysis techniques. Each of these analyses addresses a sub-question of one of the research questions defined at the beginning of this paper:

1. Reliability analysis: How consistently were reviewers able to apply the framework?
2. Validity analysis: Are the quality categories complete, parsimonious and independent determinants of process model quality?
3. Weight estimation: What was the relative influence of each quality category in decisions about the overall quality of a model?
4. Interaction analysis: What are the relationships between the quality categories?
5. Task accuracy: How accurately were participants able to identify quality defects using the quality framework?
6. Perceived ease of use: How easy did participants find the framework to use?
7. Perceived usefulness: How useful did participants find the framework?
8. Intention to use: Were the participants likely to use the framework in the future?

Analyses 1-5 evaluate *actual efficacy* in performing the task (Research Question 1), while analyses 6-8 evaluate *perceived efficacy* (Research Question 2). Analyses 1-4 are quantitative analyses, analysis 5 is qualitative, while analyses 7-9 involve a combination of the two (both closed and open questions were included in the post-task survey). Quantitative and qualitative methods have different, complementary strengths, and when used together can lead to a more comprehensive understanding of a phenomenon [20, 39].

3.2 Reliability Analysis

Reliability was evaluated by measuring the level of agreement between different reviewers of the same model. As shown in Table 1, levels of inter-rater reliability of around .6 were found for each quality category and for overall quality. This means that around 40% of the variation is due to error. While there is no definitive standard for reliability, alphas of 0.7 or above are considered to be acceptable in the literature [41]. The observed levels of reliability are clearly lower than acceptable, which suggests that the quality framework in its current form cannot be reliably applied in practice.

Table 1. Inter-rater Reliability

CONSTRUCT	CRONBACH'S α
Syntactic Quality	.6159
Semantic Quality	.5778
Pragmatic Quality	.5682
Overall Quality	.6091

3.3 Validity Analysis

To assess the validity of the quality framework, we need to address the following questions [1]:

- *Completeness (sufficiency)*: Do the set of quality categories cover all aspects of quality of process models? Are there any aspects that have been left out?
- *Parsimony (necessity)*: Are all the quality categories necessary for evaluating process models? Are they all relevant determinants of process model quality?
- *Independence*: Are the quality categories independent of each other (orthogonal dimensions of quality)?

To evaluate these properties, a regression analysis was carried out using the quality categories (syntactic, semantic, pragmatic) as predictor variables and overall quality as the predicted variable. The regression equation which resulted was:

Equation 1. *Overall Quality = .33 * Syntactic Quality + .31 * Semantic Quality + 34 * Pragmatic Quality + .07*

The details of the regression were:

- Adjusted $r^2 = 0.93$
- Significance level (p) = 0.000***

Completeness (Sufficiency) of Quality Categories. The regression was found to be highly significant ($\alpha < .01$), and the adjusted r^2 shows that the quality categories account for more than 90% of the variance in Overall Quality. This is an extremely high value for r^2, which provides strong evidence that the set of quality categories is complete. Ratings of overall quality are almost entirely explained by variations in ratings for the individual quality categories.

Parsimony (Necessity) of Quality Categories. Separate t-statistics and significance levels were also calculated for each predictor variable. These measure the effect of each quality category on overall quality, while controlling for all other quality categories. As shown in Table 2, all quality categories were found to have highly significant effects on overall quality ($\alpha < .01$). This provides strong evidence that the quality categories are all relevant determinants of the quality of a process model.

Table 2. Multiple Regression Results

QUALITY CATEGORY	T-STATISTIC	statistical significance	TOLERANCE
Syntactic	21.79	.000	.437
Semantic	17.58	.000	.345
Pragmatic	21.57	.000	.379

Independence of Quality Categories. Collinearity analysis was also conducted as part of the regression analysis, to evaluate the independence of the quality categories. As shown in Table 2, Tolerance values are well above .2 for all quality categories, which shows that they are all independent determinants of overall quality.

3.4 Weighting of Quality Categories

The regression coefficients in Equation 1 measure the separate effects of each quality category on decisions about the overall quality of a model. The results show that the categories have approximately equivalent effects on overall quality. This is surprising, as we expected that semantic quality would have the strongest influence.

3.5 Interaction Analysis

Bi-variate correlation analysis was conducted to test for interactions between the quality categories. Strong positive correlations were found between all quality categories:

- Syntactic and Semantic Quality ($r = .72$, $p = .000$)
- Syntactic and Pragmatic Quality ($r = .69$, $p = .000$)
- Semantic and Pragmatic Quality ($r = .76$, $p = .000$)

Correlations are non-directional, but our theoretical explanation is that they represent causal relationships (Figure 4):

- Improving syntactic quality will improve semantic quality, as a model that is not syntactically correct will have ambiguous semantics.
- Improving syntactic quality will improve pragmatic quality as a model that is not syntactically correct will be difficult to interpret.
- Improving pragmatic quality will improve semantic quality, as a model that is difficult to understand will be difficult to verify against user requirements.

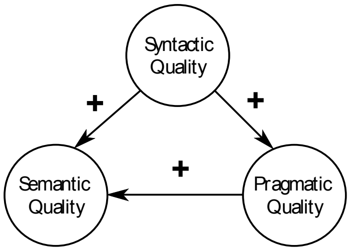


Fig. 4. Interactions Between Quality Categories

Three alternative multiple regression models were developed to test this explanation (these represent all possible causal combinations of the variables):

- Model 1: Syntactic Quality + Semantic Quality → Pragmatic Quality
- Model 2: Syntactic Quality + Pragmatic Quality → Semantic Quality
- Model 3: Semantic Quality + Pragmatic Quality → Syntactic Quality

The results of the analysis are summarised in Table 3. While all regression models were statistically significant, Model 2 (our *a priori* theory) provides the best fit (as evidenced by the higher variance explained).

Table 3. Alternative Regression Models

Model	Adjusted r^2	Significance
1	.619	.000
2	.654	.000
3	.561	.000

3.6 Task Accuracy

Task accuracy was measured by inspecting the sample reviews and comparing them to an expert review:

- Percentage of Type 1 errors (false negatives or errors of omission): where defects exist in the model but were not identified by reviewers. On average, each review reported just 2.4 defects, whereas the expert reported an average of 6.6 defects per model. Hence, on average, 64% of the defects went unreported. 20% of the participants did not report any defects at all, while not necessarily giving perfect scores for the models.
- Percentage of Type II errors (false positives or errors of commission): where defects are identified which are not defects at all. 95% of the defects reported by the participants were genuine defects according to the expert, so only 5% were Type II errors.
- Percentage of Type III errors (classification errors): where defects are correctly identified but classified in the wrong quality category. Only 9% of the errors were wrongly categorised.

There are a number of possible explanations for the high levels of underreporting. Firstly, the lack of experience of the participants means they would be less able to identify defects compared to an expert. Secondly, while IS design courses generally require students to develop models, they rarely require them to evaluate models, which requires a higher level of thinking [6]. Thirdly, the quality framework is defined at a very abstract level—this is one of its strengths (it can be applied to models of all types) but also one of its weaknesses. Decomposing the quality categories to a lower level of detail, with types of defects specific to the process modelling technique, would improve identification of defects. Finally, it was clear that many of the students used minimal effort to conduct their reviews and did not invest the time necessary to do a thorough job.

3.7 Likelihood of Adoption in Practice

Validation of the Measurement Instrument. To evaluate the results for Perceived Ease of Use, Perceived Usefulness and Intention to Use, it is first necessary to evaluate the validity and reliability of their empirical indicators.

Construct Validity. Factor analysis was conducted on the items used in the post-task survey using the principal components extraction method and varimax rotation. The initial factor analysis resulted in four factors being extracted using the Kaiser criterion. Q5 (Perceived Ease of Use) and Q14 (Perceived Usefulness) formed a fourth factor, and were therefore eliminated from the analysis. Factor weights derived from the analysis were used to estimate the value of the underlying theoretical constructs.

Item Reliability. Reliability analysis was conducted on the items used to measure Perceived Ease of Use, Perceived Usefulness and Intention to Use (excluding Q14 and Q15). As shown in Table 4, high levels of reliability were found for all constructs, with Cronbach's alpha > .7 in all cases, which is considered acceptable [41].

Table 4. Item Reliabilities for Each Construct

CONSTRUCT	CRONBACH'S α
Perceived Ease of Use	.8614
Perceived Usefulness	.7737
Intention to Use	.9036

Significance Testing. One-sample t-tests were conducted for each construct to see if they were significantly different to 3 (the "zero point" of the Likert scale used). Table 5 summarises the results of the significance testing. Overall, participants found the framework easy to use and useful, and intended to use it in the future.

Table 5. Significance of Responses

CONSTRUCT	MEAN (μ)	STDEV (δ)	SIGNIFICANCE	Y/N
Perceived Ease of Use	3.94	.764	.000	Yes
Perceived Usefulness	3.89	.597	.000	Yes
Intention to Use	3.34	.941	.019	Yes

3.8 Analysis of Causal Relationships

In the theoretical model (Figure 3), a number of causal relationships were hypothesised between the perception based variables:

- Perceived Ease of Use → Perceived Usefulness
- Perceived Ease of Use + Perceived Usefulness → Intention to Use

These relationships follow from the Method Evaluation Model (Figure 1). We evaluated the validity of these relationships using regression analysis.

Perceived Ease of Use → Perceived Usefulness. The regression equation which resulted from this analysis was:

Equation 2. *Perceived Usefulness = .33 * Perceived Ease of Use + 2.59*

The details of the regression were:

- R squared (r^2) = .18
- Significance level (p) = .004

The regression was found to be highly significant with $\alpha < .01$. This means that the causal relationship between Perceived Ease of Use and Perceived Usefulness was strongly confirmed. The r^2 statistic shows that Perceived Ease of Use accounts for almost 20% of the variance in Perceived Usefulness.

Perceived Ease of Use + Perceived Usefulness → Intention to Use. The regression equation which resulted from this analysis was:

Equation 3. *Intention to Use = .26 * Perceived Ease of Use + .84 * Perceived Usefulness – 0.03*

The details of the regression were:

- Adjusted r squared (r^2) = .26
- Significance level (p) = .001

The regression was found to be highly significant, and the r^2 statistic shows that Perceived Ease of Use and Perceived Usefulness together account for more than 25% of the variance in Perceived Usefulness. The t-statistics, significance levels and tolerance values for each independent variable are shown in Table 6 below. These measure the separate effect of each variable on Intention to Use, while controlling for the effect of the other independent variable.

Table 6. Multiple Regression Results

INDEPENDENT VARIABLE	T-STATISTIC	statistical significance	TOLERANCE
Perceived Ease of Use	.146	.885	.823
Perceived Usefulness	3.736	.001	.823

The results show that the relationship between Perceived Usefulness and Intention to Use was highly significant with $\alpha < .01$, while the relationship between Perceived Ease of Use and Intention to Use was *not* statistically significant ($\alpha < .05$). This is consistent with many of the studies of the Technology Acceptance Model, which have found that the relationship between Perceived Ease of Use and Intention to Use is not significant after controlling for the effects of Perceived Usefulness [11, 14]. No evidence of multi-collinearity was found.

4 Conclusion

This paper has conducted an empirical analysis of the conceptual model quality framework proposed by Lindland et al (1994). The validation approach combines both quantitative and qualitative research approaches. The paper illustrates a systematic approach to validating an IS design method, which addresses both its technical effectiveness and its adoption in practice—the latter issue has largely been ignored in IS design research.

4.1 Summary of Findings

Overall, the quality framework was found to be valid, with its categories found to be complete, parsimonious and independent. It was perceived to be both easy to use and useful in evaluating process models, and participants intended to use it in the future. The research findings provide clear directions for future research, refining the framework to address the limitations found, specifically with regard to reliability and identification of defects. The results for each of the research questions defined at the beginning of the paper are summarised in Table 7:

4.2 Theoretical Significance

The major theoretical contribution of this paper is that it is the first experimental study of process model quality that has so far been conducted. The observed interactions between the quality categories represent a refinement to the framework, and provides a possible theoretical contribution to semiotic theory. Finally, the paper conducts a further empirical test of the Method Evaluation Model, which showed the measurement instrument had a high level of reliability and validity and validated two of the three causal relationships in the model. This suggests that the model provides a valid basis for explaining and predicting adoption of methods in practice.

Table 7. Summary of Findings

Research question	Result
1. Task Performance (actual efficacy of the framework)	
1.1 Is the framework reliable?	No – this is an area where further research is required
1.2 Is the framework valid?	Yes – the quality categories were found to be necessary, sufficient and independent
1.3 What is the relative influence of each quality category on decisions about the overall quality of a model?	Approximately equal
1.4 What are the interactions between the quality categories?	Syntactic → Semantic Syntactic → Pragmatic Pragmatic → Semantic
1.6 How accurately were participants able to perform the evaluation task?	64% Type I errors 5% Type II errors 9% Type III errors
2. Adoption in Practice (perceived efficacy of the framework)	
2.1 Did participants find the framework easy to use?	Yes
2.2 Did participants find the framework useful?	Yes
2.3 Are participants likely to use the framework in practice?	Yes

4.3 Limitations and Further Research

The major limitation of this study, and it is a significant one, is the nature of sample population used. In general, the population from which one selects subjects for a research study should be representative of the population to which the researcher wishes to generalise results [12]. Computer Science or Information Systems students have been used in most previous experimental studies of conceptual modelling [e.g. 4, 7, 8, 17, 25, 40, 48, 49, 54]. However external validity is a significant problem in most laboratory experiments involving undergraduate students, and has been identified as a major issue in IS research in terms of its relevance to practice [16, 21, 35]. It is also one of the major limitations of experimental research in the social sciences generally [2].

The question therefore remains: would similar results have been found if practitioners (which is the population to which we wish to generalise) had been used instead? The argument could be made that undergraduate students have little no practical experience in process modelling, and therefore do not have the knowledge or expertise to properly evaluate the quality of process models. As a result, students may have been tempted to score overall quality simply as an average of their scores for syntactic, semantic, and pragmatic quality, although they were instructed that overall quality should be considered separately from the other variables. Such averaging would inflate the validity of the quality framework. Similarly, both validity and reliability might be inflated by defensive reviewing (i.e. assigning scores of 3-5 rather than 1-2 or 6-7). These are valid criticisms and difficult to argue against convincing.

For this reason, we do not see this study as the end of the validation process but only the *beginning*. This study has provided useful empirical data about the framework and pilot-tested the validation approach, but the results must be interpreted with caution. The generalisability of the results to practice is questionable because of the nature of the participants used. For this reason, a similar study is planned using experienced practitioners, possibly with a refined version of the quality framework. Some refinement of the framework has already taken place [23], but the results of this study suggest the need for other refinements, such as customising the framework to the modelling language used.

References

[1] Alexander, C. (1968): *Notes On The Synthesis Of Form*, Boston: Harvard University Press.

[2] Babbie, E.R. (1998): *The Practice of Social Research*, Belmont, California: Wadsworth Publishing.

[3] Batini, C., S. Ceri, and S.B. Navathe (1992): *Conceptual Database Design: An Entity Relationship Approach*, Redwood City, California: Benjamin Cummings.

[4] Batra, D., J.A. Hoffer, and R.P. Bostrom (1990): Comparing Representations with Relational and EER Models, *Communications of the ACM*, **33** (2), February: p. 126-139.

[5] Batra, D. and J. Davis (1992): Conceptual Data Modelling In Database Design: Similarities And Differences Between Expert And Novice Designers, *International Journal Of Man-Machine Studies*, **37**: p. 83-101.

[6] Bloom, B. (1984): *Taxonomy of Educational Objectives*, New York: Longman.

[7] Bock, D.B. and T. Ryan (1993): Accuracy in Modelling with Extended Entity Relationship and Object Oriented Data Model, *Journal Of Database Management*, **4** (4): p. 30-39.

[8] Bodart, F., A. Patel, M. Sim, and R. Weber (2001): Should the Optional Property Construct be used in Conceptual Modelling: A Theory and Three Empirical Tests, *Information Systems Research*, **12** (4).

[9] Bubenko, J.A. (1986): Information Systems Methodologies - A Research View, In T.W. Olle, H.G. Sol, and A.A. Verrijn-Stuart (Eds.), *Information Systems Design Methodologies: Improving The Practice*: North-Holland.

[10] Chaiyasut, P. and G.G. Shanks (1994): Conceptual Data Modelling Process: A Study Of Novice And Expert Data Modellers, In T. Halpin and R. Meersman (Eds.), *Proceedings Of The 1st International Conference On Object Role Modelling,* Magnetic Island, Queensland, Australia, July.

[11] Chau, P.Y.K. (1996): An Empirical Assessment of a Modified Technology Acceptance Model, *Journal of Management Information Systems*, **13** (2).

[12] Cooper, D.R. and P.S. Schindler (1998): *Business Research Methods (6th edition)*, 6th ed, Singapore: McGraw-Hill International.

[13] Curtis, B. (1986): By The Way, Did Anyone Study Any Real Programmers?, In *Empirical Studies of Programmers*, E. Soloway and S. Iyengar (Eds.), Ablex: Norward, N.J.

[14] Davis, F.D., R.P. Bagozzi, and P.R. Warshaw (1989): User Acceptance of Computer Technology: A Comparison of Two Theoretical Models*, Management Science*, **35** (8): p. 982-1003.

[15] Fitzgerald, G. (1991): Validating New Information Systems Techniques: A Retrospective Analysis, In H.E. Nissen, H.K. Klein, and R. Hirschheim (Eds.), *Information Systems Research: Contemporary Approaches And Emergent Traditions*: North-Holland.

[16] Galliers, R.D. (1994): Relevance and Rigour in Information Systems Research: Some Personal Reflections on Issues Facing the Information Systems Research Community, In *Proceedings of the IFIP TC8 Conference on Business Process Reengineering: Information Systems and Challenges,* Gold Coast, Australia.

[17] Hardgrave, B.C. and N.P. Dalal (1995): Comparing Object Oriented and Extended Entity Relationship Models*, Journal Of Database Management*, **6** (3).

[18] Hu, P.J. and P.Y.K. Chau (1999): Examining the Technology Acceptance Model Using Physician Acceptance of Telemedicine Technology*, Journal of Management Information Systems*, **16** (2), Fall: p. 91-113.

[19] Ivari, J. (1986): Dimensions Of Information Systems Design: A Framework For A Long Range Research Program*, Information Systems Journal*, (June).

[20] Kaplan, B. and D. Duchon (1988): Combining Qualitative and Quantitative Methods in Information Systems Research: A Case Study*, MIS Quarterly*, December.

[21] Keen, P.G.W. (1991): Relevance and Rigour in Information Systems Research: Improving Quality, Confidence, Cohesion and Impact, In *Information Systems Research: Contemporary Approaches and Emergent Traditions*, H.-E. Nissen, H.K. Klein, and R. Hirschheim (Eds.), Elsevier Science Publishers (North Holland).

[22] Kesh, S. (1995): Evaluating The Quality Of Entity Relationship Models*, Information And Software Technology*, **37** (12).

[23] Krogstie, J. (1998): Integrating the Understanding of Quality in Requirements Specification and Conceptual Modelling*, Software Engineering Notes*, **28** (1).

[24] Lauesen, S. and O. Vinter (2000): Preventing Requirement Defects, In Proceedings of the Sixth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'2000), Stockholm, Sweden, June 5-6 2000.

[25] Lee, H. and B.G. Choi (1998): A Comparative Study of Conceptual Data Modelling Techniques*, Journal Of Database Management*, **9** (2), Spring: p. 26-35.

[26] Levitin, A. and T. Redman (1994): Quality Dimensions of a Conceptual View*, Information Processing and Management*, **31** (1).

[27] Lindland, O.I., G. Sindre, and A. Sølvberg (1994): Understanding Quality In Conceptual Modelling*, IEEE Software*, **11** (2), March: p. 42-49.

[28] Maiden, N. and A. Sutcliffe (1992): Analysing the Novice Analyst: Cognitive Models in Software Engineering*, International Journal Of Man Machine Studies*, **367**: p. 719-740.

[29] Martin, J. (1989): *Information Engineering*, Englewood Cliffs, N.J.: Prentice Hall, 3 v.

[30] Moody, D.L. and G.G. Shanks (1994): What Makes A Good Data Model? Evaluating the Quality of Entity Relationship Models, In P. Loucopolous (Ed.) *Proceedings of the Thirteenth International Conference on the Entity Relationship Approach,* Manchester, England, December 14-17.

[31] Moody, D.L. (1998): Metrics for Evaluating the Quality of Entity Relationship Models, In T.W. Ling, S. Ram, and M.L. Lee (Eds.), *Proceedings of the Seventeenth International Conference on Conceptual Modelling (ER '98),* Singapore: Elsevier Lecture Notes in Computer Science, November 16-19.

[32] Moody, D.L. and G.G. Shanks (1998): Evaluating and Improving the Quality of Entity Relationship Models: An Action Research Programme*, Australian Computer Journal*, November.

[33] Moody, D.L. and G.G. Shanks (1998): What Makes A Good Data Model? A Framework For Evaluating And Improving The Quality Of Entity Relationship Models*, Australian Computer Journal*, August.

[34] Moody, D.L., G.G. Shanks, and P. Darke (1998): Evaluating and Improving the Quality of Entity Relationship Models: Experiences in Research and Practice, In T.W. Ling, S. Ram, and M.L. Lee (Eds.), *Proceedings of the Seventeenth International Conference on Conceptual Modelling (ER '98),* Singapore: Elsevier Lecture Notes in Computer Science, November 16-19.

[35] Moody, D.L. (2000): Building Links Between IS Research and Professional Practice: Improving the Relevance and Impact of IS Research, In R.A. Weber and B. Glasson (Eds.), *International Conference on Information Systems (ICIS'00),* Brisbane, Australia, December 11-13.

[36] Moody, D.L. (2000): Strategies for Improving the Quality of Entity Relationship Models, In *Information Resource Management Association (IRMA) Conference,* Anchorage, Alaska: Idea Group Publishing, May 21-24.

[37] Moody, D.L. (2001): *Dealing with Complexity: A Practical Method for Representing Large Entity Relationship Models (PhD Thesis)*, Melbourne, Australia: Department Of Information Systems, University of Melbourne.

[38] Morris, C.W. (1970): *Foundations of the Theory of Signs*, Chicago: Chicago University Press.

[39] Neuman, W.L. (2000): Social Research Methods - Qualitative and Quantitative Approaches (4th edition), Needham Heights, MA: Allyn and Bacon.

[40] Nordbotten, J.C. and M.E. Crosby (1999): The Effect of Graphic Style on Data Model Interpretation*, Information Systems Journal*, **9**.

[41] Nunally, J. (1978): *Psychometric Theory (2nd edition)*, New York: McGraw Hill.

[42] Olle, T.W., H.G. Sol, and A.A. Verrijn-Stuart, eds. *Information Systems Design Methodologies: A Comparative Review*. 1982, North-Holland: Amsterdam.

[43] Olle, T.W., H.G. Sol, and C.J. Tully, eds. *Information Systems Design Methodologies: A Feature Analysis*. 1983, North-Holland: Amsterdam.

[44] Olle, T.W., H.G. Sol, and A.A. Verrijn-Stuart, eds. *Information Systems Design Methodologies: Improving the Practice*. 1986, North-Holland: Amsterdam.

[45] Rosemann, M., W. Sedera, and D. Sedera (2001): Testing a Framework for the Quality of Process Models: A Case Study, In *Fifth Pacific Asia Conference on Informaiton Systems (PACIS'2001),* Seoul, Korea,

[46] Schuette, R. and T. Rotthowe (1998): The Guidelines Of Modelling: An Approach To Enhance The Quality In Information Models, In *Proceedings Of The Seventeenth International Conference On Conceptual Modelling (ER '98),* Singapore: Elsevier Lecture Notes in Computer Science, November 16-19.

[47] Shanks, G.G., G.C. Simsion, and M. Rembach (1993): The Role Of Experience In Conceptual Data Modelling, In *Fourth Australian Information Systems Conference,* Brisbane, Australia,

[48] Shoval, P. and M. Even-Chaime (1987): Database Schema Design: An Experimental Comparison Between Normalisation and Information Analysis*, Database*, **18** (3), Spring: p. 30-39.

[49] Shoval, P. and S. Shiran (1997): Entity-Relationship and Object-Oriented Data Modeling: An Experimental Comparison of Design Quality*, Data & Knowledge Engineering*, **21**: p. 297-315.

[50] Standish Group (1995): *The CHAOS Report into Project Failure*, The Standish Group International Inc. Available on-line at http://www.standishgroup.com/visitor/chaos.htm.

[51] Standish Group (1996): *Unfinished Voyages*, The Standish Group International Inc. available on-line at http://www.standishgroup.com/visitor/voyages.htm.

[52] van Vliet, H. (2000): Software Engineering: Principles and Practice (2nd edition): John Wiley & Sons.

[53] von Halle, B. (1991): Data: Asset or Liability?*, Database Programming and Design*, **4** (7), July.

[54] Weber, R.A. (1996): Are Attributes Entities? A Study Of Database Designers' Memory Structures*, Information Systems Research*, June.

[55] Weber, R.A. (1997): *Ontological Foundations Of Information Systems*, Melbourne, Australia: Coopers And Lybrand Accounting Research Methodology Monograph No. 4, Coopers And Lybrand.

[56] Westrup, C. (1993): Information Systems Methodologies in Use*, Journal of Information Technology*, **8**.

[57] Witt, G.C. and G.C. Simsion (2000): *Data Modeling Essentials: Analysis, Design, and Innovation*, The Coriolis Group.

[58] Wynekoop, J.L. and N.L. Russo (1997): Studying Systems Development Methodologies: An Examination Of Research Methods*, Information Systems Journal*, **7** (1), January.

[59] Zultner, R.E. (1992): The Deming Way: Total Quality Management for Software, In *Proceedings of Total Quality Management for Software Conference,* Washington, DC, April.

Data Quality in Web Information Systems

Barbara Pernici[1] and Monica Scannapieco[2]*

[1] Dipartimento di Elettronica e Informazione, Politecnico di Milano
Piazza Leonardo da Vinci 32, 20133 Milano, Italy
barbara.pernici@polimi.it

[2] Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
monscan@dis.uniroma1.it

Abstract. Evaluation of data quality in web information systems provides support for a correct interpretation of the contents of web pages. Data quality dimensions proposed in the literature need to be adapted and extended to represent the characteristics of data in web pages, and in particular their dynamic aspects. The present paper proposes and discusses a model and a methodological framework to support data quality in web information systems.

1 Introduction

Web Information Systems (WIS's) [9] are characterized by the presentation to a wide audience of a large amount of data, the quality of which can be very heterogeneous. There are several reasons for this variety, but a significant reason is the conflict between the need of publishing high quality information and publishing information as soon as it becomes available. Providing information with a good quality level is a traditional goal of information systems: theoretical work in database design has focused on proposing appropriate data structures and design methods for guaranteeing the quality of data (e.g., [3]); in addition, more recent research work on data quality focuses on the quality of instances of data, by defining a number of data quality dimensions (e.g., [20]) and tools to support data quality improvement. On the other hand, information systems on the web need to publish information in the shortest possible time after it is available from information sources. These two requirements are in many aspects contradictory: accurate design of data structures, and in the case of web sites, of good navigational paths between pages, and certification of data to verify its correctness are costly and lengthy activities, while publication of data on web sites requires stringent times.

Such a tradeoff between data quality and timeliness of information is a problem which has to be taken into account also in traditional publishing areas,

* The work of Monica Scannapieco is supported by *Istituto di Analisi dei Sistemi ed Informatica, Consiglio Nazionale delle Ricerche (IASI-CNR)*, Viale Manzoni 30, 00185 Roma, Italy.

such as newspapers and journals. However, WIS's present two peculiar aspects with respect to traditional paper-based information sources: first, a web site is a continuously evolving source of information, and it is not linked to a fixed release time of information; second, the process of producing information changes, additional information can be produced in different phases, and corrections to previously published information are possible. Such features lead to a different type of information with respect to traditional media.

Therefore, the persons in charge for information release on web sites are often faced with the need to decide, for single items of information, or for groups of data with similar characteristics, whether or not a given piece of information should be published at a given time.

The goal of this paper is to propose a data model and a methodological framework that support the task of data publication on the web, providing tools to associate quality information to data in web pages, both as static values, and as expected evolution of quality in time due to subsequent updates. The original contribution of the paper focuses mainly on the dynamic nature of quality information and on the need to define a methodological framework and tools to manage data quality information.

In the following of the paper, we discuss data quality in WIS's presenting examples taken from a university web site. Let us consider, for instance, the case in which a university has to present to perspective students new offerings in terms of curricula for the following academic year: the choice in terms of time of publication is between waiting that all curricula are defined in order to present the complete view on available degrees, or to present, maybe with different emphasis and details, new offerings and previously available degrees at different times. The second choice is more flexible, but it presents the disadvantage that perspective students could miss important information if they visit the university site before information they are looking for is available. On the other hand, the first option requires that all decisions are taken, and it will probably postpone the date of publication of information with respect to the second alternative; in some cases the final decisions could be taken in a time frame which is not compatible with the communication needed to publicize degree offerings for the following year.

In the previous example, a possible solution to miscommunication problems is to provide a meta-level information about the information being published. Quality information can be associated to each information item in a page (or groups of items, or pages), expressing properties related to the expected validity of each information, its completeness, and the quality of the information source. The evolution in time of quality information in WIS's is particularly important, since it provides the user of such information with a basis for deciding whether to use and how its contents. Therefore, the dynamic evolution of quality properties is also taken into account.

The structure of the paper is as follows. In Section 2, the data quality dimensions for WIS's are presented and discussed. In Section 3, a model for data quality is proposed and in Section 4, a methodological framework for data quality is described. In Section 5, an architecture realizing the proposed framework is described. Finally, in Section 6, related research work is discussed and Section 7 concludes the paper by drawing future work.

2 Definition of Data Quality Dimensions

In the literature, data quality is considered as a multidimensional concept [15], i.e., it is defined on the basis of a set of "dimensions". Many proposals concerning the set of dimensions characterizing data quality have been made, a survey of which is given in [19]. One of the reasons for such a wide variety is that it is very difficult to define a set of data quality dimensions suitable for every kind of context. In the present paper, we focus only on a "core" set of dimensions, aiming at capturing the most important aspects of data quality in WIS's, leaving to future work possible extensions of this set. The choice of this set has been guided by those that are used most frequently in the literature [17]. In the following, we give a definition for these core dimensions, and, in order to provide support for the evolving nature of information published in WIS's, we also associate metrics to indicate the expected evolution of quality in time. We consider the following four data quality dimensions:

- ❒ **Expiration**. It is defined as the time until which data remain current.
  As an example, let us consider the case of a professor of a department of a university that is going to move to another university. If a list of the members of the department is published, this situation can be represented by associating to the professor as a member of the department its expiration date, corresponding to the date when the professor leaves the university.
- ❒ **Completeness**. It is defined as the degree to which the elements of an aggregated element are present in the aggregated element instance. Therefore, completeness is defined only for elements that are an aggregation of other elements, such as lists, tables, and so on.
  Let us consider a list of courses published on the university site, and the situation in which it is known that new courses will be added to the list. A completeness measure can be associated to the list to make it explicit that the list is still partial.
- ❒ **Source reliability**. It is defined as the credibility of the source that provides data.
  Let us suppose that the starting date for a semester is available on the university web site. Before having official information (e.g., a rector's decree), it typically happens that such information is already circulating, e.g., in administrative offices. If there is a need to make this information visible before it is finally approved by decree, we can associate a `LOW` value of source reliability to the date. Conversely, if the date is provided by the official source for such information in the university, i.e., it has been validated and officially approved, it will have a `HIGH` source reliability value. The same data source may have different source reliability values for different data it provides. As an example, the school administrative office may have a `LOW`

source reliability value for starting dates of semesters; instead it may have a `HIGH` source reliability value for course programs.

❒ **Accuracy**. It is the distance between the data value *v* and *v'*, being *v'* the value considered as correct.
Let us consider the following example. A professor published on the university web site has an attribute name, the value of which is `JHN`, while the correct value is `JOHN`: this is a case of low accuracy, as `JHN` is not an admissible value according to a dictionary of English names.

2.1 Dynamics of Data Quality Dimensions

One of the aspects of data quality in WIS's is the possible evolution in time of some data. For instance, if an incomplete course list is published, the completeness dimension presented above provides only a static measure of the completeness of such a list. An important information is also the expected evolution of the list ("how and when it will be completed?"). In this section we show how to associate temporal dynamics to the expiration and completeness dimensions.

We assume instead that source reliability and accuracy are constant in time for a certain instance of data, i.e., if their values vary, it is not anymore the same data. Considering the previous example of the starting date for a semester provided by an administrative office and thus having a `LOW` source reliability value, this date may be confirmed or not by the official source; in both cases, the date communicated by the official source is a new data item with a `HIGH` source reliability value; similar considerations can be also made for accuracy.

Temporal Dynamics of Expiration: Volatility. With reference to the temporal dynamics of expiration, we consider a function corresponding to the probability that the expiration time will change in the time interval starting from the instant of publication and ending with the expiration time. More formally, let us consider the following values:

– `t_pub`: instant of publication of data,
– `t_exp`: expiration time.

We also consider a function *P_exp(t)* = probability at time t that `t_exp` will be the real expiration time, with $t \in [\texttt{t\_pub}, \texttt{t\_exp}]$. Starting from the function *P_exp(t)*, we can define a **volatility** of the published data.

Definition: Volatility is defined as $V = (t\_exp - t\_curr) - \int_{t\_curr}^{t\_exp} P\_exp(t)$, where t_curr is the time at which volatility is evaluated and $t\_curr < \texttt{t\_exp}$.

For data that are stable in time, we assume that, $\forall t \in [t\_pub, +\infty]$, *P_exp(t)* = 1 and V = 0. In Figure 1, we show an example where `p_curr` and `p_exp` are the values of *P_exp(t)* in the instants `t_curr` and `t_exp` respectively.

The volatility of the data published in the instant `t_curr` can be graphically depicted as an area; it can range between:

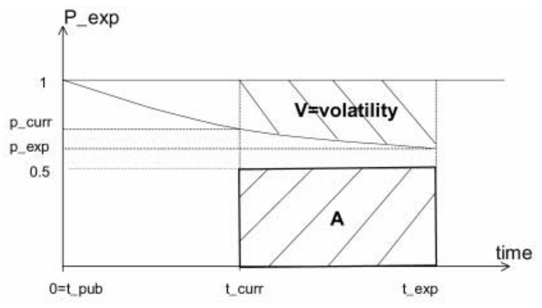


Fig. 1. A graphical representation of volatility

- a minimum value equal to 0, when *P_exp(t)* $= 1$ at all times after `t_curr`;
- a maximum value equal to (`t_exp` - `t_curr`), when *P_exp(t)* $= 0$ at all times after `t_curr`. Notice that this value should be considered as a theoretical limit, as it indicates that the probability that the expiration time `t_exp` will be the actual expiration time is 0.

We define a range `LOW`, `MEDIUM`, `HIGH` for volatility; if we consider as a reference the area $A = \frac{(t\_exp - t\_curr)}{2}$ (see Figure 1), volatility is:

- HIGH, if $V > A$;
- MEDIUM, if $V = A$;
- LOW, if $V < A$.

As an example of the usefulness of such kind of information, let us consider again the case of a professor moving to another university. It is possible to know that the date when the professor will leave is fixed or that it "may" change or that it is "very probable" that it will change. A `LOW` value of volatility corresponds to the fact that the date is definitive, a `MEDIUM` value to the fact that the date may change and a `HIGH` value to a very probable change of the date.

Temporal Dynamics of Completeness: Completability. In a similar way, we consider the temporal dynamics of completeness. We consider how completeness evolves in time, that is how a given aggregation of elements evolves towards completion. Let us consider the following given values:

- `t_max`: the maximum time within which the aggregation of elements will be completed;
- `t_pub`: the instant of publication of data,

We consider a function *C(t)* = estimate of the value of completeness in the instant t, where $t \in [\texttt{t\_pub}, \texttt{t\_max}]$. Starting from the function *C(t)*, we can define the **completability** of the published data.

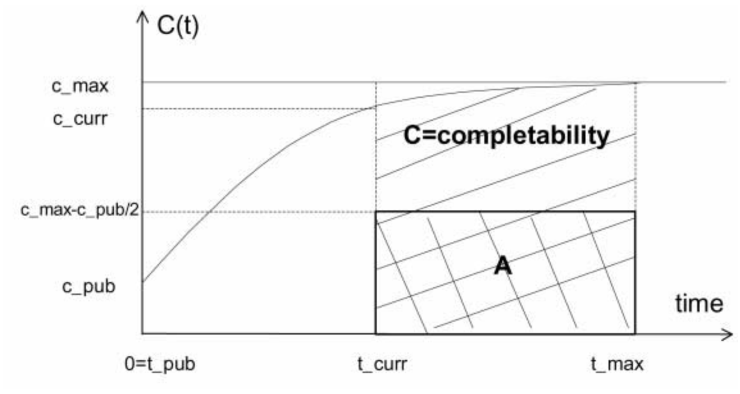


Fig. 2. A graphical representation of completability

Definition: Completability is C $= \int_{t\_curr}^{t\_max} C(t)$, where t_curr is the time at which completability is evaluated and $t\_curr <$ `t_max`.

In Figure 2, an example of completability is shown. Notice that the value corresponding to `t_curr` is indicated as `c_curr`; `c_max` is the value for completeness estimated for `t_max`. The value `c_max` is a real reachable limit that can be specified for the completeness of an aggregation of elements; if this real limit does not exist, `c_max` is equal to 1.

As in the case of volatility, we can define ranges for completability (`HIGH, MEDIUM, LOW`). We consider as a reference the area A $= (t\_max - t\_curr) * \frac{c\_max - c\_pub}{2}$ (see Figure 2). Then completability is:

- HIGH, if $C > A$;
- MEDIUM, if $C = A$;
- LOW, if $C < A$.

As an example, we can consider the list of courses published on the university web site; the completeness dimension gives the information about the current degree of completeness; the completability information gives the information about how fast such degree will grow in time, i.e., how fast the list of courses will be completed.

3 Modeling Data Quality

3.1 WIS Design Methodology

Several methodologies for WIS design (e.g., ARANEUS [12] and RMM [11, 10]) are based on the following sequence of steps (see Figure 3, left side):

- *Requirements Analysis*: it aims at understanding the information domain, what the WIS has to provide and who will be the users.

– *Hypertext Conceptual and Logical Design*: it implies the conceptual modeling of two aspects: *(i)* data that will be part of the hypertext, and *(ii)* paths to navigate among published data. This step also includes the design of some "logical" constructs, related to the ways to access information (i.e. indexes, tours, etc.) and to the structure of pages.
– *Presentation design*: it is focused on the interface design of the WIS.
– *Implementation*: it implies the generation of web pages.

To support the association of data quality information to web pages, we propose to introduce additional steps , related to the previously described ones, specifically devoted to data quality design (see Figure 3, right side):

– *Quality Requirements Analysis*: it is focused on the definition of the required quality dimensions to be associated to publishing data and of criteria and metrics associated to each dimension. The data quality dimensions in our approach are based on the definitions provided in the previous section.
– *Quality design*: the goal of this phase is to associate data quality information to pages and their contents.
– *Quality Implementation*: associating quality values to the previously defined dimensions.

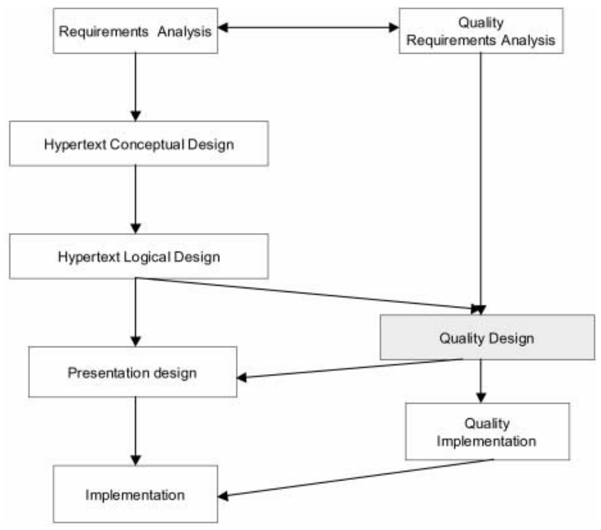


Fig. 3. The process of realization of a WIS (left side) and activities for data quality design (right side). The grey task is further considered in Section 3.2

3.2 A General Hyperbase Data Model for Data Quality

In this section, we show how to realize the step of quality design, shown in Figure 3, by proposing a model to associate quality information to web-publishing data.

Specializing a terminology introduced in [7], in the following we call as `hyperbase data model` a specific data model for hypertext representation that includes data structures, while being "purged" from the strictly navigational elements. An example of a hyperbase data model is the one of RMM [11, 10], which includes constructs such as attributes, entities, slices (i.e., groups of attributes of an entity), etc.; other constructs of RMM, such as guided tours, are not included as they are navigational elements.

We propose a `General Hyperbase Data Model` (GHDM) in order to abstract the features of hyperbase data models which are more relevant from a data quality perspective.

With respect to the possibility of associating data quality information to data elements, an important feature is the "granularity" of a data element. To clarify this concept, consider the source reliability dimension: it can be associated to a list of elements, if each element of the list derives from the same source; conversely, it can be associated to each element of the list, when the sources for them are different, and possibly with different source reliability values.

Therefore, in our general hyperbase data model we distinguish between atomic and aggregated elements and we associate data quality information to both of them by defining a *quality atomic element* and a *quality aggregated element*.

Definition: a `quality atomic element` is defined as a tuple `< atomic element, Set of Quality Dimensions (SQD) >`, in which:

– an atomic element is the schema of the smallest piece of information to be published on the Web (e.g., an attribute);
– a possible SQD is the one proposed in Section 2, and includes: *(i)* expiration, *(ii)* volatility, *(iii)* source reliability, *(iv)* accuracy.

Definition: a `quality aggregated element` is a tuple `< aggregated element (of level i), Set of Quality Dimensions (SQD) >`, in which:

– aggregated elements can have different levels of aggregation, i.e., they can be the aggregation of atomic elements or the aggregation of aggregated elements; examples are entities in the ER-model, that are an aggregation of attributes;
– a possible SQD includes: *(i)* expiration, *(ii)* volatility, *(iii)* source reliability, *(iv)* accuracy, *(v)* completeness and *(vi)* completability.

4 A Methodological Framework for Data Quality Management in WIS's

When introducing data quality in a WIS, in addition to data quality design, further issues must be considered, concerning data quality management: as an example, problems related to the updating of dynamically changing quality values and to quality improvement should be taken into account. In the present section, we describe a methodological framework for managing data quality in WIS's, discussing the principal activities that must be supported to manage data quality.

The main activities that should be made in order to cope with the requirements of an effective data quality management include:

- the definition of a set of data quality dimensions to associate to information to make available on the web;
- the definition of a conceptual model that defines quality elements according to which modeling data quality dimensions and associating quality information to data elements;
- the specification of a set of quality requirements to be satisfied by published data;
- the measurement of quality dimensions values for information to be published;
- the updating of quality dimension values that dynamically change;
- the improvement of the quality of published data.

We propose a methodological cycle to manage data quality in a WIS; the cycle consists of three iterative phases, namely: *Definition*, *Publishing* and *Cleaning* (see Figure 4). In the following subsections, each of the phases will be described.

4.1 Definition Phase

In this phase, the three following elements are to be considered:

- Data quality dimensions. The data quality dimensions that are interesting for the WIS should be chosen. A comprehensive choice should include both static and dynamic aspects of quality dimensions. The set of dimensions proposed in Section 2, and including expiration, completeness, source reliability, accuracy to catch static features and volatility and completability to catch the dynamic ones, is a potential set.
- Quality Model. A quality model has the role of modeling the chosen data quality dimensions as well as their associations to data elements in the data model. In Section 3.2, the GHDM has been proposed with the aim of addressing these features.
- Quality Constraints. It could be defined a set of quality constraints that data published in a WIS need to satisfy. A quality constraint can be seen as an expression `< dimension > < operator > < value >`, where `< dimension >`

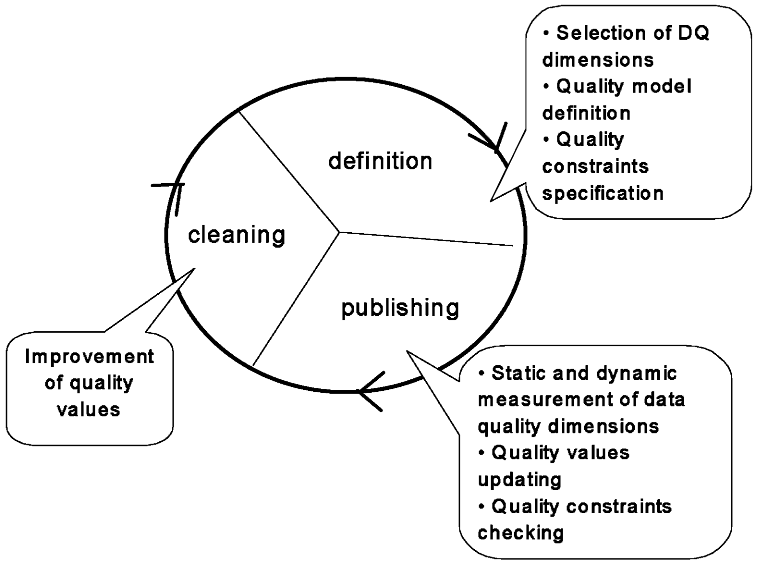

• Selection of DQ dimensions
• Quality model definition
• Quality constraints specification
definition
cleaning
publishing
Improvement of quality values
• Static and dynamic measurement of data quality dimensions
• Quality values updating
• Quality constraints checking

Fig. 4. A methodological cycle for data quality management in WIS's

is any data quality dimension, `< operator >` can be $=, <, >, \leq$ or $\geq$, and `< value >` is defined in the domain of the data quality dimension.
As an example, a data quality constraint that it is possible to state is that Course Programs are published on the university web site only if `Accuracy` $>=$ `6`, if measuring accuracy over a domain $\{1..10\}$.
A set of quality constraints can be specified: *(i)* for data which are going to be published, i.e., they will be published only if they satisfy such quality constraints; *(ii)* for published data in order to enact data cleaning actions when a fixed percentage of published data does not satisfy the specified constraints.

4.2 Publishing Phase

This phase generally includes the measurement of data quality values to publish as well as their updating. Moreover it is also possible to fix some "thresholds" for data quality dimensions values according to which to decide whether or not to publish some data. As an example, in the university web site one can choose that the information about the dates of examinations must be published only if their accuracy is `HIGH`.

Such thresholds are managed in the monitoring activity and can also trigger cleaning actions when the quality of published data goes down acceptable values.

The publishing phase is structured according to the following steps:

– static measurement;

– dynamic measurement;
– monitoring.

Static Measurement. This step implies the evaluation of the static values of data quality dimensions for data values which are going to be published. This step has to be made when data to be published have been decided; it is made once for a given set of data. Methods that can be used for measuring static data quality dimensions depend from each specific dimension chosen in the definition phase.

In the following, we describe possible methods to measure the set of quality dimensions proposed in Section 2. The expiration dimension is specified by the expiration date. For all data that expire, the expiration date should be directly associated to them. Moreover, when the expiration date for a data item is reached, two situations can occur: *(i)* expired data remain valid or *(ii)* expired data become not valid. To model this situation, expiring data have not only an expiration date associated to them, but also a `validity interval`, specifying the instant from which and the instant until which data remain valid [16].

As an example of such a situation, consider the case of a timetable of a course for a semester; supposing that the course is taught during the first semester of the year, the timetable expires at the end of the first semester but remain valid until the end of the year. Therefore, the timetable of the course has a a validity interval [`Start_1st_Sem`, `End_Year`].

Note that managers of the WIS can choose to maintain on-line not yet valid data or not, depending from the peculiarity of data themselves.

Completeness values can be calculated by considering the percentage of non-existent data values in aggregated elements. As an example, when publishing the list of the home pages of the professors of a university, all the URL's may not be available; by publishing only the available ones, the completeness of the list is a percentage of all the professors' home pages.

Source reliability values should be evaluated for each data type managed by the WIS. Data types, the values of which can be published only when an official source provide them (i.e., calls for tenders), are associated to `HIGH` source reliability values.

Accuracy can be checked by comparing data values with reference dictionaries (e.g., name dictionaries, address lists, domain related dictionaries such as product or commercial categories lists).

Dynamic Measurement. This step has to evaluate the dynamic values of data quality dimensions for publishing data. Differently from the static measurement, the dynamic measurement step is typically made many times, in specific instants that may be of two types: *(i)* pre-fixed, such as the publication instant (i.e., `t_pub`), and *(ii)* random, i.e., depending from unpredictable changes of parameters from which dynamic dimensions depend. In the data quality literature, as of our knowledge, there are no examples of dynamic data quality dimensions, therefore there are no proposals about measurement methods.

With reference to the dynamic dimensions proposed in Section 2.1, they can be measured as follows:

– volatility can be evaluated as `HIGH`, `MEDIUM` or `LOW`, according to the formulas described in Section 2.1. The pre-fixed instants of evaluation can be, beside the publication instant (i.e., `t_pub`), all other instants in the interval [`t_pub`, `t_exp`]. The random evaluation instants depend from anticipations or postponements of `t_exp`;
– completability can also be evaluated similarly to volatility. The random instants may depend from eventually changes of the maximum time within which the aggregation of elements will be completed (i.e., `t_max`) and/or the corresponding maximum value for completeness (i.e., `c_max`).

Monitoring. This step consists of controlling a set of parameters, from which data quality values associated to published web pages depend, and of enacting updates of such values. As an example, at the pre-fixed instants the evaluation (or re-evaluation) of dynamic values for data quality dimensions are enacted by monitoring conditions.

A further activity included in this step consists of the checking of the set of quality dimensions specified in the definition phase. The checking should regard both data which are going to be published and already published data, in order to enact cleaning actions.

The sequence of the publishing steps is shown in Figure 5.

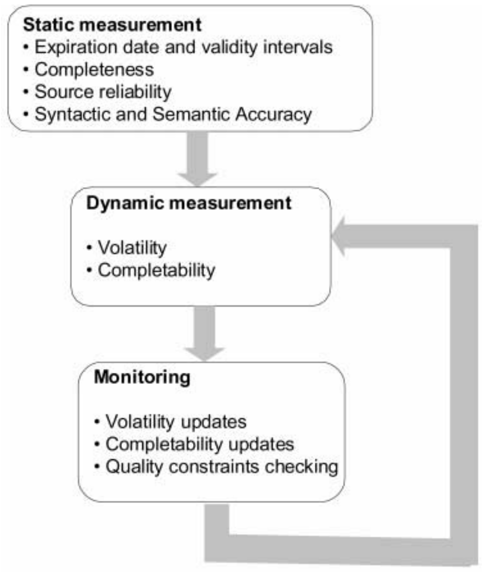


Fig. 5. Different steps in the publishing phase

4.3 Cleaning Phase

The cleaning phase concerns the improvement of the quality of source databases for published data. We propose to engage cleaning actions when the monitoring activity alerts that a specified set of quality constraints is not satisfied by published data. There are many research results concerning the cleaning of data (e.g., [6, 5]); but they only address accuracy and completeness, among the dimensions proposed in the present paper, and consistency, among the dimensions that have not been considered. No attention is paid to the improvement of source reliability and dynamic dimensions that we plan to consider in future work. Note that the cleaning activity may not succeed in improving data as much as it is necessary to satisfy the defined quality constraints. In such a case, the definition phase should be newly engaged.

5 An Architecture

The methodological framework described in Section 4 enables to manage data quality in a WIS. Such a framework can be realized by the software architecture shown in Figure 6.

The elements specifically introduced for data quality management are:

- a *quality knowledge repository* and
- a *quality manager*.

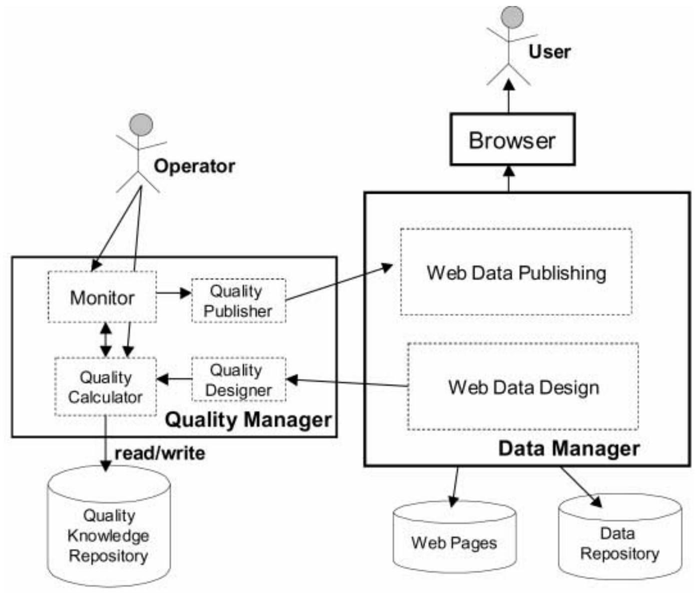


Fig. 6. An architecture supporting the quality management framework

Data values to publish are stored in the *data repository*; the *quality knowledge repository* contains, for each data value both of atomic and aggregated type, the static values of quality dimensions. Moreover, it also stores historical values for the estimation of dynamic values of quality dimensions.

The *quality manager* consists of four modules, namely *monitor*, *quality calculator*, *quality designer* and *quality publisher*. In order to describe how these modules interact, we distinguish: *(i)* a publication mode and *(ii)* an updating mode.

In the publication mode, the *quality designer* receives the conceptual elements representing data to be published together with references to data values from the module *web data design*. It also receives the specification of the quality dimensions to associate to such elements. On the basis of a quality model, such as GHDM, it associates quality elements to data elements, and produces a quality schema.

At this point, the *quality calculator* receives the produced quality schema and the set of data values references according to which: *(i)* it reads static data quality values from the *quality knowledge repository*; *(ii)* it calculates dynamic values. The calculation of dynamic dimensions should be made by interacting with an operator who provides the module with the real-time information it needs for.

In the publication mode, the *monitor* has only the task of checking the set of quality constraints that have been previously provided by the operator, against the quality values resulting from the activity of the *quality calculator*. If the checking is successful, data and the associated quality values are passed to the *quality publisher*, which acts as a bridge towards the external *web data publishing* module (that has the task of presenting both data and quality data).

If the checking is not successful, an alert is sent to the operator who has to take the opportune actions.

In the updating mode, the *monitor* plays the most important role. It has some triggers coded, such as the ones related to pre-fixed instants when dynamic dimensions should be re-calculated; in this case, it requests to the *quality calculator* to compute again dynamic dimensions, and after the quality constraint check, it gives new quality values to the *quality publisher*.

Note that when dynamic dimensions need to be re-calculated at random instant, the *monitor* is alerted by the operator. Moreover, the *monitor* has also some rules to be checked in order to eventually enact cleaning actions; such rules are derived from quality constraints on the overall quality of published information. The quality controls aiming at cleaning should be periodically realized by the *monitor*.

6 Related Work

In the literature, methodological frameworks to manage data quality within single information systems [18] and cooperative information systems [4] have already been proposed; instead, as of our knowledge, a methodological proposal to manage data quality in WIS's has not yet been addressed.

Most of the work that is relative to both data quality and web regards the assessment of the quality of web sites [1, 2, 8]. In [14], a Quality Evaluation Method (QEM) is proposed to assess the quality of academic web sites, by proposing more than a hundred characteristics. This approach allows an evaluation of the accomplishment of the characteristics required by web sites. Our perspective is different, as we give a method to *declare* the quality of the published information, more than to evaluate it "a-posteriori".

In [13], the problem of the quality of web available information has been faced in order to select data with high quality coming from distinct sources: every source has to evaluate some pre-defined data quality parameters, and to make their values available through the exposition of metadata. The approach proposed in this article is complementary to the one proposed in [13]. In fact, on one hand, our approach suggests a framework for the management of quality information associated to data. On the other hand, our focus is not on selecting sources on the basis of the quality of the provided data, but on considering the quality information of data published within a single site; in this case the published quality information can be detailed in a more specific way, including a dynamic characterization of dimensions.

7 Concluding Remarks

The data quality dimensions for data in web information systems proposed in the present paper are on one hand a subset of the most typical dimensions in data quality literature, and on the other hand provide an original view on these dimensions, since the dynamic evolution of web data is taken into account with information about the volatility and completability of web data.

The proposed data quality dimensions may be associated both to atomic and aggregated data in web pages. The main purpose is to provide clients and users accessing contents of web pages with a way to evaluate such an information with respect to its quality. The evaluation of data quality in WIS's may provide a way to guide users to select pages and to decide when to retrieve and make use of given information.

The proposed framework needs to be further refined in different directions. First of all, there is a need to study whether other dimensions traditionally proposed in data quality literature are applicable and useful in WIS's. In addition, the proposed approach associate quality attributes either to atomic elements or to aggregates. It is interesting to provide algorithms to derive quality information in aggregate web data from quality information associated to atomic elements. Another aspect which needs further investigation is the periodical nature of publication of information. For instance, the publication of course programs is periodical (i.e., yearly, or semester based), information about new degrees is published on a yearly base. Further research is needed to relate the periodical nature of information and data quality information about expiration and volatility of information.

Acknowledgments

This work is supported by MIUR, COFIN 2001 Project “DaQuinCIS - Methodologies and Tools for Data Quality inside Cooperative Information Systems” (`http://www.dis.uniroma1.it/∼dq/`). The authors would like to thank Massimo Mecella for important discussions about many issues addressed in this work.

References

[1] Atzeni P., Merialdo P., Sindoni G.: Web Site Evaluation: Methodology and Case Study. In Proceedings of DASWIS 2001: International Workshop on Data Semantics in Web Information Systems, Yokohama, Japan, 2001. 411

[2] Barnes S. J., Vidgen R. T.: Assessing the Quality of Auction Web Sites. In Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34), Maui, Hawaii, 2001. 411

[3] Batini C., Lenzerini M., Navathe S. B.: A Comparative Analysis of Methodologies for Database Schema Integration, ACM Computing Survey, vol. 15, no.4, 1984. 397

[4] Bertolazzi P., Scannapieco M.: Introducing Data Quality in a Cooperative Context. In Proceedings of the 6th International Conference on Information Quality (IQ'01), Boston, MA, USA, 2001. 410

[5] Dasu T., Johnson T., Muthukrishnan S., Shkapenyuk V.: Mining Database Structure or How to Build a Data Quality Browser. In Proceedings of the 2002 ACM/SIGMOD Conference, Madison, WI, USA, 2002. 409

[6] Galhardas H., Florescu D., Shasha D., Simon E.: An Extensible Framework for Data Cleaning. In Proceedings of the 16th International Conference on Data Engineering (ICDE 2000), San Diego, California, CA, 2000. 409

[7] Garzotto F., Mainetti L., Paolini P.: Hypermedia Design, Analysis and Evaluation Issues. Communication of the ACM, vol. 58, no. 8, 1995. 404

[8] Katerattanakul P., Siau K.: Measuring Information Quality of Web Sites: Development of an Instrument. In Proceedings of the International Conference on Information Systems (ICIS 1999), Charlotte, North Carolina, USA, 1999. 411

[9] Isakowitz T., Bieber M., Vitali F. (eds.): Web Information Systems (Special Issue). Communications of the ACM, vol.41, no.7, 1998. 397

[10] Isakowitz T., Kamis A., Koufaris M.: The Extended RMM Methodology for Web Publishing. Working Paper IS-98-18, Center for Research on Information Systems, University of Pennsylvania, Philadelphia, PA, USA, 1998. 402, 404

[11] Isakowitz T., Stohr E. A., Balasubramanian P.: RMM: a Methodology for Structured Hypermedia Design. Communications of the ACM, vol. 58, no. 8, 1995. 402, 404

[12] Mecca G., Merialdo P., Atzeni A., Crescenzi V.: The (short) ARANEUS Guide to Web-Site Development. In Proceedings of the Second International Workshop on the Web and Databases (WebDB'99) in conjunction with SIGMOD'99, Philadelphia, Pennsylvania, USA, 1999. 402

[13] Mihaila G., Raschid L., Vidal M.: Querying Quality of Data Metadata. In Proceedings of the 6th International Conference on Extending Database Technology (EDBT'98), Valencia, Spain, 1998. 411

[14] Murugesan S., Deshpande Y. (eds.): Web Engineering, Software Engineering and Web Application Development. Lecture Notes in Computer Science vol. 2016, Springer Verlag, 2001. 411

[15] Redman T. C.: Data Quality for the Information Age. Artech House, 1996. 399

[16] Tansell A., Snodgrass R., Clifford J., Gadia S., Segev A. (eds.): Temporal Databases. Benjamin-Cummings, 1993. 407

[17] Wand Y., Wang R. Y.: Anchoring Data Quality Dimensions in Ontological Foundations. Communications of the ACM, vol.39, no.11, 1996. 399

[18] Wang R. Y.: A Product Perspective on Total Data Quality Management. Communication of the ACM, vol. 41, no. 2, 1998. 410

[19] Wang R. Y., Storey V. C., Firth C. P.: A Framework for Analysis of Data Quality Research. IEEE Transaction on Knowledge and Data Engineering, vol. 7, no. 4, 1995. 399

[20] Wang R. Y., Strong D. M.: Beyond Accuracy: What Data Quality Means to Data Consumers. Journal of Management Information Systems, vol. 12, no. 4, 1996. 397

Conceptual Modeling Quality - From EER to UML Schemas Evaluation

Samira SI-SAID CHERFI* Jacky AKOKA** Isabelle COMYN-WATTIAU***

sisaid@cnam.fr, akoka@cnam.fr, isabelle.wattiau@dept-info.u-cergy.fr
* Laboratoire CEDRIC-CNAM, 292 Rue Saint Martin, F-75141 Paris Cedex 03,
** Laboratoire CEDRIC-CNAM et INT, *** Université de Cergy-Pontoise et ESSEC

Abstract. This exploratory research investigates the evaluation process of conceptual specifications developed using either Extended Entity-Relationship (EER) or Unified Modeling Language (UML) conceptual models. In this paper, we provide a comprehensive framework for evaluating EER and UML conceptual schemas. Furthermore, we define classes of metrics facilitating the evaluation process and leading to the choice of the appropriate representation among several schemas describing the same reality. Based on quality criteria proposed in the literature, we select a subset of criteria relevant to conceptual EER schema quality evaluation. For each criterion we define one or several metrics allowing the designer to measure the schema quality. We evaluate alternative EER conceptual schemas representing the same universe of discourse using the appropriate criteria and their associated metrics. Finally, we extrapolate this evaluation process to UML schemas. Following the development of our framework, we analyze a case study and provide evidence in the support of the usefulness of the framework.

Key-words: Conceptual modeling quality, quality criteria, quality metrics, user validation.

1. Introduction

Understanding users requirements may be a complicated problem for the system designer. Usually, the users express their problems using domain-oriented concepts and rules. From the users viewpoint, a conceptual schema should be supported by facilities for accessing, developing, analyzing, changing and maintaining concepts used in the construction of the conceptual schema. These five functions can be structured as three dimensions : the specification (analyzing), the usage (accessing, changing) and the implementation (developing, maintaining) dimensions. As a consequence, a conceptual schema developed using either EER or UML concepts provides a basis for the validation process. Although a conceptual schema can be consistent, it is not necessarily correct. In some cases, it can have nothing to do with the universe of discourse considered. There is therefore a need for criteria allowing us to measure the quality of a conceptual schema.

Unlike in software engineering, literature devoted to conceptual modeling quality is limited, in particular for UML schemas. This literature provides lists of desirable properties of conceptual schemas. Among general quality criteria proposed to evaluate conceptual schemas, let us mention : completeness, inherence, clarity, consistency, orthogonality, generality, abstractness and similarity. The formalization of these criteria is not yet sufficiently well understood. Abstractness measures the level of repression of detail. Similarity measures the capability of the conceptual schema to reflect the real world. But how do we measure these two criteria and how do we interpret the results obtained by the evaluation process? Although it appears very desirable to avoid conceptual schemas with low abstraction and low similarity, there is clearly a tradeoff between abstractness and similarity. How do we attain the value of this tradeoff ? A conceptual schema can be correct but not necessarily useful. Even if two conceptual schemas representing the same universe of discourse are considered to be equally correct, how do we choose the most friendly? Is it the simpler? How to characterize this simplicity? The aim of this paper is therefore to answer these questions. What are the desirable properties of a conceptual schema? How do we structure them in a global framework? How do we measure and interpret quality of EER and UML conceptual schemas? Can we perform automatically the measurement process? The underlying objective is to enrich the contribution of a CASE tool by computing a quality measure associated with each schema. This measure is to be considered only as a support for the final choice of a conceptual representation. This paper extends previous findings by providing an evaluation framework and by further examining quality criteria and classes of metrics relevant to both EER and UML conceptual schemas.

The structure of this paper is as follows : Section 2 synthesizes relevant literature. Section 3 is devoted to the presentation of the framework for conceptual quality evaluation. Criteria and their associated metrics are discussed. The results obtained by the application of the framework to a case study using EER model are presented in Section 4. Section 5 is devoted to UML schema quality evaluation using the same case study. Finally, we conclude in Section 6 and discuss areas of future research.

2. Related Literature

There exists abundant literature regarding the quality of software products. Software metrics are used to assess the quality of programs and more generally software products quality [1]. In the area of data quality, several attributes have been studied extensively by intuitive approaches : accuracy, timeliness, precision, reliability, currency, completeness, accessibility and relevancy [2, 3, 4, 5, 6, 7]. The theoretical approach uses ontologies in which attributes of data quality are derived based on data deficiencies [8]. The empirical approach collects data from data consumers to identify quality attributes [9]. Related research in conceptual modeling quality evaluation is emerging. The first structured approach dates back to the contribution of [10]. For the first time, they propose quality criteria relevant to conceptual schema evaluation (completeness, correctness, minimality, expressiveness, readability, self-explanation, extensibility, normality). In [11], the quality of schemas is evaluated along three dimensions: syntax, semantics and pragmatics. In [12], a set of criteria are proposed: homogeneity, explicitness, size, rule simplicity, rule uniformity, query simplicity and stability. Moreover, the authors describe a set of transformations aiming at improving the quality of schemas. In the context of Business Process Reengineering (BPR), [13] proposed a four dimensional framework for evaluating models and tools : functionality, ease of use, BPR trajectory and tool price and customer support. [10] extends by defining a comprehensive set of metrics taking into account the categories of actors: business users, data analysts, data administrators, and application developers [14]. These metrics have neither be validated nor implemented. However, the framework has been used to evaluate an application development project and to evaluate differences between data schemas produced by expert and novice designers [15]. [16] proposed general guidelines for modeling based on six principles: construction adequacy, language adequacy, economic efficiency, clarity, systematic design, comparability. Genero et al. have presented a set of automatically computed metrics for evaluating ER diagram complexity [17]. In addition, they have proposed an induction method for building fuzzy regression trees to evaluate the existence of relationship between the metrics and ER diagram maintainability subcharacteristics: understandability, legibility, simplicity, analysability, modifiability, stability and testability. [18] has proposed a set of measures to assess size, structure and behaviour aspects related to the complexity of object-oriented conceptual schemas. [19] have adapted and extended Lindland's framework to the comparison of reference models.

Our main contribution is to extend the literature criteria by elaborating metrics to capture the relevant properties of EER and UML conceptual schemas. As described in the next section, evaluating the schema quality means being able to exhibit the appropriate metrics associated with the criteria within a structured framework.

3. A Framework for Conceptual Quality Evaluation

Literature on conceptual schema quality mainly provides lists of desirable properties. Our framework addresses the issue of conceptual schemas quality measurement in a more systematic way. A conceptual schema quality can be measured by its capabilities (i) to provide a formal representation of the observed reality, (ii) meet users requirements, (iii) be a basis for the future information system implementation. Hence, the framework suggests to evaluate the quality of a schema in a space designed along three dimensions, namely the *specification, usage* and *implementation* dimensions (Fig. 1).

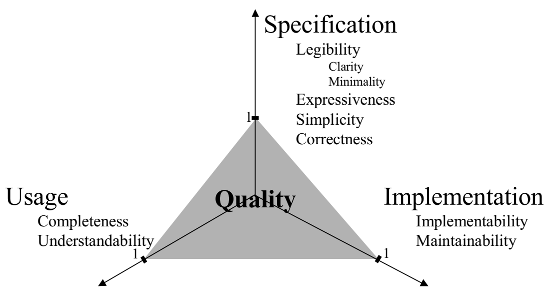


Fig. 1 The three dimensions of quality

This quality is maximal (value “1”) on each of these three dimensions. It corresponds to a schema that “perfectly” meets the desired characteristics for the specification, implementation and usage dimensions. The specification dimension is related to the conceptual phase and is described below. The implementation dimension refers to the production of a database from a conceptual specification, satisfying the various requirements. The usage dimension defines the ease of operations on the conceptual schema (accessing, analyzing, changing, etc.).

These three dimensions are not necessarily orthogonal. In this paper, due to the complexity resulting from interaction between them, we concentrate on the specification dimension only, since it is crucial for the evaluation process. A set of quality criteria is proposed for the two other dimensions. The metrics associated with this criteria set will be developed in the future. For the specification dimension, we define a set of measurable metrics and apply on several examples.

3.1. The Specification Dimension

A conceptual model is an abstract representation of the reality. This representation is the result of a specification based on a given set of notations. It is also a communication tool during the validation process. The notation provides the models with a formal semantics that enables tool-based analysis. The *specification* dimension captures the degree of requirements understanding in the conceptual schema. The literature contains several guidelines describing recommended characteristics for requirements specification [20]. We have identified the following criteria *legibility*, *expressiveness* and *simplicity*. They are the only measures of conceptual quality, which is our objective.

Legibility. To measure legibility which expresses the ease with which a conceptual schema can be read, we propose two subcriteria, namely *clarity* and *minimality*. *Clarity* is purely aesthetic criterion. It is based on the graphical arrangement of the elements composing the schema. When the number of its elements grows, a direct consequence is an increase of line-crossings which penalize readability [10]. Other criteria on the layout are described in [16]. They are interesting for clarity but difficult to measure as they imply length, angles and surface measuring. In fact, they characterize graphic choices. Our research is concerned with conceptual alternatives and their consequences on drawing. Therefore, our metric is only based on the number of line-crossings in the schema. Note that all the measures made in Section 4 are based on schemas that are equivalent regarding the other layout criteria.

Clarity = $\dfrac{\sum_R \dim(R) + NB(H) - CR}{\sum_R \dim(R) + NB(H)}$	Where NB(H) is number of inheritance links, dim(R) the dimension of R relationship and CR the number of line-crossings in the schema.

This metric is based on the heuristics stating that a schema containing N edges can have at most N crossings. In reality, it will have much less crossings. Each inheritance link provides one edge and each relationship provides a number of edges equal to its dimension. The dimension of a relationship is the number of its roles.

The second sub-criterion is *minimality*. A schema is said to be minimal when every aspect of the requirements appears only once [10]. In other words, non-minimality is due to a lack of factorization. A bad choice of entities and generalization hierarchies may lead to the replication of relationships in which several entities are involved playing the same role. To measure minimality we propose the following metric:

Minimality = $\frac{\sum_{S} wi\, NB(Ci) - wi\, NB_R(Ci)}{\sum_{S} wi\, NB(Ci)}$	Where Ci belongs to [entity-type, relationship-type, inheritance link-type]. NB(Ci) corresponds to the number of elements of type Ci, NBR(Ci) is the number of redundant elements of type Ci in the current schema S, w_i is the weight associated with Ci.

A schema containing no redundancy has a minimality value equal to 1. Minimality decreases when the number of redundancies increases. In this metric as well as in the others metrics described below, weights are associated with concepts, such as entity-type, relationship-type, and inheritance link. The motivation for adding weights is to take into account the impact of redundant concepts. For example, a redundant relationship of dimension i leads to one more node and i edges.

Expressiveness. A schema is said to be expressive when it represents users requirements in a natural way [10]. We distinguish between *concept* and *schema expressiveness*. *Concept expressiveness* measures whether the concepts are expressive enough to capture the main aspects of the reality. For example, an inheritance link is more expressive than a relationship in the EER model. Indeed, an inheritance link from entity-type E1 to entity-type E2 expresses the fact that: i) there exists a relationship between E1 and E2, ii) the set of E2 occurrences is included in the set of E1 occurrences, iii) E2 shares all properties of E1, iv) E2 participates to all relationship-types to which E1 participates. Thus we propose to associate weights with the different concepts involved. *Schema expressiveness* measures the expressiveness of the schema as a whole. It is clear that the greater is the number of concepts used the higher is the expressiveness of the conceptual schema. For example, in the "hospital management" case study (Section 4), a conceptual schema, in which several categories of doctors (Practitioner, Researcher, etc.) are specified, is more expressive than the one in which only one category (Doctor) is represented. In order to deal with both concept and schema levels, we propose the following metric :

Expressiveness= $\frac{\sum_{S} w_i\, NB(C_i)}{\sum_{Us} wj\, NB(Cj)}$	Where Ci ∈ {entity-type, relationship-type, inheritance link,..}, w_i is the weight associated with Ci , NB(Ci) is a function calculating the number of Ci concepts in a schema, S is the current schema to be measured and compared to the union of all schemas describing the same reality. This union is to be considered as the mathematical union operator in the sense that a concept present in several schemas is counted only once.

This metric provides a maximal value of expressiveness in the case of a conceptual schema constructed from the union of all the others. Such a union schema is not always correct because it is purely syntactic. It is the most expressive one and, obviously, the most redundant as it contains all the concepts of the other schemas.

Simplicity. A schema is said to be simple if it contains the minimum possible constructs. Our measure of simplicity is based on the assumption that the complexity of a conceptual schema grows with the number of relationships (including inheritance and aggregation links). As a consequence, a conceptual schema is simpler if its entities are more numerous than its relationships. Similar results can be found in [17]. We suggest the following metric to measure simplicity :

Simplicity$=\frac{NB(E)}{NB(R)+NB(H)+NB(E)}$	Where NB(E), NB(H), NB(R) correspond respectively to the number of entity-types, inheritance link-types and relationship-types.

A conceptual schema in which there are neither relationships nor inheritance links has 1 as a simplicity value, which is the maximal value for the simplicity function. On the contrary, if the number of (standard or inheritance) relationships is very high comparing to the number of entities, the value of simplicity approaches zero.

Correctness. Is used in a wide range of contexts leading to very different interpretations. A schema is syntactically correct when concepts are properly defined in the schema [10]. In the same way, a schema is semantically correct if concepts are used according to their definitions. However, another way of defining semantic correctness is to link it to conformity with requirements. Our objective is to make the quality measurement automatic. Therefore we are limited to syntactic correctness. Most of existing CASE tools support it. To measure correctness we suggest the following metric:

Correctness $=\frac{\sum_{i=1}^{N}(VERIF(C_i)-ERR(C_i))}{\sum_{i=1}^{N}VERIF(C_i)}$	Where VERIF() calculates the number of characteristics to be verified on an element Ci of the current schema. This number is the same for all the occurrences of the same type concept. ERR() calculates the errors depicted on an element Ci. N is the number of elements in the schema.

As an example on the value of the function VERIF(), let's consider that for an entity, we have defined the two following correctness rules : a naming rule and the obligation of an identifier definition. In this case, for each entity C_i of the schema VERIF(C_i) = 2. For a conceptual schema in which there is no error (ERR(Ci) = 0 for each element of the schema), the value of correctness is maximal (1). However, if no verification concludes to correctness (VERIF(Ci) = ERR(Ci), for each Ci), the value of correctness is minimal (0).

Summarizing, the first objective of a conceptual schema is to provide a formal representation of the observed reality. The specification dimension aims to measure how a conceptual schema respects some predefined specification characteristics.

3.2. The Usage Dimension

The usage dimension measures the quality of a conceptual schema according to the user's perception of both the system and the developed specification. A conceptual schema is judged to be "good" if it has the same perception of the system as the user's one. This means that the involved abstractions meet the user's vision of the reality and the users correctly interpret the schema. To take into account these two aspects, we propose two criteria, namely, *completeness* and *understandability***.**

Completeness. A schema is complete when it represents all relevant features of the application domain [10]. More specifically, the completeness can be measured by the degree of coverage of users requirements by the conceptual schema. Completeness is a very important criterion as it is crucial for the success of the future system. In other words, the degree of disparity, between user requirements and their interpretation by the designer as expressed in the conceptual schema, measures the gap between the user's and the designer's perception of the same reality.

Understandability. Understandability is defined as the ease with which the user can interpret the schema. This criterion is very important for the validation phase and consequently influences directly the measure of completeness. The understandability of a conceptual schema relies on how much modeling features are made explicit. Non-explicit names, a high level of aggregation of the modeling features, and the complexity of the defined integrity constraints are factors that decrease the schema understanding.
To summarize, the specification dimension measures the quality of the schema according to predefined modeling and representation rules. However, a good conceptual schema according to the specification dimension could not correspond to users requirements. The usage dimension measures the degree of correspondence between the conceptual schema and the user's requirements and the ease with which this correspondence can be made.

3.3. The Implementation Dimension

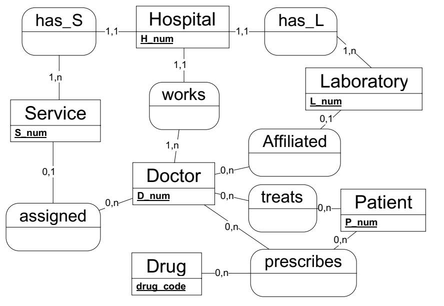
This dimension refers to the amount of effort needed to implement the conceptual schema. Two criteria are proposed to evaluate this aspect:

Implementability. Is defined as the amount of effort that a data administrator needs to provide to implement the conceptual schema. It depends on the degree of correspondence between the modeling concepts in the conceptual schema and the concepts of the target database management system.

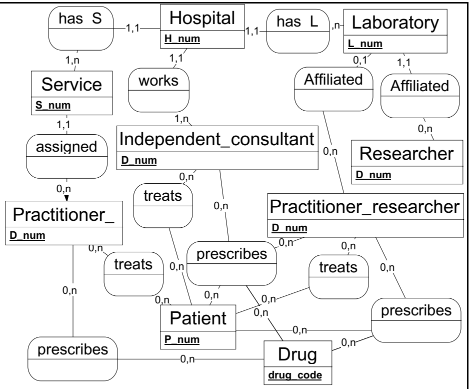
Maintainability. It measures the ease with which the conceptual schema can evolve. The maintainability of a conceptual schema implies the study of modeling elements cohesion. How do they coexist? When may they change? How frequently? The answers to these questions lead to a definition of precise characteristics necessary to insure a certain degree of maintainability.
Summarizing, the implementation dimension measures the ease with which a conceptual schema can be implemented and maintained.

4. Case Study: Hospital Management

We consider a sufficiently complex problem to illustrate the application of the framework. The quality here is measured to compare several schemas modeling the same reality. We make the assumption that all the schemas represent the same reality and are syntactically correct. We concentrate on the specification dimension. Let's consider a public organization managing the medical aspects of several independent hospitals. Each hospital has its own medical staff providing health care to patients in medical services. The research activity takes place in laboratories. There are three different functions performed by doctors : medical practice, research and consultations. More precisely, a doctor can be a practitioner in a medical service and/or researcher in a laboratory. These doctors are employees of the hospital. The others are independent-consultants whose activity is located in the hospital. Each doctor, whatever his functions, is affiliated to a unique hospital. Practitioners and independent-consultants prescribe drugs to their patients. The main difficulty in building a conceptual schema is due to the non-determinism of conceptual models. In most cases, several constructs are possible to represent a unique reality. In our case, there are several alternative categorizations of doctors. Several conceptual schemas among the most representative and resulting from this main choice are given below.

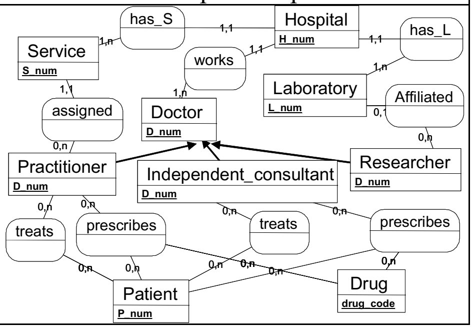


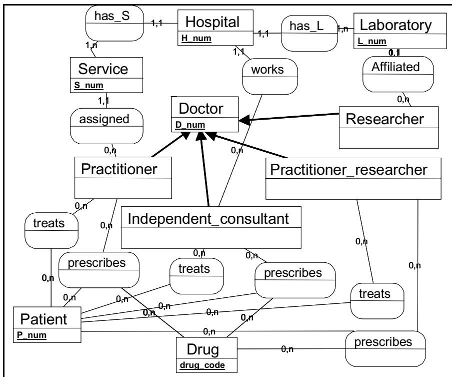
In this first schema, we don't differentiate between doctors functions. This choice is relevant when the validation is to be performed by non-professional users.



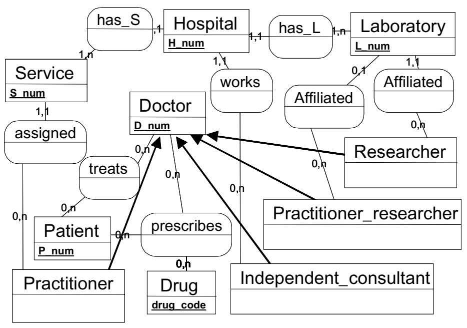
In this representation, we still don't use the generalization hierarchy. However, we distinguish between doctors functions leading to four entities. As a consequence, several relationships are replicated.

In this third schema, the generalization hierarchy is introduced to represent naturally the three different functions of doctors. This choice enriches considerably the expressiveness of the schema but leads also to some replications.

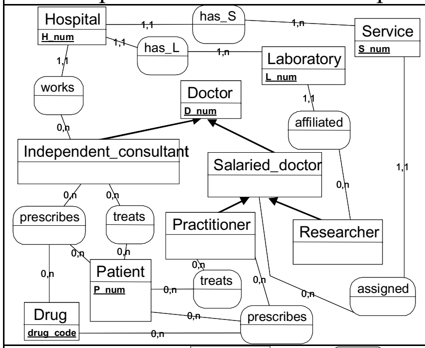




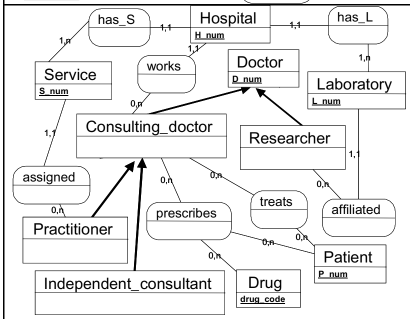
This fourth schema uses the generalization hierarchy to differentiate between the four possible doctors positions. The description resulting is more explicit but rather more complex.



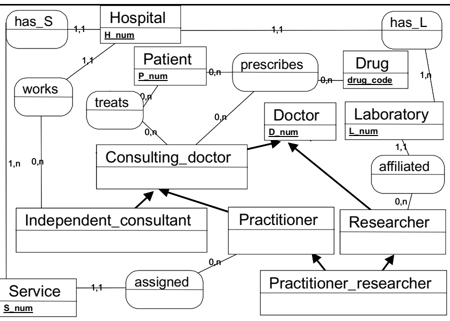
In this schema, the generalization choice is the same as the previous one. It exhibits the different categories of doctors but does not make explicit their functions.



In this sixth schema, two generalization levels are proposed. Based on the doctor employee status, the first level distinguishes between independent-consultants and employees. At the second level, employees may be practitioners or researchers (or both).



A second possibility is to distinguish at the first level between " pure" researchers and consulting doctors.



In this last schema, we introduce, at the second level of the generalization, a multiple inheritance.

We have evaluated the eight schemas using our framework and the metrics described in Section 3. The concepts weights are as follows : 1 for both entity-type and 1-1 relationship-type; 3 for a M-N relationship-type; 4 for both ternary M-N-P relationship and inheritance links. These values result from a simulation process conducted on several examples. The results are detailed in the following table.

Criterion / Schema	Legibility		Expressiveness	Simplicity	Correctness	*Average*
	Clarity	Minimality				
1	1.00	1.00	0.30	0.33	1	0.73
2	0.85	0.57	0.62	0.24	1	0.65
3	0.83	0.76	0.62	0.24	1	0.69
4	0.44	0.69	0.80	0.19	1	0.62
5	0.67	0.97	0.58	0.29	1	0.70
6	1.00	0.90	0.68	0.24	1	0.77
7	1.00	1.00	0.57	0.29	1	0.77
8	1.00	1.00	0.68	0.27	1	0.79

The correctness criterion is maximum for the eight schemas since we limited our comparison to syntactically correct schemas. The clarity measures vary from 0.44 to 1. Due to the small size of the schemas, four of them can be represented without any line-crossings. The fourth schema has the lowest clarity value (0.44). This is due to the numerous relationship replications leading to several line-crossings. This illustrates a strong dependency between the clarity and the minimality criteria : replicating relationships decreases the minimality value as well as the clarity one. Concerning the minimality measurement, let's notice that three schemas are excellent and equivalent due to the absence of redundant concepts (schemas 1, 7 and 8). They are not globally equivalent since they will differ notably on the expressiveness measurement. Schema 2 appears to be the worst since it distinguishes between four categories of doctors without using the generalization concept. This design choice maximizes the number of redundancies. According to expressiveness, schema 1 is the worst since it is based on ER concepts and represents all doctors in a unique category ignoring specific features. The best schema appears to be the fourth one. It makes explicit the greatest number of concepts contained in schemas. As for the simplicity criterion, there is no significant difference between the schemas. However, the first one is logically the best. Recall that it is based on the choice to produce a first cut schema for non-professional users. The comparison of the schemas based on the average of the different criteria values establishes three clusters of schemas. In the "best" cluster, we find the schemas 6, 7 and 8, based on two levels of generalizations. The multiple inheritance concept appears to contribute positively in the global quality evaluation of schema 8. The second cluster comprises schemas 1, 3 and 5. In particular, the first cut schema (schema 1) appears to be acceptable. Thus, one can question the usefulness of integrating complex concepts like in schema 8 to improve marginally the conceptual quality. The last cluster contains schemas 2 and 4. Although schema 4 was previously noticed as the most expressive schema. It confirms our belief that the overall quality cannot be limited to one criterion. This clustering is related to the choice of a standard average, which can be subject to discussion. Our framework can be used at least according to three viewpoints: (i) choosing the best schema based on a standard aggregation of criteria (i.e. the average of criteria discussed above), (ii) adapting the standard aggregation to a particular vision of the quality (i.e. defining a weighted average of the five measures), (iii) concentrating on a particular criterion.

5. Extending the Framework to UML Conceptual Schemas

Our state-of-the-art has shown a lack of research about UML schema quality. In this section, we describe the extension of our framework to deal with UML schemas. This extension allowed us first to validate the framework by a confrontation with another notation and second, to enrich the measures by taking into account new concepts. In this paper, we will restrict the analysis to the specification dimension.

5.1. Applying the Framework to UML Schemas

The quality criteria defined above hold for the UML conceptual schemas. The difference is that some concepts of models are different. The objective of this section is to revise the metrics and to adapt them to the concepts of the UML notation.

Assessing legibility of UML schemas. Remember that legibility is composed of two sub-criteria: *clarity* and *minimality*. Our definition of clarity is related to the number of the line-crossings in a conceptual schema. In the UML class diagram, lines may correspond to associations, links between association classes and their corresponding associations, compositions, aggregations or inheritance links. The clarity metrics is the following:

Clarity = $\frac{NB(A)+NB(AC)+NB(H)-CR}{NB(A)+NB(AC)+NB(H)}$	Where NB(H) is number of inheritance links, NB(A) the number of associations, NB(AC) the number of association classes and CR the number of line crossings in the schema.

Concerning the metric of minimality, it is unchanged as it is generic. The set of concepts (Ci) and their associated weights (*wi*) are adapted for the UML notation.

Minimality = $\frac{\sum_{S} wi\, NB(Ci) - wi\, NB_R(Ci)}{\sum_{S} wi\, NB(Ci)}$	Where Ci belongs to [class, association, inheritance link, class association, aggregation link, composition link]. NB(Ci) corresponds to the number of elements of type Ci, NBR(Ci) is the number of redundant elements of type Ci in the current schema S, wi is the weight associated with Ci.

Assessing expressiveness of UML schemas. The metric for assessing expressiveness is generic as it does not make explicit the concepts of the notation. It is consequently unchanged for the UML notation. Only the domain to which a concept belongs is adapted as well as the associated weights. Notice that the values of the weights are let to the appreciation of the designers.

Expressiveness = $\dfrac{\sum_{S} w_i\, NB(C_i)}{\sum_{Us} w_j\, NB(C_j)}$	Where C_i belongs to {class, association, inheritance link, association class}, w_i is the weight associated with C_i, NB(Ci) is a function calculating the number of Ci concepts in a schema, S is the current schema to be measured and compared to the union of all schemas describing the same reality.

Assessing simplicity of UML schemas. We apply again the principle that a conceptual schema is simpler if the concepts are more numerous than the links. Applied to UML, we say that a conceptual schema is simpler if the number of classes is higher than the number of links (associations and inheritance links) between the classes. The metric for simplicity assessment is as follows:

Simplicity = $\dfrac{NB(C)+NB(AC)}{NB(A)+NB(H)+NB(C)+NB(AC)}$	Where NB(C), NB(AC) NB(H), NB(A) correspond resp. to the number of classes, association classes, inheritances and associations (including aggregation and composition).

The metric for assessing correctness is the same as the one defined for EER.

To summarize, we can point out the fact that adapting the metrics to the UML notation induced a little rewriting. This reflects the validity of the underlying criteria. They could be in the same way adapted to any conceptual modeling tool.

5.2. Taking into Account Specific Features of UML

UML notation is devoted to object-oriented representation. Therefore, it allows the designer to refer to the main characteristics of object-orientation : abstraction, encapsulation, modularity and hierarchy [21]. Object-oriented models like UML are based on two fundamental mechanisms to cope with these characteristics : inheritance and aggregation. These mechanisms, although already present in EER schemas, are used in a more systematic way. Our framework needs to be adapted in order to take into account the effect of use or misuse of these abstraction mechanisms. As an illustration, we concentrate on the minimality criterion measure. We define the latter using two subcriteria: *inheritance effectiveness* and *aggregation effectiveness.*

Inheritance effectiveness. It measures the degree of inheritance hierarchies of a schema. The objective is to evaluate the effectiveness of inheritance use by measuring the level of factorization. The factorization process deals with the attributes as well as with the relationships and the methods.

Inheritance effectiveness = $\sum_{H}\left[\sum_{Ci}\left(1-\dfrac{DEF(Ci)}{USE(Ci)}\right)\Big/NB(Ci)\right]\Big/NB(H)$	Where H is a hierarchy, Ci Є {attribute, association, operation}, DEF(Ci) counts the number of occurrences of Ci in H, USE(Ci) counts the number of inheritances of an element Ci+1, NB(H) is the number of hierarchies, and NB(Ci) the number concepts in H.

The value of *inheritance effectiveness* is close to 1 when USE(Ci) is very high as compared to DEF(Ci) for all the concepts Ci. This is the case when the schema has deep hierarchies with many elements located at the most abstract levels of the hierarchy, leading to a high number of inheritances. As an illustration, let's consider two UML notations of the hospital management case (Fig. 2). The (a) representation is a categorization of doctors leading to a low factorization of methods. The (b) representation illustrates a judicious use of multiple inheritance.

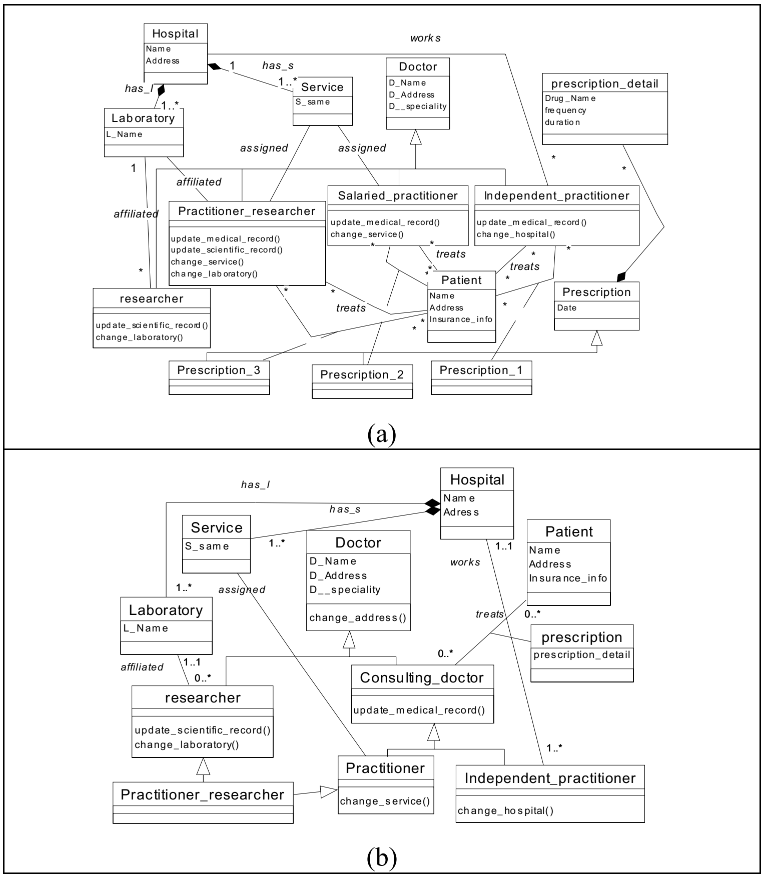


Fig. 2: UML representations of hospital management

	Concept Ci	D_name	D_address	D_speciality	change_address()	update_medical_record()	update_scientific_record()	change_service()	change_hospital()	change laboratory()	treats:association	assigned	works	affiliated	prescription:class_association	p_date	aggregation: prescription-prescription_detail	inheritance effectiveness
Schema (a)	DEF(Ci)	1	1	1	1	3	2	2	1	2	3	2	1	2	3	1	1	0.24
	USE(Ci)	5	5	5	5	3	2	2	1	2	3	2	1	2	3	4	1	
Schema (b)	DEF(Ci)	1	1	1	1	1	1	1	1	1	1	1	1	1	1			0.58
	USE(Ci)	6	6	6	6	4	2	2	1	2	4	2	1	2	4			

The results of the application of this metric are detailed above. The measures of inheritance effectiveness confirm our intuitive interpretation of the misuse of generalization hierarchy in the (a) schema. The lack of factorization in this schema leads to a poor conceptual schema. This factorization illustrated in schema (b) improves significantly the schema quality. However, this result (0.58) is yet unsatisfactory since the expected value should be close to 1.

Aggregation effectiveness. It measures the efficient use of aggregate attributes. This quality measure is based on aggregation levesl. In object-oriented paradigm, an attribute domain may be a collection of basic data types or an aggregation of such types. It can also be a class or a complex construct based on several classes. The Unnest function explicits the different aggregation levels for a given attribute. The Level function counts the number of levels. As an example, Unnest is applied to the attribute “prescription_detail” from the association class “prescription” (Fig. 2) :

Unnest(prescription_detail) = Aggregate [date ; Collection (Aggregate [frequency; duration, Aggregate [drug_name ; drug_description]]].

In other words, the attribute “prescription_detail” is characterized by five abstraction levels (Level=5). If an attribute domain is a basic data type, a Level value equals to 1. Hence, the metric of aggregation effectiveness is given below:

Aggregation effectiveness = $\sum_{ci}\left(1-\frac{1}{Level(Unnest(Ci))}\right) \Big/ NB(Ci)$	Where Level(Unnest(Ci)) counts the number of aggregation levels of an attribute Ci and NB(Ci) is the number of attributes in the schema.

Notice that if a schema does not contain any aggregate attribute, Level(Unnest(Ci))=1 for each attribute, leading to a global 0 value for this metric. It confirms our belief that these measures do not replace previous measures but must be added to the latter to enrich the overall quality evaluation.

6. Conclusion and Future Work

Our contribution is multifold. First, our evaluation framework is an extension to existing approaches by differentiating the specification dimension from the usage and implementation dimensions. Second, our approach captures the role of the specification dimension in the evaluation process. Third, our approach captures the essential tradeoff between the benefit of a criterion such as legibility and the remaining criteria such as expressiveness and simplicity. Finally the results obtained using the case study on EER schemas hold for UML schemas. Although the analysis can be seen as more complex, we have extended the framework and the associated metrics to UML schema quality evaluation. Note that the objective of integrating these measurements in a CASE tool has continually constrained our choice of metrics. Two major extensions can be made. First we should integrate the usage and implementation dimensions. Second, we will integrate the metrics to a CASE tool. Finally, we need more case studies in order to reach more definitive conclusions on the respective weights of concepts and criteria.

References

1. Davis A., Software Requirements: Analysis and Specification, Prentice Hall, 1990.
2. Zmud R., M. Lind, Young F., An attribute space for organisational communication channels. Information systems research, 1(4), 1990.
3. Kriebel C. H. Evaluating the quality of information systems. In design and implementation of computer based information systems. 1979
4. Bailey J. E., Pearson S. W., Development of a tool for measuring and analysing user satisfaction. Management science, 29(5), 1983.
5. Ives B., Olson M. H., Baroudi J . J., The measurement of user information satisfaction. Communication if the ACM, 26(10), 1983.
6. R. Y. Wang, Henry B. Kon, S. E. Madnick: Data Quality Requirements Analysis and Modeling. Int. conf. on data engineering. 1993: 670-677
7. Wang R. Y., Redy M. P., Kon H. B., Towards Quality Data: An attribute-based Approach. Decision Support Systems, 13, 1995.
8. Wand Y., Wang R. Y., Anchoring data quality dimensions in ontological foundations. In communications of the ACM, Vol. 39, No. 11 November, 1996.
9. Wang R. Y., Strong D. M., Beyond accuracy: what data quality means to data consumers. In journal of information systems (JMIS), 12(4), 1996.
10. C. Batini, S. Ceri, S.B. Navathe, Conceptual database design : An entity relationsip approach, Benjamen Cummings, Redwood City, California, 1992.
11. Lindland, O.I., G. Sindre and A. Sølvberg, Understanding quality in conceptual modelling, IEEE Software, Vol. 11, No. 2, March 1994, 42-49.
12. Assenova P., Johannesson P., Improving quality in conceptual modelling by the use of schema transformations. In the proceeding of ER'96. Cottbus, Germany, 1996.
13. Teeuw B., Van Den Berg H., On the Quality of Conceptual Models. Proceedings of the ER'97 Workshop on Behavioral Models and Design Transformations: Issues and Opportunities in Conceptual Modeling 6 - 7 Nov. 1997, UCLA, Los Angeles, California
14. Moody D.L, Metrics for Evaluating the Quality of Entity-Relationship Models, 17th Int. conf. in Conceptual Modeling (ER98), Singapore, LNCS 1507 (Ling, Ram, Lee eds).
15. Moody D.L, Shanks G.G, Darke P, Improving the quality of entity-relationship models – experience in research and practice, 17th Int. Conf. in Conceptual Modeling (ER98), Singapore, LNCS 1507 (Ling, Ram, Lee eds).
16. Reinhard Schuette, Thomas Rotthowe: The Guidelines of Modeling - An Approach to Enhance the Quality in Information Models. ER 1998: 240-254
17. Genero M., Jiménez L., Piattini M., Measuring the quality if entity relationship diagrams, in the proceedings of ER2000 conf., LNCS 1920, pp.513-526.
18. Poels G., Dedene G., Measures for dynamic aspects of object-oriented conceptual schemes, in the proceedings of ER2000 conf., LNCS 1920, pp. 499-512.
19. Misic VB, Zhao JL. Evaluating the quality of reference models. In proceeding of the 19th Conf. on Conceptual Modeling ER2000, pp. 484-498.
20. IEEE Recommended Practice for Software Requirements Specifications, (Std 830-1993).
21. Booch G., Object Oriented Design with applications, Benjamin Cummings, 1991.

Conceptual Modeling for Customized XML Schemas

Ramez Elmasri, Yu-Chi Wu, Babak Hojabri, Charley Li, and Jack Fu

Department of Computer Science and Engineering
The University of Texas at Arlington
P.O. Box 19015, Arlington, TX 76019, U.S.A.
{elmasri,yuwu,hojabri,qingli,shengfu}@cse.uta.edu

Abstract. XML was initially developed for document management, but it is becoming increasingly used for storing and exchanging all kinds of data on the Internet. In this paper, we introduce a design methodology for XML schemas that is based upon well-understood conceptual modeling methodologies. Because XML is hierarchical (tree-structured), many different XML schemas (or document structures) can be generated from the same conceptual database schema. We describe algorithms for generating customized hierarchical views from EER model, creating XML schemas from hierarchical views, and creating XML instance documents.

1 Introduction

The Extensible Markup Language (XML) [1] was initially developed for document management, but it is becoming increasingly used for storing and exchanging all kinds of data on the Internet. Some have even hypothesized that the XML format will replace databases in the future [2], although this is not very likely. XML has features from the various traditional data models, including the legacy hierarchical data model [3], as well as the relational and object oriented data models [4]. Much work has already been done related to how to store XML using relational [5,6], object-oriented, and object-relational databases [7,8,9], but little work has been done in the area of methodologies for designing XML schemas for specific applications. In this paper, we introduce a design methodology for XML schemas that is based upon well-understood conceptual modeling methodologies. Our approach is system-independent, because it is based on conceptual modeling techniques that are independent of any specific commercial system.

In general, because XML is hierarchical (tree-structured), many different XML schemas (or document structures)[1] can be generated from the same conceptual database schema, and hence from its underlying database. Therefore, there is a need for an XML schema design methodology that helps users in specifying specific XML document structures for their applications. A lot of current data sources use relational databases, whose relational schemas can be reverse engineered [10] into Extended

[1] We will use the terms *XML schema* and *XML document structure* interchangeably.

Entity Relationship (EER) schemas [4,6] (or other similar conceptual data models such as UML class diagrams [11]). By applying our methodology to EER schemas, data sources from relational databases can be converted to a variety of XML document structures, which can then be used by different user applications.

Although standard mappings from relational schemas to XML schemas exist [12,13], we will argue and demonstrate that those mappings rarely produce what the user needs. In fact, default mappings often produce XML schemas that conceptually mirror the relational schemas. In our methodology, users will be able to easily customize XML document structures for their specific applications. In particular, different hierarchical views corresponding to different XML document structures can be created by our methodology. In addition, the tedious part of writing XML Schema documents[2] will be automated from graphical EER diagrams that are easily understood and manipulated by the users.

The rest of this paper is organized as the following. In section 2 we briefly introduce the architecture of XML Schema Designer, a visual tool for designing XML Schema and instance documents using EER conceptual modeling, and describe the various ways it can be used by presenting several user scenarios. In section 3, we describe algorithms for generating customized hierarchical views from EER model, creating XML Schema documents from hierarchical views, and creating XML instance documents. In the last section of the paper, we discuss the benefits of this methodology and possible future work.

2 XML Schema Designer Architecture and Modules

XML Schema Designer is a visual XML schema designing tool that allows the user to generate XML Schema documents using well-understood conceptual modeling techniques, including EER modeling and UML class diagrams. It also automates the generation of SQL query scripts, the extraction of relational data, and the creation of XML instance documents. Only the EER model is supported in the current version of the XML Schema Designer, but future versions will support UML class diagrams. Figure 1 shows the overall architecture of XML Schema Designer.

XML Schema Designer consists of six components as shown in Figure 1. The following is a brief description of each module.

EER Reverse Engineering. This module extracts an existing relational schema from a relational database using the Data Access module, and creates a corresponding EER schema, which can then be used for designing XML schemas for parts of the relational database. The EER metadata, stored in an internal format, is used as the basis for GUI editing in the EER Designer module.

RDB Designer. Given an EER schema created using EER Designer, this module will create a corresponding RDB schema using the Data Access module. It also facilitates loading the relational data.

[2] An XML Schema document describes the structure of a set of XML documents, and conforms to the XML Schema recommendation [14].

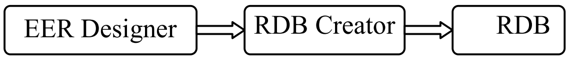
EER Designer. This module is the visual component (GUI) that a user interacts with to create and edit EER schema diagrams. The user can either retrieve the EER schema using the EER Reverse Engineering module, or they can create the EER schema from scratch. The module initiates creation of the RDB schema through RDB Designer if the EER schema is a new one.

Hierarchy/XML Schema Designer. This module allows a user to create various XML document structures based on the underlying database. The user first chooses the entity types and relationships of interest, which are passed to Hierarchy/XML Schema Designer module as input to create a hierarchy and XML schema. The user first selects a subset of the entity types and relationships from an EER diagram, then chooses a root entity type. This module creates an entity hierarchy, which represents an XML document structure, and also creates the corresponding XML Schema document.

XML Instance Document Generator. Given an XML schema, this module generates a corresponding SQL query script to retrieve data from the relational database, and it also generates the corresponding XML instance document from the query result.

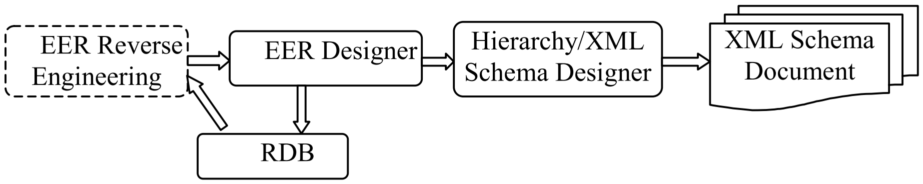
Data Access. This module is the general database access component that provides access to relational database, possible operations including querying RDB metadata, creating RDB schemas, loading RDB data, querying RDB data, and saving EER schemas.
There are basically three types of users that will benefit from using this tool.

User 1: Relational Database Schema Designer



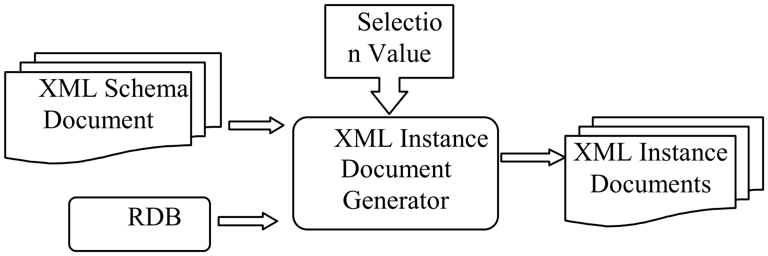
A RDB Schema Designer can use this tool to design the EER diagram graphically. The tool will automatically create the corresponding RDB schema on the RDB server. Future enhancement of this tool will also provide facilities for loading RDB data.

User 2: XML Schema Designer



This type of user can generate any number of XML schemas based on a specific RDB/EER schema. The user will either retrieve the EER schema from an existing RDB using EER Reverse Engineering module, or design it from scratch using EER Designer module, or retrieve a saved version of some previously created EER schema diagram. The user can then select entities and relationships of interest for a particular XML document structure, and a root entity. The tool will automatically generate an entity hierarchy and convert the hierarchy into an XML schema.

XML Data End User



This type of user can generate XML instance documents from an existing RDB loaded with data. The user first chooses an existing database and a saved XML schema, then enters selection values. The tool will automatically generate an SQL query, execute the query to extract the data from the RDB, and format the extracted data in XML document format.

3 Algorithms for XML Schema Designer

XML Schema [14] is the new standard for specifying the format and constraints of XML documents. One use of XML is for extracting information from relational databases and converting the information into a hierarchical XML document format [12, 8]. This makes it possible to exchange data between databases and XML documents for use on the Internet [15]. XML documents are hierarchically structured. In general, it is possible to create different hierarchical views of a relational database [16]. Since different users may need to see a specific document extracted through a particular hierarchical view, the need to extract different XML Schema documents becomes significant. This section presents three algorithms used in our tool that support the conceptual modeling of XML document structure.

3.1 Algorithms for Generating Conceptual Hierarchical View from EER Model

As a first step, we generate a conceptual hierarchical view for the subset of the entities and relationships that user has already selected for a particular XML document structure. It is likely that the user selection is an EER graph, which contains one or more cycles. In this case, we first eliminate the cycles before we proceed to generate the hierarchical view. First, we present an algorithm to break up the cycles. Then, we present the algorithm for hierarchical view formulation.

Algorithm for Eliminating Cyclic Entities/Relationships in User Selection. The purpose of this algorithm is to eliminate potential graph cycles in the EER diagram subset selected by the user. Prior to this step, the user has already selected a subset of the entities and relationships that are of interest, and the user would have already identified a root entity for the selection. In essence, this algorithm first categorizes all nodes, except root, into different node groups according to their shortest distance from the root (level). It then tries to find the furthest entity node away from the root that participates in a cycle. The algorithm then breaks the cycle by duplicating that entity node (only the entity node, not the subgraph attached to it), so that it still maintains each relationship it originally has with other nodes. At this point, we have a new graph, and we repeat the process until no cycles are detected.

```
Input: User Selection (G, subgraph of EER), with root R of the graph
identified
Output: A tree with the same root, without any cycles

 Let G be Graph with n nodes and R be selected root.
 Graph2Tree()
 {
      While (there is a cycle in graph G)
           {
              G = Break_Cycle(G);
              G = new gerented graph
           }
      Output the graph G;
 }
 Graph Break_Cycle(graph G)
 {
   1. Use Breath First Search (BFS) to define total ordering on the
   nodes based on distance from root, use Depth First Search (DFS) to
   define edge categories (tree edge or back edge).
   2. Start from the node with deepest level to top level, and do
   cycle elimination:
   Let V denote current node
   For each V {
   For each node that connect to V (NBR-neighbors) from the node with
   deepest level to top level of neighbors. {
      If there is a back edge between V and NBR {
           //Search the cycle path until back edge connected node.
           While cycle_node (the node is moving to search the cycle
           path) is not back edge connected node. {
              Find the node connected to cycle_node with tree edge in
              the search path and add cycle_node to cycle_list (the
              list contain all nodes in the cycle path).
           }
           Construct a list(c_depth_list) and sort it according the
           level of nodes in cycle_list.
           Let Copy_V be the node that is going to duplicate.
           Copy_V = the node that is deepest node in c_depth _list.
           Pre_Copy_V = find the node that is one position ahead the
           Copy_V in term of level in c_depth_list
           G = new_graph (n, Pre_Copy_V, Copy_V )
           Return G;
      }
   }
 }
   Return G;
 }
```

The process to eliminating cycles is the preprocessing procedure for generating a hierarchical view. This algorithm has been implemented and sample results are shown in Figure 6.

Algorithm for Hierarchical View Formulation. This Algorithm is for generating XML schema documents based on users' choices. In the front end, the user-friendly tool will use an EER graphical representation of the database schema to specify the criteria for document generation. The user identifies the root entity type to choose a particular hierarchical view for a specific XML Schema document. Based on the relationships between the root and other entity types, the corresponding hierarchy will be recognized by the system and the XML Schema document will be automatically generated.

Given a portion of interest of a source EER schema that has already been preprocessed to eliminate cycles, and a root entity R, we construct the hierarchical view by invoking the recursive procedure "Generate_Hierarchy (E1,AE1)". There are two input parameters for this procedure.

First parameter is for the current entity that needs to be transfered to hierarchical view.

Second parameter is the ancestor's entity. It is used when the child relationship of current entity is N:1 or 1:1.

In the procedure, we define that
The relationship that is connected to E1 be "L".
The entity type that is related to E1 though L be "E2"

The procedure Generate_Hierarchy (R, R) creates the hierarchical view. It basically merges child entity types that participate in N:1 or 1:1 relationships with their parent entity type. For 1:N or M:N relationships, the relationship attributes are migrated to the child entity type. This keeps the number of entity types in the hierarchical view to a minimum. The psuedocode of the procedure is as follows:

```
Generate_Hierarchy( EntityTypeName E1, AE1, RelationshipName Re)
{
   Let P be each path that has relationship with E1
   Let the path Re be the relationship between E1 and it's parent.
   If E1 is not R(root) { remove the path Re from P }
   For each path P
   Do{
      Let the relationship that is connected to E1 be "L".
      Let the entity type that is related to E1 though L be "E2"
      If L is M:N or 1:N relationship
      {
         Make E2 be the child of E1 in the hierarchy.
         If L is M:N relationship
         {Change L to 1:N relationship (1 to E1 and N to E2) }
         If L has it's own attributes
         { merge L's attributes to E2     }
         // Mark the attribute originally belongs to L with color 1
         Generate_Hierarchy (E2, E2, L)
      }
      else if L is N:1 or 1:1 relationship
      {
         if E1 and AE1 are not the same
         { E1 <= AE1       // Replace E1 With AE1}
         Merge E2's attributes to E1.
         If L has it's own attribute
         {  merge L's attributes to E1    }
         // mark the attributes originally belong to E2 with color 2
         // mark the attributes originally belong to L with color 1
         Generate_Hierarchy (E2, E1, L)
      }
   }
}
```

In the procedure, we color the attributes that originally belong to a relationship with color1 and the attributes that originally belong to a merged child entity with color2. As a result, we can distinguish each attribute as to whether it originally belonged to another entity or relationship.

An Example to Illustrate Hierarchical View Formulation. Consider an application that needs XML/Schema documents for student, course and grade information, to be extracted from the university database shown in Figure 2. The user is only interested in the entity sets, COURSE and STUDENT, and the relationships between them via the SECTION entity type. Then, in step 1, these three entity types and their attributes are selected, as well as the relationships S-S and C-S. The EER diagram after this step is shown in Figure 3. In step 2, based on user's choices for XML document structure, it is possible to formulate at least three possible hierarchies:

First, a user can choose COURSE as root, as illustrated in Figure 4a. Since the relationship cardinalities along the path from parent to child in this case are 1:N relationships, there is no need to merge entities. The "Grade" attribute in the S-S relationship is migrated to the STUDENT entity. This is because, by choosing COURSE as root, SECTION becomes a child of COURSE, and STUDENT becomes a child of SECTION in the hierarchy. All STUDENT elements that are children of a specific section are related to that SECTION, and hence can have a specific grade in that SECTION. In this hierarchy, a STUDENT taking more than one SECTION will have several replicas, one under each SECTION, and each will have the specific grade given in that particular section.

Second, the user can choose STUDENT as root (Figure 4b). In this view, in step 3 each section is related to one course, so the relationship between SECTION and COURSE is N:1. We can hence merge COURSE and SECTION entities as shown in Figure 4b. In addition, the "Grade" attribute is migrated to the SECTION entity. In this hierarchy, COURSE/SECTION information is replicated under each separate STUDENT who completed the section.

The third possible way is to choose SECTION as root, as shown in Figure 4c. Similar to the hierarchical view, the COURSE merges into the SECTION entity and the "Grade" attribute is migrated to the STUDENT entity.

Discussion of Hierarchical View Formulation. As we can see, even in this simple example, there can be numerous hierarchical views, each corresponding to a different XML document structure. We illustrated the process of hierarchical formulation. The advantage of this process within our tool is to provide a user-friendly interface and let the user formulate a specific hierarchical view without knowledge of the detailed structure and content of the database or XML schema constructs. The completed hierarchical view is the first stage for designing a customized XML Schema document. We next discuss the algorithm to convert a hierarchical view into an XML Schema document.

3.2 Generating XML Schema and Instance Documents from Conceptual Hierarchical View

In this section, we present the algorithm to build an XML Schema document from a hierarchical view. We will then apply the algorithm to an previous example and generate the corresponding XML Schema document.

Algorithm for Generating XML Schema Document From Conceptual Hierarchical View. The algorithm steps are as follows, and are illustrated using the hierarchy in Figure 4b:

1. Define a root element for the XML Schema document, which will be used to give a name to the entire schema document. The type of this element will be "rootType", which is a complex type element. In our example, we choose the name StudentTranscriptDoc for the root element.
2. Call the recursive procedure "Generate_XML_Schema", using the root entity of the hierarchy as its argument. In our example (Figure 4b), STUDENT is the root of the hierarchy, so we pass STUDENT as input to "Generate_XML_Schema".

We now describe the recursive procedure "Generate_XML_Schema":

```
Generate_XML_Schema (EntityTypeName E1)
{
Generate a complex type element for E1 and its type, "E1Type"
Under the E1Type element
  Do{
      For each EER attribute A of E1
      {
              Create a complex type XML element e corresponding to A
              Define a simpleContent3 element within e
              Create four XML attributes for e
                     FIELD_NAME, FIELD_TYPE, FIELD_LEN, NULLABLE.
      }
              For each entity E2 that is a child of E1, in the
              hierarchy
              Do { Call Generate_XML_Schema (E2) }
    }
}
```

An Example of Generating XML Schema Document. Now that we have described the algorithm, we use the example given in section 3.1.2 with STUDENT as root. We first define a root element for this XML Schema document, and we choose the name "StudentTranscriptDoc" to represent that this schema contains student transcript documents. Since we use STUDENT as the root of our hierarchical view, STUDENT will be the initial current entity type (E_1) that is input to "Generate_*XML*_Schema".

In "Generate_*XML*_Schema", we generate a complex element for STUDENT and its type, "STUDENTType". Under the STUDENTType element, there are three EER attributes, Ssn, Name, and Class of the STUDENT entity type, so we generate three complex elements S_SSN, S_NAME and S_S_CLASS, respectively in XML Schema document for each EER attribute, and each element will have four XML attributes that describe their properties from the database system. Next, we apply the same

[3] A SimpleContent element contains either extensions or restrictions on a complexType element with character data, or contains a simpleType element as content and contains no elements [14].

scheme recursively to the SECTION entity type, which is a child of STUDENT in the hierarchical view. Figure 5 illustrates the brief view of the generated XML Schema document.

Formulating SQL Query. In order to generate XML instance documents based on a particular hierarchical view of the database, we need to have a SQL query that extracts data from the database, in addition to some information about the hierarchy. The SQL query text can be stored within the XML Schema document or in a separate text file. This query must be formulated such that it selects all the attributes of the hierarchy ordered by the primary keys of the entities. The tables must be joined together using outer join starting from the top-level entity and in a breadth-first order. The result of the query is used to generate the XML instance document.

Generating XML Instance Documents. Some information about the hierarchy is used for generating the XML instance documents. This includes:

1. Data about the hierarchy including:
   numEntities: Number of entities in the hierarchy including the "Root" entity
   numLevels: Number of levels in the hierarchy including the "Root" entity
2. Data about each entity including:
   number: Entity's number, starting from 0 for the "Root" in a breadth-first order
   name: Entity's name, used as the XML tag name
   level: Entity's level, starting from 0 for the "Root" entity
   parent: Entity's parent number
   firstAttribute: Column number corresponding to the entity's first attribute
   numAttributes: Number of attributes for the entity
   keyIndex: Column number corresponding to the entity's key attribute
   firstChild: Entity's first child entity
   numChildren: Number of children for the entity
   tagNames: Names of the entity's attributes, used as XML tag names

To generate the XML instance documents, method "elementBuilder" is called recursively to process all the entities of the hierarchy in a depth-first order, starting from the "Root" entity. For each entity, there is a set of rows in the query result to be processed. These rows are specified by the entity's *startRow* and *endRow*. Initially, the *startRow* of each entity is the *startRow* of its parent, where the *startRow* for "Root" is 0. The *endRow* for each entity is determined by testing the *keyIndex* value of the entity, except the *endRow* for "Root" that is assigned the last row of the query result. The entity's *endRow* is determined as follows: The last row that still has the same *keyIndex* value as the entity's *startRow* will be the entity's *endRow*; where it must be less than the *endRow* of the entity's parent. Each entity has a vector named *tempVector* that is used to keep track of the entity's *keyIndex* values. Method "getEndRow" is used to find the row of the query result that must be processed as the last row for that instance of the entity. This method tests the entity's *keyIndex* value with its *tempVector* vector to find the entity's *endRow*. If it encounters a NULL value, the method returns a flag meaning that there are no instances of the entity. If it encounters a value that already exists in the entity's *tempVector,* it returns a flag indicating that there are no more instances of the entity. When the process for one instance of the entity is complete, the entity's *startRow* will be set to one after its *endRow*, and the same process will be repeated for the next instance of the entity. The algorithm terminates when the "Root" entity is processed completely. The algorithms for “elementBuilder” and “getEndRow” methods are shown in Figures 7 and 8 respectively.

4 Conclusion and Future Work

We have illustrated the process for generating customized XML schemas from EER models. The user selects the entity types and relationships of interest, and identifies the root to form a hierarchical view. According to this hierarchical view, our tool generates the corresponding XML Schema document. We described some of the algorithms used by our tool to automate the breaking of cycles, generate a hierarchy by merging entities and migrating relationship attributes, creating XML schemas, generating SQL queries, and generating XML instance documents.

Our approach for designing XML Schema documents reduces the large amount of work that is needed to access databases and transform their data to XML format. With the power of XML, users will be able to easily customize their XML document structure for their specific application. Consequently, it also establishes the interface between various databases and applications.

Our tool will also store the XML Schema documents for use by applications to create instance documents. We are currently enhancing the tool to provide more flexibility, such as providing the meaning of selecting a root by rephrasing in English, and giving the user options as to how to break the cycles in a graph.

References

[1] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E.: Extensible Markup Language (XML) 1.0, W3C Recommendation, October 2000.

[2] Wadler, P.: Et tu, XML?, 27th VLDB Conference, Roma, Italy, 2001.

[3] Ullman, J., Widom, J.: A First Course in Database Systems, Prentice Hall (1997).

[4] Elmasri, R., Navathe, S.: Fundamentals of Database Systems, 3rd Edition, Addison-Wesley (2000).

[5] Fong, J., Pang, F., Bloor, C.: Converting Relational Database into XML Document, IEEE 12th International Workshop on , 2001.

[6] Kappel, G., Kapsammer, E., Rausch-Schott, S., Retschitzegger W.: X-Ray-Towards Integrating XML and Relational Database Systems, International Conference on Conceptual Modeling (ER), LNCS 1920, Springer-Verlag (2000).

[7] Ha, S., Kim, K.: Mapping XML Documents to the Object-Relational Form, Proceedings of The 2001 International Symposium on Industrial Electronics, June 2001, IEEE.

[8] Shanmugasundaram, J., Shekita, E., et al: Efficiently Publishing Relational Data as XML Documents, Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egypt, Sept. 2000.

[9] Klettke, M., Meyer., H.: XML and Object-Relational Database Systems– Enhancing Structural Mappings Based on Statistics, International Workshop on Web and Databases (WebDB), Dallas, TX, May 2000.

[10] Vermeer, M., Apers, P.: Reverse Engineering of Relational Database Applications, Proceedings 14th International Conference on OO/ER Modeling (ER '95), LNCS 1021, Springer-Verlag (1995).

[11] Object Management Group: http://www.omg.org/technology/documents/formal/uml.htm.

[12] Bourret, R.: Mapping W3C Schemas to Object Schemas to Relational Schemas http://www.rpbourret.com/xml/SchemaMap.htm.

[13] Florescu, D., Kossmann, D.: Storing and Querying XML Data Using an RDBMS, IEEE Data Eng. Bulletin 22(3), Sep.1999.

[14] XML Schema Part 1 & 2 W3C Recommendation, May 2001: http://www.w3.org/TR/xmlschema-1/ , http://www.w3.org/TR/xmlschema-2/.

[15] Schmidt, A., Kersten, M.L., Windhouwer, M., Waas, F.: Efficient Relational Storage and Retrieval of XML Documents, International Workshop on the Web and Databases (WebDB), Dallas, TX, May 2000.

[16] Elmasri, R., Larson, J.: A Graphical Query Facility for ER Databases, Proceedings of the 4th International Conference Entity-Relationship Approach, Chicago, Illinois, October, 1985, IEEE.

Figures

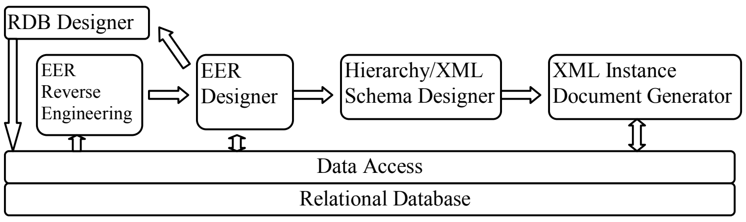


Fig. 1. Architecture Overview

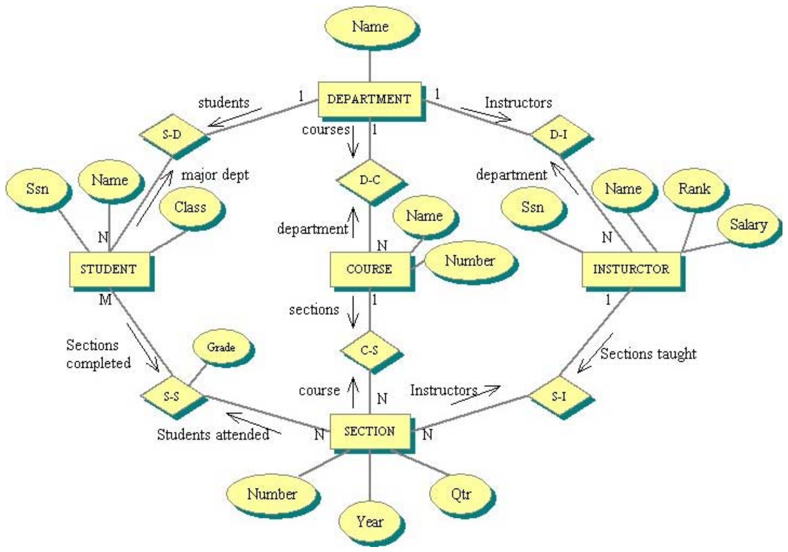


Fig. 2. EER Schema for a University Database

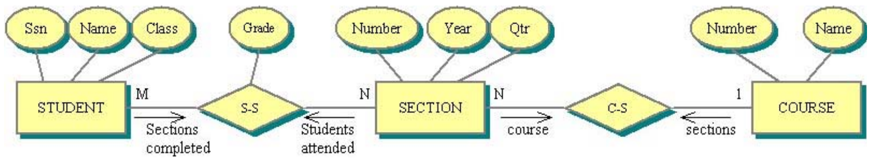


Fig. 3. The EER Diagram after User Selects Relevant Entities/Relationship Attributes

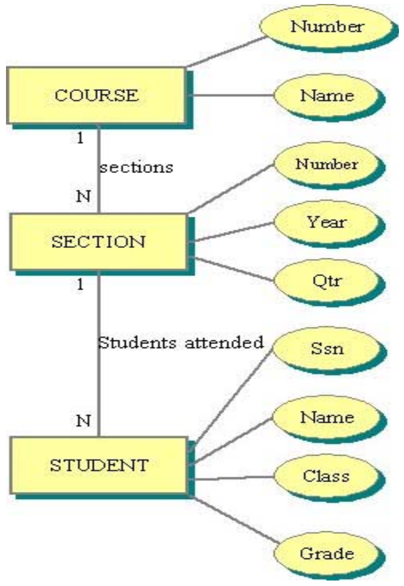


Fig. 4a. Hierarchical View with COURSE as the Root

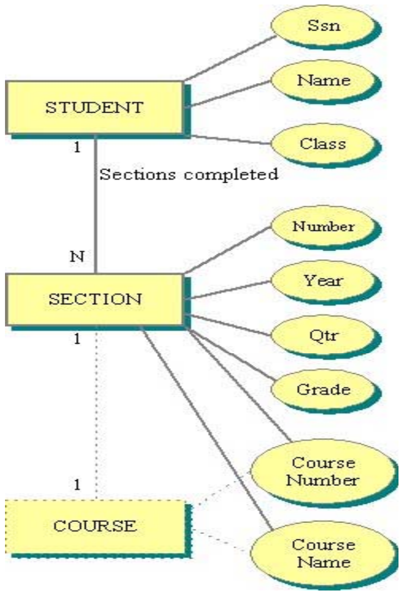


Fig. 4b. Hierarchical View with STUDENT as the Root

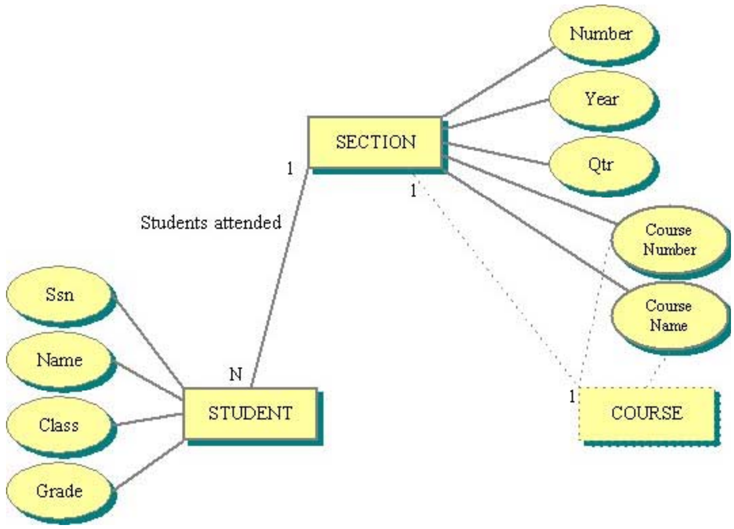


Fig. 4c. Hierarchical View with SECTION as the Root

```
<?xml version="1.0" encoding="utf-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <xsd:element name="StudentTranscriptDoc" type="rootType"/>
        <xsd:complexType name="rootType">
                <xsd:sequence>
<xsd:element        name="STUDENT"        type="STUDENTType"        minOccurs="0"
 maxOccurs="unbounded"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="STUDENTType">
                <xsd:sequence>
                        <xsd:element name="S_SSN">
                                <xsd:complexType>
                                        <xsd:simpleContent>
                                                <xsd:extension base="xsd:string">
                <xsd:attribute name="FIELD_NAME" type="xsd:string" fixed="S_SSN"/>
                <xsd:attribute name="FIELD_TYPE" type="xsd:string" fixed="CHAR"/>
                <xsd:attribute name="FIELD_LEN" type="xsd:string" fixed="11"/>
                <xsd:attribute name="NULLABLE" type="xsd:string" fixed="false"/>
                                                </xsd:extension>
                                        </xsd:simpleContent>
                                </xsd:complexType>
                        </xsd:element>
                        +<xsd:element name="S_NAME">
                        +<xsd:element name="S_CLASS">
        <xsd:element      name="SECTION"      type="SECTIONType"      minOccurs="0"
        maxOccurs="unbounded"/>
                </xsd:sequence>
        </xsd:complexType>
        +<xsd:complexType name="SECTIONType">
</xsd:schema>
```

Fig. 5. XML Schema corresponding Hierarchical View that STUDENT as the Root

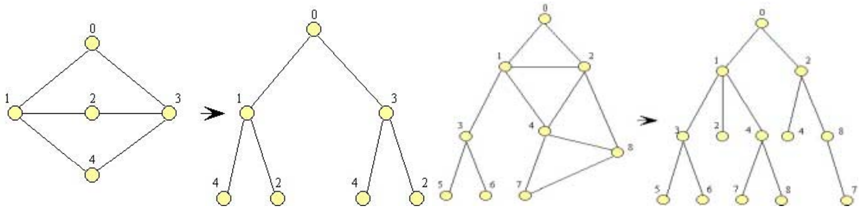


Fig. 6. Example of Algorithms for Eliminating Cycles

```
while (e's startRow <= e's parent's endRow)
 Print the opening tag for e
 Print all the simple elements of e
   if (e has at least one child)
    for (all the e's children as i)
       Empty i's tempVector
       i's startRow = i's parent's startRow
       tempEndRow = getEndRow(i)
       if (tempEndRow == -1 meaning NULL value)
          continue to the next loop iteration
       else
          i's endRow = tempEndRow
          elementBuilder(i)
 Print the closing tag for e
 if (e == 0)
    Exit, since "Root" is processed completely
 else /* search the next rows for new instances of e */
    e's startRow = e's endRow + 1
    tempEndRow = getEndRow(e)
    if (tempEndRow == -1) continue
    else if (tempEndRow == -2) return
    else e's endRow = tempEndRow
```

Fig.7. EementBuilder Algorithm

```
nextRow = e's startRow
if (e's startRow <= numRows - 1)
   if (the e's keyIndex value is NULL) return -1
 else if (the e's keyIndex is in its tempVector) return -2
   Add the e's keyIndex value to its tempVector
 nextRow = nextRow + 1
 while (nextRow <= endRow of the e's parent)
       if (the e's keyIndex value in nextRow is NULL) return -1
    else if (e's keyIndex value in nextRow is different
          from the e's keyIndex value in the e's startRow)
      break out of the loop
    else nextRow = nextRow + 1
   return nextRow - 1
else return numRows - 1
```

Fig.8. getEndRow Algorithm

A Flexible Cost Model for Abstract Object-Oriented Database Schemas

Joachim Biskup and Ralf Menzel

Fachbereich Informatik, Universität Dortmund
44221 Dortmund, Germany
{biskup,menzel}@ls6.cs.uni-dortmund.de

Abstract. Typically, the design of an object-oriented database schema starts with an analysis of the application and ends with the implementation of the application. We advocate a design process that employs an intermediate phase where the designer can choose between different abstract object-oriented database schemas. This choice influences the space and time costs that arise when the schema is implemented. We present a cost model for abstract object-oriented database schemas that allows the designer to estimate these costs. At the core of the cost model is an abstract object-oriented database machine. Access structures that are used by this abstract database machine are given by an internal schema. With this we can estimate the space costs. Queries and updates are expressed as programs of the abstract database machine. By providing cost functions that characterise cost relevant aspects of the operations of the abstract database machine we can estimate the time costs of the machine programs. Our cost model is parameterised. So, for example, it can be adopted to reflect different implementation database systems. We provide an example to show how the preferred choice of an abstract database schema changes when the parameters of the cost model vary.

1 Introduction

For an object-oriented database application a major goal of the development process is the design of a database schema. Such a nontrivial piece of software as a database application is preferably developed through multiple steps. The analysis of the application is usually the first step of a design process, while one of the last steps is the implementation. In the past, the full design process has been treated under a large variety of points of views [1, 2, 3, 4, 5, 6]. In the following, we refer to our specific version of a basic design process for object-oriented schemas [7]. Nevertheless our results appear to be easily adaptable for many other versions.

In the basic design process the *analysis* step, semantic modelling, results in an analysis schema, which can be represented by an entity-relationship schema or a UML class diagram. In the second step, *abstract logical formalisation*, the initial analysis schema is transformed into an abstract object-oriented schema, which subsequently can be transformed in order to improve it. The third step,

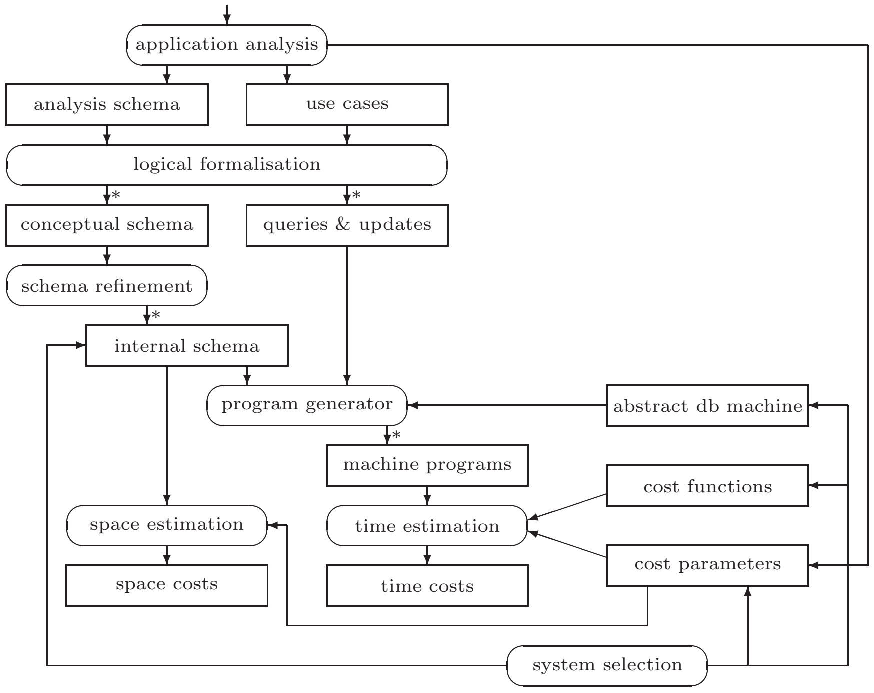


Fig. 1. Outline of cost driven database schema design

concrete class declarations, finally implements the schema using the underlying object-oriented database system. In a first attempt the steps are performed consecutively. In general, however, some iterations are required and thus a later step can indirectly influence an earlier step.

The transformations during the step of abstract logical formalisation provide the designer with a choice of different abstract object-oriented database schemas. It is possible to guide this choice by heuristics. As an alternative, in this paper, we propose a cost model that allows to estimate the space and time costs that arise when the chosen schema is implemented. This is a substantial refinement of the second step towards a cost driven design. We aim at introducing estimations of space and time costs already on the logical level, in order to make the impact of system-dependent factors as explicit as possible.

1.1 Cost Driven Design

In a recent work [8] we presented an abstract object-oriented database machine. This abstract database machine is the core of the cost model that we present in this paper. Figure 1 shows the overall structure of our refinement of the basic design process. It can be roughly outlined as follows.

One result of the *application analysis* is an analysis schema, which may be given as an entity-relationship schema or a UML class diagram. Additionally the application analysis provides use cases that describe expected user behaviour. *Logical formalisation* takes an analysis schema and translates it into what we call a conceptual schema. A conceptual schema is one part of an abstract object-oriented schema. It describes types, classes and their hierarchy, and semantic constraints. In accordance with the schema formalisation the use cases are formalised into queries and updates that fit the conceptual schema. The next step, *schema refinement*, complements the conceptual schema with an internal schema, the other part of the abstract object-oriented schema. The internal schema contains a set of access structure declarations. Given an abstract schema, that is, a conceptual schema and a corresponding internal schema, we can immediately *estimate the space costs*. For the *estimation of the time costs* we employ the abstract database machine. We *generate a machine program* for each of the queries and updates that represent the expected user behaviour. Then the time costs can be estimated by a step-by-step analysis of the machine programs.

In the figure some arrows are marked with an asterisk. At these points there is a choice between different alternatives. For example, during schema refinement different internal schemas can be selected for a given conceptual schema. Each time the costs are estimated all these choices are checked to see if any necessary improvements are possible. When all costs are low enough or otherwise optimised implementation can commence.

The cost estimations are steered by several factors. The generation of the machine programs depends on the operations that the *abstract database machine* provides. For each of the operations a series of *cost functions* outlines its time requirements. *Cost parameters* control the cost functions. These cost parameters provide control values for the space cost and time cost estimations. A part of the cost parameters describes properties of the application. These must be provided by the application analysis.

The costs of the implemented database application depend on the *selected concrete database system*. There are several points where this system selection is taken into account for the cost estimations. The choice of internal schemas can be influenced by restricting the possible types of access structures. The query (and update) processor of the database system can be characterised by choosing the operations of the abstract database machine and providing appropriate cost functions. Various cost parameters describe system properties.

In conclusion, the inputs and outputs and the overall control structure (not shown in Fig. 1) of the cost driven design can be summarised as follows. The primary input from the application analysis gives raise to alternatives for a conceptual schema, for queries and updates, for an internal schema, and for machine programs. Cost estimations must explore and compare all meaningful combinations, or at least the promising ones. The final costs, and therefore the cost estimations, depend on the system selection. Accordingly, system properties are used as a secondary input. The process generates as outputs a survey on the space and time costs of the considered combinations under the system properties, preferably together with a recommendation of a good or even a best choice.

1.2 Related Work

There are several investigations considering costs while designing databases [9, 10, 11, 12, 13]. Software performance engineering deals with the consideration of performance during software development. Nixon [14, 15] describes a Performance Requirements Framework to apply these principles to the design of information systems. When, during logical formalisation, use cases are translated into queries and updates it is important that semantic constraints are taken into account. We do not discuss this in the present paper. Others handle this task [16, 17, 18, 19]. Especially noteworthy is the work of Gupta and Widom [20] who show an approach for local verification of global integrity constraints. Bertino and Foscoli [21] specifically investigate object-oriented systems and present a model to characterise topologies of object references in object-oriented database. They present a set of parameters that are similar to the cost parameters that we use to steer our cost functions. Our abstract database machine can be seen as a special kind of object algebra [22] or as an abstract query execution language [23]. In order to find suitable cost functions it is necessary to have an idea of the way object-oriented database systems are implemented. Shekita as well as Velez and others [24, 25] describe such implementation details of object-oriented database systems.

1.3 Outline of the Paper

In Sect. 2 we sketch our abstract database machine. There we select a representative collection of database operations that basically work on streams of value lists. Section 3 contains our main contribution, the detailed description of the cost model. Some remarks conclude the paper. Section 4 contains concluding remarks. In the appendix we provide an example to illustrate the use of the cost model as part of the advocated refinement of the design process.

2 Abstract Database Machine

Here we only give a short summary of the abstract database machine. A detailed description can be found in our recent work [8]. Since we want to deal with costs of operations we consider both persistent and transient data.

The persistent database state of the abstract database machine is a *database instance*. Such an instance is based on a *conceptual schema* which describes classes, $\mathcal{C}$, a class hierarchy, $\trianglelefteq \subseteq \mathcal{C} \times \mathcal{C}$, types, $\mathcal{T}$, a type hierarchy, $\leq \subseteq \mathcal{T} \times \mathcal{T}$, the types of classes and attributes, and semantic constraints. For a every class c its type is given as $\gamma(c)$. There is a distinction between value types, $\mathcal{T}_{\mathrm{Val}}$, and object types, $\mathcal{T}_{\mathrm{Obj}}$. A database instance gives object identifiers, attribute values, class extensions and object type extensions. An *internal schema* complements a conceptual schema. It contains a set of access structure declarations, X, of the form (c, A) which declare that there should be an access structure (index) for the attribute A of class c in the implementation.

Table 1. Operations of the abstract database machine

name	number of input streams	number of output streams	access to persistent database state	may generate duplicates	sorted input required	needs entire input stream for operation
const	0	1	—	yes	n/a	n/a
duplicate	1	2	—	—	—	—
scan	0	1	read	—	n/a	n/a
activate	1	1	read	—	—	—
unnest	1	1	—	yes	—	—
nest_0	1	1	—	—	yes	—
project	1	1	—	yes	—	—
unique_0	1	1	—	—	yes	—
select	1	1	—	—	—	—
sort	1	1	—	yes	—	yes
product	2	1	—	—	—	yes
unionall	2	1	—	yes	—	—
create	1	1	—	—	—	—
write	1	0	write	n/a	—	—
delete	1	0	write	n/a	—	—
union_0	2	1	—	—	yes	—
join_0	2	1	—	—	yes	—
access	1	1	read	—	—	—

(To emphasise positive entries, we use '—' to denote 'no'.)

The transient data in the abstract database machine is represented through *streams*. The operations of the abstract database machine pass the data from one to another in the form of streams. Every stream has a type $(t_1, \ldots, t_n)$ which is a list of value types. A stream is a sequence of *value lists*. The values in the value lists are numbered by their *positions* from 1 to n. In a stream of type $(t_1, \ldots, t_n)$ all the values at a position k must be of type t_k. Additionally a stream can be lexicographically sorted, which is described by a position list $(k_1, \ldots, k_n)$.

Table 1 gives an overview of the operations. We provide a short description of their meaning at the respective cost functions below.

A *machine program* consists of one or more *steps*. Each step designates one operation and describes the input and output of the operation. For this purpose, streams are represented by program variables of the form x_k, called *channels*. Every input stream for an operation is given as a channel that stands for the output stream of an earlier step.

A machine program step is denoted as either $op(args)$ or $x_k := op(args)$ or $(x_k, x_l) := op(args)$ depending on the number of output streams of the operation. Here x_k denotes a channel, op is an operator and $args$ is a comma separated list of suitable arguments. An arguments that stands for an input stream is given as a channel x_l.

A machine program is a sequence of machine program steps that satisfies the following requirements:

- The preconditions of the operators in all steps are satisfied. All arguments are of valid types, in particular.
- Each channel is used as output by exactly one step. (The two outputs of 'duplicate' must be two different channels.)
- Each channel is used as input by at most one step.
- Each channel that is used as input by a machine program step is used as output by an earlier step.

Note, that every program can be represented by an ordered directed acyclic graph, where the nodes stand for program steps and the vertices for channels.

3 Cost Evaluation

As stated above the cost model can be adjusted by providing cost functions and cost parameters. In this section we give a set of *basic* cost functions and corresponding parameters. For special purposes it can be necessary to use more refined cost functions.

Throughout this section we present different assumptions that we make for the underlying database system. They appear when they are first needed. Furthermore we use several cost parameters that are used for the cost estimations. We introduce the cost parameters when we first need them and provide a summary in Fig. 2.

3.1 Space Cost Estimation

We first look at the space on secondary memory that we need for the persistent application data and the access structures. In the following we make the common assumption that the secondary memory consists of equally sized blocks. Based on this we measure space costs as the number of blocks needed. To capture the application data we must store the attribute values of all objects and at least the direct class extensions of all classes.

We base our estimations on the assumption that the database system uses what we call exclusive and mandatory class membership, that is, that every object is in the direct extension of exactly one class. Because of this assumption the objects on secondary memory can be organised by direct class extensions. The attribute values of each object are stored contiguously. Thus we can characterise the space needed for storing attribute values by some cost parameters. The cost parameter n_t describes the average number of blocks that are needed to store an object of type t including its attribute values. The parameter $N_{0,c}$ is an estimate for the number of objects in the direct extension of class c. The estimate of the space for the application data is:

$$\sum_{c \in \mathcal{C}} n_{\gamma(c)} \cdot N_{0,c} \qquad (1)$$

We assume that there is an implicit object identifier access structure for every class that can be used to retrieve all object identifiers of the class. The cost parameter N_c is an estimate for the number of objects in the extension of class c. Even though we can fit n_{OID} OIDs in one block, we assume that not every block of the object identifier access structure is completely filled with OIDs. The cost parameter $\theta_{\mathrm{s}:c}$ describes the average filling level of the object identifier access structure for class c. The parameter n_{a} captures the organisational overhead, for example the non-leaf nodes in a B-tree, that is needed for each object identifier. Typically this is a fraction of a block. The estimate of the space for the implicit object identifier access structures is:

$$\sum_{c \in \mathcal{C}} N_c \cdot (1/n_{\mathrm{OID}}/\theta_{\mathrm{s}:c} + n_{\mathrm{a}}) \tag{2}$$

We need space for every explicitly declared access structure, (c, A). The estimate of the space for the explicit access structures is similar to the estimate for the implicit access structures. Because there may be more (or less) than one object associated with a value for an attribute, we must use an additional cost parameter, $n_{\mathrm{A}:c,A}$, that describes the average number of elements for an attribute A of class c:

$$\sum_{(c,A) \in X} N_c \cdot n_{\mathrm{A}:c,A} \cdot (1/n_{\mathrm{OID}}/\theta_{\mathrm{s}:c} + n_{\mathrm{a}}) \tag{3}$$

Thus the total required space can be estimated as the sum of the formulas (1), (2), and (3).

3.2 Time Cost Estimation

The time costs are estimated by looking at the steps of a machine program one after the other. In this process all program steps and all channels in the program are adorned with *cost statements*.

When we have a program step it contains an operation. For the purpose of cost evaluation we must provide *cost functions* for every operation. These cost functions are applied to the parameters of the current step, to the types, sortings and the cost statements of the input channels. The results of this application are used as the cost statement of the step and as the cost statement for the output channel, respectively.

After every step is adorned with a cost statement, the total cost of the machine program results by combining the cost statements of all program steps with an *aggregation function*. We call the result of the aggregation *total cost statement*.

3.3 Cost Statements

Cost statements are attached to channels and to machine program steps. We call the one sort *channel cost statements* and the other *step cost statements*.

For the basic costs functions we use the following cost statements. The channel cost statements comprise:

– *stream length*: a description of the number of value lists in the respective stream

The step cost statements comprise:

– *block accesses*: a description of the number of blocks that are transferred to or from secondary memory during the respective machine program step.
– *value accesses*: a description of the number of values in main memory that are read or written during the respective machine program step.

Accordingly the block accesses represent the I/O costs and the value accesses represent the CPU costs.

While blocks are all of the same size, values can require rather different amounts of space. Especially the size of sets can vary widely. But for our basic cost functions we assume that a normal set of our application is small and there is no otherwise very significant difference in the size of values.

3.4 Cost Functions

By the choice of the kind of cost statements we have laid the grounds for the cost functions. The time it takes to carry out a machine operation depends on the values in the input streams. For our estimations, at the schema design time, we neither know the values nor the length of the input streams. What we exactly know is only the type of the input streams. In order to estimate the needed unknown parts, it is therefore necessary to make assumptions about the likely characteristics of 'typical' values, their occurrences in 'typical' streams and the length of such streams.

In principle, we have to distinguish values and streams stemming from persistent data, and those stemming from transient data, that is, from intermediary results that occur during the evaluation of machine programs. Our assumptions, precisely presented below, when considering the cost functions, are guided by the following rough rules:

– For persistent data, application analysis allows reasonable assumptions on the characteristics of values with respect to size, selectivity of predicates etc, and on the cardinality of class extensions which are used within first-level input streams.
– Based on that, we can reasonably estimate the length of first-level output streams.
– Additionally, we assume (or, more rigorously, we only allow) that queries and updates (which represent expected user behaviour) are 'simple'. Accordingly, and based on calculated output types we treat transient values in higher-level streams like values in first-level streams. Thus on higher levels, we use the same correlations between the length of input streams and the length of output streams as we assumed on the first level.

Now we look at the cost functions for each operation in turn. We first sum up the syntax. Then we give the cost functions for the step and any output channel. We use the notation $\|x\|$ to refer to the stream length of an input channel x. Last there may be some comments about the underlying implementation ideas.

const: $\text{const}(s_\text{c}, (t_1, \dots, t_n))$ [returns stream s_c of type $(t_1, \dots, t_n)$.]
Block accesses: 0, Value accesses: 0, Length of output stream: $\|s_\text{c}\|$

We assume that the constant stream s_c is already in main memory. Therefore there are no block or value accesses.

scan: $\text{scan}(c)$ [returns a stream of the identifiers of the objects of class c.]
Block accesses: $\lceil N_c / n_\text{OID} / \theta_{\text{s}:c} \rceil$, Value accesses: 0, Length of output stream: N_c

The OIDs are retrieved using the above mentioned implicit object identifier access structure of class c.

activate: $\text{activate}(x, k)$ [loads the objects for the identifiers at position k.]
Block accesses: $\|x\| \cdot (\theta_{\text{a}:c} + n_{t_k})$ where t_k is the type of the values at position k, Value accesses: $\|x\|$, Length of output stream: $\|x\|$

We assume that we've got physical OIDs, that allow us to directly access the blocks where the objects were stored when they were created. When the size of an object grows it might be necessary to move the object to a different location. When this happens a reference is stored at its original home. Consequently an additional block access is required to retrieve the object. Because of this we provide a correction factor $\theta_{\text{a}:c}$, the fraction of objects of class c that have been relocated.

unnest: $\text{unnest}(x, k)$ [unnests the sets at position k.]
Block accesses: 0, Value accesses: $\|x\| \cdot n_{\text{u}:t_k}$, Length of output stream: $\|x\| \cdot n_{\text{u}:t_k}$

The cost parameter $n_{\text{u}:t}$ gives the average number of elements in a set of type t.

nest$_0$: $\text{nest}_0(x, k)$ [nests the values at position k.]
Block accesses: 0, Value accesses: $(\|x\| - 1) \cdot n_{\text{q}:(t_1, \dots, t_{k-1}, t_{k+1}, \dots, t_n)}$, Length of output stream: $\|x\| / n_{\text{u}:t_k\texttt{\_Set}}$ where $t_k\texttt{\_Set}$ is the set type that corresponds with t_k.

Estimating the value accesses is a bit tricky. We must test whether two consecutive value lists are identical accept for the given position k. When they are identical we need $2(n-1)$ value accesses for such a test. When they are not we need a number of value accesses between 2 and $2(n-1)$ until we find out. The cost parameter $n_{\text{q}:(t_1, \dots, t_n)}$ gives the average number of value accesses necessary to decide whether two value lists of type $(t_1, \dots, t_n)$ are identical or not.

project: $\text{project}(x, f)$ [applies the projection function f.]
Block accesses: 0, Value accesses: 0, Length of output stream: $\|x\|$

The project operation is as much an operation as a type conversion. All it has to do is to pass on the new positions to subsequent operations. This has the effect of leaving no longer needed values in main memory. The only operations that might be effected by this are those that need the complete stream as input before they can perform. But these operations, namely sort and product, must access all positions anyway. Thus they implicitly take care of removing the no longer needed values from main memory.

unique$_0$*:* $\text{unique}_0(x)$ [removes duplicates.]
Block accesses: 0, Value accesses: $(\|x\| - 1) \cdot n_{\text{q}:(t_1,\ldots,t_n)}$, Length of output stream: $f_{\text{q}:(t_1,\ldots,t_n)}(\|x\|)$ where $(t_1, \ldots, t_n)$ is the type of the input stream

The cost parameter $f_{\text{q}:(t_1,\ldots,t_n)}(l)$ gives the average number of unique elements in a stream of type $(t_1, \ldots, t_n)$ of length l. Testing the identity of two consecutive value lists is similar to the test done for nest_0.

select: $\text{select}(x, p)$ [returns only value lists that satisfy the predicate p.]
Block accesses: 0, Value accesses: $\|x\| \cdot n_p$, Length of output stream: $\lceil \|x\| \cdot \theta_p \rceil$

The cost parameter θ_p gives the selectivity of predicate p. The cost parameter n_p gives the number of values that must be accessed to evaluate the predicate p.

sort: $\text{sort}(x, (k_1, \ldots, k_l))$ [lexicographically sorts for the positions $(k_1, \ldots, k_l)$.]
Block accesses: $2 \cdot n_{(t_1,\ldots,t_n)} \cdot \|x\| \cdot \lceil \log_q(k) \rceil$ where $k = \lceil \|x\| \cdot n_{(t_1,\ldots,t_n)}/n_{\text{mem}} \rceil$, $q = \min(\lceil n_{\text{mem}}/n_{(t_1,\ldots,t_n)} \rceil, n_{\text{files}})$ and $(t_1, \ldots, t_n)$ is the type of the input stream, Value accesses: $\|x\| \cdot \log_2(\|x\|/k) \cdot n$, Length of output stream: $\|x\|$

The cost statements are based on a merge sort using q files. The value q depends on $n_{(t_1,\ldots,t_n)}$, the average number of blocks for a value list of type $(t_1, \ldots, t_n)$, and on n_{mem}, the size of memory (in blocks) that is reserved for sorting and similar operations. There is a limit, n_{files}, to the number of files that may be simultaneously used for sorting. The value accesses are caused by the in-memory presorting of the k substreams.

product: $\text{product}(x_1, x_2)$ [returns the Cartesian product of the input streams.]
Block accesses: if $\lceil n_{(t'_1,\ldots,t'_m)} \cdot \|x_2\| \rceil \leq n_{\text{mem}}$ then 0 else $\|x_1\| \cdot \lceil n_{(t'_1,\ldots,t'_m)} \cdot \|x_2\| \rceil$ where $(t'_1, \ldots, t'_m)$ is the type of the second input stream, Value accesses: $\|x_1\| \cdot \|x_2\| \cdot (n + m)$ where the type of the first input stream is $(t_1, \ldots, t_n)$, Length of output stream: $\|x_1\| \cdot \|x_2\|$

The result of product sustains the sorting of the input stream x_1. If the second stream fits entirely into main memory the product can be computed without using any secondary memory. If this is not the case we store the entire second stream on secondary memory and read it for each value list in the first stream (except for the first, where we write it).

unionall: $\text{unionall}(x_1, x_2, (t_1, \ldots, t_n))$ [concatenates two streams.]
Block accesses: 0, Value accesses: 0, Length of output stream: $\|x_1\| + \|x_2\|$

The output stream is of type $(t_1, \ldots, t_n)$.

create: create(x, t) [appends an identifier of type t to each value list.]
Block accesses: 0, Value accesses: $\|x\|$, Length of output stream: $\|x\|$

create must only generate one new object identifier for each value list in the stream.

write: write(x, c, l, f) [inserts objects identified by position k into class c.]
Block accesses: $\|x\| \cdot \left(n_{\gamma(c)} + \eta(N_{0,c}) + \sum_{(c',A)\in X_c} \eta(N_{c'}) \cdot n_{\mathrm{A}:c',A}\right)$, Value accesses: 0, Length of output stream: none

write puts objects into a class, c. The identifiers of the objects to write are at position l of the stream x. The function f tells the position of the value for each attribute of an object.

Along with the object data we must update the implicit object identifier access structure and all relevant explicit access structures $X_c = \{(c', A) \mid (c', A) \in X \wedge c \trianglelefteq c'\}$ on the secondary storage. The cost parameter $\eta(n)$ gives the number of block accesses required to locate an element in an access structure for a set with n elements. For set valued attributes we may need to update more than one access structure entry. The cost parameter $n_{\mathrm{A}:c',A}$ gives the average number of values for an attribute A of class c'.

delete: delete(x, c, l) [deletes objects identified by position l from class c.]
Block accesses: $\|x\| \cdot (\theta_{\mathrm{d}:c} \cdot n_{\gamma(c)} + \theta_{\mathrm{D}:c} \cdot (\eta(N_{0,c}) + \sum_{(c',A)\in X_c} \eta(N_{c'}) \cdot n_{\mathrm{A}:c',A}))$,
Value accesses: 0, Length of output stream: none

When an object is deleted from class c and its subclasses it might still be a member of some other class. Then it should not be removed from secondary memory. The cost parameter $\theta_{\mathrm{d}:c}$ describes the fraction of objects in class c that are only members of c or any subclasses of c.

Like the write operation the delete operation must update the implicit and explicit access structures. But while the write operation updates only the direct extension of a class the delete operation updates the complete extension. The cost parameter $\theta_{\mathrm{D}:c}$ is a correction factor that can be used to pay regard to this fact and to attune the deletion costs.

union$_0$: union$_0(x_1, x_2, (t_1, \ldots, t_n))$ [merges two sorted streams.]
Block accesses: 0, Value accesses: $(\|x_1\| + \|x_2\| - 1) \cdot n_{\mathrm{q}:(t_1,\ldots,t_n)}$, Length of output stream: $(\|x_1\| + \|x_2\|) \cdot f_{\mathrm{q}:(t_1,\ldots,t_n)}(\|x_1\| + \|x_2\|)$

This computes an ordered union (of type $(t_1, \ldots, t_n)$) and removes duplicates, similar to unique$_0$.

join$_0$: join$_0(x_1, x_2, k, l)$ [joins two sorted streams on position k of x_1 and l of x_2.]
Block accesses: if $\|x_1\| \cdot n_{(t_1,\ldots,t_n)} < n_{\mathrm{mem}}$ then $2 \cdot \|x_2\| \cdot n_{(t'_1,\ldots,t'_m)}$
else $2 \cdot \|x_1\| \cdot n_{(t_1,\ldots,t_n)} + 2 \cdot \|x_2\| \cdot n_{(t'_1,\ldots,t'_m)}$ where $(t_1, \ldots, t_n)$ is the type of x_1 and $(t'_1, \ldots, t'_m)$ is the type of x_2, Value accesses: $\|x_1\| \cdot \|x_2\| \cdot n_{p_{k,n+l}} \cdot (n_{(t_1,\ldots,t_n,t'_1,\ldots,t'_m)})$, Length of output stream: $\|x_1\| \cdot \|x_2\| \cdot n_{p_{k,n+l}}$

This is a merge join of the two sorted input streams. The block accesses are caused by writing the streams to file and reading them. We can spare doing this

Application dependent cost parameters:

N_c:	number of objects in the extension of class c.
$N_{0,c}$:	number of objects in the direct extension of class c.
N_c:	number of objects in the extension of class c.
$N_{0,c}$:	number of objects in the direct extension of class c.
n:	'width' of a stream or type.
n_t:	average number of blocks for an object of type t.
$n_{(t_1,\ldots,t_n)}$:	average number of blocks for a value list of type $(t_1, \ldots, t_n)$.
$n_{\mathrm{u}:t}$:	average number of elements in a set of type t.
$n_{\mathrm{A}:c,A}$:	average number of values for attribute A of class c.
$n_{\mathrm{q}:(t_1,\ldots,t_n)}$:	average number of value accesses necessary to decide whether two value lists of type $(t_1, \ldots, t_n)$ are identical or not.
n_p:	the number of values that must be accessed to evaluate the predicate p.
$\theta_{\mathrm{s}:c}$:	average filling level of the object identifier access structure for class c.
$\theta_{\mathrm{a}:c}$:	portion of objects of class c that have been moved from their original second memory location.
θ_p:	selectivity of predicate p.
$\theta_{\mathrm{d}:c}$:	fraction of objects in class c that are only members of c or any subclasses of c.
$\theta_{\mathrm{D}:c}$:	correction factor for deletion of objects of class c. See the description of the cost functions for delete.
$f_{\mathrm{q}:(t_1,\ldots,t_n)}(l)$:	average number of unique elements in a stream of type $(t_1, \ldots, t_n)$ of length l.

System dependent cost parameters:

n_{a}:	fraction of a block that is needed for the non-data part of an access structure per element of the stored set.
n_{OID}:	number of object identifiers that fit into one block.
n_{mem}:	size of memory that is reserved for sorting and similar operations.
n_{files}:	maximum number of open files for sorting.
$\eta(n)$:	number of block accesses required to locate an element using an access structure for a set with n elements.

Fig. 2. Summary of Cost Paramaters

for the first stream, if it fits into memory. The predicate $p_{k,n+l}$ stands for the comparison of the values at the positions k of stream x_1 and l of stream x_2. Note, that the values that are at position l in stream x_2 appear at position $n+l$ in the result stream.

access: access$(x, (c, A), k)$
[retrieves the objects of c for the attribute values at position k.]
Block accesses: $\|x\| \cdot n_{\mathrm{A}:c,A} \cdot (\eta(N_c) + \theta_{\mathrm{a}:c} + n_{\gamma(c)})$, Value accesses: $\|x\|$, Length of output stream: $\|x\| \cdot n_{\mathrm{A}:c,A}$

access uses the access structure (c, A) to retrieve objects. For each value at position k in stream x it retrieves and appends to the value list all objects of class c that have this value for attribute A.

3.5 Aggregation Function

For our basic cost functions aggregation is readily done by adding the block accesses of all steps and by adding the value accesses of all step. As total cost statement we get a pair, containing the block accesses and the value accesses required to perform the query or update that is expressed by the machine program.

4 Conclusion

Traditionally, logical design of database schemas is primarily affected by the application analysis. Cost considerations with respect to system selections are only captured by qualitative heuristics, or by iterating the whole design process. In this paper we demonstrate how to enrich logical design with explicit cost estimations. Such cost estimations are shown to be effectively manageable, even for a reasonably expressive object model. As a prerequisite, we use an abstract database machine for the object model under consideration.

Our results are elaborated for specific versions of the design process, the object model and the underlying abstract database machine. However, these versions are considered to be representative for the many respective variants reported in the literature. Hence, we expect that our specific justification for the feasibility and usefulness of a cost driven design of object-oriented database schemas applies for the variants, too.

There are various ways for future research and development. First, it would be worthwhile to build a tool for cost estimations and to implant it in a general tool for schema design. Second, the identified cost parameters, as summarised in Fig. 2, as well as actual cost estimations should be empirically validated and adjusted with respect to specific applications running on a specific real system. Third, as a long term vision, previous insight on qualitatively ‘good’ (normalised) schemas should be refined and enhanced by quantitative aspects, as introduced in this paper.

References

[1] Batini, C., Ceri, S., Navathe, S. B.: Conceptual Database Design. Benjamin/ Cummings, Redwood City, CA (1992) 444
[2] Elmasri, R., Navathe, S. B.: Fundamentals of Database Systems. Third edn. Addison-Wesley, Reading, MA (2000) 444
[3] Embley, D. W.: Object Database Development: Concepts and Principles. Addison-Wesley, Reading, MA (1998) 444
[4] Mannila, H., Räihä, K. J.: The Design of Relational Databases. Addison-Wesley, Wokingham (1992) 444
[5] Teorey, T. J.: Database Modeling and Design. Second edn. Morgan Kaufmann, San Francisco, CA (1994) 444
[6] Thalheim, B.: Entity-Relationship Modeling: Foundations of Database Technology. Springer, Berlin (2000) 444

[7] Biskup, J., Menzel, R., Polle, T.: Transforming an entity-relationship schema into object-oriented database schemas. In Eder, J., Kalinichenko, L. A., eds.: Advances in Databases and Information Systems, Moscow 95. Workshops in Computing, Springer (1996) 109–136 444
[8] Biskup, J., Menzel, R.: An abstract database machine for cost driven design of object-oriented database schemas. In: Advances in Databases and Information Systems, Fifth East European Conf., ADBIS 2001. 2151 in LNCS, Vilnius, Lithuania, Springer, Berlin (2001) 366–380 445, 447, 458
[9] Steeg, M.: The conceptual database design optimizer CoDO – Concepts, implementation, application. In Thalheim, B., ed.: Proc. of the 15th Intl. Conf. on Conceptual Modeling. 1157 in LNCS, Cottbus, Germany, Springer (1996) 105–120 447
[10] Steeg, M., Thalheim, B.: A computational approach to conceptual database optimization. Technical report, BTU Cottbus (1995) 447
[11] van Bommel, P.: Experiences with EDO: An evolutionary database optimizer. Data & Knowledge Engineering **13** (1994) 243–263 447
[12] van Bommel, P., van der Weide, T. P.: Reducing the search space for conceptual schema transformation. Data & Knowledge Engineering **8** (1992) 269–292 447
[13] Yao, S. B.: Optimization of query evaluation algorithms. ACM Transactions on Database Systems **4** (1979) 133–155 447
[14] Nixon, B. A.: Representing and using performance requirements during the development of information systems. In Jarke, M., Bubenko, J., Jeffery, K., eds.: Advances in Database Technology—EDBT '94. 779 in LNCS, Springer, Berlin etc. (1994) 187–200 447
[15] Nixon, B. A.: Management of performance requirements for information systems. IEEE Transactions on Software Engineering **26** (2000) 1122–1146 447
[16] Jagadish, H. V., Qian, X.: Integrity maintenance in an object-oriented database. In Yuan, L. Y., ed.: Proc. of the 18th Intl. Conf. on Very Large Data Bases, British Columbia, Canada (1992) 469–480 447
[17] Lipeck, U.: Transformation of dynamic integrity constraints into transaction specifications. Theoretical Computer Science **76** (1990) 115–142 447
[18] McCune, W. W., Henschen, L. J.: Maintaining state constraints in relational databases: A proof theoretic basis. Journal of the ACM **36** (1989) 46–68 447
[19] Nicolas, J. M.: Logic for improving integrity checking in relational databases. Acta Informatica **18** (1982) 227–253 447
[20] Gupta, A., Widom, J.: Local verification of global integrity constraints in distributed databases. In Buneman, P., Jajodia, S., eds.: Proc. of the 1993 ACM SIGMOD Intl. Conf. on Management of Data. (1993) 49–58 447
[21] Bertino, E., Foscoli, P.: On modeling cost functions for object-oriented databases. IEEE Transaction on Knowledge and Data Engineering **9** (1997) 500–508 447
[22] Osborn, S. L.: Identity, equality and query optimization. In Dittrich, K. R., ed.: Advances in Object-Oriented Database Systems, 2nd Intl. Workshop on Object-Oriented Database Systems. 334 in LNCS, Bad Münster am Stein-Ebernburg, Germany, Springer (1988) 346–351 447
[23] Vance, B.: An abstract object-oriented query execution language. In Beeri, C., Ohori, A., Shasha, D. E., eds.: Proc. of the Fourth Intl. Workshop on Data Base Programming Languages – Object Models and Languages. Workshops in Computing, New York City, Springer (1993) 176–199 447
[24] Shekita, E. J.: High-Performance Implementation Techniques For Next-Generation Database Systems. PhD thesis, University Of Wisconsin, Madison (1990) 447

[25] Velez, F., Bernard, G., Darnis, V.: The O_2 object manager: An overview. In Apers, P. M. G., Wiederhold, G., eds.: Proc. of the 15th Intl. Conf. on Very Large Data Bases, Amsterdam (1989) 357–366 447

A Example

We reconsider the example we used in [8] for illustrating the abstract database machine. The example pictures a company with several departments. Every department has a number, a name and a location. In the departments work employees. Each employee has a name and a salary. Among the employees are some that manage others. These managers are given a budget to administer. Figure 3 shows our example as a UML class diagram.

A complete description of expected user activities is usually quite big. Here we choose to look only at a part of the complete user activities to demonstrate the usage of the cost model. We examine which time costs are entailed by the following two activities, one query and one update:

1. A user wants to find out the names of all employees managed by Jones.
2. The employee Smith is promoted to a manager with a budget of 10 000.

By logical formalisation we can get different conceptual schemas. For each of them there can be many internal schemas. For the purpose of this example we only consider two abstract schemas, that is, two conceptual schemas with one internal schema for each of them. Figure 4 shows the two alternatives pictured as UML class diagrams.

A.1 Programs

To simplify our example further we only consider one machine program for each of the queries.

The following machine program computes the query for the first abstract schema:

$x_1 := \mathit{const}(\langle(\text{‘Jones’})\rangle, (\texttt{string}))$
$x_2 := \mathit{access}(x_1, (\text{Manager}, \text{name}), 1)$
$x_3 := \mathit{unnest}(x_2, 7)$
$x_4 := \mathit{activate}(x_3, 7)$
$x_5 := \mathit{project}(x_4, f)$ with $f(1) = 8$

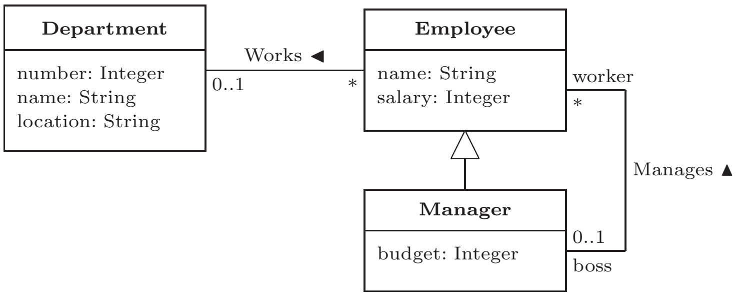


Fig. 3. A UML class diagram for an example database schema

This machine program computes the update for the first abstract schema:

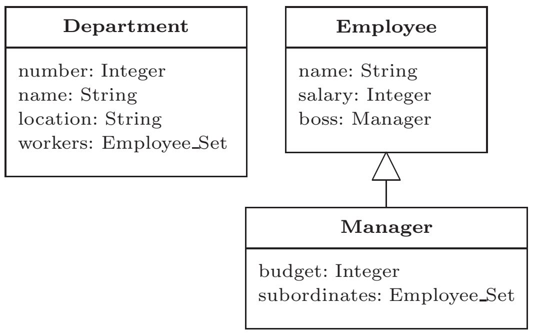
$x_1 := \textit{const}(\langle(\text{'Smith'}, 10\,000, \{\})\rangle, (\texttt{string}, \texttt{integer}, \texttt{employee\_OID\_Set}))$
$x_2 := \textit{access}(x_1, (\text{Employee}, \text{name}), 1)$
$\textit{write}(x_2, \text{Manager}, 4, f)$ with $f(\text{name}) = 5$, $f(\text{salary}) = 6$, $f(\text{budget}) = 2$, and $f(\text{workers}) = 3$

The query for the second abstract schema:

$x_1 := \textit{const}(\langle(\text{'Jones'})\rangle, (\texttt{string}))$
$x_2 := \textit{access}(x_1, (\text{Manager}, \text{name}), 1)$
$x_3 := \textit{access}(x_2, (\text{Employee}, \text{boss}), 2)$
$x_4 := \textit{project}(x_3, f)$ with $f(1) = 8$

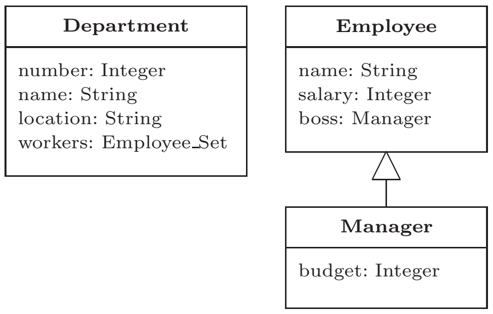
The update for the second abstract schema:

$x_1 := \textit{const}(\langle(\text{'Smith'}, 10\,000)\rangle, (\texttt{string}, \texttt{integer}))$
$x_2 := \textit{access}(x_1, (\text{Employee}, \text{name}), 1)$
$\textit{write}(x_2, \text{Manager}, 3, f)$ with $f(\text{name}) = 4$, $f(\text{salary}) = 5$, $f(\text{boss}) = 6$, and $f(\text{budget}) = 2$



Semantic constraints:
number is a key for Department,
name is a key for Employee,
Employee.boss and Manager.subordinates are inverse to each other

Access structures:
(Department, number),
(Department, workers)[a],
(Employee, name),
(Manager, name)



Semantic constraints:
number is a key for Department,
name is a key for Employee

Access structures:
(Department, number),
(Department, workers)[a],
(Employee, name),
(Employee, boss),
(Manager, name)

Fig. 4. Two alternative abstract logical formalisations shown as UML-class diagrams, each with a set of access structure declarations

A.2 Cost Parameters

Before we can estimate the costs we need to know the values of some cost parameters. Let's assume that the application analysis indicates the following constraints and parameters: the attribute number is a key for the class Department, $n_{\text{A:Department,number}} = 1$; the attribute name is a key for the class Manager, $n_{\text{A:Manager,name}} = 1$; on average 200 employees work in each department, $n_{\text{A:Department,workers}} = 200$; a manager manages 20 employees, $n_{\text{A:Employee,boss}} = 20$; and there are approximately 10 departments, $N_{\text{Department}} = N_{0,\text{Department}} = 10$; 100 managers, $N_{\text{Manager}} = N_{0,\text{Manager}} = 100$; and 2000 employees, $N_{\text{Employee}} = 2000$, $N_{0,\text{Employee}} = 1900$. On average 1 out of 100 employee objects and 1 out of 10 manager objects had to be moved from its original second memory location, $\theta_{\text{a:Employee}} = 0.01$, $\theta_{\text{a:Manager}} = 0.1$. In both abstract schemas department and employee objects have the same structure. We assume that the size of one department object is 15 blocks, $n_{\text{department}} = 15$, and that the size of an employee object is 1 block, $n_{\text{employee}} = 1$. Because in the first schema the class Manager has an additional set-valued attribute compared to the second schema, we assume that an object of type manager is the size of three blocks for the first schema, $n_{\text{manager}} = 3$, and of one block for the second schema, $n_{\text{manager}} = 1$. The filling levels of the identifier access structures are $\theta_{\text{s:Department}} = 0.95$ and $\theta_{\text{s:Employee}} = \theta_{\text{s:Manager}} = 0.7$.

One critical parameter describes the average number of elements in a set of type employee_`OID_Set`, $n_{\text{u:employee\_OID\_Set}} = 20$. The problem is that we have two kinds of sets of employees that can be considered typical: the workers of one department and workers with the same boss. This is a shortcoming of the basic cost functions. It could be eliminated by using additional channel cost statements (and respective cost functions) that supplement the stream types with information about the derivation of the values from class attributes.

For the underlying database system we assume that 16 OIDs fit into one block, $n_{\text{OID}} = 16$, the administrative overhead factor is $n_{\text{a}} = 0.1$, and that access structures are implemented through B-trees and so the number of block accesses required to locate an element is a logarithmic function, $\eta(x) = \lceil \log_{20}(x) \rceil$.

A.3 Space Costs

The estimation of the space costs for the two abstract schemas differs only at two points. The first schema needs more space to store the bigger manager objects, while the second schema has an additional access structure.

Application Data:

$\sum_{c \in \mathcal{C}} n_{\gamma(c)} \cdot N_{0,c} = n_{\text{department}} \cdot N_{0,\text{Department}} + n_{\text{employee}} \cdot N_{0,\text{Employee}} + n_{\text{manager}} \cdot N_{0,\text{Manager}}$

First abstract schema: $15 \cdot 10 + 1 \cdot 1900 + 1 \cdot 300 = 2350$

Second abstract schema: $15 \cdot 10 + 1 \cdot 1900 + 1 \cdot 100 = 2150$

Implicit Object Identifier Access Structures:

$\sum_{c \in \mathcal{C}} N_c \cdot (1/n_{\text{OID}}/\theta_{\text{s:}c} + n_{\text{a}}) = N_{\text{Department}} \cdot (1/n_{\text{OID}}/\theta_{\text{s:Department}} + n_{\text{a}}) + N_{\text{Employee}} \cdot (1/n_{\text{OID}}/\theta_{\text{s:Employee}} + n_{\text{a}}) + N_{\text{Manager}} \cdot (1/n_{\text{OID}}/\theta_{\text{s:Manager}} + n_{\text{a}}) = 10 \cdot (1/16/0.95 + 0.1) + 2000 \cdot (1/16/0.7 + 0.1) + 100 \cdot (1/16/0.7 + 0.1) = 1.66 + 378.57 + 18.93 = 399.16$

Explicit Access Structures:

$N_{\text{Department}} \cdot n_{\text{A:Department,number}} \cdot (1/n_{\text{OID}}/\theta_{\text{s:Department}} + n_{\text{a}}) = 10 \cdot 1 \cdot (1/16/0.95 + 0.1) = 1.66$

$N_{\text{Department}} \cdot n_{\text{A:Department,workers}} \cdot (1/n_{\text{OID}}/\theta_{\text{s:Department}} + n_{\text{a}}) = 10 \cdot 200 \cdot (1/16/0.95 + 0.1) = 331.58$

$N_{\text{Employee}} \cdot n_{\text{A:Employee,name}} \cdot (1/n_{\text{OID}}/\theta_{\text{s:Employee}} + n_{\text{a}}) = 2000 \cdot 1 \cdot (1/16/0.7 + 0.1) = 378.57$

$N_{\text{Manager}} \cdot n_{\text{A:Manager,name}} \cdot (1/n_{\text{OID}}/\theta_{\text{s:Manager}} + n_{\text{a}}) = 100 \cdot 1 \cdot (1/16/0.7 + 0.1) = 18.93$

$N_{\text{Employee}} \cdot n_{\text{A:Employee,boss}} \cdot (1/n_{\text{OID}}/\theta_{\text{s:Employee}} + n_{\text{a}}) = 2000 \cdot 1 \cdot (1/16/0.7 + 0.1) = 378.57$

First abstract schema: $\sum_{(c,A)\in X} N_c \cdot n_{\text{A:}c,A} \cdot (1/n_{\text{OID}}/\theta_{\text{s:}c} + n_{\text{a}}) = 1.66 + 331.58 + 378.57 + 18.93 = 730.74$

Second abstract schema: $\sum_{(c,A)\in X} N_c \cdot n_{\text{A:}c,A} \cdot (1/n_{\text{OID}}/\theta_{\text{s:}c} + n_{\text{a}}) = 1.66 + 331.58 + 378.57 + 18.93 + 378.57 = 1109.31$

Total Space Estimation:

First abstract schema: $2350 + 399.16 + 730.74 = 3479.90$

Second abstract schema: $2150 + 399.16 + 1109.31 = 3658.47$

A.4 Time Costs

Now let's analyse our example machine programs with our basic cost functions.

Query on first abstract schema:

1. $x_1 := \mathit{const}(\langle(\text{‘Jones’})\rangle, (\texttt{string}))$
   Block accesses: 0, Value accesses: 0, Length of output stream: 1
2. $x_2 := \mathit{access}(x_1, (\text{Manager}, \text{name}), 1)$
   Block accesses: $\|x_1\| \cdot n_{\text{A:Manager,name}} \cdot (\eta(N_{\text{Manager}}) + \theta_{\text{a:Manager}} + n_{\text{manager}}) = 1 \cdot 1 \cdot (\eta(100) + 0.1 + 3) = 5.1$, Value accesses: $\|x_1\| = 1$, Length of output stream: $\|x_1\| \cdot n_{\text{A:Manager,name}} = 1 \cdot 1 = 1$
3. $x_3 := \mathit{unnest}(x_2, 7)$
   Block accesses: 0, Value accesses: $\|x_2\| \cdot n_{\text{u:employee\_OID\_Set}} = 1 \cdot 20 = 20$, Length of output stream: $\|x_2\| \cdot n_{\text{u:employee\_OID\_Set}} = 1 \cdot 20 = 20$
4. $x_4 := \mathit{activate}(x_3, 7)$
   Block accesses: $\|x_3\| \cdot (\theta_{\text{a:Employee}} + n_{\text{employee}}) = 20 \cdot (0.01 + 1) = 20.2$, Value accesses: $\|x_3\| = 20$, Length of output stream: $\|x_3\| = 20$
5. $x_4 := \mathit{project}(x_3, f)$ with $f(1) = 8$
   Block accesses: 0, Value accesses: 0, Length of output stream: $\|x_3\| = 20$

Thus the estimate for the total number of block accesses is $0 + 5.1 + 20 + 20.2 + 0 = 45.3$ and the estimate for the total number of value accesses is $0 + 1 + 20 + 20 + 0 = 41$.

Update on first abstract schema:

1. $x_1 := \mathit{const}(\langle(\text{‘Smith’}, 10\,000, \{\})\rangle, (\texttt{string}, \texttt{integer}, \text{employee\_}\texttt{OID\_Set}))$
   Block accesses: 0, Value accesses: 0, Length of output stream: 1
2. $x_2 := \mathit{access}(x_1, (\text{Employee}, \text{name}), 1)$
   Block accesses: $\|x_1\| \cdot n_{\text{A:Employee,name}} \cdot (\eta(N_{\text{Employee}}) + \theta_{\text{a:Employee}} + n_{\text{employee}}) = 1 \cdot 1 \cdot (\eta(2000) + 0.01 + 1) = 4.01$, Value accesses: $\|x_1\| = 1$, Length of output stream: $\|x_1\| \cdot n_{\text{A:Manager,name}} = 1 \cdot 1 = 1$
3. $\mathit{write}(x_2, \text{Manager}, 4, f)$
   Block accesses: $\|x_2\| \cdot (n_{\text{manager}} + \eta(N_{0,\text{Employee}}) + \eta(N_{\text{Employee}}) \cdot n_{\text{A:Employee,name}} + \eta(N_{\text{Manager}}) \cdot n_{\text{A:Manager,name}}) = 1 \cdot (3 + \eta(1900) + \eta(2000) \cdot 1 + \eta(100) \cdot 1) = 11$, Value accesses: 0

Thus the estimate for the total number of block accesses is $0 + 4.01 + 11 = 15.01$ and the estimate for the total number of value accesses is $0 + 1 + 0 = 1$.

Query on second abstract schema:

1. $x_1 := \mathit{const}(\langle(\text{‘Jones’})\rangle, (\texttt{string}))$
   Block accesses: 0, Value accesses: 0, Length of output stream: 1
2. $x_2 := \mathit{access}(x_1, (\text{Manager}, \text{name}), 1)$
   Block accesses: $\|x_1\| \cdot n_{\text{A:Manager,name}} \cdot (\eta(N_{\text{Manager}}) + \theta_{\text{a:Manager}} + n_{\text{manager}}) = 1 \cdot 1 \cdot (\eta(100) + 0.1 + 1) = 3.1$, Value accesses: $\|x_1\| = 1$, Length of output stream: $\|x_1\| \cdot n_{\text{A:Manager,name}} = 1 \cdot 1 = 1$
3. $x_3 := \mathit{access}(x_2, (\text{Employee}, \text{boss}), 2)$
   Block accesses: $\|x_2\| \cdot n_{\text{A:Employee,boss}} \cdot (\eta(N_{\text{Employee}}) + \theta_{\text{a:Employee}} + n_{\text{employee}}) = 1 \cdot 20 \cdot (\eta(2000) + 0.01 + 1) = 80.2$, Value accesses: $\|x_2\| = 1$, Length of output stream: $\|x_2\| \cdot n_{\text{A:Employee,boss}} = 1 \cdot 20 = 20$
4. $x_4 := \mathit{project}(x_3, f)$ with $f(1) = 8$
   Block accesses: 0, Value accesses: 0, Length of output stream: $\|x_3\| = 20$

Thus the estimate for the total number of block accesses is $0+3.1+80.2+0 = 83.3$ and the estimate for the total number of value accesses is $0+1+1+0 = 2$.

Update on second abstract schema:

1. $x_1 := \mathit{const}(\langle(\text{‘Smith’}, 10\,000)\rangle, (\texttt{string}, \texttt{integer}))$
   Block accesses: 0, Value accesses: 0, Length of output stream: 1
2. $x_2 := \mathit{access}(x_1, (\text{Employee}, \text{name}), 1)$
   Block accesses: $\|x_1\| \cdot n_{\text{A:Employee,name}} \cdot (\eta(N_{\text{Employee}}) + \theta_{\text{a:Employee}} + n_{\text{employee}}) = 1 \cdot 1 \cdot (\eta(2000) + 0.01 + 1) = 4.01$, Value accesses: $\|x_1\| = 1$, Length of output stream: $\|x_1\| \cdot n_{\text{A:Manager,name}} = 1 \cdot 1 = 1$
3. $\mathit{write}(x_2, \text{Manager}, 3, f)$
   Block accesses: $\|x_2\| \cdot (n_{\text{manager}} + \eta(N_{0,\text{Employee}}) + \eta(N_{\text{Employee}}) \cdot n_{\text{A:Employee,name}} + \eta(N_{\text{Employee}}) \cdot n_{\text{A:Employee,boss}} + \eta(N_{\text{Manager}}) \cdot n_{\text{A:Manager,name}}) = 1 \cdot (1 + \eta(1900) + \eta(2000) \cdot 1 + \eta(2000) \cdot 20 + \eta(100) \cdot 1) = 69$, Value accesses: 0

Thus the estimate for the total number of block accesses is $0 + 4.01 + 69 = 73.01$ and the estimate for the total number of value accesses is $0 + 1 + 0 = 1$.

A.5 Cost Estimation Summary

First abstract schema:
space cost: 3479.90 blocks
time cost, query: 45.3 block accesses plus 41 value accesses
time cost, update: 15.01 block accesses plus 1 value access

Second abstract schema:
space cost: 3658.47 blocks
time cost, query: 83.3 block accesses plus 2 value accesses
time cost, update: 73.01 block accesses plus 1 value access

The cost estimation shows, that the first alternative needs less space. Additionally it needs less time for the two investigated user activities, under the assumption that a value access is not as expensive as a block access. (But without further information, we cannot decide whether the difference is significant.)

A major cause for the higher time cost of the query for the second schema lies in the fact that the machine program uses an access structure instead of an object activation. In our setting of the cost parameters we represented a database system where the use of an access structures takes time that rose with the logarithm of the number of objects in the access structure. The time costs of the query for the first schema could have been lower, if we had chosen a different cost parameter η.

Designing Valid XML Views

Ya Bing Chen, Tok Wang Ling, and Mong Li Lee

School of Computing, National University of Singapore
(chenyabi,lingtw,leeml)@comp.nus.edu.sg

Abstract. Existing systems for XML views only support selection operation applied in the views and cannot validate views. In this paper, we propose a systematic approach to design valid XML views. First, we transform the semistructured XML source documents into a semantically rich *O*bject-*R*elationship-*A*ttribute model designed for *S*emi*S*tructured data (ORA-SS). Second, we enrich the ORA-SS diagram with semantics such as participation constraints of object classes and distinguishing between attributes of object classes and relationship types, which cannot be expressed in the XML document. Third, we use the additional semantics to develop a set of rules to guide the design of valid XML views. We identify four transformation operations for creating XML views, namely, selection, projection, join and swap operation. Finally, we develop a comprehensive algorithm that checks for the validity of XML views constructed by applying the four operations.

1 Introduction

It is necessary to provide for XML views [1]. Several systems have been proposed to support XML views, including Active Views [2] and MIX [5]. While both systems provide for the definition of XML views, they do not validate the views that are created. Therefore, there is no guarantee that the views defined are valid.

In this paper, we propose a systematic approach to ensure the validity of XML views. First, we transform XML documents into ORA-SS schema diagram proposed in [6], [7]. Second, we enrich ORA-SS schema diagram with semantics, such as distinguishing between attributes of object classes and relationship types. These additional semantics will allow us to validate XML views subsequently. Third, based on the enriched ORA-SS schema diagram, we propose a set of rules to guide the design of valid XML views. We also develop a comprehensive algorithm that checks for the validity of XML views.

The rest of the paper is organized as follows. Section 2 introduces the background of our work. Section 3 describes our proposed approach to validate XML views in detail. Section 4 discusses related work and we conclude in section 5. Note there is an appendix that contains XML instance documents and XQuery used in the paper.

2 Preliminaries

2.1 ORA-SS Data Model

The ORA-SS (Object-Relationship-Attribute model for SemiStructured data) data model comprises of three basic concepts: object classes, relationship types and attributes. An object class is similar to an entity type in an ER diagram or an element in XML documents. A relationship type describes a relationship among object classes. Attributes are properties, and may belong to an object class or a relationship type. ORA-SS data model has four diagrams: the schema diagram, the instance diagram, the functional dependency diagram and the inheritance diagram. A full description of the data model can be found in [6]. In this paper, we will focus on the schema diagram because it is sufficient for our purposes.

For example, the left part of figure 1 shows an ORA-SS schema diagram, which contains three object classes – *project, supplier* and *part*, and the right part of figure 1 then shows an ORA-SS instance diagram of the schema diagram. In the schema diagram, an object class is represented as a labeled rectangle. A relationship type between two object classes in an ORA-SS schema diagram can be described by *name, n, p, c,* where *name* denotes the name of the relationship type, *n* is an integer indicating the degree of the relationship type (n = 2 indicates binary, n = 3 indicates ternary, etc.), *p* is the participation constraint of the parent object class in the relationship type, and *c* is the participation constraint of the child object class in the relationship type. The participation constraints are defined using the min:max notation.

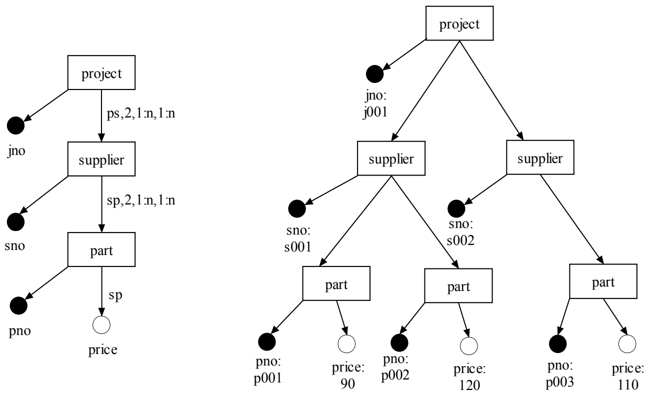


Fig. 1. An ORA-SS Schema Diagram (left) and Instance Diagram (right)

In the ORA-SS schema diagram, labeled circles denote attributes, and keys are filled circles. The attributes of an object class can be distinguished from attributes of a relationship type. The former has no label on its incoming edge while the latter has the name of the relationship type to which it belongs on its incoming edge.

It is clear that ORA-SS is a semantically rich data model. The model not only reflects the nested structure of semistructured data, but it also distinguishes between object classes, relationship types and attributes. In addition, ORA-SS provides for the specification of the participation constraints of object classes in relationship types and distinguishes between attributes of relationship types and attributes of object classes. Such information is lacking in other existing semistructured data models including OEM [3], XML DTD and XML Schema [9]. For this reason, we adopt ORA-SS as the data model for valid XML views design, because the additional semantics is essential for the verification of the validity of XML views.

2.2 View Definition Language

The World Wide Web Consortium has proposed an XML query language called XQuery [8]. XQuery provides flexible query facilities to extract data from real and virtual documents on the Web. The basic form of an XQuery expression consists of For, Let, Where and Return (FLWR) expressions. Although XQuery currently does not provide for the definition of views, we can easily extend it to include the definition of views as follows:

"Create View As view name" followed by FLWR expression.

A full description of FLWR expression in XQuery can be found in [8].

3 Valid XML Views Design

In this section, we will describe our approach to design valid XML views. When an XML view does not violate the integrity constraint and semantics of the original XML document, we say the XML view is *valid*. There are three main steps in our approach. The first two steps are preparatory stages for valid XML views design. The goal of the two steps is transform XML documents into ORA-SS schema diagram enriched with semantics, based on which, we may begin to design XML views. Therefore, we will cover them roughly in the paper.

1. Transform an XML document into an ORA-SS schema diagram.
2. Enrich the ORA-SS schema diagram with necessary semantics.
3. Define a set of rules to guide the design of valid XML views.

We will present the rough idea of the first two steps and the detailed idea of the third step.

3.1 Motivating Example

Invalid views may be produced in the case where important semantics are not expressed in the underlying data model. We will illustrate this point with an XML document shown in XDoc 1 in Appendix. The XML document is conforming to the ORA-SS schema in Figure 1. Note that there exists an implicit functional dependency in the document: *supplier, part* $\rightarrow$ *price*.

A user may use XQuery to design a view that swaps the location of the elements *supplier* and *part.* That is, *supplier* becomes a child of *part* and *part* becomes the parent of *supplier.* As a consequence, we need to decide where to place the element *price*. Since the XML document does not explicitly express the functional dependency: *supplier, part* $\rightarrow$ *price*, the element *price* may be placed under the element *part* in the designed view. This makes *price* an attribute of *part*. XDoc.2 in Appendix shows an instance of the view obtained. A new functional dependency: *part* $\rightarrow$ *price*, now holds in the view that violates the functional dependency *supplier, part* $\rightarrow$ *price* in the source document. We say that such a view is invalid. In order to obtain a valid view, the element *price* should be placed under the element *supplier* so that the original functional dependence is preserved. XDoc.3 in Appendix describes an instance of a valid view.

The above example shows that invalid views may be designed if the underlying data model does not express explicitly the necessary semantics. This includes the participation constraints of object classes in relationship types, and distinguishing between attributes of relationship types and attributes of object classes, which are available in the ORA-SS model.

3.2 Transformation of XML into ORA-SS

In this section, we will begin to introduce the two pre-processing steps for valid XML views design. First, we give a brief outline of the transformation of an XML document into an ORA-SS schema diagram:

- Map root element of the XML document into the root object class in the ORA-SS schema diagram.
- Map each element that has attributes or sub-elements into an object class in the ORA-SS schema diagram.
- Map attributes of an element into the attributes of the object class corresponding to the element.
- Map the rest of the elements, which do not have attributes or sub-elements, into attributes of their corresponding parent object classes.

3.3 Semantic Enrichment of ORA-SS

The ORA-SS diagram obtained from Section 3.2 will basically reflect the tree structure of the XML document and distinguish between object classes and attributes. In order to support the validation of XML views, we need to enrich it with the following additional semantics. Users will be allowed to input this semantics in this step.

- Identify key attributes of each object class.
- Identify attributes that belong to object classes.
- Identify relationship types among object classes.
- Identify attributes of relationship types.

3.4 Validity of XML Views

After semantically enriching the ORA-SS schema diagram, we can now design XML views and determine its validity. The XML views are designed by applying four transformation operations, which are selection, projection, join and swap. The first three are analogous to the selection, projection and join in relational databases. The fourth one is unique in XML settings because it exchanges the positions of parent and child object classes. An XML view may not be simply based on only one of the operations. For example, a view may first apply a selection operation then a join operation. We will now discuss how to guarantee valid XML views design when each operation applies.

3.4.1 Selection Operation

Selection operations basically filter data by using predicates. These are similar to selection operation in relational databases. The structure of source schema remains unchanged and will not cause any changes in the semantics of the source schema. Therefore, if an XML view only applies selection operations, it will be always valid.

Example 1
Suppose we want to design a view called *expensive-part* on the ORA-SS source schema diagram defined in Figure 1. The view definition is shown in XQuery.1 in Appendix. The view depicts *projects* for which there exist *suppliers* for which there exist *parts* with a *price* > 80. Within those *projects* it only returns *suppliers* for which there exist *parts* with a *price* > 80. Within those *suppliers* it only returns the *parts* with a *price* > 80.

Selection operations put predicates on the source schema to filter data. They do not restructure the source document. The resulting view schema will be the same as the source schema. Hence, such views will not violate semantics in the source schema. Then we do not need to set up rules to guarantee the validity of views when only selection operations are applied.

3.4.2 Projection Operation

Projection operations select or drop object classes or attributes in the source schema. They essentially extract a subset structure of the source schema. Since the structure of the source schema is changed, the source semantics may be affected. Therefore it is possible to design an invalid view that violates the semantics in the source schema. This can be detected by designing a set of rules to check for the validity. We will first illustrate how to design a view applying projection operations. Then, we will give the rules to guarantee the validity of such views.

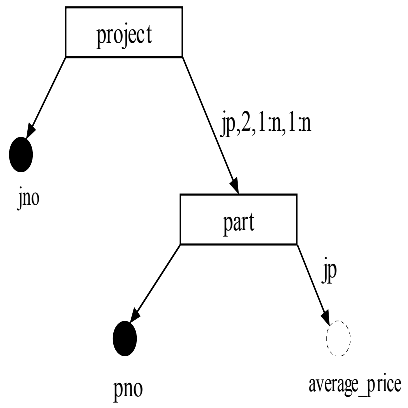


Fig. 2. ORA-SS schema of the view *Project-Part*

Example 2

Suppose we define a view called *project-part* based on the ORA-SS schema diagram in Figure 1. This view removes the intermediate object class *supplier* (see Figure 2). This implies that the attribute *sno* has to be dropped too since attributes cannot exist without its owner object class. Next, we need to remove the relationship types *–js* and *sp*, both of which involves the object class *supplier* which has been removed. The attributes of these relationship types can be dropped too. Alternatively, we can map the attribute of the relationship type *sp – price* to an aggregate attribute called *average_price*, which represents the average price of one part in a given project.

Based on the view schema, we may write a view definition in XQuery expression, which is shown in XQuery.2 in Appendix.

This example shows that flexible views can be designed based on ORA-SS with its additional semantics. However, we need to handle the semantics properly so that meaningful views are guaranteed. The following rules are critical for designing valid XML views that apply projection operations.

- **Rule Proj1.** If an object class has been dropped, then its attributes must be dropped too.
- **Rule Proj2.** If an object class has been dropped, then all relationship types containing the object class must be dropped too. The attributes of these relationship types must be dropped, or mapped into attributes with some aggregate function, such as avg, max/min or sum, or mapped into attributes typed in bag of values if they cannot be aggregated.

Rule Proj1 indicates that we cannot leave an attribute in the view if its object class has been dropped. Without its object class, the attributes will lose their meaning. When an object class is dropped, it may be because the object class itself is dropped, or because the key attribute of the object class is dropped.

On the other hand, rule Proj2 indicates that those relationship types containing the dropped object class must be dropped too. Although these relationship types will not be shown in XML document or XML schema, they need to be dropped to keep the semantics in the ORA-SS view schema consistent. The attributes of these relationship types can be dropped too. However, ORA-SS allow us to map the attributes of affected relationship types into some aggregate function attributes, such as avg, max/min, or sum, which make the view more expressive and more powerful. These modified attributes should be meaningful in the view. In cases where the type of the attributes is string that cannot be aggregated, these attributes can be changed into attributes typed in bag of value.

3.4.3 Join Operation

Join operations actually join object classes and their attributes together by key – foreign key references. There may be one referencing object class and one referenced object class in an ORA-SS source diagram. The former object class has an attribute that is actually a key attribute of the later object class. Therefore, the former is able to refer to the later by the attribute, which plays the role of a foreign key. In our notion of join operations, we first combine the two object classes together before combining all attributes of the two object classes so that they will become a single object class.

This is analogous to the join operations in relational databases, which joins two flat tables by key – foreign key references.

ORA-SS makes it possible to design such XML views as applying join operations and guarantee they are valid. This is because ORA-SS distinguishes between object classes and attributes so that two object classes can be joined. Furthermore, ORA-SS differentiates between attributes of object classes and attributes of relationship types so that attributes of relationship types will not be treated as attributes of the joined object class improperly. Next we will illustrate join operation with the following example.

Example 3
Figure 3 shows an ORA-SS source schema diagram. The object class *supplier'* under *project* refers to another object class *supplier* under *retailer* by *sno*, which is the key attribute of *supplier*. There is a relationship type between *retailer* and *supplier* called *rs*, which has an attribute *contract* under *supplier*.

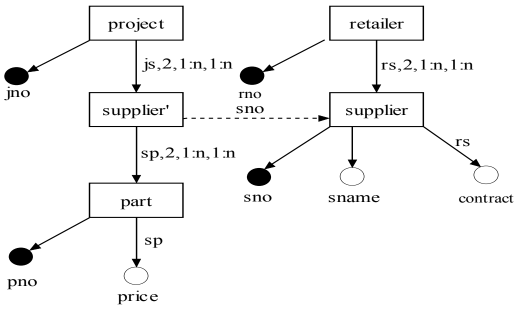


Fig. 3. An ORA-SS schema diagram on project, supplier, part and retailer

We design a view called *join-supplier* shown in figure 4. The view joins *supplier* and *supplier'* together. The attributes *sno* and *sname* of *supplier* are moved under *supplier'* in the view. However, the attribute *contract* cannot be moved in the same way because it belongs to the relationship type *rs*. If it is moved under *supplier'*, then it will become an attribute of *supplier'*. The operation then violates the original semantics and makes the view invalid. Therefore, to keep the view valid, the attribute *contract* must remain in the source schema and not be moved in the view. XQuery.3 in Appendix gives the view definition of the *join-supplier*.

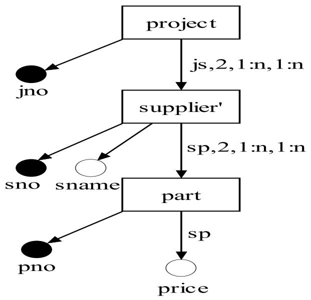


Fig. 4. ORA-SS schema of the view *join-supplier*

Based on the example above, we give the rules for designing valid views that apply join operations.

- **Rule Join1.** If a referencing object class and a referenced object class are joined together in the view, and there exists such relationship types below the referenced object class as contain those object classes above the referenced object class, then attributes of such relationship types must be dropped, or mapped into attributes with some aggregate function.
- **Rule Join2.** If a referencing object class and a referenced object class are joined together in the view, and there only exists such relationship types below the referenced object class as do not contain those object classes above the referenced object class, then attributes of such relationship types can be selected or dropped according to the view requirement.

When we design views that join one referencing object class and one referenced object class together, we need to properly handle the relationship types and their attributes below the referenced object class. These relationship types can be divided into two types.

The first type of relationship types involves object classes above the referenced object class. Rule Join1 states that such relationship types and their attributes must be dropped or mapped into attributes with some aggregate function, which is the same as Rule Proj1. It is because those object classes above the referenced object class, which are ancestors of the referenced object class, will not exist in the view any more. Therefore, these relationship types involving these object classes will not exist too. Their attributes must be dropped or modified.

The second type of relationship types only involves object classes below the referenced object class. Rule Join2 states that these relationship types and their attributes can be dropped or selected according to the view requirement. It is because the object classes below the referenced object class may still exist in the view. Then the corresponding relationship types may be included in the view also.

3.4.4 Swap Operation

Swap operations restructure the source schema by exchanging the positions of a parent object class and one of its child object class. We think this type of operation will be widely applied in XML views design because of the hierarchical nature of XML data. Therefore we include it as one of four types of operations. The following example illustrates how to design valid XML views when swap operation is applied.

Example 4
Given the source schema in Figure 1, we design a view shown in Figure 5 called *swap-supplier-part*, which swaps the object class *supplier* and *part* hierarchically.

After the object classes have been swapped, we need to ensure that their attributes are relocated properly. The attributes *pno* and *sno* are also swapped in order to preserve their parent object classes. However, the attribute *price*, which belongs to the relationship *sp*, must stay with the new child object class *supplier* in order to preserve the semantics of the source schema. If it moves with the object class *part*, then it will violate the semantics in the source schema.

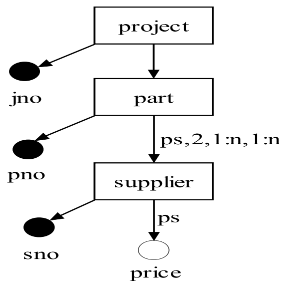


Fig. 5. ORA-SS schema of the view *swap-supplier-part*

An XQuery expression of the *swap-supplier-part* view is described in XQuery.4 in Appendix. The following rules guarantee that the design of views is valid when swap operations are applied.

- **Rule Swap1.** If two object classes are swapped in the view, then the attributes of each of the object classes must stay with the object class.
- **Rule Swap2.** If two object classes are swapped in the view, then the attributes of relationship types involving the two object classes must stay below the lowest participating object class in the relationship types.

When a swap operation is applied, the two swapped object classes may not involve any relationship type. In this case, we simply swap them and move their attributes with them, as stated in Rule Swap1. If there is any relationship type involving the two object classes, then we must keep the attributes of the relationship types below the lowest participating object class of the relationship types. If these attributes move with one of the object class, they will not belong to the relationship types and become attributes of the object class, which then violates the semantics in the source schema and lead to a meaningless view.

3.4.5 Design Rules for IDentifier Dependency Relationship

The previous sections present the design rules when projection, join and swap operations are applied in XML views. However, these rules are not enough when the views contain IDD (IDentifier Dependency) relationship types. An IDD relationship type is defined as follows:

Definition 1. An object class A is said to be ID dependent on its parent object class B if A does not have a key attribute, and an A object can be identified by its parent's key value (say k1) together with some of its own attributes (say k2). That is, the key of A is {k1, k2}. The relationship type between A and B is then called IDD relationship type.

Example 5
Figure 6 shows an IDD relationship type between the object class *employee* and *child*. The object class *child* does not have a key attribute, but can be identified by the key attribute of *employee* – *eno* and its own attribute – *cname*. When we design a view over the IDD relationship type, additional rules are needed to keep the view meaningful.

Based on Figure 6, we design a view applying a swap operation, which swaps the object class *employee* and *child* (see Figure 7). Unlike the previous view applying swap operations, this view still duplicates the key attribute of *employee – eno* for the object class *child* so that *eno* and *cname* can combine a key for the object class *child*. It is because the object class *child* cannot be identifiable without *eno*. Note this view need to be enforced with a constraint, which says the *eno* under the object class *child* must be the same as the *eno* under the object class *employee*. The straight line between the incoming edges of the attributes *eno* and *cname* denotes {*eno, cname*} is a composite key for the object class *child*.

We can also design a view applying projection operation. For example, Figure 8 depicts a view that drops the object class *employee*. To make the object class *child* identifiable, the key attribute of *employee – eno* is also combined with the attribute *cname* to construct a key for the object class *child*.

The similar situation exists if a join operation is applied in a source schema containing an IDD relationship type.

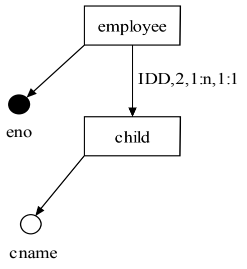


Fig.6. ORA-SS source schema diagram of an IDD relationship type

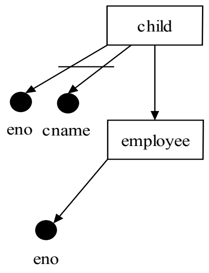


Fig.7. ORA-SS schema of the view swapping employee and child

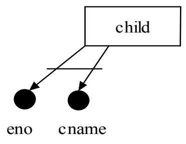


Fig.8. ORA-SS schema of the view dropping employee

These examples show that when we design a view that destroys an IDD relationship type, the key attribute of the parent object class of the IDD relationship type should be added to the child object class to construct a key for the child. The following additional rules indicate for each operation, how XML views should be designed when IDD relationship types are involved.

Rule Proj_IDD. If an object class is a parent object class of an IDD relationship type and is dropped in the view, then its key attribute must be added to the child object class of the IDD relationship type to construct a key for the child.

Rule Join_IDD. If an object class is a child of an IDD relationship type and is referenced by another object class in the source schema, and a view is designed to join the two object classes together, then the key attribute of the parent object class of the IDD relationship type must be added to the child to construct a key for the child.

Rule Swap_IDD. If two object classes compose an IDD relationship type and are swapped in the view, then the key attribute of the parent object class must be added to the child object class to construct a key for the child.

In summary, a theorem may be derived based on the above sections.

Theorem 1. XML views designed based on all the above rules do not violate the integrity constraints and semantics of the original XML documents.

3.4.6 View Validation Algorithm

In this section, we summarize all the design rules into an algorithm to validate XML views. Our algorithm will automatically modify related part of the view schema according to different operations so that the view is guaranteed to be valid.

```
   Algorithm ValidateView
   Input: ORA-SS schema diagram
   Output: A valid ORA-SS view schema diagram
Do
  Switch (Operation) {
   Case (Drop an object class):   //Projection operation
          if (the object class is a parent object class of an IDD relationship type){
                add the key attribute of the parent to the child object class of the IDD
                         relationship type to construct a key for the child;
          }   //Rule Proj_IDD
          drop attributes of the object class;   //Rule Proj1
          drop relationship types containing the object class;   //Rule Proj2
          handle the attributes of relationship types (drop or modify according to
                         the view requirement);   //Rule Proj2
          break;

       Case (Join two object classes):   //Join operations
           if (the referenced object class is a child object class of an IDD relationship
                type){
                add the key attribute of the parent object class of the IDD relationship
                         type to the child to construct a key for the child;
           }   //Rule Join_IDD
          drop the relationship types that are below the referenced object class but
                  contain object classes above the referenced object class; //Rule Join1
           handle the attributes of such relationship types (drop or modify according
                  to the view requirement;   //Rule Join1
           handle the relationship types and their attributes that are below the
                  referenced object class and do not contain object classes above the
                  referenced object class (drop or keep according to the view require-
                   ment);   //Rule Join2
       break;
       Case (Swap two object classes):   //Swap operations
            If (the two object classes compose an IDD relationship type){
                 add the key attribute of the parent object class to the child object class
                          to construct a key for the child;
            }   //Rule Swap_IDD
            move the attributes of each object class with them;   //Rule Swap1
            keep attributes of relationship types containing the two object classes
                   below the lowest participating object class in the relationship types;
                   //Rule Swap2
            break;
   }
while (view design is not done);
```

The algorithm first uses a do-while clause to monitor the process of designing view until the view is done. Then it uses a selection statement – switch clause to handle the three operations – projection, join and swap, which may be repeated in the view. Once an operation is applied in the view, the algorithm first checks if an IDD relationship type is involved. If so, then it applies the corresponding additional rule for the operation. After that, the algorithm applies the normal rules for the operation. In this way, the view will be guaranteed to be valid once it is done.

4 Related Work

Several prototype systems have been developed to support the design of XML views. The Active Views system [2] is built on top of Ardent Software's XML repository [4], which is based on the object-oriented O2 system. In the Active Views system, a view is presented as an object, which allows not only data, but also methods. MIX (Mediation of Information using XML) [5] is another system that offers a virtual XML view from its underlying heterogeneous sources. Table 1 compares our approach with the Active View system and MIX.

Our approach adopts the semantically rich ORA-SS data model to express both the source and view schemas. This allows us to support a richer set of views compared to Active Views and MIX. The Active Views system uses the Object Query Language as a view definition language, and the Lorel language [3] as its query language over the views. This requires the users to be familiar with two different languages. MIX develops its own XMAS language as the view definition language and query language. In contrast, our approach directly adopts the W3C standard, XQuery as the query/view language over the views. A view definition is differentiated from a query by its additional view declaration clause before FLWR expression. Finally, both the Active Views system and MIX system do not provide for the validation of views. As a consequence, these two systems cannot support valid XML views that apply projection, join and swap operations.

Table 1. Comparison of ActiveViews system, MIX system and our approach

	Active Views system [2]	MIX system [5]	Our approach
Data model	XML	XML DTD	ORA-SS
View definition language	OQL-style language	XMAS language	XQuery language
Query language	Lorel language	XMAS language	XQuery language
Support projection, join and swap operations	No	No	Yes
Support view validation	No	No	Yes
Support graphical views design	No	No	Yes

5 Conclusions

In this paper, we have proposed a systematic approach for valid XML views design. The approach is composed of three steps. The first two steps are preparatory stages. In first step, we transform an XML document into an ORA-SS schema diagram. In second step, we enrich the ORA-SS schema diagram with necessary semantics for valid XML views design. In third step, we develop a set of rules to guide the design of valid XML views. We also give an algorithm to validate views. We have implemented our approach into a CASE tool for designing XML views. In the future work, we will give more formal grounding, such as query algebra underlying view definition. We will also design query translation algorithm and provide support to update XML views in the future.

References

[1] S. Abiteboul. On views and XML. In Proceedings of the Eighteenth ACM Symposium on Principles of Database Systems, ACM Press, pages 1-9, 1999.

[2] S. Abiteboul, B. Amann, S. Cluet, A. Eyal, L. Mignet, and T. Milo. Active views for electronic commerce. In Int. Conf. on Very Large DataBases (VLDB), Edinburgh, Scotland, pages 138-149,1999.

[3] S. Abiteboul, D. Quass, J. McHugh, J.Widom, and J. L. Wiener. The lorel query language for semistructured data. International Journal of Digital Libraries, Volume 1, No. 1, pages 68-88, 1997.

[4] Ardent Software. http://www.ardentsoftware.com.

[5] C. Baru, A. Gupta, B. Ludaescher, R. Marciano, Y. Papakonstantinou, and P. Velikhov. XML-Based Information Mediation with MIX. ACM-SIGMOD, Philadelphia, PA, pages 597-599, 1999.

[6] Gillian Dobbie, Xiaoying Wu, Tok Wang Ling, Mong Li Lee. ORA-SS: An Object-Relationship-Attribute Model for Semi-Structured Data. Technical Report TR21/00, School of Computing, National University of Singapore, 2000.

[7] Tok Wang Ling, Mong Li Lee, Gillian Dobbie. Application of ORA-SS: An Object-Relationship-Attribute Model for Semi-Structured Data. In Proceedings of the Third International Conference on Information Integration and Web-based Applications & Services (IIWAS), Linz, Austria, 2001.

[8] http://www.w3.org/TR/xquery.

[9] http://www.w3.org/XML/Schema.

6 Appendix: XML Document and XQuery in the Paper

```
<db>
   <project jno=”j001”>
     <supplier sno=“s001”>
       <part pno=“p001”>
           <price> 100</price>
       </part>
     </supplier>
     <supplier sno=“s002”>
       <part pno=“p001”>
          <price> 100</price>
        </part>
     </supplier>
   </project>
</db>
```

XDoc.1. An XML document conforming to the schema in Figure 1

```
<db>
   <project jno=”j001”>
     <part pno=“p001”>
        <price>100</price>
        <supplier sno=“S001”/>
        <supplier sno=“S002”/>
     </part>
   </project>
</db>
```

XDoc.2. Invalid view instance of the XML document in XDoc.1

```
<db>
   <project jno=”j001”>
     <part pno=“p001”>
       <supplier sno=“S001”>
           <price>100</price>
       </supplier>
       <supplier sno=“S002”/>
           <price>100</price>
       </supplier>
     </part>
   </project>
</db>
```

XDoc.3. Valid view instance of the XML document in XDoc.2

```
   Create View As expensive-part
   Let   $p:= document(“spj.xml”)
         //part[price>80]
   Return  filter($p/../.. | $p/.. | $p |
$p/price)
```

XQuery.1. XQuery expression of the view *expensive-part*

```
Create View As project-part
For    $j In document("spj.xml")
        //project
Return
   <project jno={$j/@jno}>
      {For $pn In
         distinct($j//part/@pno)
       Let  $p := $j//part[@pno=$pn]
       Return
         <part pno={$pn}>
            <average_price>
               {avg($p/price)}
            </average_price>
         </part>
      }
   </project>
```

XQuery.2. XQuery expression of the view project-part

```
Create View As join-supplier
For $j in document("spjr.xml")
     //project
Return
  <project jno={$j/@jno}>
    {For $s In $j/supplier,
       $ref_s In document("spjr.xml")

//retailer/supplier[@sno=$s/@sno]
     Return
       <supplier sno={$ref_s/@sno}
              sname={$ref_s/@sname}>
          {$s/part}
       </supplier>
    }
    </project>
```

XQuery.3. XQuery expression of the view join-supplier

```
Create View As swap-supplier-part
For $j In document("spj.xml")//project
Return
  <project jno={$j/@jno}>
   {For $pn In  distinct($j//part/@pno)
    Return
      <part pno={$pn}>
      {For $s In $j/supplier[part/@pno=$pn]
       Return
         <supplier sno={$s/@sno}>
           {$s/part[@pno=$pn]/price}
         </supplier>
      }
      </part>
   }
  </project>
```

XQuery.4. XQuery expression of the view *swap-supplier-part*

Author Index