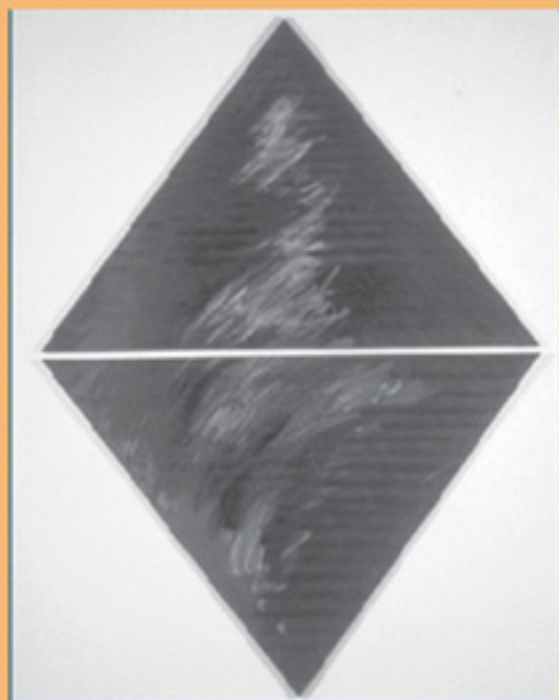


Advanced Topics in Database Research

Volume 5



Keng Siau

Advanced Topics in Database Research Volume 5

Keng Siau
University of Nebraska-Lincoln, USA



IDEA GROUP PUBLISHING

Hershey • London • Melbourne • Singapore

Acquisitions Editor: Michelle Potter
Development Editor: Kristin Roth
Senior Managing Editor: Amanda Appicello
Managing Editor: Jennifer Neidig
Copy Editor: Lisa Conley
Typesetter: Jessie Weik
Cover Design: Lisa Tosheff
Printed at: Integrated Book Technology

Published in the United States of America by
Idea Group Publishing (an imprint of Idea Group Inc.)
701 E. Chocolate Avenue, Suite 200
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@idea-group.com
Web site: <http://www.idea-group.com>

and in the United Kingdom by
Idea Group Publishing (an imprint of Idea Group Inc.)
3 Henrietta Street
Covent Garden
London WC2E 8LU
Tel: 44 20 7240 0856
Fax: 44 20 7379 0609
Web site: <http://www.eurospanonline.com>

Copyright © 2006 by Idea Group Inc. All rights reserved. No part of this book may be reproduced, stored or distributed in any form or by any means, electronic or mechanical, including photocopying, without written permission from the publisher.

Product or company names used in this book are for identification purposes only. Inclusion of the names of the products or companies does not indicate a claim of ownership by IGI of the trademark or registered trademark.

Advanced Topics in Database Research, Volume 5 is a part of the Idea Group Publishing series named *Advanced Topics in Database Research* (Series ISSN 1537-9299).

ISBN 1-59140-935-7
Paperback ISBN 1-59140-936-5
eISBN 1-59140-937-3

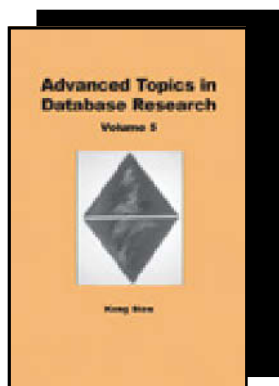
British Cataloguing in Publication Data

A Cataloguing in Publication record for this book is available from the British Library.

All work contributed to this book is new, previously-unpublished material. The views expressed in this book are those of the authors, but not necessarily of the publisher.

Advanced Topics in Database Research Series

ISSN: 1537-9299



Series Editor

Keng Siau

University of Nebraska-Lincoln, USA

Advanced Topics in Database Research, Volume 5
1-59140-935-7 (h/c) • 1-59140-936-5 (s/c) • copyright 2006

Advanced Topics in Database Research, Volume 4
1-59140-471-1 (h/c) • 1-59140-472-X (s/c) • copyright 2005

Advanced Topics in Database Research, Volume 3
1-59140-255-7 (h/c) • 1-59140-296-4 (s/c) • copyright 2004

Advanced Topics in Database Research, Volume 2
1-59140-063-5 (h/c) • copyright 2003

Advanced Topics in Database Research, Volume 1
1-930708-41-6 (h/c) • copyright 2002

Visit us today at www.idea-group.com !



IDEA GROUP PUBLISHING

Hershey • London • Melbourne • Singapore

Advanced Topics in Database Research

Volume 5

Table of Contents

Preface	viii

Section I: Analysis and Evaluation of Database Models

Chapter I

A Rigorous Framework for Model-Driven Development	1

*Liliana Favre, Universidad Nacional del Centro de la Provincia de
Buenos Aires, Argentina*

Chapter II

Adopting Open Source Development Tools in a Commercial Production

Environment: Are We Locked in?	28

*Anna Persson, University of Skövde, Sweden
Henrik Gustavsson, University of Skövde, Sweden
Brian Lings, University of Skövde, Sweden
Björn Lundell, University of Skövde, Sweden
Anders Mattsson, Combitech AB, Sweden
Ulf Ärlig, Combitech AB, Sweden*

Chapter III

Classification as Evaluation: A Framework Tailored for Ontology

Building Methods	41

*Sari Hakkarainen, Norwegian University of Science and Technology,
Norway
Dariusz Strasunskas, Norwegian University of Science and Technology,
Norway, & Vilnius University, Lithuania
Lillian Hella, Norwegian University of Science and Technology, Norway
Stine Tuxen, Bekk Consulting, Norway*

Chapter IV	
Exploring the Concept of Method Rationale: A Conceptual Tool to Understand Method Tailoring	63
<i>Pär J. Ågerfalk, University of Limerick, Ireland</i>	
<i>Brian Fitzgerald, University of Limerick, Ireland</i>	
Chapter V	
Assessing Business Process Modeling Languages Using a Generic Quality Framework	79
<i>Anna Gunhild Nysetvold, Norwegian University of Science and Technology, Norway</i>	
<i>John Krogstie, Norwegian University of Science and Technology, Norway</i>	
Chapter VI	
An Analytical Evaluation of BPMN Using a Semiotic Quality Framework	94
<i>Terje Wahl, Norwegian University of Science and Technology, Norway</i>	
<i>Guttorm Sindre, Norwegian University of Science and Technology, Norway</i>	
Chapter VII	
Objectification of Relationships	106
<i>Terry Halpin, Neumont University, USA</i>	
Chapter VIII	
A Template-Based Analysis of GRL	124
<i>Patrick Heymans, University of Namur, Belgium</i>	
<i>Germain Saval, University of Namur, Belgium</i>	
<i>Gautier Dallons, DECIS SA/NV, Belgium</i>	
<i>Isabelle Pollet, SmalS-MvM/Egov, Belgium</i>	

Section II: Database Designs and Applications

Chapter IX	
Externalisation and Adaptation of Multi-Agent System Behaviour	148
<i>Liang Xiao, Queen's University Belfast, UK</i>	
<i>Des Greer, Queen's University Belfast, UK</i>	
Chapter X	
Reuse of a Repository of Conceptual Schemas in a Large Scale Project	170
<i>Carlo Batini, University of Milano Bicocca, Italy</i>	
<i>Manuel F. Garasi, Italy</i>	
<i>Riccardo Grosso, CSI-Piemonte, Italy</i>	

Chapter XI	
The MAIS Approach to Web Service Design	187
<i>Marzia Adorni, Francesca Arcelli, Carlo Batini, Marco Comerio, Flavio De Paoli, Simone Grega, Paolo Losi, Andrea Maurino, Claudia Raibulet, Francesco Tisato, Università di Milano Bicocca, Italy Danilo Ardagna, Luciano Baresi, Cinzia Cappiello, Marco Comuzzi, Chiara Francalanci, Stefano Modafferi, & Barbara Pernici, Politecnico di Milano, Italy</i>	
Chapter XII	
Toward Autonomic DBMSs: A Self-Configuring Algorithm for DBMS Buffer Pools	205
<i>Patrick Martin, Queen's University, Canada Wendy Powley, Queen's University, Canada Min Zheng, Queen's University, Canada</i>	
Chapter XIII	
Clustering Similar Schema Elements Across Heterogeneous Databases: A First Step in Database Integration	227
<i>Huimin Zhao, University of Wisconsin-Milwaukee, USA Sudha Ram, University of Arizona, USA</i>	
Chapter XIV	
An Efficient Concurrency Control Algorithm for High-Dimensional Index Structures	249
<i>Seok Il Song, Chungju National University, Korea Jae Soo Yoo, Chungbuk National University, Korea</i>	
Section III: Database Design Issues and Solutions	
Chapter XV	
Modeling Fuzzy Information in the IF₂O and Relational Data Models	273
<i>Z. M. Ma, Northeastern University, China</i>	
Chapter XVI	
Evaluating the Performance of Dynamic Database Applications	294
<i>Zhen He, La Trobe University, Australia Jérôme Darmont, Université Lumière Lyon 2, France</i>	
Chapter XVII	
MAMDAS: A Mobile Agent-Based Secure Mobile Data Access System Framework	320
<i>Yu Jiao, Pennsylvania State University, USA Ali R. Hurson, Pennsylvania State University, USA</i>	

Chapter XVIII	
Indexing Regional Objects in High-Dimensional Spaces	348
<i>Byunggu Yu, University of Wyoming, USA</i>	
<i>Ratko Orlandic, University of Illinois at Springfield, USA</i>	

Section IV: Semantic Database Analysis

Chapter XIX	
A Concept-Based Query Language Not Using Proper Association Names	374
<i>Vladimir Ovchinnikov, Lipetsk State Technical University, Russia</i>	

Chapter XX	
Semantic Analytics in Intelligence: Applying Semantic Association	
Discovery to Determine Relevance of Heterogeneous Documents	401
<i>Boanerges Aleman-Meza, University of Georgia, USA</i>	
<i>Amit P. Sheth, University of Georgia, USA</i>	
<i>Devanand Palaniswami, University of Georgia, USA</i>	
<i>Matthew Eavenson, University of Georgia, USA</i>	
<i>I. Budak Arpinar, University of Georgia, USA</i>	

Chapter XXI	
Semantic Integration in Multidatabase Systems: How Much	
Can We Integrate?	420
<i>Te-Wei Wang, University of Illinois, USA</i>	
<i>Kenneth E. Murphy, Willamette University, USA</i>	

About the Editor	440

About the Authors	441

Index	453

Preface

INTRODUCTION

Database management is an integral part of many business applications, especially considering the current business environment that emphasizes data, information, and knowledge as crucial components to the proper utilization and dispensing of an organization's resources. Building upon the work of previous volumes in this book series, we are once again proud to present a collection of high-quality and state-of-the-art research conducted by experts from all around the world .

This book is designed to provide researchers and academics with the latest research-focused chapters on database and database management; these chapters will be insightful and helpful to their current and future research. The book is also designed to serve technical professionals and aims to enhance professional understanding of the capabilities and features of new database applications and upcoming database technologies.

This book is divided into four sections: (I) Analysis and Evaluation of Database Models, (II) Database Designs and Applications, (III) Database Design Issues and Solutions, and (IV) Semantic Database Analysis.

SECTION I: ANALYSIS AND EVALUATION OF DATABASE MODELS

Chapter I, “A Rigorous Framework for Model-Driven Development,” describes a rigorous framework that comprises the NEREUS metamodeling notation, a system of transformation rules to bridge the gap between UML/OCL and NEREUS and, the definition of MDA-based reusable components and model/metamodeling transformations. This chapter also shows how to integrate NEREUS with algebraic languages using the Common Algebraic Specification Language.

Chapter II, “Adopting Open-Source Development Tools in a Commercial Production Environment: Are We Locked in?” explores the use of a standardized interchange format for increased flexibility in a company environment. It also reports on a case study in which a systems development company has explored the possibility of complementing its current proprietary tools with open-source products for supporting its model-based development activities.

Chapter III, “Classification as Evaluation: A Framework Tailored for Ontology Building Methods,” presents a weighted classification approach for ontology-building guidelines. A sample of Web-based ontology-building method guidelines is evaluated in general and experimented with when using data from a case study. It also discusses directions for further refinement of ontology-building methods.

Chapter IV, “Exploring the Concept of Method Rationale: A Conceptual Tool to Understand Method Tailoring,” starts off explaining why systems development methods also encapsulate rationale. It goes on to show how the combination of two different aspects of method rationale can be used to enlighten the communication and apprehension methods in systems development, particularly in the context of tailoring of methods to suit particular development situations.

Chapter V, “Assessing Business Process Modeling Languages Using a Generic Quality Framework,” evaluates a generic framework for assessing the quality of models and modeling languages used in a company. This chapter illustrates the practical utility of the overall framework, where language quality features are looked upon as a means to enable the creation of other models of high quality.

Chapter VI, “An Analytical Evaluation of BPMN Using a Semiotic Quality Framework,” explores the different modeling languages available today. It recognizes that many of them define overlapping concepts and usage areas and consequently make it difficult for organizations to select the most appropriate language related to their needs. It then analytically evaluates the business process modeling notation (BPMN) according to the semiotic quality framework. Its further findings indicate that BPMN is easily learned for simple use, and business process diagrams are relatively easy to understand.

Chapter VII, “Objectification of Relationships,” provides an in-depth analysis of objectification, shedding new light on its fundamental nature, and providing practical guidelines on using objectification to model information systems.

Chapter VIII, “A Template-Based Analysis of GRL,” applies the template proposed by Opdahl and Henderson-Sellers to the goal-oriented Requirements Engineering Language GRL. It then further proposes a metamodel of GRL that identifies the constructs of the language and the links between them. The purpose of this chapter is to improve the quality of goal modeling.

SECTION II: DATABASE DESIGNS AND APPLICATIONS

Chapter IX, “Externalisation and Adaptation of Multi-Agent System Behaviour,” proposes the adaptive agent model (AAM) for agent-oriented system development. It then explains that, in AAM, requirements can be transformed into externalized business rules that represent agent behaviors. Collaboration between agents using these rules can be modeled using extended UML diagrams. An illustrative example is used here to show how AAM is deployed, demonstrating adaptation of inter-agent collaboration, intra-agent behaviors, and agent ontologies.

Chapter X, “Reuse of a Repository of Conceptual Schemas in a Large-Scale Project,” describes a methodology and a tool for the reuse of a repository of conceptual schemas. The methodology described in this chapter is applied in a project where an

existing repository of conceptual schemas, representing information of interest for central public administration, is used in order to produce the corresponding repository of the administrations located in a region.

Chapter XI, “The MAIS Approach to Web Service Design,” presents a first attempt to realize a methodological framework supporting the most relevant phases of the design of a value-added service. The framework has been developed as part of the MAIS project. It describes the MAIS methodological tools available for different phases of service life cycle and discusses the main guidelines driving the implementation of a service management architecture that complies with the MAIS methodological approach.

Chapter XII, “Toward Autonomic DBMSs: A Self-Configuring Algorithm for DBMS Buffer Pools,” introduces autonomic computing as a means to automate the complex tuning, configuration, and optimization tasks that are currently the responsibility of the database administrator.

Chapter XIII, “Clustering Similar Schema Elements Across Heterogeneous Databases: A First Step in Database Integration,” proposes a cluster analysis-based approach to semi-automating the interschema relationship identification process, which is typically very time-consuming and requires extensive human interaction. It also describes a self-organizing map prototype the authors have developed that provides users with a visualization tool for displaying clustering results and for incremental evaluation of potentially similar elements from heterogeneous data sources.

Chapter XIV, “An Efficient Concurrency Control Algorithm for High-Dimensional Index Structures,” introduces a concurrency control algorithm based on link-technique for high-dimensional index structures. This chapter proposes an algorithm that minimizes delay of search operations in high-dimensional index structures. The proposed algorithm also supports concurrency control on reinsert operations in such structures.

SECTION III: DATABASE DESIGN ISSUES AND SOLUTIONS

Chapter XV, “Modeling Fuzzy Information in the IF₂O and Relational Data Models,” examines some conceptual data models used in computer applications in non-traditional area. Based on a fuzzy set and possibility distribution theory, different levels of fuzziness are introduced into IFO data model and the corresponding graphical representations are given. IFO data model is then extended to a fuzzy IFO data model, denoted IF₂O. This chapter also provides the approach to mapping an IF₂O model to a fuzzy relational database schema.

Chapter XVI, “Evaluating the Performance of Dynamic Database Applications,” explores the effect that changing access patterns has on the performance of database management systems. The studies indicate that all existing benchmarks or evaluation frameworks produce static access patterns in which objects are always accessed in the same order repeatedly. The authors in this chapter instantiate the Dynamic Evaluation Framework, which simulates access pattern changes using configurable styles of change, into the Dynamic object Evaluation Framework that is designed for object databases.

Chapter XVII, “MAMDAS: A Mobile Agent-Based Secure Mobile Data Access System Framework,” recognizes that creating a global information-sharing environment in the presence of autonomy and heterogeneity of data sources is a difficult task.

The constraints on bandwidth, connectivity, and resources worsen the problem when adding mobility and wireless medium to the mix. The authors in this chapter designed and prototyped a mobile agent-based secure mobile data access system (MAMDAS) framework for information retrieval in large and heterogeneous databases. They also proposed a security architecture for MAMDAS to address the issues of information security.

Chapter XVIII, “Indexing Regional Objects in High-Dimensional Spaces,” reviews the problems of contemporary spatial access methods in spaces with many dimensions and presents an efficient approach to building advanced spatial access methods that effectively attack these problems. It also discusses the importance of high-dimensional spatial access methods for the emerging database applications.

SECTION IV: SEMANTIC DATABASE ANALYSIS

Chapter XIX, “A Concept-Based Query Language Not Using Proper Association Names,” focuses on a concept-based query language that permits querying by means of application domain concepts only. It introduces constructions of closures and contexts as applied to the language which permit querying some indirectly associated concepts as if they were associated directly and adopting queries to users’ needs without rewriting. The author of this chapter believes that the proposed language opens new ways of solving tasks of semantic human-computer interaction and semantic data integration.

Chapter XX, “Semantic Analytics in Intelligence: Applying Semantic Association Discovery to Determine Relevance of Heterogeneous Documents,” describes an ontological approach for determining the relevance of documents based on the underlying concept of exploiting complex semantic relationships among real-world entities. This chapter builds upon semantic metadata extraction and annotation, practical domain-specific ontology creation, main-memory query processing, and the notion of semantic association. It also discusses how a commercial product using Semantic Web technology, Semagix Freedom, is used for metadata extraction when designing and populating an ontology from heterogeneous sources.

Chapter XXI, “Semantic Integration in Multidatabase Systems: How Much Can We Integrate?” reviews the semantic integration issues in multidatabase development and provides a standardized representation for classifying semantic conflicts. It then explores the idea further by examining semantic conflicts and proposes taxonomy to classify semantic conflicts in different groups.

These 21 chapters provide a sample of the cutting edge research in all facets of the database field. This volume aims to be a valuable resource for scholars and practitioners alike, providing easy access to excellent chapters which address the latest research issues in this field.

Keng Siau
University of Nebraska-Lincoln, USA
January 2006

Section I:

Analysis and Evaluation of Database Models

Chapter I

A Rigorous Framework for Model-Driven Development

Liliana Favre,
Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina

ABSTRACT

The model-driven architecture (MDA) is an approach to model-centric software development. The concepts of models, metamodels, and model transformations are at the core of MDA. Model-driven development (MDD) distinguishes different kinds of models: the computation-independent model (CIM), the platform-independent model (PIM), and the platform-specific model (PSM). Model transformation is the process of converting one model into another model of the same system, preserving some kind of equivalence relation between them. One of the key concepts behind MDD is that models generated during software developments are represented using common metamodeling techniques. In this chapter, we analyze an integration of MDA metamodeling techniques with knowledge developed by the community of formal methods. We describe a rigorous framework that comprises the NEREUS metamodeling notation (open to many other formal languages), a system of transformation rules to bridge the gap between UML/OCL and NEREUS, the definition of MDA-based reusable components, and model/metamodeling transformations. In particular, we show how to integrate NEREUS with

algebraic languages using the Common Algebraic Specification Language (CASL). NEREUS focuses on interoperability of formal languages in MDD.

INTRODUCTION

The model-driven architecture (MDA) is an initiative of the Object Management Group (OMG, www.omg.org), which is facing a paradigm shift from object-oriented software development to model-centric development. It is emerging as a technical framework to improve portability, interoperability, and reusability (MDA, www.omg.org/docs/omg/03-06-01.pdf). MDA promotes the use of models and model-to-model transformations for developing software systems. All artifacts, such as requirement specifications, architecture descriptions, design descriptions, and code, are regarded as models and are represented using common modeling languages. MDA distinguishes different kinds of models: the computation-independent model (CIM), the platform-independent model (PIM), and the platform-specific model (PSM). Unified Modeling Language (UML, www.uml.org) combined with Object Constraint Language (OCL, www.omg.org/cgi-bin/doc?ptc/2003-10-14) is the most widely used way to specify PIMs and PSMs.

A model-driven development (MDD) is carried out as a sequence of model transformations. Model transformation is the process of converting one model into another model of the same system, preserving some kind of equivalence relation between them. The high-level models that are developed independently of a particular platform are gradually transformed into models and code for specific platforms.

One of the key concepts behind MDA is that all artifacts generated during software developments are represented using common metamodeling techniques. Metamodels in the context of MDA are expressed using meta object facility (MOF) (www.omg.org/mof). The integration of UML 2.0 with the OMG MOF standards provides support for MDA tool interoperability (www.uml.org). However, the existing MDA-based tools do not provide sophisticated transformations because many of the MDA standards are recent or still in development (CASE, www.omg.org/cgi-bin/doc?ad/2001-02-01). For instance, OMG is working on the definition of a query, view, transformations (QVT) metamodel, and to date there is no way to define transformations between MOF models (<http://www.sce.carleton.ca/courses/sysc-4805/w06/courseinfo/OMdocs/MOF-QVT-ptc-05-11-01.pdf>). There is currently no precise foundation for specifying model-to-model transformations.

MDDs can be improved by means of other metamodeling techniques. In particular, in this chapter, we analyze the integration of MDA with knowledge developed by the formal method community. If MDA becomes a commonplace, adapting it to formal development will become crucial. MDA can take advantage of the different formal languages and the diversity of tools developed for prototyping, model validations, and model simulations. Currently, there is no way to integrate semantically formal languages and their related tools with MDA. In this direction, we define a framework that focuses on interoperability of formal languages in MDD. The framework comprises:

- The metamodeling notation NEREUS;
- A “megamodel” for defining MDA-based reusable components;
- A bridge between UML/OCL and NEREUS; and
- Bridges between NEREUS and formal languages.

Considering that different modeling/programming languages could be used to specify different kinds of models (PIMs, PSMs, and code models) and different tools could be used to validate or verify them, we propose to use the NEREUS language, which is a formal notation suited for specifying UML-based metamodels. NEREUS can be viewed as an intermediate notation open to many other formal specifications, such as algebraic, functional, or logic ones.

The “megamodel” defines reusable components that fit with the MDA approach. A “megamodel” is a set of elements that represent and/or refer to models and metamodel (Bezivin, Jouault, & Valduriez, 2004). Metamodels that describe instances of PIMs, PSMs, and code models are defined at different abstraction levels and structured by different relationships. The “megamodel” has two views, one of them in UML/OCL and the other in NEREUS.

We define a bridge between UML/OCL and NEREUS consisting of a system of transformation rules to convert automatically UML/OCL metamodels into NEREUS specifications. We also formalize model/metamodel transformations among levels of PIMs, PSMs, and implementations.

A bridge between NEREUS and algebraic languages was defined by using the common algebraic specification language (CASL) (Bidoit & Mosses, 2004), that has been designed as a general-purpose algebraic specification language and subsumes many existing formal languages.

Rather than requiring developers to manipulate formal specifications, we want to provide rigorous foundations for MDD in order to develop tools that, on one hand, take advantage of the power of formal languages and, on the other hand, allow developers to directly manipulate the UML/OCL models that they have created.

This chapter is structured as follows. We first provide some background information and related work. The second section describes how to formalize UML-based metamodels in the intermediate notation NEREUS. Next, we introduce a “megamodel” to define reusable components in a way that fits MDA. Then, we show how to bridge the gap between UML/OCL and NEREUS. An integration of NEREUS with CASL is then described. Next, we compare our approach with other existing ones, and then discuss future trends in the context of MDA. Finally, conclusions are presented.

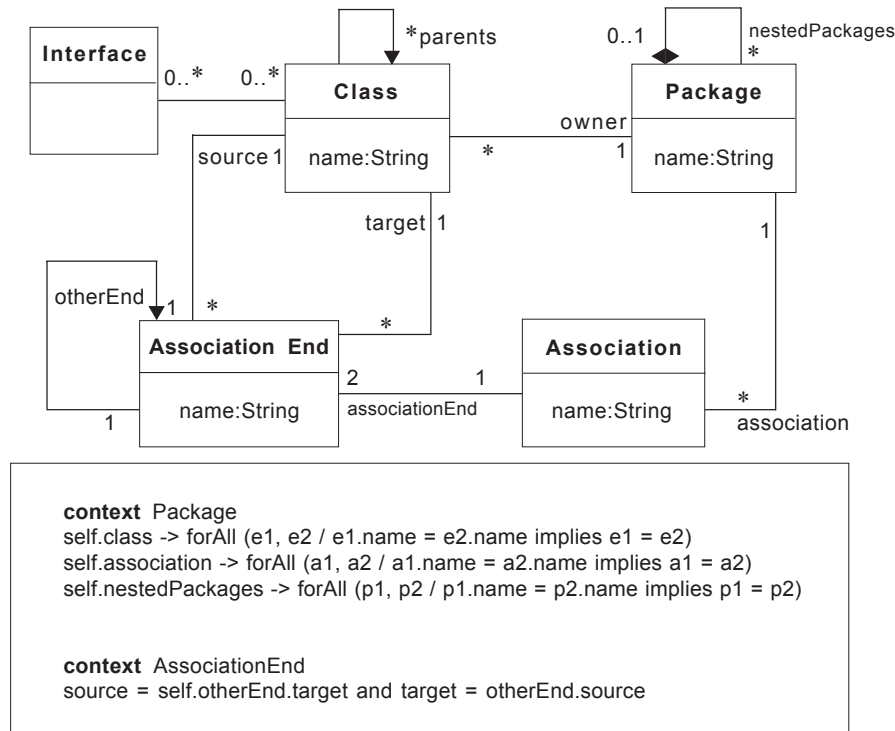
BACKGROUND

The Model-Driven Architecture

MDA distinguishes different kinds of models: the computation-independent model (CIM), the platform-independent model (PIM), the platform-specific model (PSM), and code models. A CIM describes a system from the computation-independent viewpoint that focuses on the environment of and the requirements for the system. In general, it is called a domain model and may be expressed using business models. A PIM is a model that contains no reference to the platforms that are used to realize it. A PSM describes a system in the terms of the final implementation platform, for example, .NET or J2EE. UML combined with OCL is the most widely used way of writing either PIMs or PSMs.

The transformation for one PIM to several PSMs is at the core of MDA. A model-driven development is carried out as a sequence of model transformations that includes,

Figure 1. A simplified UML metamodel



at least, the following steps: construct a CIM; transform the CIM into a PIM that provides a computing architecture independent of specific platforms; transform the PIM into one or more PSMs, and derive code directly from the PSMs (Kleppe, Warmer, & Bast, 2003).

Metamodeling has become an essential technique in model-centric software development. The UML itself is defined using a metamodeling approach. The metamodeling framework for the UML is based on an architecture with four layers: meta-metamodel, metamodel, model, and user objects. A model is expressed in the language of one specific metamodel. A metamodel is an explicit model of the constructs and rules needed to construct specific models. A meta-metamodel defines a language to write metamodels. The meta-metamodel is usually self-defined using a reflexive definition and is based on at least three concepts (entity, association, and package) and a set of primitive types. Languages for expressing UML-based metamodels are based on UML class diagrams and OCL constraints to rule out illegal models.

Related OMG standard metamodels and meta-metamodels such as Meta Object Facility (MOF) (www.omg.org/mof), software process engineering metamodel (SPEM, www.omg.org/technology/documents/formal/spem.htm), and common warehouse metamodel (CWM) (www.omg.org/cgi-bin/doc?ad/2001-02-01) share a common design

philosophy. Metamodels in the context of MDA are expressed using MOF. It defines a common way for capturing all the diversity of modeling standards and interchange constructs that are used in MDA. Its goal is to define languages in a same way and then integrate them semantically. MOF and the core of the UML metamodel are closely aligned with their modeling concepts. The UML metamodel can be viewed as an “instance of” the MOF metamodel. OMG is working on the definition of a query, view, transformations (QVT) metamodel for expressing transformations as an extension of MOF.

Figure 1 depicts a “toy” metamodel that includes the core modeling concepts of the UML class diagrams, including classes, interfaces, associations, association-ends, and packages. As an example, Figure 1 shows some OCL constraints that also complement the class diagram.

MDA-Based Tools

There are at least 100 UML CASE tools that differ widely in functionality, usability, performance, and platforms. Currently, about 10% of them provide some support for MDA. Examples of these tools include OptimalJ, ArcStyler, AndroMDA, Ameos, and Codagen, among others. The tool market around MDA is still in flux. References to MDA-based tools can be found at www.objectsbydesign.com/tools. As an example, OptimalJ is an MDA-based environment to generate J2EE applications. OptimalJ distinguishes three kinds of models: a domain model that correspond to a PIM model, an application model that includes PSMs linked to different platforms (Relational-PSM, EJB-PSM and web-PSM), and an implementation model. The transformation process is supported by transformation and functional patterns. OptimalJ allows the generation of PSMs from a PIM and a partial code generation.

UML CASE tools provide limited facilities for refactoring on source code through an explicit selection made for the designer. However, it will be worth thinking about refactoring at the design level. The advantage of refactoring at the UML level is that the transformations do not have to be tied to the syntax of a programming language. This is relevant since UML is designed to serve as a basis for code generation with the MDA approach (Sunnyé, Pollet, Le Traon, & Jezequel, 2001).

Many UML CASE tools support reverse engineering; however, they only use more basic notational features with a direct code representation and produce very large diagrams. Reverse engineering processes are not integrated with MDDs either.

Techniques that currently exist in UML CASE tools provide little support for validating models in the design stages. Reasoning about models of systems is well supported by automated theorem provers and model checkers; however, these tools are not integrated into CASE tools environments.

A discussion of limitations of the forward engineering processes supported by the existing UML CASE tools may be found in Favre, Martinez, and Pereira (2003, 2005).

The MDA-based tools use MOF to support OMG standards such as UML and XML metadata interchange (XMI). MOF has a central role in MDA as a common standard to integrate all different kinds of models and metadata and to exchange these models among tools. However, MOF does not allow the capture of semantic properties in a platform-independent way, and there are no rigorous foundations for specifying transformations among different kinds of models.

MDA and Semi-Formal/Formal Modeling Techniques

Various research analyzed the integration of semiformal techniques and object-oriented designs with formal techniques. It is difficult to compare the existing results and to see how to integrate them in order to define standard semantics since they specify different UML subsets and are based on different formalisms. Next, we mention only some of numerous existing works. U2B transforms UML models to B (Snook & Butler, 2002). Kim and Carrington (2002) formalize UML by using OBJECT-Z. Reggio, Cerioli, and Astesiano (2001) present a general framework of the semantics of UML, where the different kinds of diagrams within a UML model are given individual semantics and then such semantics are composed to get the semantics on the overall model. McUmbert and Cheng (2001) propose a general framework for formalizing UML diagrams in terms of different formal languages using a mapping from UML metamodels and formal languages. Kuske, Gogolla, Kollmann, and Kreowski (2002) describe an integrated semantics for UML class, object, and state diagrams based on graph transformation.

UML CASE tools could be enhanced with functionality for formal specification and deductive verification; however, only research tools provide support for advanced analysis. For example, the main task of USE tool (Gogolla, Bohling, & Ritchers, 2005) is to validate and verify specifications consisting of UML/OCL class diagrams. Key (Ahrendt et al., 2005) is a tool based on Together (CASE, www.omg.org/cgi-bin/doc?ad/2001-02-01) enhanced with functionality for formal specification and deductive verification.

To date, model-driven approaches have been discussed at several workshops (Abmann, 2004; Evans, Sammut, & Willans, 2003; Gogolla, Sammut, & Whittle, 2004). Several metamodeling approaches and model transformations have been proposed to MDD (Atkinson & Kuhne, 2002; Bezivin, Farcet, Jezequel, Langlois, & Pollet, 2003; Buttner & Gogolla, 2004; Caplat & Sourrouille, 2002; Cariou, Marvie, Seinturier and Duchien, 2004; Favre, 2004; Gogolla, Lindow, Richters, & Ziemann, 2002; Kim & Carrington, 2002).

Akehurst and Kent (2002) propose an approach that uses metamodeling patterns that capture the essence of mathematical relations. The proposed technique is to adopt a pattern that models a transformation relationship as a relation or collections of relations, and encode this as an object model. Hausmann (2003) defined an extension of a metamodeling language to specify mappings between metamodels based on concepts presented in Akehurst and Kent (2002). Kuster, Sendall, and Wahler (2004) compare and contrast two approaches to model transformations: one is graph transformation and the other is a relational approach. Czarnecki and Helsen (2003) describe a taxonomy with a feature model to compare several existing and proposed model-to-model transformation approaches. To date, there is no way to integrate semantically formal languages and their related tools with Model-Driven Development.

FORMALIZING METAMODELS: THE NEREUS LANGUAGE

A combination of formal specifications and metamodeling techniques can help us to address MDA. A formal specification clarifies the intended meaning of metamodel/

models, helps to validate model transformations, and provides reference for implementation. In this light, we propose the intermediate notation NEREUS that focuses on interoperability of formal languages. It is suited for specifying metamodels based on the concepts of entity, associations, and systems. Most of the UML concepts for the metamodels can be mapped to NEREUS in a straightforward manner. NEREUS is relation-centric; that is, it expresses different kinds of relations (dependency, association, aggregation, composition) as primitives to develop specifications.

Defining Classes in NEREUS

In NEREUS the basic unit of specification is the class. Classes may declare types, operations, and axioms that are formulas of first-order logic. They are structured by three different kinds of relations: importing, inheritance, and subtyping. Figure 2 shows its syntax.

NEREUS distinguishes variable parts in a specification by means of explicit parameterization. The elements of `<parameterList>` are pairs `C1:C2` where `C1` is the formal generic parameter constrained by an existing class `C2` (only subclasses of `C2` will be actual parameters). The `IMPORTS` clause expresses clientship relations. The specification of the new class is based on the imported specifications declared in `<importList>` and their public operations may be used in the new specification.

NEREUS distinguishes inheritance from subtyping. Subtyping is like inheritance of behavior, while inheritance relies on the module viewpoint of classes. Inheritance is expressed in the `INHERITS` clause; the specification of the class is built from the union of the specifications of the classes appearing in the `<inheritsList>`. Subtypings are declared in the `IS-SUBTYPE-OF` clause. A notion closely related with subtyping is polymorphism, which satisfies the property that each object of a subclass is at the same time an object of its superclasses.

NEREUS allows us to define local instances of a class in the `IMPORTS` and `INHERITS` clauses by the following syntax `ClassName [<bindingList>]` where the elements of `<bindingList>` can be pairs of class names `C1: C2` being `C2` a component of `ClassName`; pairs of sorts `s1: s2`, and/or pairs of operations `o1: o2` with `o2` and `s2` belonging to the own part of `ClassName`.

NEREUS distinguishes deferred and effective parts. The `DEFERRED` clause declares new types or operations that are incompletely defined. The `EFFECTIVE` clause either declares new types or operations that are completely defined or completes the definition of some inherited type or operation.

Figure 2. Class syntax in NEREUS

CLASS <code>className</code> [<code><parameterList></code>] IMPORTS <code><importList></code> INHERITS <code><inheritsList></code> IS-SUBTYPE-OF <code><subtypingList></code> GENERATED-BY <code><basicConstructors></code> ASSOCIATES <code><associatesList></code> DEFERRED TYPES <code><typesList></code>	FUNCTIONS <code><functionList></code> EFFECTIVE TYPES <code><typesList></code> FUNCTIONS <code><functionList></code> AXIOMS <code><varList></code> <code><axiomList></code> END-CLASS

Figure 3. The collection class

<pre> CLASS Collection [Elem:ANY] IMPORTS Boolean, Nat GENERATED-BY create, add DEFERRED TYPE Collection FUNCTIONS create : → Collection add : Collection x Elem → Collection count : Collection x Elem → Nat iterate : Collection x (Elem x Acc: ANY) x (-> Acc) -> Acc EFFECTIVE FUNCTIONS isEmpty: Collection -> Boolean size: Collection → Nat includes: Collection x Elem -> Boolean includesAll: Collection x Collection -> Boolean excludes: Collection x Elem -> Boolean forAll : Collection x (Elem -> Boolean) -> Boolean exists : Collection x (Elem -> Boolean) -> Boolean select: Collection x (Elem -> Boolean) -> Collection </pre>	<pre> AXIOMS c : Collection; e : Elem; f : Elem -> Boolean; g : Elem x Acc -> Acc; base : -> Acc isEmpty (c) = (size (c) = 0) iterate (create, g, base) = base iterate (add (c, e), g, base) = g (e, iterate (c, g, base)) count (c,e) = LET FUNCTIONS f1: Elem x Nat -> Nat AXIOMS e1:Elem; i:Nat f1(e1, i) = if e = e1 then i+1 else i IN iterate (c, f1, 0) END-LET includes (create , e) = False includes (add (c, e), e1) = if e = e1 then True else includes (c, e1) forAll (create , f) = True forAll (add(c,e), f) = f (e) and forAll (c, f) exists (create, f) = False exists (add (c, e)) = f (e) or exists (c, f) select (create, f) = create select (add (c,e), f) = if f(e) then add (select(c,f),e) else select (c, f)... END-CLASS </pre>

Operations are declared in the **FUNCTIONS** clause that introduces the operation signatures, the list of their arguments, and result types. They can be declared as total or partial. Partial functions must specify its domain by means of the **PRE** clause that indicates what conditions the function's arguments must satisfy to belong to the function's domain. NEREUS allows us to specify operation signatures in an incomplete way. NEREUS supports higher order operations (a function f is higher order if functional sorts appear in a parameter sort or the result sort of f). In the context of OCL Collection formalization, second-order operations are required. In NEREUS, it is possible to specify any of the three levels of visibility for operations: public, protected, and private. NEREUS provides the construction **LET... IN** to limit the scope of the declarations of auxiliary symbols by using local definitions.

Several useful predefined types are offered in NEREUS, for example, Collection, Set, Sequence, Bag, Boolean, String, Nat, and enumerated types. Figure 3 shows the predefined type OCL-Collection.

Defining Associations

NEREUS provides a taxonomy of constructor types that classifies binary associations according to kind (aggregation, composition, association, association class, qualified association), degree (unary, binary), navigability (unidirectional, bidirectional), and connectivity (one-to-one, one-to-many, many-to-many). Figure 4 partially depicts the hierarchy of Binary Associations.

Figure 4. The binary association hierarchy

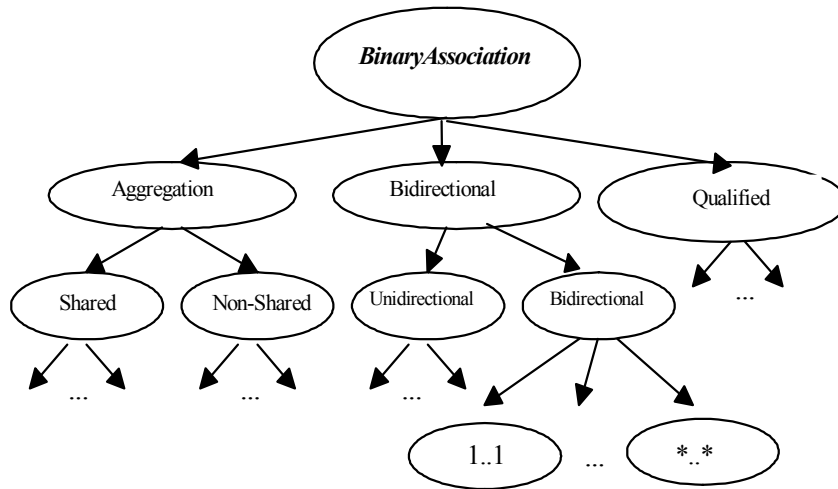


Figure 5. Association syntax in NEREUS

```

ASSOCIATION <relationName>
IS <constructorTypeName> [...: Class1; ...: Class2; ...: Role1; ...: Role2;
...: mult1; ...: mult2; ...: visibility1; ...: visibility2]
CONSTRAINED-BY <constraintList>
END

```

Figure 6. Package syntax

```

PACKAGE packageName
IMPORTS <importsList>
INHERITS <inheritsList>
<elements>
END-PACKAGE

```

Generic relations can be used in the definition of concrete relations by instantiation. New associations can be defined by means of the syntax shown in Figure 5.

The IS paragraph expresses the instantiation of <constructorTypeName> with classes, roles, visibility, and multiplicity. The CONSTRAINED-BY clause allows the specification of static constraints in first-order logic. Relations are defined in a class by means of the ASSOCIATES clause.

Figure 7. A simplified UML metamodel in NEREUS

<pre> PACKAGE Core CLASS TheClass ASSOCIATES <<ClassPackage>>, << ClassClass >>, << SourceAssociationEnd >>, <<TargetAssociationEnd>>, <<ClassInterface>>,... TYPES TheClass FUNCTIONS name: TheClass -> String ... END-CLASS CLASS ThePackage ASSOCIATES <<PackagePackage>>, <<ClassPackage>>, << PackageAssociation >>... TYPE ThePackage FUNCTIONS name: ThePackage -> String ... END-CLASS CLASS TheAssociation ASSOCIATES <<PackageAssociation>>, <<AssociationAssociationEnd>> TYPES TheAssociation FUNCTIONS name: TheAssociation -> String ... END-CLASS CLASS TheAssociationEnd ASSOCIATES <<AssociationAssociationEnd>>, <<AssociationEndAssociationEnd>>, <<SourceAssociationEnd>>, <<TargetAssociationEnd>>, ... END-CLASS </pre>	<pre> CLASS TheInterface ASSOCIATES <<ClassInterface>>,... END-CLASS ASSOCIATION PackagePackage IS Composition-2 [ThePackage: Class1; ThePackage : Class2; thepackage :Role1; nestedPackages: Role2; 0..1: mult1; *: mult2; +: visibility1; +: visibility2] END ASSOCIATION ClassPackage IS Bidirectional-2 [TheClass: Class1; ThePackage: Class2; theClass: role1; owner: role2; *: mult1; 1: mult2; +: visibility1; +: visibility2] END ASSOCIATION ClassClass IS Unidirectional-3 [TheClass: Class1; TheClass: Class2; theClass: role1; parents: role2; *: mult1; *: mult2; +: visibility1; +: visibility2] END ASSOCIATION ClassInterface IS Bidirectional-4 [TheClass: Class1; TheInterface: Class2; theClass:role1; implementedInt: role2; 0..*: mult1; 0..*: mult2; +: visibility1; +: visibility2] END ASSOCIATION SourceAssociationEnd ... ASSOCIATION TargetAssociationEnd ... ASSOCIATION PackageAssociation ... ASSOCIATION AssociationEndAssociationEnd ... END-PACKAGE </pre>

Defining Packages

The package is the mechanism provided by NEREUS for grouping classes and associations and controlling its visibility. Figure 6 shows the syntax of a package.

<importsList> lists the imported packages; <inheritList> lists the inherited packages and <elements> are classes, associations, and packages. Figure 7 partially shows the NEREUS specification of Figure 1.

DEFINING REUSABLE COMPONENTS: A “MEGAMODEL”

Developing reusable components requires a high focus on software quality. The traditional techniques for verification and validation are still essential to achieve software quality. The formal specifications are of particular importance for supporting

testing of applications, for reasoning about correctness and robustness of models, for checking the validity of a transformation and for generating code “automatically” from abstract models. MDA can take advantages of formal languages and the tools developed around them. In this direction, we propose a “megamodel” to define MDA reusable components. A “megamodel” is a set of elements that represent and/or refer to models and metamodels at different levels of abstraction and structured by different relationships (Bezivin, Jouault, & Valduriez, 2004). It relates PIMs, PSMs, and code with their respective metamodels specified both in UML/OCL and NEREUS. NEREUS represents the transient stage in the process of conversion of UML/OCL specifications to different formal specifications.

We define MDA components at three different levels of abstraction: platform-independent component model (PICM), platform-specific component model (PSCM), and implementation component model (ICM). The PICM includes a UML/OCL metamodel that describes a family of all those PIMs that are instances of the metamodel. A PIM is a model that contains no information of the platform that is used to realize it. A platform is defined as “a set of subsystems and technologies that provide a coherent set of functionality, which any application supported by that platform can use without concern for the details of how the functionality is implemented” (www.omg.org/docs/omg/03-06-01.pdf, p.2.3).

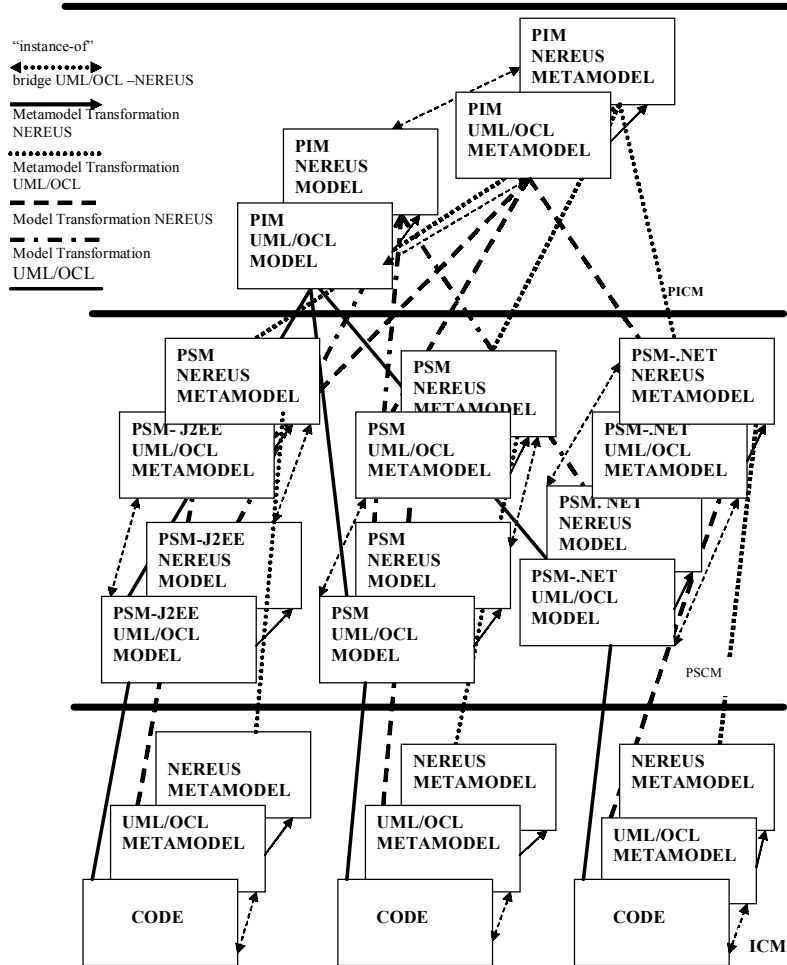
A PICM-metamodel is related to more than one PSCM-metamodel, each one suited for different platforms. The PSCM metamodels are specializations of the PICM-metamodel. The PSCM includes UML/OCL metamodels that are linked to specific platforms and a family of PSMs that are instances of the respective PSCM-metamodel. Every one of them describes a family of PSM instances. PSCM-metamodels correspond to ICM-metamodels. Figure 8 shows the different correspondences that may be held between several models and metamodels. A “megamodel” is based on two views, one of them in UML/OCL and the other in NEREUS. A metamodel is a description of all the concepts that can be used in the respective level (PICM, PSCM, and ICM). The concepts of attribute, operations, classes, associations, and packages are included in the PIM-metamodel. PSM-metamodels constrain a PIM-metamodel to fit a specific platform, for instance, a metamodel linked to a relational platform refers to the concepts of table, foreign key, and column. The ICM-metamodel includes concepts of programming languages such as constructor and method.

A model transformation is a specification of a mechanism to convert the elements of a model that are instances of a particular metamodel into elements of another model, which can be instances of the same or different metamodel. A metamodel transformation is a specific type of model transformations that impose relations between pairs of metamodels. We define a bridge between UML/OCL and NEREUS. For a subsequent translation into formal languages, NEREUS may serve as a source language. In the following sections, we describe how to bridge the gap between NEREUS and formal languages. In particular, we analyze how to translate NEREUS into CASL.

A BRIDGE BETWEEN UML AND NEREUS

We define a bridge between UML/OCL static models and NEREUS. A detailed analysis may be found in Favre (2005a). The text of the NEREUS specification is

Figure 8. A “megamodel” for MDA



completed gradually. First, the signature and some axioms of classes are obtained by instantiating the reusable schemes `BOX_` and `ASSOCIATION_`. Next, OCL specifications are transformed using a set of transformation rules. Then, a specification that reflects all the information of UML models is constructed. Figure 9 depicts the main steps of this translation process.

Figure 10 shows the reusable schemes `BOX_` and `ASSOCIATION_`. In `BOX_`, the attribute mapping requires two operations: an access operation and a modifier. The access operation takes no arguments and returns the object to which the receiver is mapped. The modifier takes one argument and changes the mapping of the receiver to that argument. In NEREUS, no standard convention exists, but frequently we use names such as `get_` and `set_` for them. Association specification is constructed by instantiating the scheme `ASSOCIATION_`.

Figure 9. From UML/OCL to NEREUS

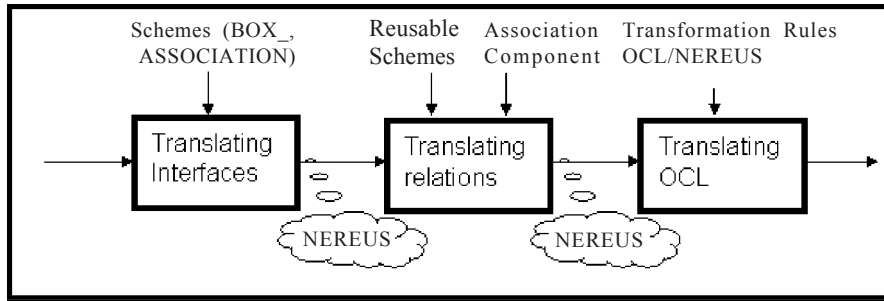


Figure 10. The reusable schemes BOX_ and ASSOCIATION_

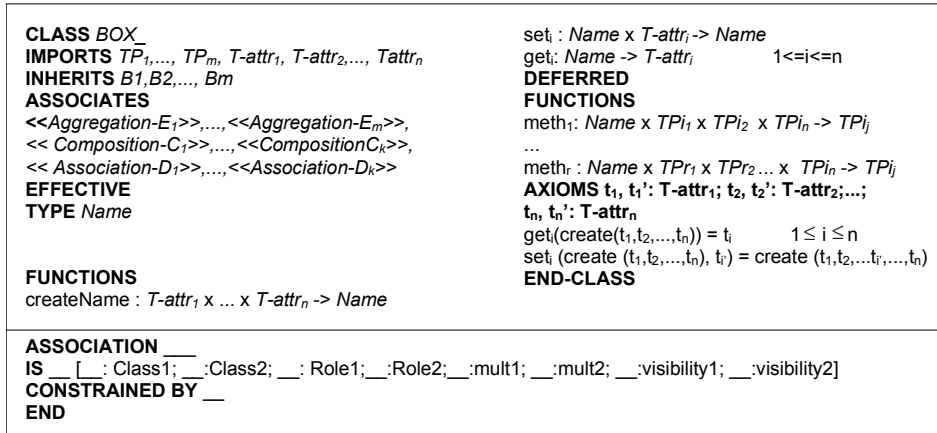


Figure 11. The package P&M

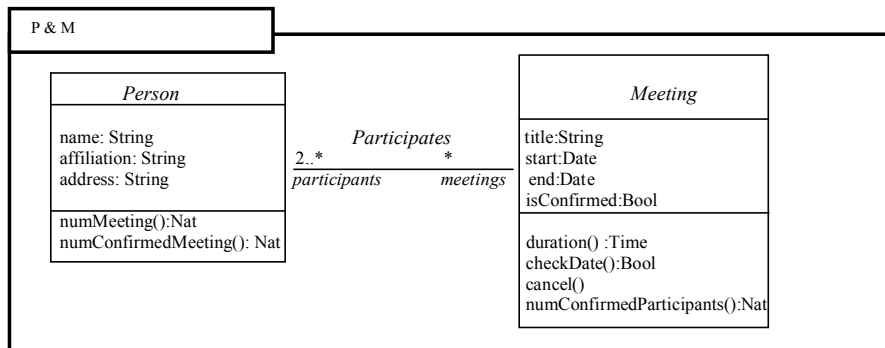


Figure 12. The package P&M: Translating interfaces and relations into NEREUS

<pre> PACKAGE P&M CLASS Person IMPORTS String, Nat ASSOCIATES <<Participates>> EFFECTIVE TYPES Person GENERATED-BY createPerson FUNCTIONS createPerson: String x String x String -> Person name: Person -> String affiliation: Person -> String address: Person -> String set-name: Person x String -> Person set-affiliation : Person x String -> Person set-address: Person x String -> Person AXIOMS p:Person; m: Meeting; s, s1, s2, s3: String; pa: Participates name(createPerson(s1,s2, s3)) = s1 affiliation (createPerson (s1, s2, s3)) = s2 address (createPerson (s1, s2, s3)) = s3 set-name (createPerson (s1, s2, s3), s) = createPerson (s,s2,s3)) set-affiliation (createPerson(s1,s2, s3), s) = createPerson (s1, s, s3)) ... END-CLASS CLASS Meeting IMPORTS String, Date, Boolean, Time ASSOCIATES <<Participates>> </pre>	<pre> EFFECTIVE TYPES Meeting GENERATED-BY createMeeting FUNCTIONS createMeeting: String x Date x Date x Boolean -> Meeting title: Meeting -> String start : Meeting -> Date end : Meeting -> Date isConfirmed : Meeting -> Boolean set-title: Meeting x String -> Meeting set-start : Meeting x Date -> Meeting set-end: Meeting x Date -> Meeting set-isConfirmed: Meeting x Boolean -> Boolean AXIOMS s: String; d, d1,: Date; b:Boolean;... title (createMeeting (s, d, d1, b)) = s start (createMeeting (s, d, d1, b)) = d end (createMeeting (s, d, d1, b)) = d1 isConfirmed (createMeeting (s, d, d1, b)) = b ... END-CLASS ASSOCIATION Participates IS Bidirectional-Set [Person: Class1; Meeting: Class2; participates: Role1; meetings: Role2; *: mult1; * : mult2; + : visibility1; +: visibility2] END END-PACKAGE </pre>

Figure 11 shows a simple class diagram P&M in UML. P&M introduces two classes (Person and Meeting) and a bidirectional association between them. This example was analyzed by Hussmann, Cerioli, Reggio, and Tort (1999), Padawitz (2000), and Favre (2005a). We have meetings in which persons may participate. The NEREUS specification of Figure 12 is built by instantiating the scheme BOX_ and the scheme ASSOCIATION_ (see Figure 10).

The transformation process of OCL specifications to NEREUS is supported by a system of transformation rules. Figure 13 shows how to translate some OCL expressions into NEREUS.

By analyzing OCL specifications, we can derive axioms that will be included in the NEREUS specifications. Preconditions written in OCL are used to generate preconditions in NEREUS. Postconditions and invariants allow us to generate axioms in NEREUS. Figure 14 shows how to map OCL specifications of P&M onto NEREUS.

An operation can be specified in OCL by means of pre- and post-conditions. *self* can be used in the expression to refer to the object on which the operation was called, and the name *result* is the name of the returned object, if there is any. The names of the parameter (parameter1,...) can also be used in the expression. In a postcondition, the

Figure 13. Transforming OCL into NEREUS: A system of transformation rules

OCL	NEREUS
v (variable)	v (variable)
Type-> operationName (parameter1: Type1,...): Rtype	operationName : TypexType1x...-> Rtype
v. operation (v')	operation (v, v')
v->operation (v')	operation (v, v')
v.attribute	attribute (v)
context A object.rolename	get_ rolename (a, object) <i>Let a:A</i>
OCLexp1 = OCLexp2	Translate _{NEREUS} (OCLexp1) = Translate _{NEREUS} (OCLexp2)
e.op	op (Translate _{NEREUS} (e)) <i>Let Translate_{NEREUS} be functions that translate logical expressions of OCL into first-order formulae in NEREUS.</i>
collection-> op (v:Elem boolean-expr-with-v) op ::=select forAll reject exists	LET FUNCTIONS f: Elem -> Boolean AXIOMS v : Elem f (v)= Translate _{NEREUS} (boolean-expr-with-v) IN op (collection, f) END-LET ----- op _v (collection, [Translate _{NEREUS} (boolean-expr-with-v)] <i>Equivalent concise notation</i>

expression can refer to two sets of values for each property of an object: the value of a property at the start of the operation and the value of a property upon completion of the operation. To refer to the value of a property at the start of the operation, one has to postfix the property name with “@”, followed by the keyword “pre”. For example, the following OCL specification:

```
AddPerson (p:Person)
pre: not meetings -> includes(p)
post: meetings = meetings@pre -> including(p)
```

is translated into:

```
AddPerson: Participates (a) x Person (p) -> Participates
pre: not includes(getMeetings(a), p)
AXIOMS a: Participates; p:Person;....
getMeetings(AddPerson(a,p)) =including(getMeetings(a), p)
```


Figure 14. The package P&M: Transforming OCL contracts into NEREUS

OCL	
context Meeting:: checkDate():Bool post: result = self.participants->collect(meetings) ->forall(m m<> self and m.isConfirmed implies (after(self.end,m.start) or after(m.end,self.start)))	
context Meeting::isConfirmed () post: result= self.checkdate() and self.numConfirmedParticipants > = 2	
context Person:: numMeeting () : Nat post: result = self.meetings -> size	
context Person :: numConfirmedMeeting () : Nat post: result= self.meetings -> select (isConfirmed) -> size	
RULES	
Rule 1 $T \rightarrow \text{Op} (<\text{parameterList}>) : \text{ReturnType}$ post: <i>expr</i>	AXIOMS $t : T, \dots$ $\text{Translate}_{\text{NEREUS}}(\text{exp})$
Rule 2 $T \rightarrow \text{forall op} (v:\text{Type} \mid \text{bool-expr-with-v})$ $\text{op} ::= \text{exists} \mid \text{select} \mid \text{reject}$	$\text{forall op} (\text{Translate}_{\text{NEREUS}}(T), \text{Translate}_{\text{NEREUS}}(\text{bool-expr-with-v}))$
Rule 3 $T \rightarrow \text{collect} (v:\text{type} \mid v.\text{property})$	$\text{collect}_v (\text{Translate}_{\text{NEREUS}}(T), \text{Translate}_{\text{NEREUS}}(v.\text{property}))$
NEREUS	
CLASS Person... AXIOMS p:Person; s,s':String; Pa: Participates numConfirmedMeetings (p) = size(select _m (getMeetings(Pa,p), [isConfirmed (m)]) Rule 1, 2 numMeetings (p) = size (getMeetings (Pa, p)) Rule 1 END-CLASS	
CLASS Meeting... AXIOMS m,m1:Meeting; s,s':String; d,d',d1,d1':Date; b,b':Boolean; Pa:Participates isConfirmed (cancel(m)) = False isConfirmed (m)=checkDate(m) and NumConfirmedParticipants (m) >= 2 Rule 1 checkDate(m) = Rules 1, 2, 3 forall _{me} (collect _p (getParticipants(Pa,m), [getMeetings (Pa, p)]), [consistent (m,me)]) consistent(m,m1)= not (isConfirmed(m1)) or (end(m) < start(m1) or end(m1) < start(m)) NumConfirmedParticipants (m) = size (getParticipants(Pa,m)) END-CLASS	

INTEGRATING NEREUS WITH ALGEBRAIC LANGUAGES: FROM NEREUS TO CASL

In this section, we examine the relation between NEREUS and algebraic languages using Common Algebraic Specification Language (CASL) as a common algebraic language (Bidoit & Mosses, 2004).

CASL is an expressive and simple language based on a critical selection of known constructs, such as subsorts, partial functions, first-order logic, and structured and architectural specifications. A basic specification declares sorts, subsorts, operations,

and predicates, and gives axioms and constraints. Specifications are structured by means of specification-building operators for renaming, extension, and combining. Architectural specifications impose structure on implementations, whereas structured specifications only structure the text of specifications. It allows loose, free, and generated specifications.

CASL is at the center of a family of specification languages. It has restrictions to various sublanguages and extensions to higher order, state-based, concurrent, and other languages. CASL is supported by tools and facilitates interoperability of prototyping and verification tools.

Algebraic languages do not follow similar structuring mechanisms to UML or NEREUS. The graph structure of a class diagram involves cycles such as those created by bidirectional associations. However, the algebraic specifications are structured hierarchically and cyclic import structures between two specifications are avoided. In the following, we describe how to translate basic specification in NEREUS to CASL, and then analyze how to translate associations (Favre, 2005b).

Translating Basic Specifications

In NEREUS, the elements of `<parameterList>` are pairs `C1:C2` where `C1` is the formal generic parameter constrained by an existing class `C2` or `C1: ANY` (see Figure 2). In CASL, the first syntax is translated into `[C2]` and the second in `[sort C1]`. Figure 15 shows some examples.

NEREUS and CASL have a similar syntax for declaring types. The sorts in the IS-SUBTYPE paragraph are linked to subsorts in CASL.

The signatures of the NEREUS operations are translated into operations or predicates in CASL. Datatype declarations may be used to abbreviate declarations of types and constructors.

Any NEREUS function that includes partial functions must specify the domain of each of them. This is the role of the PRE clause that indicates what conditions the function's arguments must satisfy to belong to the function's domain. To indicate that a CASL function may be partial, the notation uses `->?`; the normal arrow will be reserved for total functions. The translation includes an axiom for restricting the domain. Figure 16 exemplifies the translation of a partial function `remove` (see Figure 2).

In NEREUS, it is possible to specify three different levels of visibility for operations: public, protected, and private. In CASL, a private visibility requires hiding the operation by means of the operator `Hide`. On the other hand, a protected operation in a class is

Figure 15. Translating parameters

NEREUS	CLASS CartesProd [E: ANY; E1 : ANY]
CASL	spec CARTESPROD [sort E] [sort E1]
NEREUS	CLASS HASH [T: ANY; V: HASHABLE]
CASL	spec HASH [sort T] [HASHABLE]

Figure 16. Translating partial functions

NEREUS remove: Bidirectional (b) x Class1(c1) x Class2 (c2)-> Bidirectional pre: isRelated (b,c1,c2)
CASL remove: Bidirectional (b) x Class1 x Class2 ->? Bidirectional ... forall b:Bidirectional, c1:Class1; c2: Class2 def remove(b,c1,c2) <=> isRelated (b,c1,c2)

Figure 17. Translating importing relations

```

spec SN [SP1] [SP2]... [SPn]
given SP1', SP2',..., SPm' = SP1" and SP2" and ...
then
    SP
end

```

included in all the subclasses of that class, and it is hidden by means of the operator Hide or the use of local definitions.

The IMPORTS paragraph declares imported specifications. In CASL, the specifications are declared in the header specification after the keyword given or like unions of specifications. A generic specification definition SN with some parameters and some imports is depicted in Figure 17.

SN refers to the specification that has parameter specifications SP₁, SP₂, ... SP_n, (if any). Parameters should be distinguished from references to fixed specifications that are not intended to be instantiated such as SP₁', SP₂', ..., SP_m' (if any). SP₁", SP₂", ... are references to import that can be instantiated. Unions also allow us to express inheritance relations in CASL. Figure 18 exemplifies the translation of inheritance relations. References to generic specifications always instantiate the parameters. In NEREUS, the instantiation of parameters [C : B]—where C is a class already existing in the environment and B is a component of A, and C is a subclass of B—constructs an instance of A in which the component B is substituted by C. In CASL, the intended fitting of the parameter symbols to the argument symbols may have to be specified explicitly by means of a fit C|-> B.

NEREUS and CASL have the similar syntax for defining local functions. Then, this transformation is reduced to a simple translation.

NEREUS distinguishes incomplete and complete specifications. In CASL, the incomplete specifications are translated to loose specifications and complete ones to free

Figure 18. Translating inheritance relations

NEREUS CLASS A INHERITS B, C
CASL spec A = B and C end

Figure 19. Translating higher order functions

<pre> spec Operation [sort X] = Z1 and Z2 and ... Zr then pred f1_j : X 1 ≤ j ≤ m f2_j : X 1 ≤ j ≤ n f3_j : X 1 ≤ j ≤ k f4_j : X 1 ≤ j ≤ l ops base_j : -> Z_j 1 ≤ j ≤ r g_j : Z_j x X -> Z_j 1 ≤ j ≤ r end spec Collection [sort Elem] given NAT= OPERATION [Elem] then generated type Collection ::= create add (Collection ; Elem) pred isEmpty : Collection includes : Collection x Elem includesAll : Collection x Collection forAll_i : Collection 1 ≤ i ≤ k exists_i : Collection 1 ≤ i ≤ l iterate_i : Collection -> Z_j 1 ≤ i ≤ r ops size : Collection -> Nat select_i : Collection -> Collection 1 ≤ i ≤ m reject_i : Collection -> Collection 1 ≤ i ≤ n </pre>	<pre> forall c,c1:Collection; e:Elem isEmpty (create) includes(add (c,e),e1) = if e=e1 then true else includes(c,e1) select_i (create) = create select_i (add (c,e)) = if f1_i(e) then add (select_i(c),e) else select_i (c) 1 ≤ i ≤ m includesAll (c,add (c1,e)) = includes (c,e) and includesAll (c,c1) reject_i (create) = create reject_i (add(c,e))= if not f2_i(e) then add (reject_i (c), e) else reject_i (c) 1 ≤ i ≤ n forAll_i(add(c,e))= f3_i(e) and for-all_i(c) 1 ≤ i ≤ k exists_i (add (c,e))= f4_i (e) or exists_i(c) 1 ≤ i ≤ l iterate_i (create) = base_j iterate_j (add (c,e)) = g_j(e, iterate_j (c)) 1 ≤ i ≤ r local ops f2 : Elem x Nat -> Nat forall e: Elem; i: Nat f2(e,i) = i+1 within size(c) = iterate (c, f2, 0) end-local end </pre>

specifications. If the specification has basic constructors, it will be translated into generated specifications. However, if it is incomplete, it will be translated into loose generated specifications. Both NEREUS and CASL allow loose extensions of free specifications.

The classes that include higher order operations are translated inside parameterized first-order specifications. The main difference between higher order specifications and parameterized ones is that, in the first approach, several function-calls can be done with

the same specification and parameterized specifications require the construction of several instantiations. Figure 19 shows the translation of the Collection specification (see Figure 3) to CASL. Take into account that there are as many functions f_1 , f_2 , f_3 , and f_4 as functions `select`, `reject`, `forAll` and `exists`. There are also as many functions `base` and `g` as functions `iterate`.

Translating Associations

NEREUS and UML follow similar structuring mechanisms of data abstraction and data encapsulation. The algebraic languages do not follow these structuring mechanisms in an UML style. In UML, an association can be viewed as a local part of an object. This interpretation cannot be mapped to classical algebraic specifications, which do not admit cyclic import relations.

We propose an algebraic specification that considers associations belonging to the environment in which an actual instance of the class is embedded. Let `Assoc` be a bi-directional association between two classes called `A` and `B`; the following steps can be distinguished in the translation process. We exemplify these steps with the transformation of `P&M` (see Figure 11).

Step1:

Regroup the operations of classes `A` and `B` distinguishing operations local to `A`, local to `B` and, local to `A` and `B` and `Assoc` (Figure 20).

Step 2:

Construct the specifications `A'` and `B'` from `A` and `B` where `A'` and `B'` include local operations to `A` and `B` respectively (Figure 21).

Step 3:

Construct specifications `Collection[A']` and `Collection[B']` by instantiating reusable schemes (Figure 22).

Step 4:

Construct a specification `Assoc` (with `A'` and `B'`) by instantiating reusable schemes in the component `Association` (Figure 23).

Step 5:

Construct the specification `AssocA+B` by extending `Assoc` with `A'`, `B'` and the operations local to `A'`, `B'` and `Assoc` (Figure 24).

Figure 25 depicts the relationships among the specifications built in the different steps.

Figure 20. Translating Participates association. Step 1.

Local to...	Operations/Attributes
Person	name
Meeting	title, start, end, duration
Person, Meeting, Participates	cancel, isConfirmed, numConfirmedMeetings, checkDate, numMeetings

Figure 21. Translating Participates association. Step 2.

<pre> spec PERSON given STRING, NAT = then generated type Person ::= create-Person (String) ops name: Person -> String set-name :Person x String -> Name end </pre>
<pre> spec MEETING given STRING, DATE = then generated type Meeting ::= create-Meeting (String; Date; Date) ops title: Meeting -> String set-title: Meeting x String -> Meeting start : Meeting -> Date set-start: Meeting x Date -> Meeting isEnd: Meeting -> Date set-end: Meeting x Date -> Meeting end </pre>

Figure 22. Translating Participates association. Step 3.

<pre> spec SET-PERSON given NAT= PERSON and BAG[PERSON] and ... then generated type Set[Person] :: = create including (Set[Person]; Person) ops union : Set[Person] x Set[Person] -> Set [Person] intersection : Set[Person] x Set[Person] -> Set [Person] count: Set[Person] x Person -> Nat ... </pre>
<pre> spec SET-MEETING given NAT = MEETING and BAG[MEETING] and ... then generated type Set [Meeting] :: = create including (Set[Meeting]; Meeting) ... </pre>

Figure 23. Translating Participates association. Step 4.

```

spec PARTICIPATES = SET-PERSON and SET-MEETING
and BINARY-ASSOCIATION [PERSON][MEETING]
with BinaryAssociation |-> Participates
pred
  isRightLinked: Participates x Person
  isLeftLinked: Participates x Meeting
  isRelated: Participates x Person x Meeting
ops
  addLink: Participates x Person x Meeting -> Participates
  getParticipants: Participates x Meeting -> Set [Person]
  getMeetings: Participates x Person -> Set[Meeting]
  remove: Participates x Person x Meeting -> Participates
    a : Participates; p,p1: Person; m,m1: Meeting
  def addLink (a,p,m) < = > not isRelated (a,p,m)
  def getParticipants (a, m) < = > isLeftLinked (a,m)
  def getMeetings (a, m) < = > isRightLinked ( a, m)
  def remove (a,p,m) < = > isRelated (a, p, m)
end

```

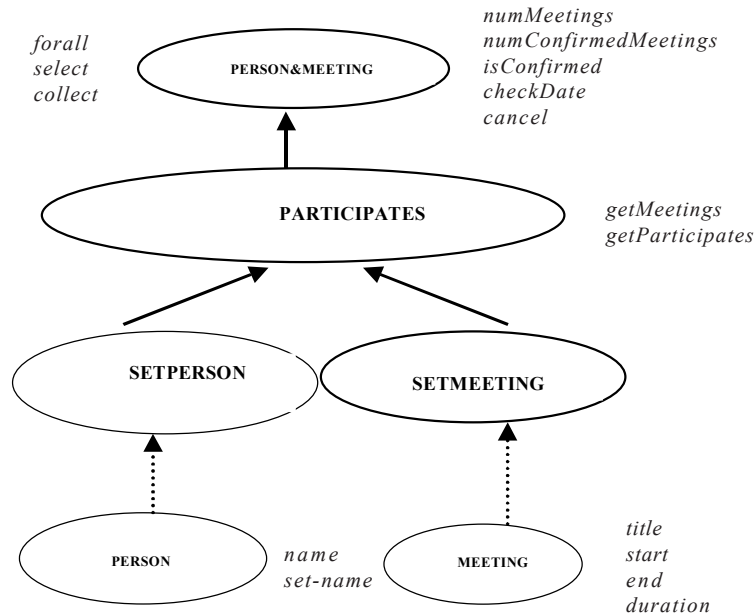
Figure 24. Translating Participates association. Step 5.

```

spec PERSON&MEETING = PARTICIPATES
then ops
  numMeeting: Participates x Person -> Nat
  numConfirmedMeeting: Participates x Person -> Nat
  isConfirmed: Participates x Meeting -> Boolean
  numConfirmedParticipants: Participates x Meeting -> Nat
  checkDate: Participates x Meeting -> Participates
  select: Participates x Set[Meeting] -> Set[Meeting]
  collect: Participates x Set[Person] -> Bag[Meeting]
  pred forall: Participates x Set[Meeting] x Meeting
  ∀s : Set[Meeting]; m:Meeting; pa:Participates; p:Person; m:Meeting; sp:Set[Person];
  bm: Bag[Meeting]
  forall (pa, including(s,m),m1) = isConsistent(pa, m,m1) and forall(pa, s, m1)
  select( pa, create-Meeting) = create-Meeting
  select ( pa, including (s, m)) = including(select(pa,s), m) when isConfirmed (pa, m)
    else select (pa,s)
  collect (pa, create-Person,s) = asBag (create-Person)
  collect (pa, including (sp, p) ) = asBag (including (collect (pa,sp), p))
  numMeeting( pa, p) = size (getMeetings(pa, p))
  isConfirmed (pa, m) = checkDate (pa,m) and NumConfirmedParticipants (pa,m) > = 2
  numConfirmedMeeting (pa, p) = size (select (getMeetings (pa,p))
  checkDate (pa, m) = forall (pa, collect (pa, getParticipants(pa,m), m)
  isConsistent (pa, m, m1) = not (isConfirmed (pa,m1)) or (end(m) < start (m1) or
  end (m1) < start(m))
  numParticipantsConfirmed (pa, m) = size( getParticipants (pa, m))
end

```

Figure 25. Translating Participates association into CASL



BENEFITS OF THE RIGOROUS FRAMEWORK FOR MDA

Formal and semiformal techniques can play complementary roles in MDA-based software development processes. We consider it beneficial for both semiformal and formal specification techniques. On one hand, semiformal techniques lack precise semantics; however, they have the ability to visualize language constructions, allowing a great difference in the productivity of the specification process, especially when the graphical view is supported by good tools. On the other hand, formal specifications allow us to produce a precise and analyzable software specification and automate model-to-model transformations; however, they require familiarity with formal notations that most designers and implementers do not currently have, and the learning curve for the application of these techniques requires considerable time.

UML and OCL are too imprecise and ambiguous when it comes to simulation, verification, validation, and forecasting of system properties and even when it comes to generating models/implementations through transformations. Although OCL is a textual language, OCL expressions rely on UML class diagrams, that is, the syntax context is determined graphically. OCL does also not have the solid background of a classical formal language. In the context of MDA, model transformations should preserve correctness. To achieve this, the different modeling and programming languages involved in a MDD must be defined in a consistent and precise way. Then, the combination

of UML/OCL specifications and formal languages offers the best of both worlds to the software developer. In this direction, we define NEREUS to take advantage of all the existing theoretical background on formal methods, using different tools such as theorem provers, model checkers, or rewrite engines in different stages of MDD.

In contrast to other works, our approach is the only one focusing on the interoperability of formal languages in model-driven software development. There are UML formalizations based on different languages that do not use an intermediate language. However, this extra step provides some advantages. NEREUS would eliminate the need to define formalizations and specific transformations for each different formal language. The metamodel specifications and transformations can be reused at many levels in MDA. Languages that are defined in terms of NEREUS metamodels can be related to each other because they are defined in the same way through a textual syntax.

We define only one bridge between UML/OCL and NEREUS by means of a transformational system consisting of a small set of transformation rules that can be automated. Our approach avoids defining transformation systems and the formal languages being used. Also, intermediate specifications may be needed for refactoring and for forward and reverse engineering purposes based on formal specifications.

We have applied the approach to transform UML/OCL class diagrams into NEREUS specifications, which, in turn, are used to generate object-oriented code. The process is based on the adaptation of MDA-based reusable components. NEREUS allows us to keep a trace of the structure of UML models in the specification structure that will make it easier to maintain consistency between the various levels when the system evolves. All the UML model information (classes, associations, and OCL specifications) is overturned into specifications having implementation implications. The transformation of different kinds of UML associations into object-oriented code was analyzed, as was the construction of assertions and code from algebraic specifications. The proposed transformations preserve the integrity between specification and code. The transformation of algebraic specifications to object-oriented code was prototyped (Favre, 2005a). The OCL/NEREUS transformation rules were prototyped (Favre et al., 2003).

FUTURE TRENDS

Currently, OMG is promoting a transition from code-oriented to MDA-based software development techniques. The existing MDA-based tools do not provide sophisticated transformation from PIM to PSM and from PSM to code. To date, they might be able to support forward engineering and partial round-trip engineering between PIM and code. However, it will probably take several years before a full round-trip engineering based on standards occurs (many authors are skeptical about this).

To solve these problems, a lot of work will have to be carried out dealing with the semantics for UML, advanced metamodeling techniques, and rigorous transformation processes. If MDA becomes commonplace, adapting it to formal development will become crucial. In this light, we will investigate the NEREUS language for integrating formal tools. NEREUS would allow different formal tools to be used in the same development environment to translate models expressed in different modeling languages into the intermediate language, and back, by using NEREUS as an internal representation that is shared among different formal languages/tools. Any number of source languages

(modeling language) and target languages (formal language) could be connected without having to define explicit model/metamodel transformations for each language pair.

Techniques that currently exist in UML CASE tools provide little support for generating business models. In the light of the advances of the MDA paradigm, a new type of UML tool that does a more intelligent job might emerge. Probably, the next generation of tools might be able to describe the behavior of software systems in terms of business models and translate it into executable programs on distributed environment.

CONCLUSION

In this chapter, we describe a uniform framework for model-driven development that integrates UML/OCL specifications with formal languages. It is comprised of a “megamodel” for defining MDA components, a metamodeling notation NEREUS, and the definition of metamodeling/model transformations using UML/OCL and NEREUS.

A “megamodel” integrates PIMs, PSMs and code models with their respective metamodels. We formalize UML-based metamodels in NEREUS, which is an intermediate notation particularly suited for metamodeling. We define a system of transformation rules to bridge the gap between UML/OCL models and NEREUS. We propose to specify metamodel transformations independently of any technology. We investigate the way to define them using UML/OCL and NEREUS.

We want to define foundations for MDA tools that permit designers to directly manipulate the UML/OCL models they have created. However, meta-designers need to understand metamodels and metamodel transformations.

We are validating the “megamodel” through forward engineering, reverse engineering, model refactoring, and pattern applications.

We foresee the integration of our results in the existing UML CASE tools, experimenting with different platforms such as .NET and J2EE.

REFERENCES

- Abmann, U. (Ed.). (2004). *Proceedings of Model-Driven Architecture: Foundations and applications*. Switzerland: Linköping University. Retrieved February 28, 2006, from <http://www.ida.liv.se/henla/mdafa2004>
- Ahrendt, W., Baar, T., Beckert, B., Bubel, R., Giese, M., Hähnle, R., et al. (2005). The key tool. *Software and Systems Modeling*, 4, 32-54.
- Akehurst, D., & Kent, S. (2002). A relational approach to defining transformations in a metamodel. In J. M. Jezequel, H. Hussmann, & S. Cook (Eds.), *Lecture Notes in Computer Science 2460* (pp. 243-258). Berlin: Springer-Verlag.
- Atkinson, C., & Kuhne, T. (2002). The role of metamodeling in MDA. In J. Bezivin & R. France (Eds.), *Proceedings of UML 2002 Workshop in Software Model Engineering (WiSME 2002)*, Dresden, Germany. Retrieved February 28, 2006, from <http://www.metamodel.com/wisme-2002>
- Bézivin, J., Farcet, N., Jézéquel, J., Langlois, B., & Pollet, D. (2003). Reflective model driven engineering. In P. Stevens, J. Whittle, & G. Booch (Eds.), *Lecture Notes in Computer Science 2863* (pp. 175-189). Berlin: Springer-Verlag.

- Bézivin, J., Jouault, F., & Valduriez, P. (2004). On the need for megamodels. In J. Bettin, G. van Emde Boas, A. Agrawal, M. Volter, & J. Bezivin (Eds.), *Proceedings of Best Practices for Model-Driven Software Development (MDSD 2004), OOSPLA 2004 Workshop*, Vancouver, Canada. Retrieved February 28, 2006, from <http://www.softmetaware.com/oospla2004>
- Bidoit, M., & Mosses, P. (2004). CASL user manual- Introduction to using the Common Algebraic Specification Language. In *Lecture Notes in Computer Science 2900* (p. 240). Berlin: Springer Verlag.
- Büttner, F., & Gogolla, M. (2004). Realizing UML metamodel transformations with AGG. In R. Heckel (Ed.), *Proceedings of ETAPS Workshop Graph Transformation and Visual Modeling Techniques (GT-VMT 2004)*. Retrieved February 28, 2006, from <http://www.cs.uni-paderborn.de/cs/ag-engels/GT-VMT04>
- Caplat, G., & Sourrouille, J. (2002). Model mapping in MDA. In J. Bezivin & R. France (Eds.), *Proceedings of UML 2002 Workshop in Software Model Engineering (WiSME 2002)*. Retrieved February 28, 2006, from <http://www.metamodel.com/wisme-2002>
- Cariou, E., Marvie, R., Seinturier, L., & Duchien, L. (2004). OCL for the specification of model transformation contracts. In J. Bezivin (Ed.), *Proceedings of OCL&MDE'2004, OCL and Model Driven Engineering Workshop*, Lisbon, Portugal. Retrieved February 28, 2006, from <http://www.cs.kent.ac.uk/projects/ocl/oclmdewsu104>
- Czarnecki, K., & Helsen, S. (2003). Classification of model transformation approaches. In J. Bettin et al. (Eds.), *Proceedings of OOSPLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture*. Retrieved February 28, 2006, from <http://www.softmetaware.com/oopspla.2003/mda-workshop.html>
- Evans, A., Sammut, P., & Willans, S. (Eds.). (2003). *Proceedings of Metamodeling for MDA Workshop*, York, UK. Retrieved February 28, 2006, from <http://www.cs.york.uk/metamodel4mda/onlineProceedingsFinal.pdf>
- Favre, J. (2004). Towards a basic theory to model driven engineering. In M. Gogolla, P. Sammut, & J. Whittle (Eds.), *Proceedings of WISME 2004, 3rd Workshop on Software Model Engineering*. Retrieved February 28, 2006, from <http://www.metamodel.com/wisme-2004>
- Favre, L. (2005a). Foundations for MDA-based forward engineering. *Journal of Object Technology (JOT)*, 4(1), 129-154.
- Favre, L. (2005b). A rigorous framework for model-driven development. In T. Halpin, J. Krogstie, & K. Siau (Eds.), *Proceedings of CAISE'05 Workshops. EMMSAD '05 Tenth International Workshop on Exploring Modeling Method in System Analysis and Design* (pp. 505-516). Porto, Portugal: FEUP Editions.
- Favre, L., Martinez, L., & Pereira, C. (2003). Forward engineering and UML: From UML static models to Eiffel code. In L. Favre (Ed.), *UML and the unified process* (pp. 199-217). Hershey, PA: IRM Press.
- Favre, L., Martinez, L. & Pereira, C. (2005). Forward engineering of UML static models. In M. Khosrow-Pour (Ed.), *Encyclopedia of information science and technology* (pp. 1212-1217). Hershey, PA: Idea Group Reference.
- Gogolla, M., Bohling, J., & Richters, M. (2005). Validating UML and OCL models in USE by automatic snapshot generation. *Journal on Software and System Modeling*. Retrieved from <http://db.informatik.uni-bremen.de/publications>

- Gogolla, M., Lindow, A., Richters, M., & Ziemann, P. (2002). Metamodel transformation of data models. In J. Bézivin & R. France (Eds.), *Proceedings of UML'2002 Workshop in Software Model Engineering (WiSME 2002)*. Retrieved February 28, 2006, from <http://www.metamodel.com/wisme-2002>
- Gogolla, M., Sammut, P., & Whittle, J. (Eds.). (2004). *Proceedings of WISME 2004, 3rd Workshop on Software Model Engineering*. Retrieved February 28, 2006, from <http://www.metamodel.com/wisme-2004>
- Hausmann, J. (2003). Relations-relating metamodels. In A. Evans, P. Sammut, & J. Williams (Eds.), *Proceedings of Metamodeling for MDA. First International Workshop*. Retrieved February 28, 2006, from <http://www.cs.uni-paderborn.de/cs/ag-engels/Papers/2004/MM4MD/Ahausmann.pdf>
- Hussmann, H., Cerioli, M., Reggio, G., & Tort, F. (1999). *Abstract data types and UML models* (Tech. Rep. No. DISI-TR-99-15). University of Genova, Italy.
- Kim, S., & Carrington, D. (2002). A formal model of the UML metamodel: The UML state machine and its integrity constraints. In *Lecture Notes in Computer Science 2272* (pp. 477-496). Berlin: Springer-Verlag.
- Kleppe, A., Warner, J., & Bast, W. (2003). *MDA explained. The model driven architecture: Practice and promise*. Boston: Addison Wesley Professional.
- Kuske, S., Gogolla, M., Kollmann, R., & Kreowski, H. (2002, May). An integrated semantics for UML class, object and state diagrams based on graph transformation. In *Proceedings of the 3rd International Conference on Integrated Formal Methods (IFM'02)*, Turku, Finland. Berlin: Springer-Verlag.
- Kuster, J., Sendall, S., & Wahler, M. (2004). Comparing two model transformation approaches. In J. Bézivin et al. (Eds.), *Proceedings of OCL&MDE'2004, OCL and Model Driven Engineering Workshop*, Lisbon, Portugal. Retrieved February 28, 2006, from <http://www.cs.kent.ac.uk/projects/ocl/oclmdewsu104>
- McUmber, W., & Cheng, B. (2001). A general framework for formalizing UML with formal languages. In *Proceedings of the IEEE International Conference on Software Engineering (ICSE01)*, Canada (pp. 433-442). IEEE Computer Society.
- Padawitz, P. (2000). Swinging UML: How to make class diagrams and state machines amenable to constraint solving and proving. In A. Evans & S. Kent (Eds.), *Lecture Notes in Computer Science 1939* (pp. 265-277). Berlin: Springer-Verlag.
- Reggio, G., Cerioli, M., & Astesiano, E. (2001). Towards a rigorous semantics of UML supporting its multiview approach. In *Proceedings of Fundamental Approaches to Software Engineering (FASE 2001)* (LNCS 2029, pp. 171-186). Berlin: Springer-Verlag.
- Snook, C., & Butler, M. (2002). *Tool-supported use of UML for constructing B specifications*. Technical report, Department of Electronics and Computer Science, University of Southampton, UK.
- Sunyé, G., Pollet, D., LeTraon, Y., & Jezequel, J-M. (2001). Refactoring UML models. In M. Gogolla & C. Kobryn (Eds.), *Lecture Notes in Computer Science 2185* (pp. 134-148). Berlin: Springer-Verlag.

ENDNOTE

- ¹ This work is partially supported by the Comisión de Investigaciones Científicas (CIC) de la Provincia de Buenos Aires in Argentina.

Chapter II

Adopting Open Source Development Tools in a Commercial Production Environment: Are We Locked in?

Anna Persson, University of Skövde, Sweden

Henrik Gustavsson, University of Skövde, Sweden

Brian Lings, University of Skövde, Sweden

Björn Lundell, University of Skövde, Sweden

Anders Mattsson, Combitech AB, Sweden

Ulf Ärlig, Combitech AB, Sweden

ABSTRACT

Many companies are using model-based techniques to offer a competitive advantage in an increasingly globalised systems development industry. Central to model-based development is the concept of models as the basis from which systems are generated, tested, and maintained. The availability of high-quality tools and the ability to adopt and adapt them to the company practice are important qualities. Model interchange between tools becomes a major issue. Without it, there is significantly reduced flexibility and a danger of tool lock-in. We explore the use of a standardised interchange format (XMI) for increasing flexibility in a company environment. We report on a case study in which a systems development company has explored the possibility of

complementing its current proprietary tools with open-source products for supporting its model-based development activities. We found that problems still exist with interchange and that the technology needs to mature before industrial-strength model interchange becomes a reality.

INTRODUCTION

The nature of the information systems development industry is changing under the pressures brought about by increased globalisation. There is competition to offer cheaper but higher quality products faster. To stay competitive, many companies are using model-based techniques to offer rapid prototyping, fast response to requirements change, and improved systems quality. Central to model-based development is the concept of models as the major investment artefacts; these are then used as the basis for automatic system generation and test. Tools for the development, maintenance, and transformation of models are therefore at the heart of the tool infrastructure for environments which support model-based development practice.

One potential danger for companies is tool lock-in. Tool lock-in exists if the models developed within a tool are accessible only through that tool. It has long been recognised that the investment inherent in design artefacts must be protected against tool lock-in, not least for maintenance of a long-lived application. Such lock-in effects are recognised as a risk, which can have severe consequences for an individual company (Statskontoret, 2003). The tool market is dynamic, and there is no guarantee that a tool or tool version used to develop a product will remain usable for the lifetime of the product (Lundell & Lings, 2004a, 2004b). In order to protect against such problems, models must be stored together with the version of the tool with which they were created. Even this is not guaranteed to succeed — hardware changes may mean that old versions of tools can no longer be run — unless hardware is also maintained with the tool. Such lock-ins are therefore undesirable for tool users. This may not be the case for some tool vendors, who may view lock-in as a tactic to ensure future business by keeping customers tied to their products (Statskontoret, 2003).

The availability of high-quality modelling tools and the ability to adopt and adapt them to a company context are also important qualities. A variety of different development tools can be applied during a systems development project, including tools for the design of UML diagrams, tools for storing models for persistence, and tools for code generation (Boger, Jeckle, Mueller, & Fransson, 2003). The ability to seamlessly use and combine the various tools used within a project is highly desirable (Boger et al., 2003). The reality for many designers is an environment in which a mix of tools is used, and many companies are considering a mix of proprietary and open source tools to flexibly cover their needs. The interchange of design artefacts between tools becomes critical in such environments. One special case of this is geographically distributed development where partners in different locations are working in different environments, using different tool sets.

Model interchange functionality can therefore significantly increase flexibility and reduce exposure to lock-in effects. There are two accepted ways in which model interchange can be undertaken: via software bridges, and via an open interchange

standard. For example, the i-Logix Rhapsody® tool offers a VB software bridge to allow the import of models from the IBM Rose® tool, utilising its proprietary API. Such ad hoc provision is neither guaranteed nor universal and can inhibit the development and organisational adoption of new tools. Neither does a bridge lessen the burden of having to save a tool with the models produced by it — a bridge requires the tool to be running in order to allow access to the model.

The more flexible (and scalable) approach is to support interchange through an open interchange standard. In an International Technology Education Association report, open-source software is seen as one way of avoiding dependence on a single vendor (ITEA, 2004). Adherence to open standards has always been viewed as central to the open-source movement, and key to achieving interoperability (Fuggetta, 2003). An implied message to the open-source community is that adoption of open-source tools will depend heavily on their ability to interchange models with other tools using an open-data standard.

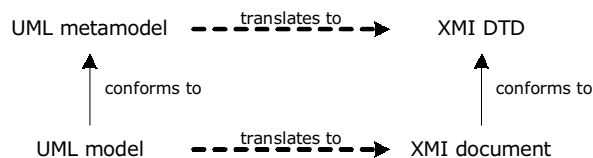
BACKGROUND

Over the years, many standardised interchange technologies have been proposed. Current interest centres on the Object Management Group's XML Metadata Interchange format (XMI) (OMG, 2000a, 2000b, 2002, 2003). In theory, any model within a tool can be exported in XMI format and imported into a different tool also supporting XMI.

In principle, XMI allows for the interchange of models between modelling tools in distributed and heterogeneous environments and eases the problem of tool interoperability (Brodsky, 1999; Obrenovic & Starcevic, 2004; OMG, 2000a, 2000b, 2002, 2003). As most major UML modelling tools currently offer model interchange using XMI (Jeckle, 2004; Stevens, 2003), tool lock-in should not be a problem. This could offer the prospect of an invigorated tool market, with niche suppliers offering specialised functionality knowing that lock-in is not a factor in potential purchasing.

Although XMI can be used for the interchange of models in any modelling notation, according to OMG (2000a) one of the main purposes of XMI is to serve as an interchange format for UML models. The interchange of XMI-based UML models between tools is realized by the export and import of XMI documents. An XMI document consists of two parts: an XML document of tagged elements, and a Document Type Declaration (DTD) — or schema in XMI version 2.0 — specifying the legal tags and defining structure.

Figure 1. Generation of XMI document for a UML model (from Stevens, 2003)



Exporting a model into an XMI document is done by traversing the model and building an XML tree according to a DTD. The XML tree is then written to a document. Other tools can recreate the model by parsing the resulting XMI document. An overview of how an XMI document for an UML model is generated is shown in Figure 1.

The OMG states that “in principle, a tool needs only to be able to save and load the data it uses in XMI format in order to inter-operate with other XMI capable tools” (OMG, 2000a). From this description, tool integration using XMI-based model interchange may seem to be simple. However, a number of reports have suggested that, in practice, having a tool with XMI support is no guarantee for a working interchange, something we wished to explore in a case study.

For example, Damm, Hansen, Thomsen, and Tyrsted (2000) encountered some problems with XMI-based model interchange when integrating Knight, their UML modelling tool, with two proprietary UML-modelling tools. One problem was incompatibility between tools that support different versions of XMI. Today, there are four versions of XMI recognised by OMG: Versions 1.0, 1.1, 1.2 and 2.0 (OMG, 2000a, 2000b, 2002, 2003), and different tool producers have adopted different versions of XMI. What should be a straightforward export/import situation instead requires extra transformations between versions of XMI. Damm et al. state that “The IBM Toolkit and the Rose plug-in produce XMI files that are compatible, but neither of them is compatible with ArgoUML which uses an earlier version of the XMI specification.” (Damm et al., 2000, p. 102).

XMI-based model interchange may also be troublesome between tools supporting the same version of XMI, as discussed by Süß, Leicher, Weber, and Kutsche (2003) and Stevens (2003). According to Süß et al., “Most modelling tools support an XMI dialect that more or less complies with the XMI specification” (2003, p. 35). According to Stevens, “Some incompatibilities between XMI written by different tools still exist” (2003, p. 9), since two tools using the same version of XMI and UML do not necessarily generate the same XMI representation of a certain model.

In this chapter, we consider the use of XMI in UML-modelling tools for model interchange. We report on a case study in which a systems development company has explored the possibility of addressing tool lock-in and complementing its current proprietary tools with open-source tools for supporting its model-based development activities. The use of open-source software is appealing to many organisations, given reports of “very significant cost savings” (Fitzgerald & Kenny, 2003, p. 325). The study concentrated on UML models and, specifically, class diagrams — among the most widely used UML diagramming techniques and with “the greatest range of modeling concepts” (Fowler, 2003, p. 35).

In the case study, we consider class diagrams taken from commercial development projects in order to investigate whether XMI-based model interchange is a current option for the company. One aspect of the study was to explore whether it would be possible to use open-source modelling tools to complement its current (proprietary) tool usage within the company context.

THE CASE STUDY

Combitech Systems AB (hereafter referred to as Combitech) is a medium sized, geographically distributed enterprise working with advanced systems design and software development, electronic engineering, process optimisation, and staff training. It has approximately 230 employees and covers a broad spectrum of business areas such as defence, aviation, automotive, medical, and telecoms.

The company has a long experience of systematic method work and model-based systems development. In several development projects, UML is used (e.g., Mattsson, 2002), but other modelling techniques are used as well. The company uses three of the major case tools supporting both UML and time-discrete modelling: Rose Realtime® (from IBM), Rhapsody® (from i-Logix), and TAU® (from Telelogic).

Combitech has an interest in exploring the potential of open-source tools to complement its current tool suite and is also sensitive to the potential problem of tool lock-in. With this in mind, a case study was set up to explore the potential of XMI-based export and import to offer a strategy for tool integration and tool-independent storage formats. For the purposes of the case study, the company chose to look at existing UML class diagrams developed using the Rhapsody tool.

Rhapsody is a proprietary development tool that supports all diagram types developed according to UML Version 2.0 (for information, see Douglass, 2003). Interchange of UML models is supported by export and import of XMI Version 1.0 for UML Version 1.1 and 1.3 (i-Logix, 2004). Apart from UML modelling, requirements modelling, design-level debugging, forward engineering (generation of C, C++ and Ada source code), and automatic generation of test cases are also supported in the tool. The version of Rhapsody used currently by the company and in this study is 5.0.1.

Two production models developed by Combitech, hereafter referred to as “Model A” and “Model B,” were used in the study. The two models, developed in different versions of Rhapsody (Version 3.x and Version 4.x respectively), consist of approximately 170 and 60 classes, respectively. The classes have different kinds of attributes and operations and make use of all common association types available in a UML class diagram.

Model A describes a device manager for an application platform used in an embedded system and was developed using a “pair programming” activity. The model is one of many developed in a two-year project that, in total, involved about 50 system developers divided into nine teams.

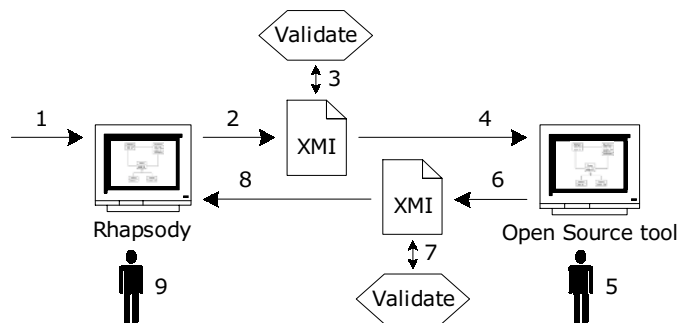
Model B is a high-level architectural model of an airborne laser-based bathymetry system for hydrographic surveys and was itself developed by a single developer. The model is taken from a development project of about four years.

To explore the open-source aspects, three open-source UML modelling tools have been used in the study: ArgoUML v.0.16.1 (hereafter referred to as Argo; for information, see Robbins & Redmiles, 2000), Fujaba (a recent nightly build, as the most recent stable version does not support XMI; for information, see Nickel, Niere, & Zundorf, 2000), and Umbrello UML Modeler v.1.3.0 (hereafter referred to as Umbrello; for information, see Ortenburger, 2003). These tools were selected for the study because they support UML class diagrams and interchange of such diagrams using XMI. A systematic review of available open-source modelling tools revealed no other tools with these properties. The tools, all supporting UML v.1.3, are presented in Table 1.

Table 1. Open-source UML modelling tools used in the study

	Argo http://argouml.tigris.org	Fujaba http://www.fujaba.de	Umbrello http://uml.sourceforge.net
XMI version	1.0	1.1	1.2
Storage format	Project-specific	Project-specific	XMI
UML models	All except object	Class, state, activity	All except object
Forward eng.	Java, C++, PHP	Java	Java, C++, PHP, ...
Reverse eng.	Java	Java	C++
Platform	All (Java-based)	All (Java-based)	Linux (with KDE)
Active developers	Approx. 25	Approx. 35	Approx. 5
License	BSD Open Source	GNU Lesser General Public	GNU General Public

Figure 2. Overview of model interchange



It should be noted that only 5% of open source projects are developed by more than five developers (Zhao & Elbaum, 2003), so all of these are sizeable developments (information as published on each tool's mailing list in August 2004).

In order to explore interchange fully, a round-trip interchange scenario was devised. Each of the two models, developed at different times and by different developers within the company, were to be exported as an XMI document for import into an open-source tool and then exported by that tool for re-import into Rhapsody (see Figure 2). If such a test succeeded with no semantic loss, then we could conclude that interchange of the model was possible — and lock-in absent. Round-trip is necessary to counter the possibility that lock-in was simply extended to two tools.

In what follows, our approach to model interchange is described. The procedure described applies for both models used and also for a third (small) test model created as a control. Using this third model, we were able to check the basic export/import functionality in each tool. The numbered steps relate to the numbering in Figure 2.

RESULTS FROM THE CASE

Step 1: Bring Up Models in Rhapsody

The models used were developed in two different and earlier versions of the Rhapsody tool used in the exploration. The first step was to bring up each model in the current version of Rhapsody (5.0.1) for visual inspection, ready for export.

Step 2: Export Models from Rhapsody

Each model was then exported from Rhapsody into an XMI 1.0 document for UML 1.3. The document representing Model A consisted of 174,445 lines of XMI code, that for Model B 36,828 lines.

Step 3: Validate XMI Documents

At this stage, we checked whether each document conformed to the XMI DTD (specified by OMG) by using two independent XML validation tools: XMLSPY (www.altova.com) and ExamXML (<http://www.a7soft.com>). Export from Model B was found to be valid, but that from Model A was not. Both validation tools stated that the exported XMI document for Model A had a structure that deviated from the standard specified by OMG. The problem related to non-conformance with an ordering dependency in the XMI DTD. This was repaired manually in order to allow tests to continue. Such repair is extremely difficult without specialised tool support because the file consists of 174,445 lines of XMI code, which in any case is very difficult for a human reader to comprehend.

Step 4: Import Models into Open-Source Tools

An attempt was made to import each XMI document into each of the three open-source tools, resulting in a model as represented in the tool's internal storage format and available for inspection through its presentation layer.

Neither of the XMI documents exported from Rhapsody (and modified in the case of Model A) could be imported into either Fujaba or Umbrello. This was not unexpected, as Fujaba and Umbrello support only the import of later XMI versions than that used in Rhapsody, and it was evident from inspection of the documentation that backwards compatibility was not a feature of the XMI versions. This is because later versions have very different structure from XMI v. 1.0. For both models, Fujaba simply hangs, while in Umbrello nothing happens, and control is returned to the user without feedback.

It is possible to translate between versions of XMI. At the time of this writing, no open-source converters were available to allow further testing with these tools. However, the Poseidon tool from Gentleware (www.gentleware.com) — which is based on the code base from ArgoUML — claims to import Versions 1.0, 1.1 and 1.2 of XMI and export Version 1.2 (Gentleware, 2004). We therefore attempted to use Poseidon (Community Edition, Version 2.6) to import Rhapsody's exported XMI 1.0 file with a view to exporting XMI v. 1.2 for import into Umbrello. The XMI v. 1.0 file exported from Rhapsody for Model B could not initially be imported into Poseidon. However, after deleting an offending line — detected after inspection of Poseidon's log files — import was successful. Poseidon's exported XMI v. 1.2 file was used for further tests with Umbrello.

Testing continued by attempting to import Rhapsody's exported XMI v.1.0 documents for Models A and B (modified in the case of A) into Argo, and import the XMI v.1.2 document exported from Poseidon into Umbrello. Success was expected with the first two tests, since the structure of the XMI documents representing the models were each confirmed as conforming to the XMI v.1.0 standard by both validation tools, and Argo and Rhapsody both claim to support this version of XMI. Successful transfer via Poseidon was considered less likely, as several transformations are involved.

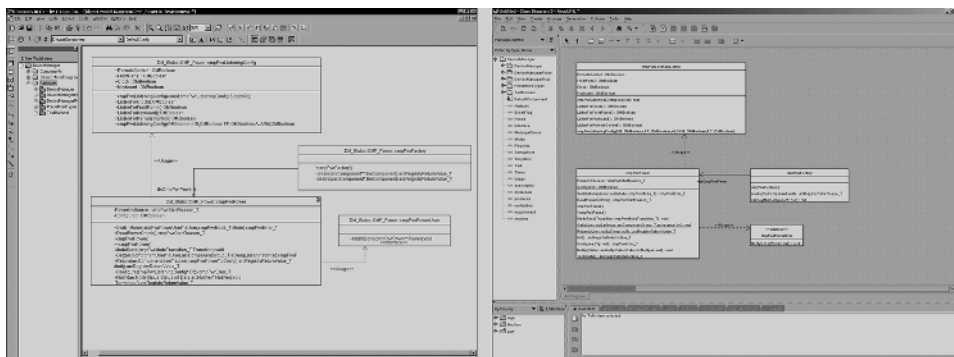
Even after repair, import of Model A into Argo failed. There are many comments attached to various elements in the UML model, and these were exported into XMI format. Although valid according to the XMI DTD, some of these caused problems for the Argo importer. It is unclear why only certain attachments caused problems. After significant experimentation, the XMI document was modified (with semantic loss) such that import into Argo became possible. The XMI v.1.0 document for Model B was successfully imported into Argo.

The XMI v.1.2 document exported from Poseidon was not valid, and so could not be imported into Umbrello. As a final test, a small test model developed in Poseidon was exported. Even this could not be successfully imported into Umbrello, and no further tests were made with that tool. Subsequent to the test, we found that the problem lies with illegal characters generated in Poseidon's IDs, and this has been noted in the vendor's forum as an issue to be resolved.

Step 5: Visual Inspection

A visual inspection was performed to compare each model as imported with its original in Rhapsody. A part of Model A is shown in Figure 3, firstly in Rhapsody and then in Argo (see the Appendix for larger versions of the screen shots). It should be noted that versions of UML earlier than 2.0 do not cater to the exchange of presentation information, so comparison will be of content only. Given the size of the models, this is not a simple task, and some manipulation of the presentations was made to help in the visual checks.

Figure 3. Screen shots from Model A in Rhapsody (left) and Argo (right)



Step 6: Models Exported from Open-Source Tool

Each model was exported from Argo into an XMI document in order to test its export facility. This generated a new XMI v.1.0 file for each of Models A and B.

Step 7: Visual Inspection

At this stage, we again checked whether the documents conformed to the XMI DTD. Neither exported XMI document was valid. This was due to a misspelling generated by Argo in the exported XMI. Once corrected (using a separate text editor), the documents became valid.

Step 8: Model Import to Rhapsody

Each model exported from Argo was imported into Rhapsody to complete a round-trip interchange. In each case, import (of the repaired XMI) was successful.

Step 9: Visual Inspection

A visual inspection was performed to determine whether the content of each model was identical to the original version of it in Rhapsody. Once again, it is extremely difficult for models of this size to be checked for semantic loss, particularly as presentation information is not preserved with XMI versions available in the tools. However, in the visual inspection, using some manual repositioning in the Rhapsody tool to assist the process, no inconsistencies were found.

Step 10: Final Test

As a final test, each model (revised as necessary) was repeatedly put through the complete cycle. It was observed that the XMI file grew through the addition of an extra enclosing package on each export (by Rhapsody and by Argo). This makes no semantic difference to the model but can be considered an inconvenience.

FUTURE TRENDS

Commercial tools offer proprietary bridges to other tools, particularly market leaders, and may even make efforts to improve XMI interchange possible by catering to product-specific interpretations of XMI. However, the OSS community can be expected to offer high conformance with any open standard and not to resort to tool-specific bridging software. Further, it could be argued that a goal for OSS tools should be to offer reliable import and export of documents conforming to any of the XMI versions, in this way offering both openness and an important role in the construction of interchange adapters — especially useful for legacy situations. As a special case, one hopes that OSS tools will lead the way in conformance with XMI 2.0 and UML 2.0. With the advent of UML 2.0 and XMI 2.0, there is a real possibility of standard interchange, both horizontally and vertically within the tool chain.

CONCLUSION

Like many companies, Combitech is currently committed to tools provided by more than one vendor. Although its current tool mix seems highly appropriate, Combitech's experience is that the tool market is dynamic — products come and go, and market leaders change over time. Most projects within the company involve many man years of effort, and the company is very aware of the need to protect its own and its customers' investments. It is also aware of the need to take full advantage of technology advances.

Further, in the company's experience, different developers prefer different aspects of tools, and it is quite likely that a particular developer may prefer a specific tool for a particular task. In fact, the company view is that some current open-source tools have clear potential for supporting aspects of its UML-modelling activities, and it envisions a hybrid tool mix as the most likely scenario in the future. Combitech is also increasingly finding that customers are knowledgeable about UML and envisages a future scenario in which parts of solutions are developed at customers' sites (perhaps using specialised tools). All of this heightens the company's interest in model interchange between tools, and XMI is currently the most commonly supported open-data standard.

It can be noted that OMG describes XMI as a general interchange standard and does not, in this respect, distinguish between different XMI versions, stating that "XMI allows metadata to be interchanged as streams or files with a standard format based on XML" (OMG, 2000a). This raises the question of whether XMI should actually be referred to as *a* standard interchange format. If tools supporting different XMI versions cannot interchange their XMI documents, then the interchange format may seem weakly standardized, and it is the different versions of XMI by themselves that are standardized, not the overall XMI format. It is also worthy of note that this distinction is not made clear by all manufacturers of products, many making interchange claims for their products which are not sustainable in practice. It is important that companies are well aware of the exact position with XMI, as it can feature highly in adoption decisions — as witnessed, for example, in OMG News (2002), where one company focused on adherence to standards (including XMI) when adopting the Rhapsody tool.

Although OSS tools offer support for XMI-based model interchange equal to that in commercial tools, better could be expected. It is interesting to note that no open-source tool yet offers conformity with the latest version of the standard or offers the ability to import documents formatted in more than one version of XMI. It is also interesting that a major commercial tool only offers conformance with XMI v.1.0.

Compatibility between XMI versions is not the only requirement for successful XMI-based model interchange between tools. Tools must guarantee the export of XMI documents that conform to any normative XMI document structure specified by OMG. As apparent from this study, this is not yet guaranteed. Export of invalid XMI documents is a serious issue that tool developers need to address.

The results of the study also show that complexity of models may cause interchange problems: less complex models seem easier for tools to handle. It is important that future studies should explore interchange issues using medium to large-scale models in order to subject tools to realistic modelling constructs from real usage contexts. Architectures for model-based systems development rely heavily on model interchange. To support such development in a globally distributed environment, robust and general export/import functionality must be provided. This will require effective and continued feedback

from practice on the actual and attempted use of open-data standards in systems development.

ACKNOWLEDGMENT

This research has been financially supported by the European Commission via FP6 Co-ordinated Action Project 004337 in priority IST-2002-2.3.2.3 “Calibre” (<http://www.calibre.ie>).

REFERENCES

- Boger, M., Jeckle, M., Mueller, S., & Fransson, J. (2003). Diagram interchange for UML. In J.-M. Jezequel, H. Hussmann, & S. Cook (Eds.), *Proceedings of UML 2002 — Unified Modeling Language: Model Engineering, Concepts, and Tools* (pp. 398-411). Berlin: Springer-Verlag.
- Brodsky, S. (1999). *XMI opens application interchange*. Retrieved April 15, 2005, from <http://www-4.ibm.com/software/ad/standards/xmiwhite0399.pdf>
- Damm, C. E., Hansen, K. M., Thomsen, M., & Tyrsted, M. (2000). Tool integration: Experiences and issues in using XMI and component technology. In *Proceedings of 33rd International Conference on Technology of Object-Oriented Languages and Systems TOOLS 33* (pp. 94-107). Los Alamitos, CA: IEEE Computer Society.
- Douglass, B. P. (2003). *Model driven architecture and Rhapsody*. Retrieved April 15, 2005, from http://www.omg.org/mda/mda_files/MDAandRhapsody.pdf
- Fitzgerald, B., & Kenny, T. (2003). Open-source software in the trenches: Lessons from a large-scale OSS implementation. In S. T. March, A. Massey, & J. I. DeGross (Eds.), *Proceedings of 2003 — Twenty-Fourth International Conference on Information Systems* (pp. 316-326). Seattle, WA: Association for Information Systems.
- Fowler, M. (2003). *UML distilled: A brief guide to the standard object modeling language* (3rd ed.). Boston: Addison-Wesley.
- Fuggetta, A. (2003). Open source software: An evaluation. *Journal of Systems and Software*, 66(1), 77-90.
- Gentleware. (2004). *Gentleware Product Description: Community Edition 2.6*. Retrieved April 15, 2005, from <http://www.gentleware.com>
- i-Logix. (2004). *XMI TOOLKIT VERSION 1.7.0 README FILE*. i-Logix Inc. Retrieved from <http://www.ilogix.com>
- ITEA. (2004). *International Technology Education Association Report on Open Source Software*. Retrieved November 10, 2005, from <http://www.iteaconnect.org/index.html>
- Jeckle, M. (2004, March 25). OMG’s XML metadata interchange format XMI. In *Proceeding of XML Interchange Formats for Business Process Management (XML4BPM 2004): 1st Workshop of German Informatics Society e.V. (GI), in conjunction with the 7th GI Conference, “Modellierung 2004,”* Marburg, Germany (pp. 25-42). Boon: Gesellschaft für Informatik.
- Lundell, B., & Lings, B. (2004a). Changing perceptions of CASE-technology. *Journal of Systems and Software*, 72(2), 271-280.
- Lundell, B., & Lings, B. (2004b). Method in action and method in tool: A stakeholder perspective. *Journal of Information Technology*, 19(3), 215-223.

- Mattsson, A. (2002). *Modellbaserad utveckling ger stora fördelar, men kräver mycket mer än bara verktyg*. Retrieved April 15, 2005, from <http://www.ontime.nu> (in Swedish)
- Nickel, U., Niere, J., & Zundorf, A. (2000). The FUJABA environment. In *Proceedings of the 2000 International Conference on Software Engineering: ICSE 2000, the New Millennium* (pp. 742-745). New York: ACM Press.
- Obrenovic, Z., & Starcevic, D. (2004). Modeling multimodal human-computer interaction. *IEEE Computer*, 37(9), 65-72.
- Ortenburger, R. (2003, August). Software modeling with UML and the KDE Umbrello tool: One step at a time. *Linux Magazine*, 40-42.
- OMG. (2000a). *OMG-XML Metadata Interchange (XMI) Specification, version 1.0*. Retrieved April 15, 2005, from <http://www.omg.org/docs/formal/00-06-01.pdf>
- OMG. (2000b). *OMG-XML Metadata Interchange (XMI) Specification, version 1.1*. Retrieved April 15, 2005, from <http://www.omg.org/docs/formal/00-11-02.pdf>
- OMG. (2002). *XML Metadata Interchange (XMI) Specification, version 1.2*. Retrieved April 15, 2005, from <http://www.omg.org/cgi-bin/doc?formal/2002-01-01>
- OMG. (2003) *XML Metadata Interchange (XMI) Specification, version 2.0*. Retrieved April 15, 2005, from <http://www.omg.org/docs/formal/03-05-02.pdf>
- OMG News. (2002). *OMG News: The architecture for a connected world*. Retrieved April 15, 2005, from <http://www.omg.org>
- Robbins, J. E., & Redmiles, D. F. (2000). Cognitive support, UML adherence, and XMI interchange in Argo/UML. *Information and Software Technology*, 42(2), 79-89.
- Statskontoret. (2003). *Free and open source software – A feasibility study 2003:8a*. Retrieved April 15, 2005, from <http://www.statskontoret.se/upload/Publikationer/2003/200308A.pdf>
- Stevens, P. (2003). Small-scale XMI programming: A revolution in UML tool use? *Automated Software Engineering*, 10(1), 7-21.
- Süß, J. G., Leicher, A., Weber, H., & Kutsche, R.-D. (2003). Model-centric engineering with the evolution and validation environment. In P. Stevens, J. Whittle, & G. Booch (Eds.), *Proceedings of UML 2003 — The Unified Modelling Language: Modelling Languages and Applications* (pp. 31-43). Berlin: Springer-Verlag.
- Zhao, L., & Elbaum, S. (2003). Quality assurance under the open source development model. *Journal of Systems and Software*, 66(1), 65-75.

APPENDIX

Figure 4a. Screen shot from Model A in Rhapsody

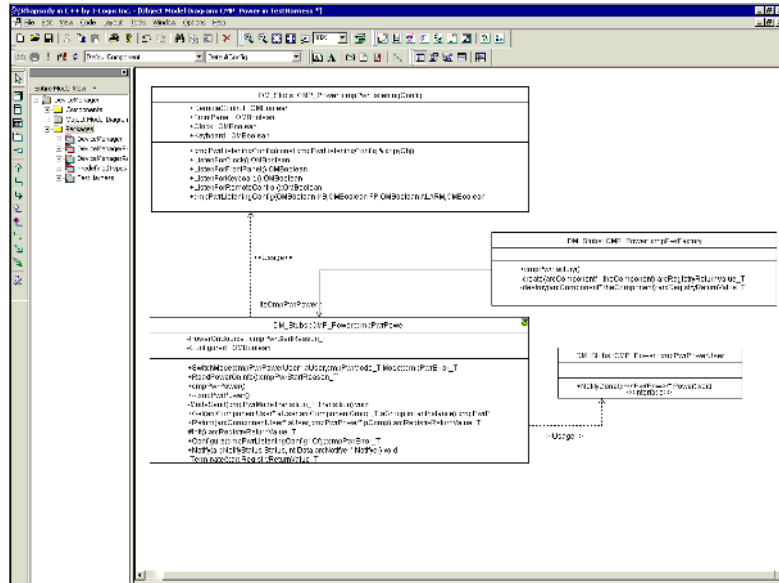
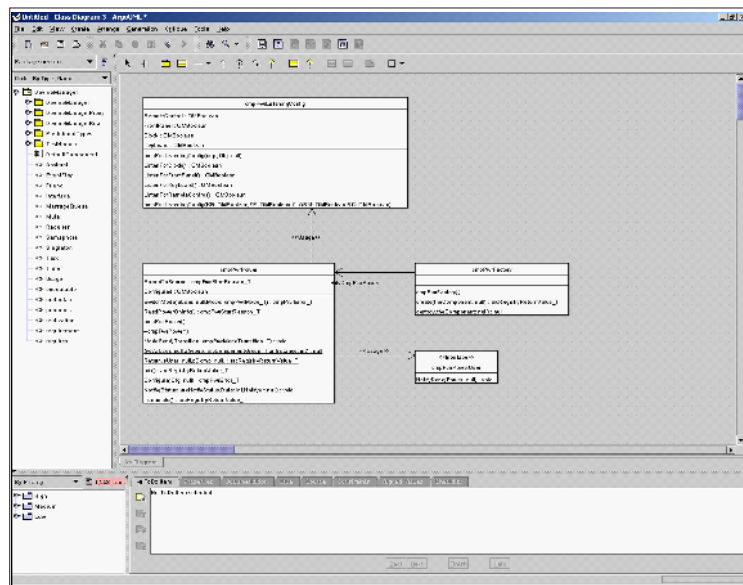


Figure 4b. Screen shot from Model A in Argo



Chapter III

Classification as Evaluation: A Framework Tailored for Ontology Building Methods

Sari Hakkarainen, Norwegian University of Science and Technology, Norway

Dariusz Strasunskas, Norwegian University of Science and Technology,
Norway, & Vilnius University, Lithuania

Lillian Hella, Norwegian University of Science and Technology, Norway

Stine Tuxen, Bekk Consulting, Norway

ABSTRACT

Ontology is the core component in Semantic Web applications. The employment of an ontology building method affects the quality of ontology and the applicability of ontology language. A weighted classification approach for ontology building guidelines is presented in this chapter. The evaluation criteria are based on an existing classification scheme of a semiotic framework for evaluating the quality of conceptual models. A sample of Web-based ontology building method guidelines is evaluated in general and experimented with using data from a case study in particular. Directions for further refinement of ontology building methods are discussed.

INTRODUCTION

The vision for the next generation Web is the *Semantic Web* (Berners-Lee, Handler, & Lassila, 2001), where information is accompanied by meta-data about its interpretation so that more intelligent information-based services can be provided. A core component in Semantic Web applications will be ontologies. An ontology can be seen as an explicit representation of a shared conceptualization (Gruber, 1993) that is formal (Uschold & Gruninger, 1996), and will thus encode the semantic knowledge enabling the sophisticated services. The quality of a Semantic Web application will thus be highly dependent on the quality of its underlying ontology. The quality of the underlying ontology will again depend on factors such as (1) the appropriateness of the language used to represent the ontology, and (2) the quality of the method guidelines provided for building the ontology by means of that language. There are also other factors, such as the complexity of the specific task at hand and the competence of the persons involved.

With a small number of developers, the need for rigid method guidelines may be smaller than for larger projects. Similarly, with highly skilled modelling experts, the need for method guidelines may be smaller than for less experienced people. Method guidelines can thus be seen as an important means to make ontology building possible for a wider range of developers, for example, not only for a few expert researchers in the ontology field but also for companies wanting to develop Semantic Web applications for internal or external use.

However, the current situation is that while many ontology representation languages have been proposed, there is much less to find in terms of method guidelines for how to use these languages — especially for the newer Web-based ontology specification languages. Similarly, if there is little about method guidelines for Web ontology building, there is even less about evaluating the appropriateness of these method guidelines. As observed not only for Web ontology building but also for conceptual modelling in general, there is an “abundance of techniques (and lack of comparative measures)” (Gemino & Wand, 2003, p. 80).

The quality of the interoperation and views management will depend on the quality of the used ontology. The quality of the underlying ontology will, in turn, depend on factors such as (1) the appropriateness of the language used to represent the ontology, and (2) the quality of the engineering environment, including tool support and method guidelines for creating the ontology by means of that language. Method guidelines can thus be seen as an important means to make ontology building possible for a wider range of developers, for example, not only for a few expert researchers in the ontology field but also for companies wanting to develop an ontology for internal or external use.

The objectives of this chapter are to inspect available method guidelines for Web-based ontology specification languages and to evaluate these method guidelines using a coherent framework. The rest of the chapter is structured as follows. The next section describes related work, followed by a section describing a classification framework. Then, the existing method guidelines and their means to achieve quality goals are analyzed in general. A case study taken from industry is then presented where the method guidelines are evaluated in particular. Finally, the chapter concluded with suggested directions for future work and for further refinement of ontology building method guidelines.

RELATED WORK

Related work for this chapter comes from two sides: (a) work on ontology representation languages and method guidelines for these, and (b) work on evaluating conceptual modelling approaches (i.e., languages, method guidelines, and tools). The intersection between these two is limited; the work on Web ontology languages has contained little about evaluation, and the work on evaluating conceptual modelling approaches has concentrated on mainstream approaches for systems analysis and design. However, the newer Web-based ontology languages are becoming mature enough to allow comparative analysis of their guidelines, given a suitable instrument.

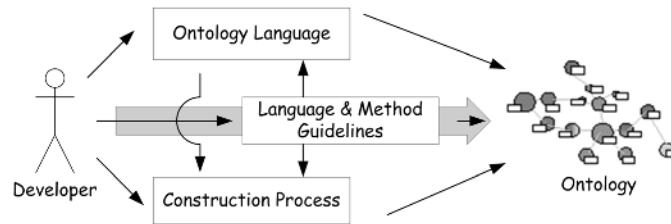
During the last decade, a number of ontology representation languages have been proposed. The so-called traditional ontology specification languages include: CycL (Lenat & Guha, 1990), Ontolingua (Gruber, 1993), F-logic (Kifer, Lausen, & Wu, 1995), CML (Schreiber, Wielinga, Akkermans, van de Velde, & de Hoog, 1994), OCML (Shadbolt, Motta, & Rouge, 1993), Telos (Mylopoulos, Borgida, Jarke, & Koubarakis, 1990), and LOOM (MacGregor, 1991). There are Web standards that are relevant for ontology descriptions for Semantic Web applications, such as XML and RDF. Finally, there are the newer Web ontology specification languages that are based on the layered architecture for the Semantic Web, such as OIL (Decker et al., 2000), DAML+OIL (Horrocks, 2002), XOL (Karp, Chaudhri, & Thomere, 1999), SHOE (Luke & Heffin, 2000), and OWL (Antonioni & van Harmelen, 2003). The latter are in the focus of this study.

There exist several methodologies to guide the process of Web ontology building, which vary in both generality and granularity. Some of the methodologies describe an overall ontology development process yet do not provide details on the ontology creation. Such methodologies are primarily intended to support the knowledge elicitation and management of the ontologies in a basically, centralised environment:

- Fernández, Gómez-Pérez, and Juriso (1997) propose an evolving prototype methodology with six states as ontology life-cycle and include activities related to project management and ontology management.
- Uschold (1996) proposes a general framework for the ontology building process consisting of four steps including quality criteria for ontology formalisation.
- Sure and Studer (2002) propose an application-driven ontology development process in five steps, emphasizing the organisational value, integration possibilities, and the cyclic nature of the development process.

The above methodologies provide only a few user guidelines for carrying out the steps and for creating the ontology. Yet, in order to increase the number and scale of practical applications of the Semantic Web technologies, the developers need to be provided with detailed instructions and general guidelines for ontology creation. A limited selection of method guidelines were found for the newer Web-based ontology specification languages, which are at the foci of this study:

- Knublauch, Musen, and Noy (2003) present a tutorial containing method guidelines for making ontologies in the representation language Web Ontology Language (OWL) by means of the open-source ontology editor Protégé.
- Denker (2003) presents a user guide with method guidelines for making ontologies in the representation language DAML+OIL, again by means of Protégé.

Figure 1. Factors that affect a final ontology

- Noy and McGuinness (2001) present method guidelines for making ontologies called “Ontology Development 101.” Unlike the previous two, this method is independent of any specific representation language.

There are several factors that affect the quality of ontology. Most difficult to control are human factors. A developer constructs the ontology based on individual perception and interpretation of reality, experience, and perception of model quality. The human factors influence the use of the ontology language through the construction process and, consequently, the resulting ontology (see Figure 1). Different ontology languages may incur different ontology variations because of differences in their expressive power and set of constructs used. The ontology construction process is related to ontology language but does not depend on it. Both the chronological order of the ontology building activities and the rules applied for mapping the entities and phenomena from UoD to ontological constructs are important aspects in the ontology construction process. Usually, Web ontology languages do not entail precise rules that define how to map real-world phenomena into the ontological constructs. Thus, method guidelines are important for the quality of ontology, as the guidelines explain how language constructs should be used and define stepwise the construction process.

As for evaluation of ontology specification approaches, a comprehensive evaluation of representation languages was done by Su and Ilebrette (2005), covering all the languages mentioned above except OWL. The paper also evaluates some tools for ontology building: Ontolingua, WebOnto, WebODE, Protégé 2000, OntoEdit, and OilEd. Similarly, Davies, Green, Milton, and Rosemann (2005) and Gómez-Pérez and Corcho (2002) evaluate various ontology languages. These studies concentrate on evaluating the representation languages (and partly tools), not hands-on instructions or ontology building guidelines. Given the argumentation above, such studies are targeting the audience of highly skilled modelling experts rather than the wide spectrum of potential developers of Semantic Web applications.

In the field of conceptual modelling, there are, however, a number of frameworks suggested for evaluating modelling approaches in general. For instance, the Bunge-Wand-Weber ontology (Wand & Weber, 1990) has been used on several occasions as a basis for evaluating modelling techniques, for example, NIAM (Weber & Zhang, 1996) and UML (Opdahl & Henderson-Sellers, 2002), as well as ontology languages in Davies

et al. (2005). The semiotic quality framework first proposed in Lindland, Sindre, and Sølvsberg (1994) for the evaluation of conceptual models has later been extended for evaluation of modelling approaches and used for evaluating UML and RUP (Krogstie, 2001). This framework was also the one used in the evaluation of ontology languages and tools in Su and Illebrekke (2005). The framework suggested by Pohl (1994) is particularly meant for requirements specifications, but is still fairly general. There are also more specific quality evaluation frameworks, for example, Becker, Rosemann, and von Uthmann (1999) for process models, and Moody, Shanks, and Darke (1998) and Schuette (1999) for data / information models.

The framework used in Krogstie (2001) builds on an earlier framework described by Lindland, Sindre, and Sølvsberg (1994). This early version distinguished between three *quality categories* for conceptual models (syntactic, semantic, pragmatic) according to steps on the semiotic ladder (Falkenberg et al., 1997). The *quality goals* corresponding to the categories were syntactic correctness, semantic validity and completeness, and comprehension (pragmatic). The framework also took care to distinguish between goals and *means to reach the goals* (where, e.g., various types of method guidelines would be an example of the latter). In later extensions by Krogstie, more quality categories have been added so that the entire semiotic ladder is included, for example, *physical, empirical, syntactic, semantic, pragmatic, social, and organizational quality*.

Here, the framework is used for evaluating something different, namely, *method guidelines* for ontology building. Moreover, an interesting question is to which extent it is suitable for this new evaluation task, so customizations to the framework are suggested in order to improve its relevance for evaluating method guidelines in general, and method guidelines for ontology building in particular. The framework has been adapted to evaluating specification languages by means of five categories (Krogstie, 1995) adopted for evaluation of method guidelines as follows.

CLASSIFICATION OF ONTOLOGY BUILDING METHODS

As argued in the introduction above, the developers typically need instructions and guidelines for ontology creation in order to support the learning and cooperative deployment of the Semantic Web enabling languages in practice. Krogstie (1995) describes a methodology classification framework consisting of seven categories: *weltanschauung, coverage in process, coverage in product, reuse of product and process, stakeholder participation, representation of product and process, and maturity*. We use the categories for classification of the ontology building method guidelines.

The principle modification here is that the concept of application system (as the end product of the development process) is consequently replaced by ontology (as the end product of applying the method guidelines). In the following, the adapted criteria for each category are described briefly and the method guidelines are classified accordingly.

The experiences from the case study (Hella & Tuxen, 2003) suggested that numerical values could be used for the classification and thus qualify weighted selection techniques such as the PORE methodology (Maiden & Ncube, 1998). Therefore, we adapt PORE methodology here and define the coverage weights -1, 1, and 2 for each category. The method guidelines are classified accordingly in the next section.

Let CF be a classification framework such that CF has a fixed set ζ of categories ζ , where $\zeta = \{\zeta_1, \zeta_2, \zeta_3, \zeta_4, \zeta_5, \zeta_6, \zeta_7\}$ and $\zeta_i \in \zeta$. Each ζ is a quadruple $\langle id, descriptor, C, cw \rangle$, where id is the name of the category, $descriptor$ is a natural language description, C is a set of selection criteria c , and cw defines a function of S that return -1, 1, or 2 as coverage weight, where S is a set of satisfied elements c in the selection criteria C of each category in ζ . Intuitively, we define a number of selection criteria alongside an associated coverage weight function for each category in the classification framework. The categories are as follows.

Weltanschauung describes the underlying philosophy or view to the world. For a method, we may examine why the ontology construction is addressed in a particular way in a specific methodology. In the FRISCO report (Falkenberg et al., 1997), three different views are described: the objectivistic, the constructivistic and the mentalistic view. The objectivistic view claims that reality exists independently of any observer. The relation between reality and the model is trivial or obvious. The constructivistic view claims that reality exists independently of any observer, but what each person possesses is only a restricted mental model. The relationship between reality and models of this reality are subject to negotiations among the community of observers and may be adapted from time to time. The mentalistic view claims that reality and the relationship to any model is totally dependent on the observer. We can only form mental constructions of our perceptions. In many cases, when categorizing a method, the Weltanschauung will not be stated directly, but exists indirectly. Weltanschauung can be $\zeta_1 c_1$ — *explicit*, that is, stated in the document, $\zeta_1 c_2$ — *implicit*, that is, derivable from the documentation, or $\zeta_1 c_3$ — *undefined*, that is, non derivable.

$$cw_1(S_1) = \begin{cases} 2, & \text{if } \zeta_1 c_1 \in S_1. \\ 1, & \text{if } \zeta_1 c_2 \in S_1. \\ -1, & \text{if } \zeta_1 c_3 \in S_1. \end{cases} \quad (1)$$

Coverage in process concerns the method's ability to address $\zeta_2 c_1$ — *planning for changes*, $\zeta_2 c_2$ — *single and co-operative development* of ontology or aligned ontologies, which includes analysis, requirements specification, design, implementation and testing; $\zeta_2 c_3$ — *use and operations* of ontologies; $\zeta_2 c_4$ — *maintaining and evolution* of ontologies; and $\zeta_2 c_5$ — *management of planning, development, operations, and maintenance* of ontologies.

$$cw_2(S_2) = \begin{cases} -1, & \text{if } |S_2| = 0. \\ 1, & \text{if } 0 < |S_2| \leq 2. \\ 2, & \text{if } 2 < |S_2| \leq 5. \end{cases} \quad (2)$$

Coverage in product is described as the method that concerns planning, development, usage and maintenance of and operates on $\zeta_3 c_1$ — *one single ontology*, $\zeta_3 c_2$ — *a family of related ontologies*, $\zeta_3 c_3$ — *a whole portfolio of ontologies* in an organization,

and $\zeta_3\zeta_4$ — a totality of the goals, business process, people and technology used within the organization.

$$cw_3(S_3) = \begin{cases} -1, & \text{if } |S_3| = 0. \\ 1, & \text{if } 0 < |S_3| \leq 2. \\ 2, & \text{if } 2 < |S_3| \leq 4. \end{cases} \quad (3)$$

Reuse of product and process is important to avoid re-learning and recreation. A method may support reuse of ontologies as products or reuse of method as processes. There are six dimensions of reuse:

- $\zeta_4\zeta_1$ — *Reuse by motivation* answers the question “Why is reuse done?” Different rationale may be, for example, productivity, timeliness, flexibility, quality, and risk management goals.
- $\zeta_4\zeta_2$ — *Reuse by substance* answers the question, “What is the essence of the items to be reused?” A product is a reuse of all the deliverables that are produced during a project, such as models, documentation, and test cases. Reusing a development or maintenance method is process reuse.
- $\zeta_4\zeta_3$ — *Reuse by development scope* answers the question, “What is the coverage of the form and extent of reuse?” The scope may be either external or internal to a project or organization.
- $\zeta_4\zeta_4$ — *Reuse by management mode* answers the question, “How is reuse conducted?” The reuse may be planned in advance with existing guidelines and procedures defined, or it can be ad hoc.
- $\zeta_4\zeta_5$ — *Reuse by technique* answers the question, “How is reuse implemented?” The reuse may be compositional or generative.
- $\zeta_4\zeta_6$ — *Reuse by intentions* answers the question, “What is the purpose of reused elements?” There are different intentions of reuse. The elements may be used as they are, slightly modified, used as a template, or just used as an idea.

$$cw_4(S_4) = \begin{cases} -1, & \text{if } 0 < |S_4| \leq 2. \\ 1, & \text{if } 2 < |S_4| \leq 4. \\ 2, & \text{if } 4 < |S_4| \leq 6. \end{cases} \quad (4)$$

Stakeholder participation reflects the interests of different actors in the ontology building activity. The stakeholders may be categorized into those who are $\zeta_5\zeta_1$ — *responsible for developing* the method, those with $\zeta_5\zeta_2$ — *financial interest*, and those who have $\zeta_5\zeta_3$ — *interest in its use*. Further, there are different forms of participation. *Direct* participation means every stakeholder has an opportunity to participate. *Indirect* participation uses representatives; thus every stakeholder is represented through other representatives that are supposed to look after his or her interests.

$$cw_5(S_5) = \begin{cases} -1, & \text{if } |S_5| = 0. \\ 1, & \text{if } 0 < |S_5| \leq 1. \\ 2, & \text{if } 1 < |S_5| \leq 3. \end{cases} \quad (5)$$

Representation of product and process can be based on linguistic and non-linguistic data such audio and video. Representation languages can be $\zeta_6 c_1$ — *informal*, $\zeta_6 c_2$ — *semi-formal*, or $\zeta_6 c_3$ — *formal*, having logical or executorial semantics.

$$cw_6(S_6) = \begin{cases} -1, & \text{if } |S_6| = 1. \\ 1, & \text{if } |S_6| = 2. \\ 2, & \text{if } |S_6| = 3. \end{cases} \quad (6)$$

Maturity is characterized on different levels of completion. Some methodologies have been used for a long time; others are only described in theory and never tried out in practice. Several conditions influence maturity of a method, namely, if the method is $\zeta_7 c_1$ — *fully described*, if the method lends itself to $\zeta_7 c_2$ — *adaptation, navigation and development*, if the method is $\zeta_7 c_3$ — *used and updated through practical applications*, if it is $\zeta_7 c_4$ — *used by many organizations*, and if the method is $\zeta_7 c_5$ — *altered* based on experience and scientific study of its use.

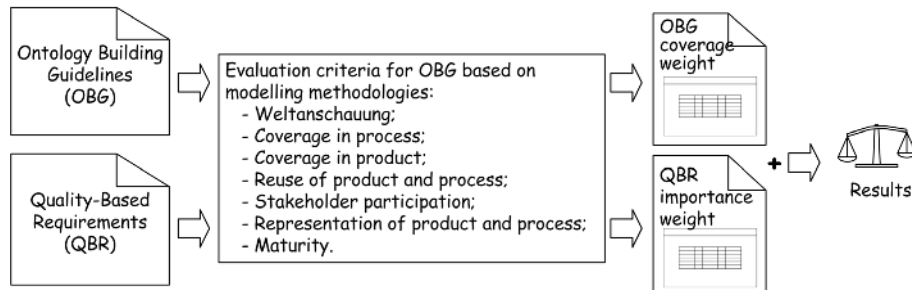
$$cw_7(S_7) = \begin{cases} -1, & \text{if } |S_7| = 0. \\ 1, & \text{if } 0 < |S_7| \leq 3. \\ 2, & \text{if } 3 < |S_7| \leq 5. \end{cases} \quad (7)$$

The selection criteria are exhaustive and mutually exclusive in the categories ζ_i and ζ_6 , and exhaustive in ζ_5 , whereas the set of satisfied criteria S of the remaining categories may also be the empty list $\{\}$. The coverage weight cw is independent of any category-wise prioritisation. Since the intervals are decisive for the coverage weight, they can be adjusted based on preferences of the evaluator. However, when analysing different evaluation occurrences, the intervals need to be fixed in comparison, but may be used as dependent variables.

METHOD GUIDELINES FOR ONTOLOGY BUILDING: GENERAL COVERAGE

Three method guidelines among the newer Web-based ontology specification languages are categorized, namely, that presented by Knublauch, Musen, and Noy (2003), which is based on OWL and Protégé; that of Denker (2003) which is based on DAML+OIL and Protégé; and that of Noy and McGuinness (2001), which is language

Figure 2. The approach for the ontology building guidelines classification and evaluation



independent yet uses Protégé in the examples. Protégé20001 is an open-source ontology editor developed at Stanford University and built with Java technology. All the method guidelines meet the selection criteria as supporting Semantic Web enabling language(s) and assume RDF/XML notation as the underlying Web standard.

The evaluation framework presented by Krogstie (2001) provides a means to evaluate quality and development perspectives of a methodology dependent on a specific ontology language. As illustrated in Figure 2, the framework provides some guidelines of what may be contained in an evaluation process. Different levels of appropriateness allow important aspects to be considered and make it possible to consider important aspects such as the domain to be modelled, the participants' previous knowledge, and the extent to which participants are able to express their knowledge. Each method guideline is shortly described and characterised in the sequel followed by an analysis of the observations and an explanation to the table.

The classification according to the Krogstie (1995) categories is summarized in Table 1, where the columns are the classification criteria as above and the rows are the method guidelines where the intersection describes how the method covers the criteria. Each method guideline is shortly described and characterised in the sequel, followed by an analysis of the observations and an explanation to the table.

- **OWL-Tutorial** (Knublauch, Musen, & Noy, 2003) is a tutorial that was originally created for the 2nd International Semantic Web Conference. The ontology building method is based on OWL as the ontology application language and assumes Protégé as the ontology development tool. The ontology building process consists of seven iterative steps: determine scope, consider reuse, enumerate terms, define classes, define properties, create instances, and classify ontology. *Overall comment:* The development activity requires some experience and foresight, communication between domain experts and developers, and a tool that is considered easy-to-understand, yet powerful, including support of ontology evolution.
- **DAML+OIL Tutorial** (Denker, 2003) is a user's guide of the DAML+OIL plug-in for Protégé2000. The ontology building method is based on DAML+OIL as the ontology application language and Protégé as the ontology development tool. The

Table 1. Classification of method guidelines

Evaluation criteria	Category name	Coverage weight
Explanation		
DAML+OIL - Tutorial		
ζ ₁	Weltanschauung	-1
Undefined. The method does not explicitly state its worldview, and it is not possible to implicitly deduce the worldview.		
ζ ₂	Coverage in process	1
The method contains no explicit description of the development process, yet the sequence of the sections in the documentation indicates how to proceed in order to create an ontology. The importance of reuse is not covered, and it does not describe how to plan for changes. The evolution and use of Protégé are described. The coherence between the development tool and the ontology language is considered.		
ζ ₃	Coverage in product	1
A single ontology. However, it describes situations where the user would like to import concepts created in another ontology. The method does not allow references to resources located in another ontology except for four explicitly stated URIs (see the discussion that follows the table).		
ζ ₄	Reuse of product and process	-1
Considers only the technical aspect of reuse and describes only the import of DAML+OIL files.		
ζ ₅	Stakeholder participation	-1
The tutorial is available through the Artificial Intelligence Center at SRI International, and is linked through the DAML homepage. The physical editor(s)/author(s) are unknown other than the contact person regarding the plug-in and the user guide.		
ζ ₆	Representation of product and process	-1
The document is basically written in natural language on top of screenshots that explain the ontology building method with Protégé. The user/participant does not need to be aware of the underlying syntax of the ontology language.		
ζ ₇	Maturity	2
The tutorial is based on DAML+OIL as ontology language, released in December 2000. It has been subject of evaluation. Protégé is used by a large community and is a well-examined system. The method is not complete. The method guideline describes the uncovered or unimplemented functionalities.		
OWL - Tutorial		
ζ ₁	Weltanschauung	2
Constructivistic. The first step in the development method is to determine the scope. By doing that, the domain that is to be covered in the ontology will be explicitly stated. The method states that communication between domain experts and developers is necessary.		
ζ ₂	Coverage in process	2
Defines seven iterative steps. It has a detailed yet unstructured and incomplete description of ontology development. The first three steps — determine scope, consider reuse and enumerate terms—are just mentioned. The tool guidance does not follow the steps in the building process, but is presented rather ad hoc. There are no explicit procedures to prepare for changes.		
ζ ₃	Coverage in product	1
Protégé is described as a toolset for constructing ontologies that is scalable to very large knowledge bases and enables embedding of stand-alone applications in the Protégé knowledge environment. It does not describe the relationship between heterogeneous ontologies, nor the requirements the tool should fulfill prior to use in larger context.		
ζ ₄	Reuse of product and process	1
The tutorial considers reuse partially in the ontology building activity. The <i>development scope</i> and <i>technical</i> prerequisite of reuse are covered, but not why, when, or how to consider reuse. It does not provide examples of how reuse is carried out in practice. It describes how to import existing OWL files that are developed with another tool or developed with some previous version of Protégé. It lists formats from which ontologies may be read (imported), written to (exported), or inter-converted.		

Table 1. continued

Ç5	Stakeholder participation	2
The tutorial is comprehensible for inexperienced stakeholders with development or financial interests and supports the interests of novice users/participants. Since it is written by those <i>responsible for developing</i> the tool, the guide has a deep and detailed description of practical use. Several members of the user community, namely, those who have <i>interest in its use</i> , have contributed to the method <i>indirectly</i> through material such as visualization systems, inference engines, means of accessing external data sources, and user-interface features.		
Ç6	Representation of product and process	1
It is mostly <i>informal</i> , written in natural language yet presents a narrow description of the Semantic Web and ontologies. On the <i>visual</i> part, it has a multitude of screenshots that explain and make the semi-structured tool concepts and the <i>formal</i> language elements comprehensible. The development process is covered in a graphical representation, yet not explained. In overall, the method is mostly <i>informal</i> and provides feasible graphical representation.		
Ç7	Maturity	1
The tutorial is based on OWL, the newest contribution in this field. The language itself has hardly been examined yet. However, guidance for OWL modeling benefits from experiences with guidelines for Protégé, RDF, and OIL. The plug-in that is used in Protégé is also new, but the core Protégé is well-examined. The method covers the latest release, and is up-to-date in regarding both the language and the tool. The method is not complete, since not all the steps in the development process are fully described.		
Ontology development 101		
Ç1	Weltanschauung	2
Constructivistic. It presents a list of different reasons for creating an ontology, for example, to make domain assumptions explicit. The method argues that an explicit specification is useful for new users.		
Ç2	Coverage in process	2
It covers seven iterative steps, each of which is described in detail. For example, there are several guidelines for developing a class hierarchy. This feature provides participants with a checklist to avoid mistakes such as creating cycles in a class hierarchy. It has good coverage in process. Reuse is considered, but there is no plan for changes. The actual implementation of an ontology is not covered.		
Ç3	Coverage in product	1
The method is an initial guide to help creating a single new ontology. There is awareness of the possible integration to other ontologies and applications. Further, translating an ontology from one formalism to another is not considered a difficult task; however, instructions for this are not provided.		
Ç4	Reuse of product and process	2
It covers reuse in Step 2. Reusing existing ontologies is a requirement if the system needs to interact with applications that have already committed to some ontologies. Reuse is not fully covered, yet references to available libraries of ontologies are given.		
Ç5	Stakeholder participation	1
The method guideline provides introduction to ontologies and describes why they are necessary. The method is suitable for experienced as well as novice participants since it mainly uses informal languages, yet provides comprehensive descriptions.		
Ç6	Representation of product and process	1
It makes no explicit reference to any specific ontology language. It is written in natural language, with only a few logical or executable statements. The language is informal and the method offers adequate description of each concept. There are illustrations based on screenshots from Protégé to support comprehensibility. A semi-structured scenario is given and used as a reference throughout the guideline.		
Ç7	Maturity	2
Published in 2001. Many researchers in the field reference the method guideline, many readers examine it, and acknowledged Web sites such as the Protégé Web site provide hyperlinks to it. The method does not claim it has been tried out in practice, but several projects that use the method can be located by searching on the Web. However, it has not been updated in response to such experiences.		

ontology building process consists of three basic steps: create a new ontology, load existing ontologies, and save ontology. The creation of new ontology consists of five types of instructions: define classes, properties (slots), instances, restrictions, and Boolean combinations. *Overall comment:* The method does not contain any explicit description of the development process. However, the sequence of the sections in the documentation indicates how to create an ontology.

- **Ontology Development 101** (Noy & McGuinness, 2001) is a guide to building ontologies. The ontology building method is ontology application language independent and ontology development tool independent, yet it uses Protégé in the examples. The ontology building process consists of seven iterative steps: determine the domain and scope of the ontology, consider reusing existing ontologies, enumerate important terms in the ontology, define the classes and the class hierarchy, define the properties of classes — slots, define the facets of the slots, and create instances. *Overall comment:* The methodology provides three fundamental rules that are used to make development decisions: (1) there is no correct way to model a domain, (2) ontology development is necessarily an iterative process, and (3) concepts in the ontology should be close to objects, physical or logical, and relationships in the domain of interest.

The *Weltanschauung* is similar in the studied methods. *OWL-Tutorial* is based on *constructivistic* worldview. The first step in the development method is to determine the scope. By doing that, the domain that is to be covered in the ontology will be explicitly stated. Further, the method states that communication between domain experts and developers are necessary. *DAML+OIL-Tutorial* is based on *undefined* worldview. The method does not explicitly state its worldview, and it is not possible to implicitly deduce the worldview. The method does not describe the term ontology, and it does not describe why an ontology is needed. *Ontology Development 101* is based on *constructivistic* worldview. It presents a list of different reasons for creating an ontology, for example, to make domain assumptions explicit. The method argues that an explicit specification is useful for new users. Thus, there is a need for explanation, where the relation between the domain and the model is not obvious.

The *coverage in process* varies clearly between the methods. *OWL-Tutorial* covers seven iterative steps. It has a detailed yet unstructured and incomplete description of ontology development. The first three steps — determine scope, consider reuse and enumerate terms — are just mentioned. It describes the evolution and use of Protégé. The tool guidance does not follow the steps in the building process, but is presented rather ad hoc. There are no explicit procedures to prepare for changes. The process is described as iterative, which indicates the method awareness of and the need for modification. *DAML+OIL-Tutorial* covers three plus five steps. It has an unstructured and incomplete description of ontology development. The method contains no explicit description of the development process, yet the sequence of the sections in the documentation indicates how to proceed in order to create an ontology. A detailed yet incomplete description of how to create a DAML+OIL ontology with Protégé is provided. The importance of reuse is not covered and it does not describe how to plan for changes. It describes the evolution and use of Protégé. It links to the syntax of DAML+OIL when its concepts in the development are described. Further, the coherence between the development tool and

the ontology language is considered important, that is, resolving differences between the concepts of DAML+OIL and the representation in Protégé. There are explicit rules, for example, that DAML+OIL properties are mapped to Slots in Protégé. *Ontology Development 101* covers seven iterative steps, each of which is described in detail. It has a good coverage in process. For example, Step 1 (determine the domain and scope) is illustrated in different domains, and the competency questions technique is suggested as a method to determine the scope. Reuse is considered, but there is no plan for changes. The actual implementation of ontology is not covered. The method is an initial guide to help create a single new ontology. It provides three fundamental rules in ontology design in order to make decisions. The process steps are covered in sufficient detail. For example, there are several guidelines for developing a class hierarchy. This feature provides participants with a checklist to avoid mistakes such as creating cycles in a class hierarchy.

The *coverage in product* is low (covers a single ontology) in both *DAML+OIL-Tutorial* and *Ontology Development 101*. *OWL-Tutorial* includes an example scenario that describes the use of ontologies in relation to agents with reasoning mechanisms. It has medium coverage in product. Protégé is described as a toolset for constructing ontologies that is scalable to very large knowledge bases and enables embedding of stand-alone applications in the Protégé knowledge environment. It does not describe the relationship between heterogeneous ontologies nor the requirements the tool should fulfill prior to use in larger context. It refers to yet does not explain description logics. *DAML+OIL-Tutorial* describes situations where the user would like to import concepts created in another ontology. The method does not allow references to resources located in another ontology except for four explicitly stated URIs: <http://www.daml.org/2001/03/daml+oil#>; <http://www.w3.org/1999/02/22-rdf-syntax-ns#>; <http://www.w3.org/2000/01/rdf-schema#>; and <http://www.w3.org/2000/10/XMLSchema#>. The method covers single ontology. *Ontology Development 101* regards an ontology as a model of reality and the concepts in the ontology must reflect this reality. It mentions projects built with ontologies, and ontologies developed for specific domains and existing broad general-purpose ontologies. Reuse is considered important if the ontology owner needs to interact with other applications that have committed to particular ontologies or controlled vocabularies. Thus, there is an awareness of the possible integration to other ontologies and applications. Further, translating an ontology from one formalism to another is not considered a difficult task. Instructions for this are not provided.

The *reuse of product and process* varies among the methods. *OWL-Tutorial* considers reuse partially in the ontology building activity. The *development scope* and *technical* prerequisite of reuse are covered. It does not describe why, when, or how to consider reuse. It does not provide examples of how reuse is carried out in practice. It describes how to import existing OWL files that are developed with another tool or developed with some previous version of Protégé. It also lists formats from which ontologies may be read (imported), written to (exported), or inter-converted (transformed). *DAML+OIL-Tutorial* only consider the technical aspect of reuse. It explains how to import existing DAML+OIL files that are developed with another tool or developed with a previous version of Protégé. The process is described with images that guide the participants. However, the support tool, that is, the plug-in, only reads DAML+OIL ontologies and only allows such files to be manipulated and saved. This is

a drawback and reduces the opportunity for reuse. *Ontology Development 101* covers reuse in Step 2 that is called “consider reusing existing ontologies.” Reusing existing ontologies may be a requirement if the system needs to interact with other applications that have already committed to particular ontologies. If considering the assumption that no relevant ontologies exist, we might conclude that reuse is not covered yet, for example, references to available libraries of ontologies are given.

The *stakeholder participation* further discriminates the methods. *OWL-Tutorial* was developed by members of the Protégé team at the Stanford University School of Medicine. The method assumes use of Protégé and provides a number of screenshots from the development tool. The tutorial is comprehensible for inexperienced stakeholders with development or financial interests and supports the interests of novice users/participants. Since it is written by those *responsible for developing* the tool, the guide has a deep and detailed description of practical use. Several members of the user community, namely, those who have an *interest in its use*, have contributed to the method *indirectly* through material such as visualization systems, inferencing engines, means of accessing external data sources, and user-interface features. *DAML+OIL-Tutorial* is available through the Artificial Intelligence Center at SRI International, and is linked through the DAML homepage. The physical editor(s)/author(s) are unknown, other than the contact person regarding the plug-in and the user guide. In *Ontology Development 101*, one co-author is a member of the Protégé team and the other is co-editor of the Web Ontology Language (OWL). The method guideline provides an introduction to ontologies and describes why they are necessary. Since it uses mainly informal language yet provides detailed descriptions, we suggest that the method is suitable for experienced as well as novice participants.

The *representation of product and process* is only partially covered in all the methods. *OWL-Tutorial* is based on OWL and Protégé and the representations are influenced by these notations. It is mostly *informal*, written in natural language, yet it presents a narrow description of the Semantic Web and ontologies. On the *visual* part, it has a multitude of screenshots that explain and make the semi-structured tool concepts and the *formal* language elements comprehensible. The development process is covered in a graphical representation yet not explained. Overall, the method is informal and provides feasible graphical representation. *DAML+OIL-Tutorial* is influenced by the representations of DAML+OIL and Protégé. The document is basically written in natural language on top of screenshots that explain the ontology building method with Protégé. The user/participant needs to be aware of the underlying syntax of the ontology language. The document is accessed through links to the different sections that are to be opened/printed separately. The overall language and layout of the methodology are informal. *Ontology Development 101* makes no explicit reference to specific ontology language. It is written in natural languages, with only a few logical or executable statements. The language is informal and the method offers an adequate description of each concept presented. There are illustrations based on screenshots from Protégé that support comprehensibility. A semi-structured scenario is given and used as a reference throughout the paper.

The *maturity* is covered on a medium level in all the methods. *OWL-Tutorial* is based on OWL as the ontology language, which is the newest contribution in this field. However, guidance for OWL modeling benefits from experiences with guidelines for

Protégé, RDF, and OIL. The plug-in that is used in Protégé is also new, but the core Protégé is a well-examined system. The method covers the latest release of the methodology and is up-to-date in both the language used and the development tool. The method is not complete, since not all the steps in the development process are properly described. *DAML+OIL-Tutorial* is based on DAML+OIL as the ontology language, which was released in December 2000. Compared to OWL, it has been available for a while and thus been under evaluation. Protégé is used by a large community and is a well-examined system; however, the method is not complete. As a sign of maturity, the method guideline describes the uncovered or unimplemented functionalities. *Ontology Development 101* was published in March 2001, and is older than the other two method guidelines. It is still valid when using ontology languages developed after the methodology was published, for example, OWL. The method guideline is referenced from many sites on the Web, it has been examined by many readers, and it is referred to from acknowledged sites such as the Protégé Web site. The method does not explicitly state that it has been tried out in practice, but several projects that claim to be using the method can be located by searching on the Web. However, it has not been updated as a response to such experiences.

METHOD GUIDELINE FOR ONTOLOGY BUILDING: THE EDI CASE

The case study is based on edi (engaging, dynamic innovation), which is a system developed by a student project group. edi is intended to support exchange of business ideas between the employees within an oil company, which is an integrated oil and gas company with business operations in 25 countries. At the end of 2002, there were 17,115 employees in the company. Consequently, the amount of information and knowledge provided by the employees is rapidly increasing; thus there is a need for more effective retrieval and sharing of knowledge. edi will become a tool and motivator to generate ideas, as well as enabling the employees to focus on the relevant aspects of their activities.

The overall idea of the edi system is to create a connection for communication and knowledge sharing between employees from different business areas, domain experts, and department managers. The plan is to utilize Semantic Web and Web service technology for that purpose. Ontologies will play a crucial part in edi, supporting common access to information and enabling implementation of Web and ontology-based search. There will be participants with different qualities and knowledge that are experts on creativity and processes that support creativity.

EDI Requirements

The status is that the overall functional requirements for edi have been analysed. However, before the system can be developed, a much more thorough analysis needs to be conducted, and a decision about the purpose of the ontology has to be made. Information about the domain plays an important role in this process. It can be gathered in many ways and, unavoidably, there will be many different participants involved in such a process; for instance, end users as possible idea contributors and people in the edi network evaluating ideas. This can be similar to software development in general, hence

starting with an ontology requirements specification (Davies et al., 2005). Generally, this specification should describe what the ontology should support, sketching the planned area of the ontology application and listing, for example, valuable knowledge sources. The oil industry is in constant change, and the internationality of the company makes the changes even more complex. edi needs to have high durability, be adaptable to changes in the environment, be maintainable, and have high reliability in order to secure the investment. Thus, a careful analysis needs to be made early in the process that places elaborate requirements on the ontology development environment.

Quality-Based Requirements

An ontology should be built in a way that supports automatic reasoning and provides a basis for high quality, Web-based information services. The underlying assumption is that a high quality engineering process assures a high quality end product. The quality of ontology building process depends on the environmental circumstances under which the ontology is used. Further, a model is defined to have high degree of quality if it is developed according to its specifications (Krogstie, 2003). Similarly, a method guideline has high quality degree if it describes a complete set of steps and instructions for how to arrive at a model that is valid with respect to the language(s), it supports.

In the following, the quality requirements are categorized according to the categories of the classification framework (Krogstie, 1995). We adopt the PORE methodology (Maiden & Ncube, 1998) to prioritise the classification criteria based on edi requirements (Hella & Tuxen, 2003) in order to evaluate the ontology building guidelines in this particular situation. Importance weights for each appropriateness category are calculated as follows. Let $R(CF)$ be a set of weighted requirements such that R has a fixed set R_C of categories r_c , where categories in R_C correspond with categories C of an evaluation framework EF , i.e., $R_C = C$, and $c \in C$, $r_c \in R_C$. r_c is a triple $\langle id, descriptor, iw \rangle$, where id is the name of the appropriateness requirement category, $descriptor$ is a natural language description of the appropriateness requirement, and iw_{r_c} defines a function of I that returns 1, 3, or 5 as importance weight based on priorities and policy of the company, where I is a set of importance judged elements r_c in the selection criteria C of each category in R_C .

$$iw_{r_c}(I) = \begin{cases} 0, & \text{if } r_c \text{ may be satisfied, is optional;} \\ 3, & \text{if } r_c \text{ should be satisfied, is recommended;} \\ 5, & \text{if } r_c \text{ must be satisfied, is essential.} \end{cases} \quad (8)$$

Based on the edi requirements, the stakeholder prioritises the evaluation factors according to the quality-based requirements, where an importance weight (0, 3, or 5) is assigned to each appropriateness and classification category as in Equation 8. In Table 2, the columns are requirement category id, name, and importance weight, and every other row is a NL description of the requirement.

In summary, Table 2 shows that the key criteria for meeting edi requirements with high utility are *coverage in process*, *reuse of product and process*, and *representation*

Table 2. Classification of edi requirements

Category of requirements	Category name	Importance weight
Description of requirements		
rζ_1	Weltanschauung	3
Constructivistic worldview — however this is not a crucial requirement. The end users may have different models of the reality depending on, for example, their geographical location or the business area in which they are involved.		
rζ_2	Coverage in process	5
Ontology building method for edi must be extensively covered to support large development teams and heavily illustrated to support inexperienced project participants.		
rζ_3	Coverage in product	0
Development of a single ontology in a stand-alone application may be supported.		
rζ_4	Reuse of product and process	5
Important, must be integrated in the process. Feasible guidance including illustrative examples should be provided. Ontology building method for edi should provide feasible guidance including illustrative examples, and the procedures should be integrated into steps in the development process.		
Rζ_5	Stakeholder participation	3
Ontology building method for edi should cover the participants' development and financial interests of the involved creators of the method, as well as the low experience of its user-group participants.		
rζ_6	Representation of product and process	3
Informal (natural language) representation and rich illustration are important. Independent of the method, the language should cover the required level of formality in the product to support automated reasoning.		
rζ_7	Maturity	3
Ontology building method for edi should be widely adopted and well-examined in order to support evolution, co-operation, and management of the ontology.		

of product and process. The discriminating criteria are *coverage in process*, and *reuse of product and process*, with the assigned importance weight equal to 5. The least discriminating criterion is *coverage in product*, where the weight is equal to 0.

Finally, a total coverage weight Tw is calculated for each ontology building method guideline. Recall the coverage weights (-1, 1, and 2) from Table 1 expressing how well the guidelines satisfy the evaluation factors. Intuitively, the importance weights from Table 2 are multiplied by the coverage weights from Table 1. The total weights in Table 3 are calculated as in Equation 9.

$$Tw_i = \sum_{\zeta \in \zeta} (cw_{\zeta} \times iw_{r\zeta}) \quad (9)$$

On its *Weltanschauung*, an ontology building method for **edi** should be based on a constructivistic view. The end users may have different models of the reality depending on, for example, their geographical location or the business area in which they are involved. Both *OWL-Tutorial* and *Ontology Development 101* meet this requirement, whereas it is undefined for *DAML+OIL-Tutorial*. On its *coverage in process*, an

ontology building method for edi should be extensively covered to support large development teams and heavily illustrated to support inexperienced project participants. Both *OWL-Tutorial* and *M3* meet this requirement — *OWL-Tutorial* partially, whereas it is not well covered by *DAML+OIL-Tutorial*. On its *coverage in product*, an ontology building method for edi should cover a single ontology. Each studied method guides creation of a complete ontology.

When it comes to *reuse of product and process*, an ontology building method for edi should provide feasible guidance including illustrative examples, and the procedures should be integrated into steps in the development process. Both *OWL-Tutorial* and *Ontology Development 101* meet this requirement — *Ontology Development 101* partially, whereas it is not well covered by *DAML+OIL-Tutorial*. On its *stakeholder participations*, an ontology building method for edi should cover the participants' development and financial interests of the involved creators of the method, as well as the low experience of its user-group participants. Both *OWL-Tutorial* and *Ontology Development 101* meet this requirement — *Ontology Development 101* partially, whereas it is not covered or unknown by *DAML+OIL-Tutorial*. On its *representation of product and process*, an ontology building method for edi should cover informal (natural language) representation and rich illustration. Each of the studied methods uses both natural language and rich illustrations to support novice participants. Independent of the method, the language will cover the required level of formality in the product to support automated reasoning. On its *maturity*, an ontology building method for edi should be widely adopted and well-examined in order to support evolution, cooperation, and management of the ontology. Relative to the other methods, *Ontology Development 101* cover best the maturity criteria.

In summary, Table 3 colligates the situated evaluation in favor of *Ontology Development 101*, with the total coverage weight $Tw_{OntDev101} = 38$. Next most relevant is *OWL-Tutorial*, with the score $Tw_{OWL\ Tutorial} = 33$. Moreover, out of the key requirements for edi, the discriminating criteria are *coverage in process*, and *reuse of product and process*. The *Ontology Development 101* tutorial meets the both criteria completely, and *OWL-Tutorial* partially, whereas *DAML+OIL-Tutorial* has shortages in both cases. All the guidelines support *coverage in product* on the level as required for edi ($iw=0$) and

Table 3. Evaluation of method guidelines according to importance of edi requirements

Evaluation criteria	Importance weight (<i>iw</i>)	DAML+OIL-Tutorial		OWL-Tutorial		Ontology development 101	
		Coverage weight (<i>cw</i>)	Total	Coverage weight (<i>cw</i>)	Total	Coverage weight (<i>cw</i>)	Total
¶ ₁	3	-1	-3	2	6	2	6
¶ ₂	5	1	5	2	10	2	10
¶ ₃	0	1	0	1	0	1	0
¶ ₄	5	-1	-5	1	5	2	10
¶ ₅	3	-1	-3	2	6	1	3
¶ ₆	3	-1	-3	1	3	1	3
¶ ₇	3	2	6	1	3	2	6
$Tw_{DAML+OIL\ Tutorial}$			-3	$Tw_{OWL\ Tutorial}$	33	$Tw_{OntDev101}$	38

support the *representation of product and process* in a range, where *Ontology Development 101* and *OWL-Tutorial* meet the requirements completely, and *DAML+OIL-Tutorial* only partially. Out of the remaining categories of edi requirements, *DAML+OIL-Tutorial* fails to meet any of them, *OWL-Tutorial* meets two completely and fails in one, and *Ontology Development 101* meets two completely and one partially. Thus, according to our metrics, *Ontology Development 101* seems most suitable to guide the edi ontology creation.

CONCLUSION

An evaluation of three method guidelines for Semantic Web ontology building was conducted using the framework presented by Hakkarainen, Hella, Tuxen, and Sindre (2004) and Krogstie (1995). Evaluation of method guidelines was performed in two steps: one general evaluation, namely, their applicability for building ontologies in general, and one particular, namely, their appropriateness for ontology development in a real-world project — how applicable is the framework in practice. The main results are as follows:

- The method classification part of the framework (Krogstie, 1995) has potential for evaluating method guidelines. Use of the numerical values for the weights and adoption of the PORE methodology (Maiden & Ncube, 1998) produce the more explicit evaluation results.
- The categorization according to *Weltanschauung*, that is, the applied modelling worldview, was expected to be the same for all the method guidelines, but turned out to be discriminating as selection criteria in the case study. However, the *Weltanschauung* most probably is the same for the studied guidelines, since they support languages that all are constructivistic; it was merely not derivable for one of the guidelines.
- In both steps — the general classification and the evaluation against the situated requirements — the method *Ontology Development 101* (Noy & McGuinness, 2001) came out on top, since it met most of the evaluation criteria. This was also the only method guideline that is independent of any specific representation language and has the longest history.
- Major weaknesses were identified for all the methods, as expected because of the current immaturity of the field of Web-based ontology construction. None of the method guidelines are complete concerning *coverage in product*, whereas all of them cover *representation of product and process* fairly well.

The contribution of this chapter is twofold. First, an existing evaluation framework was tried out with other evaluation objects than has been used for previously. Second, numerical values and metrics were incorporated into the classification framework for the classification, thus supporting qualification of weighted selection. The experimental case study suggests that, given the small adjustments, the framework intended for model classification is applicable in evaluation of method guidelines regardless of whether the classification is used for their selection, quality assurance, or engineering.

The concrete ranking of methods may be of limited use as new ontology languages and method guidelines are developed, the existing languages evolve, and some of them

became more mature. Nevertheless, it can be useful in terms of guiding the current and future creators of such languages and their method guidelines. By drawing attention to the weakness of current proposals, they can be mended in future proposals so that there will be higher quality languages and method guidelines to choose from in the future. The underlying assumption for our work is that high quality method guidelines may increase and widen the range and scalability of the Semantic Web ontologies and applications.

There are several interesting topics for future work, such as supplementing the theoretical evaluations with empirical ones as larger scale Semantic Web applications arise utilizing the empirical nature of Krogstie (1995), as well as evaluating more methods as they emerge, for example, those presented by Knublauch (2004), Pepper (2004), Smith, Welty, and McGuinness (2004). Further possibilities are the investigation of the appropriateness of the formalisation quality criteria presented in Uschold (1996), and unified methodology as a complement to the semiotic quality framework (Lindland, Sindre, & Sølvsberg, 1994) in order to conduct evaluation of the process-oriented methodological frameworks that were out of the scope of this chapter.

REFERENCES

- Antoniou, G., & van Harmelen, F. (2003). Web ontology language: OWL. In S. Staab & R. Studer (Eds.), *Handbook on ontologies in information systems* (pp. 67-92). Berlin: Springer-Verlag.
- Becker, J., Rosemann, M., & von Uthmann, C. (1999). Guidelines of business process modeling. In W. Aalst, J. Desel, & A. Oberweis (Eds.), *Business process management: Models, techniques and empirical studies* (LNCS 1806, pp. 30-49). Springer-Verlag.
- Berners-Lee, T., Handler, J., & Lassila, O. (2001). The semantic Web. *Scientific American*, 34-43.
- Davies, I., Green, P., Milton, S., & Rosemann, M. (2005). Using meta-models for the comparison of ontologies. In *Proceedings of the 8th CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design, (EMMSAD '03)*, Velden, Austria (pp. 16-17).
- Decker, S., Fensel, D., van Harmelen, F., Horrocks, I., Melnik, S., Klein, M., & Broekstra, J. (2000). Knowledge representation on the Web. In *Proceedings of the 2000 International Workshop on Description Logics (DL2000)*, Aachen, Germany. Retrieved February 27, 2006, from <http://citeseer.ist.psu.edu/decker00knowledge.html>
- Denker, G. (2003, July 8). *DAML+OIL Plug-in for Protégé 2000 — User's guide*. SRI International AI Center Report.
- Falkenberg, E. D., Hesse, W., Lindgreen, P., Nilsson, B. E., Han Oei, J. L., Rolland, C., et al. (1997). *FRISCO — A framework of information systems concepts*. IFIP WG 8.1 Technical Report.
- Fernandes, M., Gómez-Pérez, A., & Juriso, N. (1997, March 24-26). METHONTOLOGY: From ontological art towards ontological engineering. In *Proceedings of AAAI-97 Spring Symposium on Ontological Engineering*. Stanford University, CA: AAAI Press.
- Gemino, A., & Wand, Y. (2003). Evaluating modelling techniques based on models of learning. *Communications of the ACM*, 46(10), 79-84.

- Gómez-Pérez, A., & Corcho, O. (2002). Ontology specification languages for the semantic Web. *IEEE Intelligent Systems*, 17(1), 54-60.
- Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 199-220.
- Hakkarainen, S., Hella, L., Tuxen, S. M., & Sindre, G. (2004). Evaluating the quality of Web-based ontology building methods: A framework and a case study. In *Proceedings of 6th International Baltic Conference on Databases and Information Systems (Baltic DBIS'04)* University of Latvia, Riga, Latvia (CSIT Vol. 672 pp. 451-466).
- Hella, L., & Tuxen, S. M. (2003). *An evaluation of ontology building methodologies — An analysis and a case study*. TDT4730 Information Systems Specialization, Study Report, NTNU.
- Horrocks, I. (2002). DAML+OIL: A description logic for the semantic Web. *IEEE Data Engineering Bulletin*, 25(1), 4-9.
- Karp, P. D., Chaudhri, V. K., & Thomere, J. (1999). *XOL: An XML-based ontology exchange language, Version 0.3, July 3*. Retrieved from <http://www.ai.sri.com/pkarp/xol/xol.html>
- Kifer, M., Lausen, G., & Wu, J. (1995). Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4), 741-843.
- Knublauch, H. (2004). *Protégé OWL tutorial*. Presentation at the 7th International Protégé Conference, Maryland. Retrieved February 27, 2006, from <http://protege.stanford.edu/plugins/owl/publications/2004-07-06-OWL-Tutorial.ppt>
- Knublauch, H., Musen, M. A., & Noy, N. F. (2003, October 20). *Creating semantic Web (OWL) ontologies with Protégé*. Presentation at the 2nd International Semantic Web Conference, Sanibel Island, FL. Retrieved February 27, 2006, from <http://iswc2003.semanticweb.org/pdf/Protege-OWL-Tutorial-ISW03.pdf>
- Krogstie, J. (1995). *Conceptual modeling for computerized information system support in organizations*. PhD Thesis 1995:87 NTH, Trondheim, Norway.
- Krogstie, J. (2001). Using a semiotic framework to evaluate UML for the development of models of high quality. In K. Siau & T. Halpin (Eds.), *Unified modeling language: Systems analysis, design, and development issues* (pp. 89-106). Hershey, PA: Idea Group Publishing.
- Krogstie, J. (2003). Evaluating UML using a generic quality framework. In L. Favre (Ed.), *UML and the unified process* (pp. 1-22). Hershey, PA: Idea Group Publishing.
- Lenat, D. B., & Guha, R. V. (1990). Building large knowledge-based systems. *Representation and inference in the Cyc project*. Reading, MA: Addison-Wesley.
- Lindland, O. I., Sindre, G., & Sølvyberg, A. (1994). Understanding quality in conceptual modeling. *IEEE Software*, 11(2), 42-49.
- Luke, S., & Heffin, J. (2000). Shoe 1.01 proposed specification, Shoe project. Retrieved February 27, 2006, from <http://www.cs.umd.edu/projects/plus/SHOE/spec.html>
- MacGregor, R. M. (1991). Inside the LOOM description classifier. *ACM SIGART Bulletin*, 2(3), 88-92.
- Maiden, N. A. M., & Ncube, C. (1998, March/April). Acquiring COTS software selection requirements. *IEEE Software*, 46-56.
- Moody, D. L., Shanks, G. G., & Darke P. (1998). Evaluating and improving the quality of entity relationship models: Experiences in research and practice. In T. Wang Ling, S. Ram, & M.-L. Lee (Eds.), *Proceedings of the 17th International Conference on Conceptual Modelling (ER'98)* (LNCS 1507, pp. 255-276). Berlin: Springer-Verlag.

- Mylopoulos, J., Borgida, A., Jarke, M., & Koubarakis, M. (1990). Telos: A language for representing knowledge about information systems. *ACM Transactions on Information Systems*, 8(4), 325-362.
- Noy, N. F., & McGuinness, D. L. (2001). *Ontology Development 101: A guide to creating your first ontology* (Technical Report KSL-01-05). Stanford Knowledge Systems Laboratory.
- Opdahl, A. L., & Henderson-Sellers, B. (2002). Ontological evaluation of the UML using the Bunge-Wand-Weber model. *Software and Systems Modelling (SoSyM)*, 1(1), 43-67.
- Pepper, S. (2004). *The TAO of topic maps — Finding the way in the age of infoglut*. Ontopia AS, Oslo, Norway. Retrieved from <http://www.ontopia.net/topicmaps/materials/tao.html>
- Pohl, K. (1994). Three dimensions of requirements engineering: a framework and its applications. *Information Systems*, 19(3), 243-258.
- Schreiber, A. Th., Wielinga, B., Akkermans, J.M., van De Velde, W., & de Hoog, R. (1994). CommonKADS. A comprehensive methodology for KBS development. *IEEE Expert*, 9(6), 28-37.
- Schuette, R. (1999). Architectures for evaluating the quality of information models — A meta and an object level comparison. In J. Akoka, M. Bouzeghoub, I. Comyn-Wattiau, & E. Métais (Eds.), *Proceedings of the 18th International Conference on Conceptual Modelling (ER '99)*, Paris (LNCS 1728, pp. 490-505). Berlin: Springer-Verlag.
- Shadbolt, N., Motta, E., & Rouge, A. (1993). Constructing knowledge-based systems. *IEEE Software*, 10(6), 34-38.
- Smith, M. K., Welty, C., & McGuinness, D. L. (2004). *OWL Web ontology language guide*. W3C Recommendation, World Wide Web Consortium.
- Su, X., & Ilebrikke, L. (2005). Using a semiotic framework for a comparative study of ontology languages and tools. In J. Krogstie, T. Halpin, & K. Siau (Eds.), *Information modeling methods and methodologies* (pp. 278-299). Hershey, PA: Idea Group Publishing.
- Sure, Y., & Studer, R. (2002). *On-To.knowledge methodology — Final version*. Institute AIFB, University of Karlsruhe, Germany.
- Uchold, M. (1996, December 16-18). Building ontologies: Towards a unified methodology. In *Proceedings of the 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems*, Cambridge, UK.
- Uchold, M., & Gruninger, M. (1996). Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, 11(2), 93-155.
- Wand, Y., & Weber, R. (1990). Mario Bunge's ontology as a formal foundation for information systems concepts. In P. Weingartner & G. Dorn (Eds.), *Studies on Mario Bunge's Treatise*. Atlanta, GA: Rodopi.
- Weber, R., & Zhang, Y. (1996). An analytical evaluation of NIAM's grammar for conceptual schema diagrams. *Information Systems Journal*, 6(2), 147-170.

ENDNOTE

- ¹ Here abbreviated Protégé as in <http://www-protege.stanford.edu/>

Chapter IV

Exploring the Concept of Method Rationale: A Conceptual Tool to Understand Method Tailoring

Pär J. Ågerfalk, University of Limerick, Ireland
Brian Fitzgerald, University of Limerick, Ireland

ABSTRACT

Systems development methods are used to express and communicate knowledge about systems and software development processes, that is, methods encapsulate knowledge. Since methods encapsulate knowledge, they also encapsulate rationale. Rationale can, in this context, be understood as the reasons and arguments for particular method prescriptions. In this chapter, we show how the combination of two different aspects of method rationale can be used to shed some light on the communication and apprehension of methods in systems development, particularly in the context of tailoring of methods to suit particular development situations. This is done by clarifying how method rationale is present at three different levels of method existence. By mapping existing research on methods onto this model, we conclude the chapter by pointing at some research areas that deserve attention and where method rationale could be used as an important analytic tool.

INTRODUCTION

Systems development methods are used as a means to express and communicate knowledge about the systems/software development process. The idea is that methods encapsulate knowledge of good design practice so that developers can be more effective, efficient, and confident in their work. Despite this, it is a well known fact that many software organizations do not use methods at all (Iivari & Maansaari, 1998; Nandhakumar & Avison, 1999), and when methods are used they are not used literally “out of the box,” but are tailored to suit the particular development situation (Fitzgerald, Russo, & O’Kane, 2003). This tension between the method “as documented” (or as inter-subjectively agreed upon) and the method “in use” has been described as a “method usage tension” between “method-in-concept” and “method-in-action” (Lings & Lundell, 2004). This tension has given rise to an array of different approaches, ranging from contingency factor-driven method engineering (van Slooten & Hodes, 1996) through method tailoring and configuration (Cameron, 2002; Fitzgerald et al., 2003; Karlsson & Ågerfalk, 2004) to the various agile methods, such as XP (Beck, 2000) and SCRUM (Schwaber & Beedle, 2002).

A basic condition for a method to be accepted and used is that method users perceive it to be useful in their development practice (Riemenschneider, Hardgrave, & Davis, 2002). For someone to regard a piece of knowledge as valid and useful, the knowledge must be possible to rationalize, that is, the person needs to be able to make sense of it and incorporate it into his or her view of the world. Ethno-methodologists refer to this property of human behaviour as “accountability” (Dourish, 2001; Eriksén, 2002; Garfinkel, 1967); people require an account of the truth or usefulness of something in order to accept it as valid.¹ This is particularly true in the case of method prescriptions since method users are supposed to use these as a basis for future actions, and thus use the method description as a partial account of their own actions. Hence, we follow Goldkuhl’s (1999) lead and use the term “action knowledge” to refer to the type of knowledge that is codified as method descriptions.

In order to better understand the rationalization of system development methods, the concept of method rationale has been suggested (Ågerfalk & Åhlgren, 1999; Ågerfalk & Wistrand, 2003; Oinas-Kukkonen, 1996; Rossi, Ramesh, Lyytinen, & Tolvanen, 2004). Method rationale concerns the reasons and arguments behind method prescriptions and why method users (e.g., systems developers) choose to follow or adapt a method in a particular way. This argumentative dimension is an important but often neglected aspect of systems development methods (Ågerfalk & Åhlgren, 1999; Ågerfalk & Wistrand, 2003; Rossi et al., 2004). One way of approaching method rationale is to think of it as an instance of “design rationale” (MacLean, Young, Bellotti, & Moran, 1991) that concerns the design of methods, rather than the design of computer systems (Rossi et al., 2004). This aspect of method rationale captures how a method may evolve and what options are considered during the design process, together with the argumentation leading to the final design (Rossi et al., 2004), and thus provides insights into the process dimension of method development. A complementary view on method rationale is based on the notion of purposeful-rational action. This aspect of method rationale focuses on the underlying goals and values that make people chose options rationally (Ågerfalk & Åhlgren, 1999; Ågerfalk & Wistrand, 2003) and provides an understanding of the overarching conceptual structure of a method’s underlying philosophy.

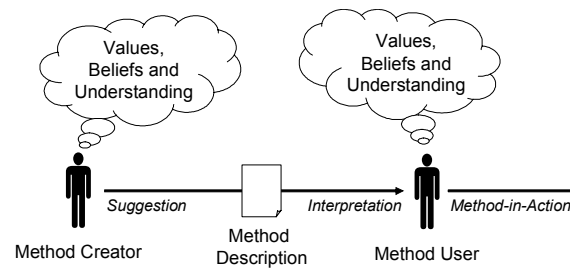
In this chapter, we show how the combination of these two aspects of method rationale can be used to shed some light on the communication, apprehension, and rationalization of methods in software and systems development. This will be done by clarifying how method rationale is present at three different levels of method existence. By mapping existing research on methods onto this three-level model, we conclude the chapter by pointing at some areas that deserve attention and where method rationale could be an important analytic tool.

The chapter proceeds as follows. The next section elaborates the concept of action knowledge and how methods represent an important instance of such knowledge. The subsequent section looks at how methods as action knowledge exist at different levels of abstraction in systems/software development. It also relates these levels to the corresponding actor roles taking part in the communication, interpretation, and refinement of this knowledge. The following two sections elaborate the concept of method rationale as a way of representing the rationality dimension of methods as action knowledge. The final two sections reflect upon the existing research in systems/software development methodology and discuss how method rationale can be used as a tool in creating a more integrated understanding of methods, method configuration/tailoring, and agile development practices.

METHODS AS ACTION KNOWLEDGE

When we think of software and systems development methods, what usually spring to mind are descriptions of ideal typical software processes. Such descriptions are used by developers in practical situations to form what can be referred to as methods-in-action (Fitzgerald, Russo, & Stolterman, 2002). A method description is a linguistic entity and an instance of what can be referred to as action knowledge (Ågerfalk, 2004; Goldkuhl, 1999). The term “action knowledge” refers to theories, strategies, and methods that govern people’s action in social practices (Goldkuhl, 1999). The method description is a result of a social action² performed by the method creator directed towards intended users of the method. A method description should thus be understood as a suggestion by the method creator for how to perform a particular development task. This “message” is received and interpreted by the method user and acted upon by following or not following the suggestion (see Figure 1); that is, by transforming the method description (or “formalized method” [Fitzgerald et al., 2002] or “method-in-concept” [Lings & Lundell, 2004]) into a method-in-action. The “method as message” is formulated based on the method creator’s understanding of the development domain and on his or her fundamental values and beliefs. Similarly, the interpretation of a method by a method user is based on the user’s understanding, beliefs, and values.

It is possible to distinguish between five different aspects of action knowledge: a subjective, an inter-subjective, a linguistic, an action and a consequence (Ågerfalk, 2004; Goldkuhl, 1999), each of which are briefly discussed below. Subjective knowledge is part of a human’s “subjective world” and is related to the notion of “tacit knowledge” (Polanyi, 1958). This would correspond to the two “clouds” in Figure 1. This would also correspond to someone’s personal interpretation and understanding of a method. Inter-subjective knowledge is “shared” by several people in the sense that they attach the same meaning to it. This could imply that some of the elements of the “clouds” in Figure

Figure 1. Method descriptions in a communication context

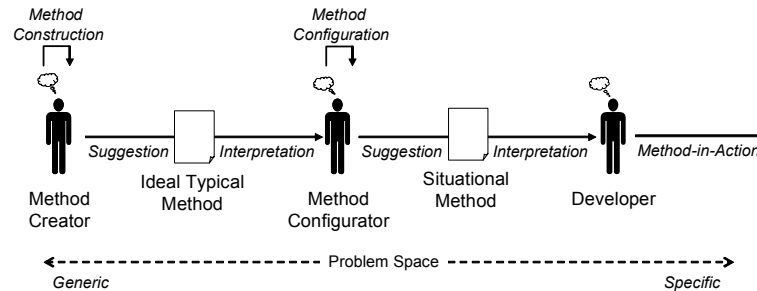
l are agreed upon by the communicator (method creator) and interpreter (method user), and that they thus attach the same meaning to at least parts of a particular method. Linguistic knowledge is expressed as communicative signs, for example, as the written method description in Figure 1. As the name suggests, action knowledge is expressed or manifested in action. This is the action aspect of knowledge, or “method-in-action.” Finally, traces of the action knowledge might be found in materialized artefacts, which constitute a consequence aspect of the knowledge. This would correspond to, for example, produced models and documentation as well as the actual software.

ABSTRACTION LEVELS OF METHODS

As stated above, it is a well-known fact that a method-in-action usually deviates significantly from the ideal typical process described in method handbooks and manuals (Fitzgerald et al., 2003; Iivari & Maansaari, 1998; Nandhakumar & Avison, 1999). Such adaptations of methods can be made more or less explicit and be based on more or less well-grounded decisions.

Methods need to be tailored to suit particular development situations since a method, as described in a method handbook, is a general description of an ideal process. Such an ideal type³ needs to be aligned with a number of situation-specific characteristics or “contingency factors” (Karlsson & Ågerfalk, 2004; van Slooten & Hodes, 1996). The process of adapting a method to suit a particular development situation has been referred to as method configuration⁴ (Karlsson & Ågerfalk, 2004). Method configuration can be understood as a particular form of situational method engineering taking one specific method as a base for configuration. This is in contrast to most method engineering approaches, which assume that a situational method is to be arrived at by assembling a (usually quite large) number of “atomic” method fragments (Brinkkemper, Saeki, & Harmsen, 1999; Harmsen, 1997). This latter form of method engineering allows for construction of situational methods based on a coherent integration of fragments from different methods. In many situations, a more relevant question to ask is “What parts of the method can be omitted?” (Fitzgerald et al., 2003), bearing in mind that omitting a particular part of a method may lead to undesired consequences later in the process, a typical example of which would be if a particular artefact is not produced when it is needed to proceed successfully with a subsequent activity.

Figure 2. Levels of method abstraction in methods as action knowledge



When a situational method has been “configured” or “engineered” and is used by developers in a practical situation, it is likely that different developers disagree with the method description and adapt the method further to suit their particular hands-on situational needs (as indicated above, it is actually impossible for a method-following action to be identical to the action prescribed and linguistically expressed by the method — they represent different aspects of the same knowledge). As a consequence, the method-in-action will deviate not only from the ideal typical method but also from the situational method.

Altogether this gives us three “abstraction levels” of method: (a) the ideal typical method that abstracts details and addresses a generic problem space, (b) the situational method that takes project specifics into account and thus addresses a more concrete problem space, and (c) the method-in-action, which is the “physical” manifestation of developers’ actual behaviour “following” the method in a concrete situation. It follows from this that both the ideal typical method (a) and the situational method (b) exist as linguistic expressions of knowledge about the software development process. On the contrary, the method-in-action represents an action aspect of that knowledge, which may of course be reconstructed and documented post facto (in addition to the way it is manifested in different developed artefacts along the way).

Figure 2 depicts these three abstraction levels of method and corresponding actions and communication between the actors involved. In Figure 2, the Method User of Figure 1 has been specialized into the Method Configurator (or process engineer) and the Developer. Method configurators use the externalized knowledge expressed by the method creator in the ideal typical method as one basis for method configuration and subsequently communicate a situational method to developers. What is not shown in Figure 2 is that method construction, method configuration, and method-in-action rely on the actors’ interpretation of and assumptions about the development context. The developer “lives” directly with this context and thus focuses his or her tailoring efforts on a specific problem space. The method creator, on the other hand, has to rely on an abstraction of an assumed development context and thus focuses on a generic problem space. Finally, the method configurator supposedly has some interaction with the actual development context, which provides a more concrete basis for configuring a situational method.

In both method construction and method configuration, the method communicated is a result of social action aimed towards other actors as a basis for their subsequent actions. This means that method adaptation, both in construction and in-action, relies on the values, beliefs, and understanding of the different actors involved — and this is where method rationale comes into play.

THE CONCEPT OF METHOD RATIONALE

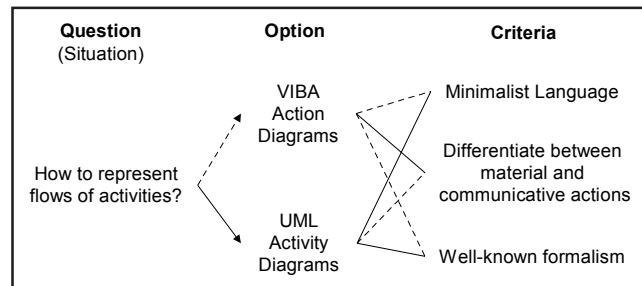
Since methods represent knowledge they also represent rationale. Therefore, a method user “inherits” both the knowledge expressed by the method and the rationale of the method constructor (Ågerfalk & Åhlgren, 1999). It can be argued that regardless of the grounds, method tailoring (both during configuration and in-action) are rational from the point-of-view of the method user (Parnas & Clements, 1986); they are based on some sort of argument for whether to follow, adapt, or omit a certain method or part thereof. Such adaptations are driven by the possibility of reaching “rationality resonance” between the method constructor and method user (Stolterman & Russo, 1997). That is, they are based on method users’ efforts to understand and ultimately internalize the rationale expressed by a method description.

From a process perspective, method rationale can be thought of as having to do with the choices one makes in a process of design (Rossi et al., 2004). Thus, we can capture this kind of method rationale by paying attention to the questions or problematic situations that arise during method construction. For each question, we may find one or more options, or, solutions, to the question.

As an example, consider the construction of a method for analysing business processes. In order to graphically represent flows of activities in business processes, we may consider the option of modelling flows as links between activities, as in UML Activity Diagrams (Booch, Rumbaugh, & Jacobson, 1999). Another option would be to use a modelling language that allows for explicitly showing results of each action and how those results are used as a basis for subsequent actions, as in VIBA⁵ Action Diagrams (Ågerfalk & Goldkuhl, 2001). To help explore the pros and cons of each option, we may specify a number of criteria as guiding principles. Then, for each of the options, we can assess whether it contributes positively or negatively with respect to each criterion. Let us, for example, assume that one criterion (a) is that we want to create a visual modelling language (notation) with as few elements as possible in order to simplify models (a minimalist language). Another criterion (b) might be that we want a process model that is explicit on the difference between material actions and communicative actions⁶ in order to focus developers’ attention on social aspects and material/instrumental aspects respectively (thus a more expressive language). Finally, a third criterion (c) might be that we would favour a well-known modelling formalism. The UML Activity Diagram option would have a positive impact on criterion (a) and (c), and a negative impact on criterion (b), while the VIBA Activity Diagram option would have a positive impact on criterion (b), and a negative impact on criterion (a) and (c). Thus, given that we do not regard any of the criterion to be more important than any other, we would likely choose the UML Activity Diagram option.

Figure 3 depicts this notion of method rationale as based on explicating the choices made throughout method construction. The specific example shown is the choice

Figure 3. Method rationale as choosing between options VIBA action diagrams and UML activity diagrams for modelling activity flows (based on the question, option, criteria model of design space analysis [MacLean et al., 1991]). The solid arrow between “situation” and “option” indicates the preferred choice; a solid line between an option and a criterion indicates a positive impact, while a dashed line indicates a negative impact.



between the VIBA Action Diagram and UML Activity Diagram. This model of method rationale is explicitly based on the Question, Option, Criteria Model of Design Space Analysis (MacLean et al., 1991). Other approaches to capture method rationale in terms of design decisions are, for example, IBIS/gIBIS (Conklin & Begeman, 1988; Conklin, Selvin, Shum, & Sierhuis, 2003) and REMAP (Ramesh & Dhar, 1992). The process-oriented view of method rationale captured by these approaches is important, especially when acknowledging method engineering as a continuous evolutionary process (Rossi et al., 2004). However, as we shall see, another complementary approach to method rationale, primarily based on Max Weber's (1978) notion of practical rationality, has been put forth as means to understand why methods prescribe the things they do (Ågerfalk & Åhlgren, 1999; Ågerfalk & Wistrand, 2003).

According to Weber (1978), rationality can be understood as a combination of means in relation to ends, ends in relation to values, and ethical principles in relation to action. This means that rational social action is always possible to relate to the means (instruments) used to achieve goals, and to values and ethical principles to which the action conforms. Weber's message is that we cannot judge whether or not means and ends are optimal without considering the value base upon which we judge the possibilities.

In this view of method rationale, all fragments of a method (prescribed concepts, notations, and actions) are related to one or more goal. This means that if a fragment is proposed as part of a method, it should have at least one reason to be there. This idea, which is based on Weber's (1978) concept of "instrumental rationality," is referred to as goal rationale. Each goal is, in turn, related to one or more values. This means that if a goal is proposed as the argument for a method fragment, it should have at least one reason to be there. The reason in this latter case is the goal's connection to a "value base" underpinning the method. This idea, which is based on Weber's concept of "rationality of choice," is referred to as value rationale.

Figure 4. Method rationale as consisting of interrelated goals and values as arguments for method fragments (Ågerfalk & Wistrand, 2003)

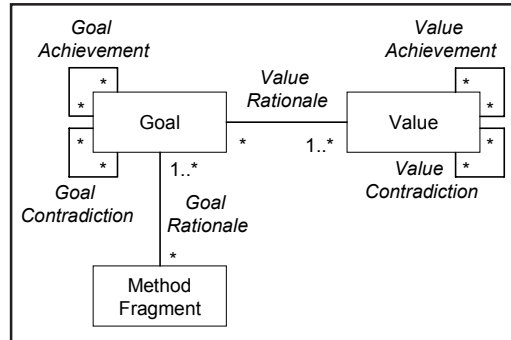


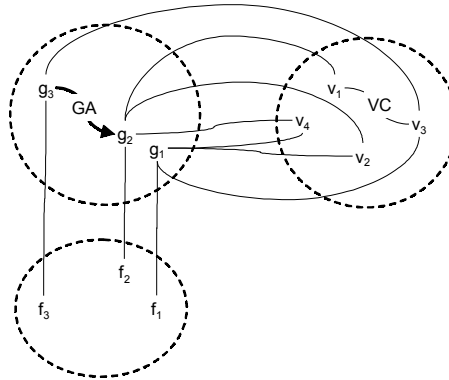
Figure 4 depicts this notion of method rationale, which also includes the idea that goals and values are related to other goals and values in networks of achievements and contradictions.

To illustrate how these two concepts of method rationale fit together, we will return to the example introduced above. Assume we have a model following Figure 4 populated as follows (the classes in the model can be represented as sets and associations as relations between sets, that is, as sets of pairs with elements from the two related sets):

A set of method fragments $F = \{f_1: \text{Representation of the class concept}; f_2: \text{Representation of the activity link concept}; f_3: \text{Representation of the action result concept}\}$; a set of goals $G = \{g_1: \text{Classes are represented in the model}; g_2: \text{activity links are represented in the model}; g_3: \text{Activity results are represented in the model}\}$; a set of values $V = \{v_1: \text{Model only information aspects}; v_2: \text{Minimalist design of modelling language}; v_3: \text{Focus on instrumental v. communicative}; v_4: \text{Use well-known formalisms}\}$; Goal rationale $R_G = \{(f_1, g_1), (f_2, g_2), (f_3, g_3)\}$; Value rationale $R_V = \{(g_1, v_2), (g_1, v_3), (g_1, v_4), (g_2, v_1), (g_2, v_2), (g_2, v_4), (g_3, v_3)\}$; Goal achievement $GA = \{(g_3, g_2)\}$; Value contradiction $VC = \{(v_1, v_3)\}$; $VA = GC = \emptyset$.

A perhaps more illustrative graphical representation of the model is shown in Figure 5. If we view each method fragment in the model as possible options to consider, then the goals and values can be used to compare with the criteria in a structured way. Given that we know that what we want to describe in our notation is a flow of activities (or more precisely the link between activities), we can disregard f_1 outright, since its only goal is not related to what we are trying to achieve. When considering f_2 and f_3 , we notice that each is related to a separate goal. However, since there is a goal achievement link from g_3 to g_2 , we understand that both f_2 and f_3 would help satisfy the goal of representing visually a link between two activities (if we model results as output from one activity and input to another, we also model a link between the two), since these two goals are based on different underlying and contradictory values. Since g_2 is related to v_1 , and g_3 to v_3 , we must choose the goal that best matches our own value base. This should be expressed by the criteria we use. If we, for example, believe that it is important to direct attention

Figure 5. Graphical representation of the method rationale model showing the tree method fragments, the three goals, the three values, and their relationships. The goal achievement relation is represented by an arrow to indicate the direction of the “goal contribution.” All other relationships are represented by non-directed edges since the direction of reading is arbitrary.



to instrumental versus communicative aspects (v_3), then we should choose g_3 and consequently f_3 . If, on the other hand, we are only concerned with modelling information flows, then g_2 and consequently f_2 would be the options to choose.

The concept of method rationale described above applies to both construction of methods and refinement of methods-in-action (Rossi et al., 2004). Since method descriptions are means of communicating knowledge between method creators and method users, the concept of method rationale could be used as a bridge between the two and thus as an important tool in achieving rationality resonance, as discussed above.

USING METHOD RATIONALE

From the example in the previous section, we can see that method rationale is related to both the choices we make during method construction and to the goals and values that underpin the method constructs from which we choose. In the theory of method fragments (Brinkkemper et al., 1999; Harmsen, 1997), method fragments are thought of as existing on different layers of granularity, from the atomic “concept level” through “diagram,” “model,” and “stage,” to the complete “method.” The example used above was at a very detailed level, focusing on rationale in relation to method fragments at the concept layer of granularity. The same kind of analysis could be performed at any layer of granularity and may consider both process and product fragments (i.e., both activities and deliverables).

In order to clarify the issue, we analyse the application of method rationale in an in-depth case study of the use of agile methods in a global context (Fitzgerald & Hartnett, 2005). Briefly summarising, proponents of agile methods — the method creators — have stressed that the principles underpinning agile methods are not radically new, as well as

the philosophy that it is the synergistic combination of all the elementary principles that creates the large impact (Beck, 2000). Thus, their contention is that an “a la carte” cherry-picking of fragments of these methods by method configurators invalidates the overall approach. However, in the situational context of the use of agile methods-in-action being discussed here (Fitzgerald & Hartnett, 2005), the original rationale of the method creators is not borne out in the manner originally anticipated.

For example, one of the key practices of eXtreme Programming (XP) is the Planning Game. However, in the case study, this was not practiced as part of XP, since this was already catered to in the complementary SCRUM method that was also in use. Thus, the overall method-in-action can be bigger than a single method, and the overall logical goal — that of ensuring adequate planning — was being achieved. Another key XP practice, the 40-Hour Week, was seen as a good aspiration, but it was not consistently achievable given the trans-Atlantic development context, where the discrepancy in time zones between Europe and the U.S. caused an inevitable extension in working hours. However, the goal of this practice is to prevent burn-out and exhaustion, and other compensatory mechanisms were in place to combat this. In terms of method rationale, other means had been selected that achieved that goal. Another key XP practice is that of the on-site customer. The rationale here is to try to ensure that the development team can gain an in-depth understanding of the actual customer requirements, and that these can be elicited and nuanced in an ongoing fashion as development unfolds. However, this was simply not possible in this case. In this context, the software being developed was embedded in silicon chips in new product development, and typically, there were no specific customers during the early conceptual stages. Thus, the product marketing group acted as a customer proxy, prioritizing features based on potential revenue. Again, the goal was operationalized in another way than that suggested by the method creator.

Altogether, these examples show that although the actual practices (method fragments) of XP were not always followed, the goals to which XP aspires were achieved. Hence, by understanding the method rationale of XP, other means could be selected to arrive at a method-in-action that realised the XP values and goals and which, at the same time, was tailored to the specific needs of the organization.

As a further example, let us return to the use of agile methods for globally distributed software development. This may indeed seem counter-intuitive in many ways. One example is that agile methods usually stress the importance of having the development team co-located, even, as discussed above, with an always present on-site customer (Beck, 2000). This would obviously be impossible were the team geographically distributed across the globe, as in the case above. However, by analysing the reasons behind this method prescription (i.e., the suggestion by the method creator), we may find that we can operationalize the intended goals of co-location (such as increased informal communication) into other method prescriptions, say utilizing more advanced communication technologies. This way we could make sure that the method rationale of this particular aspect of an agile method is transferred into the rationale of a method tailored for globally distributed development. Thus, we may be able to adhere to agile values even if the final method does look quite different from the original method. That is to say, the principles espoused by the method creators may be logically achieved to the extent that they are relevant in the particular situational context of the final method.

It is important to see that method rationale is present at all three levels of method abstraction: ideal typical, situational, and in-action. At the ideal typical level, method

rationale can be used to express the method creator's intentions, goals, values, and choices made. This would serve as a basis for method configurators (i.e., those who perform method configuration) and developers in understanding the method and tailoring it properly. In the communication between configurator and developer, method rationale would also express why certain adaptations were made when configuring the situational method. Finally, if we understand different developers' personal rationale, we might be able to better configure or assemble situational methods.

Combining the two aspects of method rationale gives us a structured approach to using method rationale — both as a tool to express and document a method's rationale and as a tool to analyse method rationale as the basis for method construction, assembly, configuration, and use.

METHOD RATIONALE RESEARCH

Method rationale has not received much attention in the literature so far, except for a few studies on why methods-in-action deviate from ideal typical and situational methods (although the latter distinction is usually not maintained). Obvious exceptions are the sources cited above, but the uptake by other researchers has so far been limited. It is interesting to note that there seems to be two strands of method research that largely pursue their own agendas without many cross-references. We intentionally construct two ideal types here.

On the one hand, we have the method engineering research that, as stated above, has to a large extent concentrated on the engineering of situational methods from “atomic” method fragments forming larger “method chunks” (e.g., Brinkkemper, 1996; Brinkkemper et al., 1999; Harmsen, 1997; Ralyté, Deneckère, & Rolland, 2003; Rolland & Prakash, 1996; Rolland, Prakash, & Benjamin, 1999; ter Hofstede & Verhoef, 1997). This strand of method research has not paid much attention to what actually happens in systems and software development projects where the situational method is used.

On the other hand, we have the method-in-action research that focuses on the relationship between linguistically expressed methods and methods-in-action (e.g., Avison & Fitzgerald, 2003; Introna & Whitley, 1997; Nandhakumar & Avison, 1999; Russo & Stolterman, 2000). This research, while having contributed extensively to our understanding of method use and rationality resonance, seems to neglect the intricate task of defining and validating consistent method constructs.

Another way to put it is that there has been a lot of research on (a) the construction of situational methods out of existing method parts, and (b) the relationship between linguistically expressed methods (ideal typical methods and situational methods) on the one hand and methods-in-action on the other. The basic flaw in the research of Type (a) is that it does not pay sufficient attention to actual method use. The focus is perhaps too much on what people should do, rather than on what they actually do. The basic flaw in research of Type (b) is that it does not pay sufficient attention to the formality (rigour) required to ensure method consistency; that is, too little focus on how to codify successful practice into useful methods. Another flaw is that (b) does not acknowledge the two different forms of linguistically expressed method-abstraction levels.

There seems to be much to be gained from a systematic effort of integrating these research interests, and method rationale could be an important link between the two. It

is not enough simply to state that a purported objectivistic and instrumental perspective inherent in the method engineering approach (sometimes somewhat derisively referred to as *method-ism* [Introna & Whitley, 1997]) is fundamentally flawed if we are to understand methods-in-action properly. Methods are linguistic expressions as result of and basis for social action. Therefore, we need to understand the complex social reality that shapes methods-in-action. Equally important, though, is to find ways to use that understanding as a basis for being able to better cope with the formal construction, verification, and validation of methods at all three levels of method abstraction. The concept of method rationale can be used as an important conceptual and analytic tool in such a research effort. The reason is that it gives us one construct that can be used to understand method construction and use as social activity. At the same time, it can be used to create a frame of reference for method engineering in terms of analysing, validating, and communicating methods.

CONCLUSION

In this chapter, we have presented a communicative view on systems/software development methods. From this perspective, method descriptions are conceived of as linguistic expressions. As such, they are not just descriptions of ideal typical development processes, but expressions of method creators' suggestions as to how system development should be performed. Such descriptions are subsequently interpreted and (possibly) rationalized by method users. This is also a way of clarifying the distinction between method-in-concept and method-in-action (Lings & Lundell, 2004) by highlighting that there are in fact several methods-in-concept (at least one per actor) involved in method formulation, communication, and use. A method description is here seen as the linguistic expression of the method creator's method-in-concept. This description is then interpreted by method users when forming their own method-in-concept, which is a basis for their method-in-action.

With this foundation, we have also presented a comprehensive concept of method rationale by integrating two different method-rationale aspects. Our conclusion is that method rationale exists as the goals and values upon which we choose what method fragments should belong to a particular method, method configuration, or method assembly. Method rationale exists as an expression of the method creator's values, beliefs, and understanding of the development context. This "intrinsic" method rationale is then compared with method user's values, beliefs, and understanding in method configuration and systems development.

This method rationale existence maps directly to three abstraction levels of methods: the ideal typical method (as expressed by the method creator), the situational method (as adapted by a process engineer/method configurator), and the method-in-action (as manifested by actual method-following actions). The first two levels constitute a linguistic aspect of method, and the last an action aspect.

A method, at any of the three levels, represents knowledge about software and systems development processes. Therefore, method rationale is present at all three levels. Method rationale can be made explicit, which may aid in communication between method creators and method users; a communication that is usually performed through method handbooks and modelling tools.

Finally, we have discussed how method rationale may be an important tool in better understanding the relationships between the three method levels and in synthesising important (past, current, and future) research on method engineering and method-in-action.

ACKNOWLEDGMENT

This work has been financially supported by the Science Foundation Ireland Investigator Programme, (Building a Bi-Directional Bridge Between Software Theory and Practice (B4-STEP).

REFERENCES

- Ågerfalk, P. J. (2004). *Grounding through operationalization: Constructing tangible theory in IS research*. Paper presented at the 12th European Conference on Information Systems (ECIS 2004), Turku, Finland.
- Ågerfalk, P. J., & Åhlgren, K. (1999). Modelling the rationale of methods. In M. Khosrowpour (Ed.), *Managing information technology resources in organizations in the next millennium. Proceedings of the 10th Information Resources Management Association International Conference* (pp. 184-190). Hershey, PA: Idea Group Publishing.
- Ågerfalk, P. J., & Goldkuhl, G. (2001). Business action and information modelling: The task of the new millennium. In M. Rossi & K. Siau (Eds.), *Information modeling in the new millennium* (pp. 110-136). Hershey, PA: Idea Group Publishing.
- Ågerfalk, P. J., & Wistrand, K. (2003). *Systems development method rationale: A conceptual framework for analysis*. Paper presented at the 5th International Conference on Enterprise Information Systems (ICEIS 2003), Angers, France.
- Avison, D. E., & Fitzgerald, G. (2003). Where now for development methodologies. *Communications of the ACM*, 46(1), 79-82.
- Beck, K. (2000). *Extreme programming explained: Embrace change*. Reading, MA: Addison-Wesley.
- Booch, G., Rumbaugh, J., & Jacobson, I. (1999). *The unified modeling language user guide*. Harlow, UK: Addison-Wesley.
- Brinkkemper, S. (1996). Method engineering: Engineering of information systems development methods and tools. *Information and Software Technology*, 38(4), 275-280.
- Brinkkemper, S., Saeki, M., & Harmsen, F. (1999). Meta-modelling based assembly techniques for situational method engineering. *Information Systems*, 24(3), 209-228.
- Cameron, J. (2002). Configurable development processes: Keeping the focus on what is being produced. *Communications of the ACM*, 45(3), 72-77.
- Conklin, J., & Begeman, M. L. (1988). gIBIS: A hypertext tool for exploratory policy discussion. *ACM Transactions on Office Information Systems*, 6(4), 303-331.
- Conklin, J., Selvin, A., Shum, S. B., & Sierhuis, M. (2003). Facilitated hypertext for collective sensemaking: 15 years on from gIBIS. In H. Weigand, G. Goldkuhl, & A. de Moor (Eds.), *Proceedings of the 8th International Working Conference on the*

- Language-Action Perspective on Communication Modelling (LAP 2003)* (pp. 1-22). Tilburg, The Netherlands: Tilburg University.
- Dourish, P. (2001). *Where the action is: The foundations of embodied interaction*. Cambridge, MA: MIT Press.
- Eriksén, S. (2002). Designing for accountability. In *Proceedings of the Second Nordic Conference on Human-Computer Interaction (NordiCHI 2002)* (pp. 177-186). New York: ACM Press.
- Fitzgerald, B., & Hartnett, G. (2005, May 8-11). A study of the use of agile methods within Intel. In L. M. R Baskerville, J. Pries-Heje, & J. DeGross (Eds.), *Proceedings of IFIP 8.6 International Conference on Business Agility and IT Diffusion*, Atlanta, GA (pp. 187-202). New York: Springer.
- Fitzgerald, B., Russo, N. L., & O'Kane, T. (2003). Software development method tailoring at Motorola. *Communications of the ACM*, 46(4), 65-70.
- Fitzgerald, B., Russo, N. L., & Stolterman, E. (2002). *Information systems development: Methods in action*. Berkshire, UK: McGraw-Hill.
- Garfinkel, H. (1967). *Studies in ethnomethodology*. Cambridge, UK: Polity Press.
- Goldkuhl, G. (1999). *The grounding of usable knowledge: An inquiry in the epistemology of action knowledge*. Linköping, Sweden: Linköping University, CMTO Research Papers 1999:03.
- Harmsen, A. F. (1997). *Situational method engineering*. Doctoral dissertation, Moret Ernst & Young Management Consultants, Utrecht, The Netherlands.
- Iivari, J., & Maansaari, J. (1998). The usage of systems development methods: Are we stuck to old practice? *Information and Software Technology*, 40(9), 501-510.
- Introna, L. D., & Whitley, E. A. (1997). Against method-ism: Exploring the limits of method. *Information Technology & People*, 10(1), 31-45.
- Karlsson, F., & Ågerfalk, P. J. (2004). Method configuration: Adapting to situational characteristics while creating reusable assets. *Information and Software Technology*, 46(9), 619-633.
- Lings, B., & Lundell, B. (2004, April 14-17). *Method-in-action and method-in-tool: Some implications for CASE*. Paper presented at the 6th International Conference on Enterprise Information Systems (ICEIS 2004), Porto, Portugal.
- MacLean, A., Young, R. M., Bellotti, V. M. E., & Moran, T. P. (1991). Questions, options, and criteria: Elements of design space analysis. *Human-Computer Interaction*, 6(3/4), 201-250.
- Nandhakumar, J., & Avison, D. E. (1999). The fiction of methodological development: A field study of information systems development. *Information Technology & People*, 12(2), 176-191.
- Oinas-Kukkonen, H. (1996). Method rationale in method engineering and use. In S. Brinkkemper, K. Lyytinen & R. Welke (Eds.), *Method engineering: Principles of method construction and support* (pp. 87-93). London: Chapman & Hall.
- Parnas, D. L., & Clements, P. C. (1986). A rational design process: How and why to fake it. *IEEE Transactions on Software Engineering*, 12(2), 251-257.
- Polanyi, M. (1958). *Personal knowledge: Towards a post-critical philosophy*. Chicago: Routledge & K. Paul.
- Ralyté, J., Deneckère, R., & Rolland, C. (2003, June 16-18). Towards a generic model for situational method engineering. In J. Eder & M. Missikoff (Eds.), *Proceedings of*

- 15th International Conference on Advanced Information Systems Engineering (CAiSE 2003)*, Klagenfurt, Austria (pp. 95-110). Heidelberg, Germany: Springer-Verlag.
- Ramesh, B., & Dhar, V. (1992). Supporting systems development by capturing deliberations during requirements engineering. *IEEE Transactions on Software Engineering*, 18(6), 498-510.
- Riemenschneider, C. K., Hardgrave, B. C., & Davis, F. D. (2002). Explaining software developer acceptance of methodologies: A comparison of five theoretical models. *IEEE Transactions on Software Engineering*, 28(12), 1135-1145.
- Rolland, C., & Prakash, N. (1996). A proposal for context-specific method engineering. In S. Brinkkemper, K. Lyytinen, & R. Welke (Eds.), *Method engineering: Principles of method construction and tool support* (pp. 191-208). London: Chapman & Hall.
- Rolland, C., Prakash, N., & Benjamin, A. (1999). A multi-model view of process modelling. *Requirements Engineering*, 4(4), 169-187.
- Rossi, M., Ramesh, B., Lyytinen, K., & Tolvanen, J.-P. (2004). Managing evolutionary method engineering by method rationale. *Journal of the Association for Information Systems*, 5(9), 356-391.
- Russo, N. L., & Stolterman, E. (2000). Exploring the assumptions underlying information systems methodologies: Their impact on past, present and future ISM research. *Information Technology & People*, 13(4), 313-327.
- Schwaber, K., & Beedle, M. (2002). *Agile software development with SCRUM*. Upper Saddle River, NJ: Prentice-Hall.
- Searle, J. R. (1969). *Speech acts: An essay in the philosophy of language*. Cambridge, UK: Cambridge University Press.
- Stolterman, E., & Russo, N. L. (1997). *The paradox of information systems methods: Public and private rationality*. Paper presented at the British Computer Society 5th Annual Conference on Methodologies, Lancaster, UK.
- ter Hofstede, A. H. M., & Verhoef, T. F. (1997). On the feasibility of situational method engineering. *Information Systems*, 22(6/7), 401-422.
- van Slooten, K., & Hodes, B. (1996). Characterizing IS development projects. In S. Brinkkemper, K. Lyytinen, & R. Welke (Eds.), *Method engineering: Principles of method construction and tool support* (pp. 29-44). London: Chapman & Hall.
- Weber, M. (1978). *Economy and society*. Berkeley, CA: University of California Press.

ENDNOTES

- ¹ According to ethnomethodologist Harold Garfinkel (1967), actions that are accountable are “visibly-rational-and-reportable-for-all-practical-purposes.”
- ² According to sociologist Max Weber (1978), social action is that human behaviour to which the actor attaches meaning and which takes into account the behaviour of others, and thereby is oriented in its course.
- ³ Max Weber (1978) introduced the notion of an “ideal type” as an analytic abstraction. Ideal types do not exist as such in real life, but are created to facilitate discussion. We use the term here to emphasize that a formalized method, expressed in a method description, never exists as such as a method-in-action. Rather, the

method-in-action is derived from an ideal typical formalized method. At the same time, a formalized method is usually an ideal type created as an abstraction of existing "good practice" (Ågerfalk & Åhlgren, 1999).

- ⁴ Process configuration (Cameron, 2002) and method tailoring (Fitzgerald et al., 2003) are other terms used to describe this.
- ⁵ Versatile Information and Business Analysis is a requirements-analysis method based on language/action theory (Ågerfalk & Goldkuhl, 2001).
- ⁶ Material actions are actions that produce material results, such as painting a wall, while communicative actions result in social obligations, such as a promise to paint a wall in the future. The latter thus corresponds to what Searle (1969) termed "speech act."

Chapter V

Assessing Business Process Modeling Languages Using a Generic Quality Framework

Anna Gunhild Nysetvold,
Norwegian University of Science and Technology, Norway

John Krogstie, Norwegian University of Science and Technology, Norway

ABSTRACT

We describe in this chapter an insurance company that has recently wanted to standardize on business process modeling language. To perform the evaluation, a generic framework for assessing the quality of models and modeling languages was specialized to the needs of the company. Three different modeling languages were evaluated according to the specialized criteria. The work illustrates the practical utility of the overall framework, where language quality features are looked upon as means to enable the creation of models of high quality. It also illustrates the need for specializing this kind of general framework based on the requirements of the specific organization.

INTRODUCTION

There exists a large number of business process modeling languages. Deciding which modeling language to use for a specific task is often done in an ad hoc fashion by different organizations. In this chapter, we present the work done within an insurance company that had a perceived need for using process modeling to support the integration of its business systems across different functions of the organization.

We have earlier developed a general framework for assessment of quality of models, where criteria for the language to be used for modeling are among the means to support quality goals at different levels. We have termed this *language quality* (Krogstie, 2001). This chapter presents an example of using and specializing this part of the quality framework for the evaluation and selection of a modeling language for enterprise process modeling for the insurance company. The need for such specialization is grounded on work on task-technology fit (Goodhue & Thompson, 1995). A similar use of the framework for comparing process modeling languages in an oil company has been reported in Krogstie and Arnesen (2004). Although similar, we will see that due to different goals of process modeling, the criteria derived from the quality framework by the oil company was different in the work reported in this chapter.

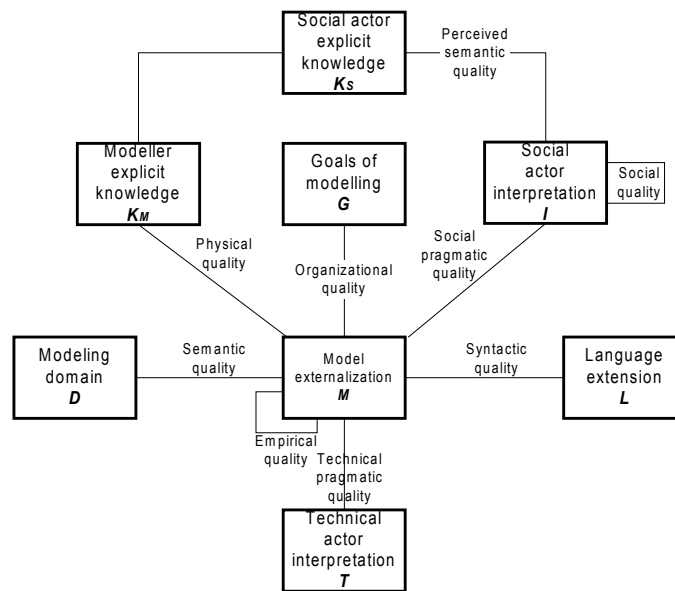
The chapter is structured as follows. The next section describes the quality framework, with a focus on language quality. Then, the case study is described in more detail, followed by the results of the evaluation. The conclusion highlights some of our experiences from using and specializing the quality framework for evaluating modeling languages for business process modeling.

FRAMEWORK FOR QUALITY OF MODELS

The model quality framework (Krogstie, 2001; Krogstie, Lindland, & Sindre, 1995; Krogstie & Sølvsberg, 2003) is used as a starting point for the discussion on language quality. The main concepts of the framework and their relationships are shown in Figure 1. We have taken a set-theoretic approach to the discussion of model quality at different semiotic levels. Different aspects of model quality have been defined as the correspondence between statements belonging to the following sets:

- **G**: the (normally organizational) goals of the modeling task.
- **L**: the language extension, that is, the set of all statements that can be made according to the graphemes, vocabulary, and syntax of the modeling languages used.
- **D**: the domain, that is, the set of all statements that can be made about the situation at hand.
- **M**: the externalized model.
- **K_s**: the relevant explicit knowledge of the set of stakeholders being involved in modeling (the audience **A**). A subset of the audience is those actively involved in modeling, and their knowledge is indicated by **K_M**.
- **I**: the social actor interpretation, that is, the set of all statements that the audience at a given time thinks of as comprising an externalized model.
- **T**: the technical actor interpretation, that is, the statements in the model as “interpreted” by different modeling tools.

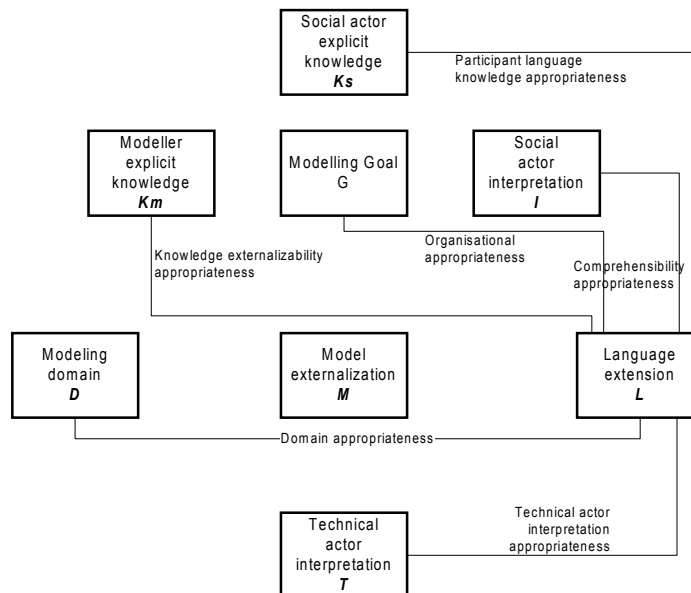
Figure 1. Main parts of the quality framework



The solid lines between the sets in Figure 1 indicate the model quality types.

- **Physical quality:** The basic quality goals on the physical level are externalization (that the knowledge K of the domain D of some social actor has been externalized by the use of a modeling language) and internalizeability (that the externalized model M is persistent and available to the audience).
- **Empirical quality:** Deals with HCI-ergonomics for models and modeling tools.
- **Syntactic quality:** The correspondence between the model M and the language extension L of the language in which the model is written.
- **Semantic quality:** The correspondence between the model M and the domain D . The framework contains two semantic goals:
 1. Validity, which means that all statements made in the model are correct relative to the domain; and
 2. Completeness, which means that the model contains all the statements that are found in the domain.
- **Perceived semantic quality:** The similar correspondence between the audience interpretation I of a model M and their current knowledge K of the domain D .
- **Pragmatic quality:** The correspondence between the model M and the audience's interpretation of it (I).
- **Social quality:** The goal defined for social quality is agreement among audience members' interpretations I .
- **Organizational quality:** The organizational quality of the model relates to the fact that all statements in the model directly or indirectly contribute to fulfilling the

Figure 2. Language quality related to the quality framework



goals of modeling (organizational goal validity), and that all the goals of modeling are being addressed through the model (organizational goal completeness).

Language Quality

- **Language quality** relates the modeling languages used to the other sets. It is distinguished between two types of criteria:
 1. Criteria for the underlying (conceptual) basis of the language (i.e., what is represented in the abstract language model [meta-model] of the language).
 2. Criteria for the external (visual) representation of the language (i.e., the notation).

As illustrated in Figure 2, six areas for language quality are identified, with aspects related both to the meta-model and the notation. They are:

1. **Domain appropriateness:** Ideally, the conceptual basis must be powerful enough to express anything in the domain, that is, not having construct deficit (Wand & Weber, 1993). On the other hand, you should *not* be able to express things that are not in the domain; i.e., what is termed construct excess (Wand & Weber, 1993). The only requirement to the external representation is that it does not destroy the underlying basis. Domain appropriateness is primarily a means to achieve physical quality and, through this, to be able to achieve semantic quality.
2. **Participant language knowledge appropriateness:** This area relates the knowledge of the stakeholder to the language. The conceptual basis should correspond

as much as possible to the way individuals perceive reality. This will differ from person to person according to an individual's previous experience, and thus will initially be directly dependent on the stakeholder or modeler. On the other hand, the knowledge of the stakeholder is not static; that is, it is possible to educate persons in the use of a specific language. In that case, one should base the language on experiences with languages for the relevant types of modeling and languages that have been used successfully earlier in similar tasks. Participant language knowledge appropriateness is primarily a means of achieving physical and pragmatic quality.

3. **Knowledge externalizability appropriateness:** This area relates the language to the participant knowledge. The goal is that there are no statements in the explicit knowledge of the modeler that cannot be expressed in the language. Knowledge externalizability appropriateness is primarily a means of achieving physical quality.
4. **Comprehensibility appropriateness:** This area relates the language to the social actor interpretation. For the conceptual basis we have the following criteria:
 - The phenomena of the language should be easily distinguishable from each other (versus construct redundancy [Wand & Weber, 1993]).
 - The number of phenomena should be reasonable. If the number has to be large, the phenomena should be organized hierarchically and/or in sub-languages of reasonable size linked to specific modeling tasks or viewpoints.
 - The use of phenomena should be uniform throughout the whole set of statements that can be expressed within the language.
 - The language must be flexible in the level of detail.

As for the external representation, the following aspects are important:

- Symbol discrimination should be easy.
- It should be easy to distinguish which graphical mark each symbol belongs to in each model (what Goodman [1976] terms syntactic disjointness).
- The use of symbols should be uniform, that is, a symbol should not represent one phenomenon in one context and another one in a different context. Different symbols should not be used for the same phenomenon in different contexts.
- One should strive for symbolic simplicity.
- One should use a uniform writing system — all symbols (at least within each sub-language) should be within the same writing system (e.g., non-phonological, such as pictographic, ideographic, logographic; or phonological, such as alphabetic).
- The use of emphasis in the notation should be in accordance with the relative importance of the statements in the given model

Comprehensibility appropriateness is primarily a means of achieving empirical quality and, through that, potentially improved pragmatic quality.

5. **Technical actor interpretation appropriateness:** This area relates the language to the technical actor interpretation. For the technical actors, it is especially important

that the language lend itself to automatic reasoning. This requires formality (i.e., both formal syntax and semantics, with the formal semantics being operational, logical, or both), but formality is not sufficient, since the reasoning must also be efficient to be of practical use. This is covered by what we term analyzability (to exploit the mathematical semantics of the language, if any) and executability (to exploit the operational semantics of the language, if any). Different aspects of technical actor interpretation appropriateness are a means of achieving syntactic, semantic, and pragmatic quality (through formal syntax, mathematical semantics, and operational semantics, respectively).

6. **Organizational appropriateness:** This area relates the language to standards and other organizational needs within the organizational context of modeling. These are means of supporting organizational quality.

A number of subareas are identified for each of the six areas of language quality, and in Østbø (2000), approximately 70 possible criteria were identified. We will return to how this extensive list has been narrowed down and specialized for the task at hand.

DESCRIPTION OF THE CASE

The insurance company in our case has a large number of life insurance and pension insurance customers. The insurances are managed by a large number of systems of different ages and based on different technology. The business processes of the company go across systems, products, and business areas, and the work pattern is dependant on the system being used. The company has modernized its IT architecture. The IT architecture is service-oriented, based on a common communication bus and an EAI-system to integrate the different systems. To be able to support complete business processes in this architecture, there is a need for tools for development, evolution, and enactment of business processes.

Goals for Business Process Modeling

Before discussing the needs of the case organization specifically, we will outline the main uses of enterprise process modeling. Five main categories for enterprise modeling can be distinguished:

1. **Human-sense making and communication:** To make sense of aspects of an enterprise and communicate this with other people.
2. **Computer-assisted analysis:** To gain knowledge about the enterprise through simulation or deduction.
3. **Business process management:** To follow up and evolve company processes.
4. **Model deployment and activation:** To integrate the model in an information system and thereby make it actively take part in the work performed by the organization.
5. **To give the context for a traditional system development project:** To provide the business background for understanding the relevance of system requirements and design.

Table 1. Overview of evaluation criteria

No.	Requirement	Type of Requirement
1.1	The language should support the following concepts: (a) processes that must be possible to decompose (b) activities (c) actors/roles (d) decision points (e) flow between activities, tasks, and decision points	Domain appropriateness
1.2	The language should support: (a) system resources (b) states	''
1.3	The language should support basic control patterns (van der Aalst, 2003)	''
1.4	The language should support advanced branching and synchronization patterns	''
1.5	The language should support structural patterns	''
1.6	The language should support patterns involving multiple instances	''
1.7	The language must support state-based flow patterns	''
1.8	The language must support cancellation patterns	''
1.9	The language must include extension mechanisms to fit the domain	''
1.10	Elements in the process model must be possible to link to a data/information model	''
1.11	It must be possible to make hierarchical models	''
2.1	The language must be easy to learn, preferably being based on a language already being used in the organization	Participant language knowledge appropriateness
2.2	The language should have an appropriate level of abstraction	''
2.3	Concepts should be named the same as they are in the domain	''
2.4	The external representation of concepts should be intuitive to the stakeholders.	''
2.5	There should be good guidelines for the use of the language	''
4.1	It must be easy to differentiate between different concepts	Comprehensibility appropriateness
4.2	The number of concepts should be reasonable	''
4.3	The language should be flexible in precision	''
4.4	It must be easy to differentiate between the different symbols in the language	''
4.5	The language must be consistent, not having one symbol to represent several concepts or more than one symbol to express the same concept.	''
4.6	One should strive for graphical simplicity	''
4.7	It should be possible to group related statements	''
5.1	The language should have a formal syntax	Technical Actor appropriateness
5.2	The language should have a formal semantics	''
5.3	It must be possible to generate BPEL documents from the model	''

Table 1. *continued*

5.4	It must be possible to represent Web services in the model	”
5.5	The language should lend itself to automatic execution and testing	”
6.1	The language must be supported by tools that are either already available or can easily be made available in the organization	Organizational appropriateness
6.2	The language should support traceability between the process model and any automated process support system	”
6.3	The language should support the development of models that can improve the quality of the process.	”
6.4	The language should support the development of models that help in the follow-up of separate cases	”

Company Requirements

A general set of requirements to a modeling language based on the previous discussion of language quality is outlined in Østbø (2000). These were looked at relative to the requirements of the case organization, and their importance was evaluated. The analysis together with the case organization resulted in the requirements listed in Table 1.

THE EVALUATION APPROACH

The overall approach to the evaluation began with the identification of a short list of relevant languages by the authors and the case organization. The chosen languages were then evaluated on a 0-3 scale, according to the selected criteria. To look upon this in more detail, all languages were used for the modeling of several real cases using a modeling tool that could accommodate all the selected languages (which in our case was METIS). By showing the resulting models and evaluation results to company representatives, we got feedback and corrections both on the models and our grading. The models were also used specifically to judge the participant language knowledge appropriateness.

Based on discussions with persons in the case organization and experts on business process modeling, three languages were selected as relevant languages. These will be briefly described (for a more in-depth description, see the report by Nysetvold [2004] and the cited references).

Extended Enterprise Modeling Language (EEML)

Extended Enterprise Modeling Language (EEML) was originally developed in the EU-project EXTERNAL (1999) as an extension of APM (Carlsen, 1997), and has been further developed in the EU projects, Unified Enterprise Modelling Language (UEML) and ATHENA (ongoing). The language has constructs to support all modeling categories previously mentioned.

The following main concepts are provided:

- Task with input and output ports (which are specific types of decision points);
- General decision-points;
- Roles (Person role, Organization role, Tool role, Object role); and
- Resources (Persons, Organizations and groups of persons, Tools (manual and software), Objects (material and information)).

A flow links two decision points and can carry resources. A task has several parts: An in-port and an out-port, and potentially a set of roles and a set of sub-tasks. Roles “is filled by” resources of the corresponding type. Figure 3 provides a meta-model of the main concepts.

In addition, EEML contains constructs for goal modeling, organizational modeling, and data-modeling.

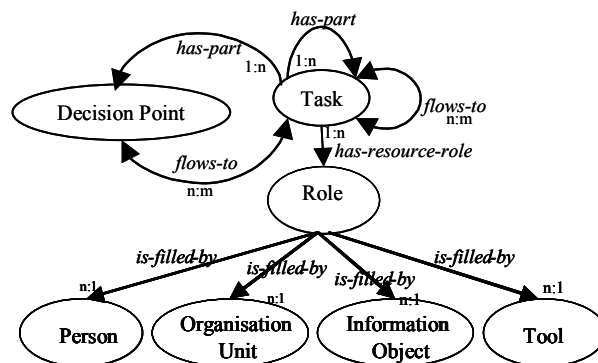
Unified Modeling Language (UML) 2.0 Activity Diagrams

An activity diagram (Fowler, 2004) can have the following symbols:

- Start,
- End,
- Activity,
- Flow (between activities, either as control or as object-flows),
- Decision-points, and
- Roles using swimlanes.

In addition, a number of constructs are provided to support different kinds of control-flow. Given that it is expected that UML activity diagrams are well known, we do not describe these in further detail here.

Figure 3. Main concepts of EEML



Business Process Modeling Notation (BPMN)

Business process modeling notation (BPMN) (<http://www.bpmn.org>) is a notation aiming to be easily understandable and usable to both business users and system developers. It also tries to be formal enough to be easily translated into executable code. By being formally defined, it is meant to create a connection between the design and the implementation of business processes.

BPMN defines business process diagrams (BPDs), which can be used to create graphical models that are especially useful for modeling business processes and their operations. It is based on a flowchart technique — models are networks of graphical objects (activities) with flow controls between them.

The four basic categories of elements are (White, 2004):

- flow objects,
- connecting objects,
- swimlanes, and
- artifacts (not included here).

Flow Objects

This category contains the three core elements used to create BPDs, as illustrated in Table 2.

Connecting Objects

Connecting Objects are used to connect Flow Objects to each other, as illustrated in Table 3.

Swimlanes

Swimlanes are used to group activities into separate categories for different functional capabilities or responsibilities (e.g., a role/participant), as shown in Table 4.

Table 2. Basic BPD flow objects

Event	There are three event-types: <i>Start</i> , <i>Intermediate</i> , and <i>End</i> , respectively, as shown in the figure to the right.	
Activity	Activities contain work that is performed, and can be either a <i>Task</i> (atomic) or a <i>Sub-Process</i> (non-atomic/compound).	
Gateway	Gateways are used for decision-making, forking, and merging of paths.	

Table 3. BPD connecting objects

Sequence Flow	This is used to show the order in which activities are performed in a Process.	
Message Flow	This represents a flow of messages between two Process Participants (business entities or business roles).	
Association	Associations are used to associate data, text and other Artifacts with Flow Objects.	

Table 4. BPD swimlane objects

Pool	A Pool represents a Participant in a Process, and partitions a set of activities from other Pools by acting as a graphical container.	
Lane	Pools can be divided into Lanes, which are used to organize and categorize activities	

Overview of Evaluation Results

Below, the main result of the evaluation is summarized. For every language, every requirement is scored from 0 - 3, according to the scale that follows (earlier evaluations of this sort (Krogstie and Arnesen 2004) have used a 1-10 scale):

- 0 There is no support of the requirement
- 1 The requirement is partly supported
- 2 There is satisfactory support of the requirement
- 3 The requirement is completely supported

The reasoning behind the grading can be found in Nysetvold (2004) and is not included here due to space limitations. The three last rows of Table 5 summarize the results.

None of the languages satisfies all the requirements, but BPMN is markedly better overall. With 72.5 points, BPMN scores 75% of the maximum score, whereas the others score around 66%.

Table 5. Comparison table with all the evaluations collected

No.	Requirement description	Grading of languages		
		UML-AD	BPMN	EEML
1.1	The language should support the listed concepts	3	3	3
1.2	The language should support the listed concepts	2	2	3
1.3	The language should support basic control patterns	3	3	3
1.4	The language should support advanced branching and synchronization patterns	0	0,5	3
1.5	The language should support structural patterns	0	1,5	1,5
1.6	The language should support patterns involving multiple instances	1,5	1,5	2
1.7	The language must support state-based flow patterns	1	1	2
1.8	The language must support cancellation patterns	3	3	3
1.9	The language must include extension mechanisms to fit the domain	3	1	1
1.10	Elements in the process model must link to a data/information model	3	1	3
1.11	It must be possible to make hierarchical models	3	3	3
2.1	The language must be easy to learn, preferably being based on a language already being used in the organization	2	3	1
2.2	The language should have an appropriate level of abstraction	3	3	3
2.3	Concepts should be named is the same as they are in the domain	1	3	2
2.4	The external representation of concepts should be intuitive to the stakeholders	2	2	2
2.5	There should be good guidelines for the use of the language	2	2	1
4.1	It must be easy to differentiate between different concepts	3	3	2
4.2	The number of concepts should be reasonable	3	3	1
4.3	The language should be flexible in precision	1	2	3
4.4	It must be easy to differentiate between the different symbols in the language	2	2	1
4.5	The language must be consistent, not having one symbol to represent several concepts, or more than one symbol expressing the same concept.	3	3	3
4.6	One should strive for graphical simplicity	3	2	1
4.7	It should be possible to group related statements	1	1	2
5.1	The language should have a formal syntax	3	3	3
5.2	The language should have a formal semantics	1	3	2
5.3	It must be possible to generate BPEL documents from the model	2	3	0
5.4	It must be possible to represent Web services in the model	1	3	1
5.5	The language should lend itself to automatic execution and testing	1	3	2
6.1	The language must be supported by tools that are either already available or can easily be made available in the organization	3	3	1
6.2	The language should support traceability between the process model and any automated process support system	2	3	1
6.3	The language should support the development of models that can improve the quality of the process.	1	1	1
6.4	The language should support the development of models that can help in the follow-up of separate cases	1	1	2
	Sum	63,5	72,5	63,5
	Sum without technical actor appropriateness	55,5	57,5	55,5
	Sum without participant language knowledge appropriateness	53,5	59,5	53,5

BPMN has the highest score in all categories, except for domain appropriateness, which is the category with highest weight due to the importance of being able to express the relevant business process structures. EEML is found to have the best domain appropriateness, but loses to BPMN on technical actor appropriateness and participant knowledge appropriateness.

Comprehensibility appropriateness is the category that has the second-highest weight (number of criteria), since the organization regards it to be very important that it is possible to use the language across the different areas of the organization to improve communication between the IT department and the business departments. In this category, BPMN and activity diagrams score the same, which is not surprising given that they use the same kind of swimlane metaphor as a basic structuring mechanism. EEML got a lower score, primarily due to the graphical complexity of the visualization of some of the concepts, combined with the fact that EEML has a larger number of concepts than the others.

Participant language knowledge appropriateness and technical actor appropriateness were scored equally high, and BPMN scores somewhat surprisingly high in both areas. When looking at the evaluation without taking technical actor appropriateness into account, we see that the three languages score almost equal. Thus, in this case, the focus towards the relevant implementation platforms (BPEL and Web services) is putting BPMN on top. On the other hand, we see that this focus on technical aspect does not destroy the language as a communication tool between people, at least not as it is regarded in this case.

In the category organizational appropriateness, BPMN and Activity diagrams score almost the same. The organization had used activity diagrams for some time, but it also appeared that tools supporting BPMN were available to the organization. The organization concluded that it wanted to go forward using BPMN for this kind of modeling in the future.

CONCLUSION AND FURTHER WORK

In this chapter, we have described the use of a general framework for discussing the quality of models and modeling languages in a concrete case of evaluating business process modeling languages.

The case presented illustrates how our generic framework can (and must) be specialized to a specific organization and type of modeling to be useful, which it was found to be by the people responsible for these aspects in the organization in the case study. In an earlier use of the framework, with a different emphasis, UML activity diagrams got a much higher score than EEML, whereas here, they scored equally high (Krogstie & Arnesen, 2004).

It can be argued that the actual valuation is somewhat simplistic (flat grades on a 0-3 scale that is summarized). On the other hand, different kinds of requirements are weighted taking into account the number of criteria in the different categories. An alternative to flat grading is to use pair-wise comparison and analytical hierarchy process (AHP) on the alternatives (Krogstie, 1999). The weighting between expressiveness, technical appropriateness, organizational appropriateness, and human understanding

can also be discussed. For later evaluations of this sort, we would like to use several variants of grading schemes to investigate if and to what extent this would affect the result.

This said, we should not forget that language quality properties are never more than means for supporting the model quality (where the modeling task typically has specific goals of its own). Thus, instead of only evaluating modeling languages objectively on the generic language quality features of expressiveness and comprehension, it is very important that these language quality goals are linked to model quality goals to more easily adapt such a generic frameworks to the task at hand. This is partly achieved by the inclusion of organizational appropriateness, which is not used in earlier work applying the framework. The evaluation results are also useful when a choice has been made, since those areas where the language does not score high can be supported through appropriate tools and modeling methodologies.

REFERENCES

- Carlsen, S. (1997). *Conceptual modeling and composition of flexible workflow models*. Unpublished PhD thesis. Norwegian University of Science and Technology, Trondheim, Norway.
- EXTERNAL (1999). *EXTERNAL — Extended enterprise resources, networks and learning, EU Project, IST-1999-10091, new methods of work and electronic commerce, dynamic networked organizations*. Retrieved November 14, 2005, from <http://research.dnv.com/external/default.htm>
- Fowler, M. (2004). *UML distilled: A brief guide to the standard object modeling language* (3rd ed.). Reading, MA: Addison-Wesley.
- Goodhue, D., & Thompson, R. (1995, June). Task-technology fit and individual performance. *MIS Quarterly*, 14(2).
- Goodman, N. (1976). *Languages of art: An approach to a theory of symbols*. Indianapolis, IN: Hackett.
- Krogstie, J. (1999, June 14-15). Using quality function deployment in software requirements specification. In A. L. Opdahl, K. Pohl, & E. Dubois (Eds.), *Proceedings of the Fifth International Workshop on Requirements Engineering: Foundations for Software Quality (REFSQ '99)* (pp. 171-185). Heidelberg, Germany.
- Krogstie, J. (2001). Using a semiotic framework to evaluate UML for the development of models of high quality. In K. Siau & T. Halpin (Eds.), *Unified modeling language: Systems analysis, design, and development issues* (pp. 89-106). Hershey, PA: Idea Group Publishing.
- Krogstie, J., & Arnesen, S. (2004). Assessing enterprise modeling languages using a generic quality framework. In J. Krogstie, K. Siau, & T. Halpin (Eds.), *Information modeling methods and methodologies* (pp. 63-79). Hershey, PA: Idea Group Publishing.
- Krogstie, J., Lindland, O. I., & Sindre, G. (1995, March 28-30). Defining quality aspects for conceptual models. In E. D. Falkenberg, W. Hesse, & A. Olive (Eds.), *Proceedings of the IFIP8.1 Working Conference on Information Systems Concepts (ISCO3)*;

- Towards a consolidation of views*, Marburg, Germany (pp. 216-231). London: Chapman & Hall.
- Krogstie, J., & Sølvberg, A. (2003). *Information systems engineering: Conceptual modeling in a quality perspective*. Trondheim, Norway: Kompendiumforlaget.
- Nysetvold, A. G. (2004, November). *Proessorientert IT-arkitektu.*, Project thesis (in Norwegian), IDI, NTNU.
- Østbø, M. (2000, June 20). *Anvendelse av UML til dokumentering av generiske systemer*. Unpublished master's thesis (in Norwegian). Høgskolen i Stavanger, Norway.
- van der Aalst, W. M. P., ter Hofstede, A. H. M., Kiepuszewski, B., & Barros, A. P. (2003). Workflow patterns. *Distributed and Parallel Databases*, 5-52.
- Wand, Y., & Weber, R. (1993). On the ontological expressiveness of information systems analysis and design grammars. *Journal of Information Systems* 3(4), 217-237.
- White, S. A. (2004). *Introduction to BPMN*. White Plains, NY: IBM Corporation.

Chapter VI

An Analytical Evaluation of BPMN Using a Semiotic Quality Framework

Terje Wahl, Norwegian University of Science and Technology, Norway

Guttorm Sindre, Norwegian University of Science and Technology, Norway

ABSTRACT

Evaluation of modelling languages is important both to be able to select the most suitable languages according to the needs and to improve existing languages. In this chapter, business process modeling notation (BPMN) is presented and analytically evaluated according to the semiotic quality framework. BPMN is a functionally oriented language well suited for modeling within the domain of business processes, and probably general processes outside of the business domain. The evaluation indicates that BPMN is easily learned for simple use, and business process diagrams (BPDs) are relatively easy to understand. Tools can fairly easily map BPDs into the Web Services Business Process Execution Language (WS-BPEL) (formerly known as BPEL4WS) format, but executable systems then require creation of Web services representing the activities in BPDs. An evaluation according to the Bunge-Wand-Weber (BWW) ontology is useful for finding ontological discrepancies, and the semiotic framework is useful for evaluating quality on a relatively general level. Thus, these methods complement each other.

INTRODUCTION

Currently there exist a large number of different modelling languages. Many of them define overlapping concepts and usage areas, and consequently it is difficult for organizations to select the most appropriate language related to their needs. Traditionally, the research community has focused more on creating new modelling languages than evaluating existing ones. However, evaluation of languages is important both to be able to select the most suitable ones and to improve existing languages.

Conceptual modelling languages can be evaluated analytically and empirically. As Gemino and Wand (2003) discuss, analytical and empirical analyses of modelling techniques complement each other. We can also distinguish between analyses of single languages and comparative analyses of several languages. In this chapter, we present business process modeling notation (BPMN) and perform an analytical evaluation of the quality of BPMN according to the semiotic quality framework (Krogstie, 2003; Lindland, Sindre, & Sølvyberg, 1994). We also discuss how an analytical evaluation according to the Bunge-Wand-Weber (BWW) ontology may be performed as a complement to this evaluation.

In the next section, we present BPMN and its notation, providing some examples of business process diagrams (BPDs) and relating BPMN to the Web Services Business Process Execution Language (WS-BPEL). The subsequent section presents the semiotic framework, divided into parts for evaluating the quality of conceptual models and the quality of conceptual modelling languages. An analytical evaluation of BPMN according to the semiotic framework is then discussed, followed by a short summary of what the BWW ontology is, how it may be used to evaluate conceptual modelling languages, and in what ways this can complement the evaluation according to the semiotic framework. We then discuss related work, present suggestions for future work, and finally, our conclusion.

BUSINESS PROCESS MODELLING NOTATION

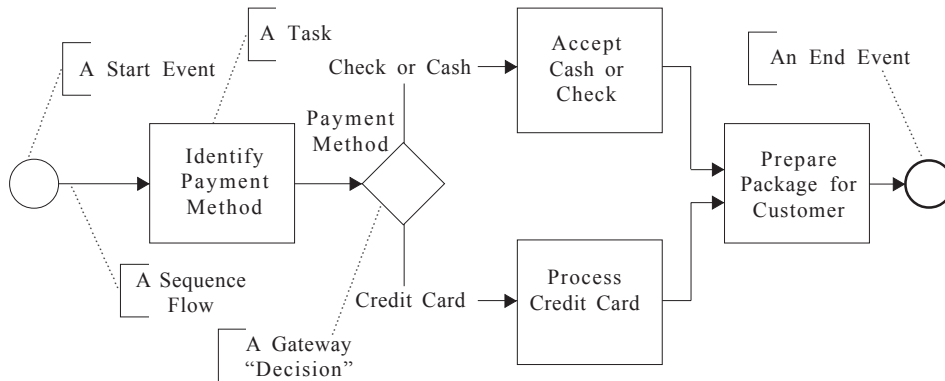
Overview

Business process modelling notation (BPMN) is a notation aiming to be easily understandable and usable to both business users and technical system developers (White, 2004). It also tries to be formal enough to be easily translated into executable code. By being adequately formally defined, it can create a connection between the design and the implementation of business processes.

BPMN defines business process diagrams (BPD), which can be used to create graphical models especially useful for modelling business processes and their operations. It is based on a flowchart technique — models are networks of graphical objects (activities) with flow controls between them.

The BPMN 1.0 specification was developed by the Business Process Management Initiative (BPMI) and was released in May 2004. BPMN is based on the revision of other notations and methodologies, especially unified modeling language (UML) activity

Figure 1. A simple example of a business process, as shown in White (2004)



diagram, UML EDOC business process, IDEF, ebXML BPSS, activity-decision flow (ADF) diagram, RosettaNet, LOVeM, and event-process chains (EPCs).

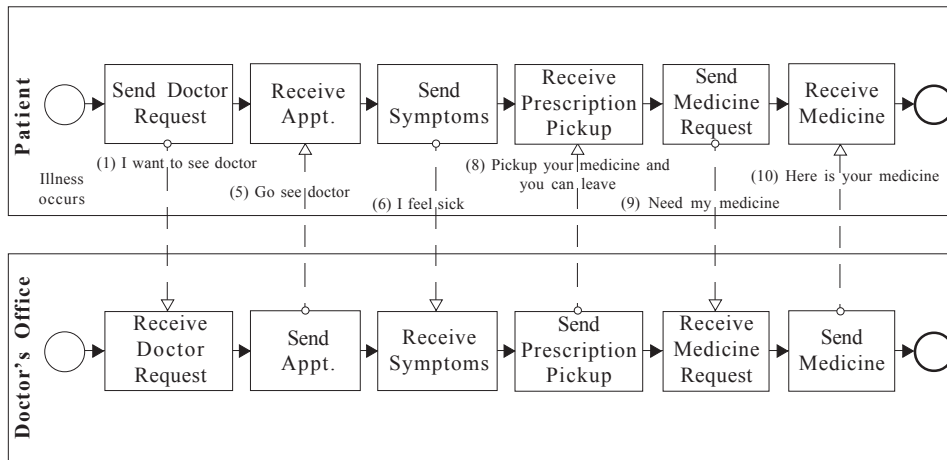
Basic Notation

The graphical elements that are defined by BPMN for use in BPDs are divided into a small number of categories so that they can be easily recognized, even if a user is not immediately familiar with a specific graphical element (White, 2004). The four basic categories of elements are *Flow Objects*, *Connecting Objects*, *Swimlanes* and *Artefacts* (White, 2004):

- **Flow Objects** contain the three core elements used to create BPDs: *Event* (Start, Intermediate, and End), *Activity* (atomic Task and compound Sub-Process) and *Gateway* (decision-making, forking, and merging of paths).
- **Connecting Objects** are used to connect Flow Objects to each other through arrows representing *Sequence Flow*, *Message Flow*, and *Association*.
- **Swimlanes** are used to group activities into separate categories for different functional capabilities or responsibilities (e.g., a role/participant). A *Pool* represents a Participant in a Process, and Pools can be divided into *Lanes* (e.g., between divisions in a company). Pools are used when a Process involves two or more business entities or participants. Activities within Pools must constitute self-contained Processes. Because of this, Sequence Flow may not cross from one Pool to another; instead, Message Flow goes between Pools to indicate the communication between participants. See the *Examples*-subsection for an example of this.
- **Artefacts** may be added to a diagram where deemed appropriate. The following three Artefacts are defined: *Data Object*, *Group* and *Annotation* (to be used for comments and explanations).

For further introduction to BPMN, White (2004) is recommended.

Figure 2. An example of a BPD that uses two Pools, as shown in White (2004)



Metamodel

A metamodel is defined for BPMN (<http://www.bpmn.org>). It contains 55 concepts, some attributes, and many relations between the concepts. Because of its relative complexity, its further description is out of the scope of this chapter.

Examples

To be better able to understand what BPDs are, two examples are shown here. Figure 1 shows a simple process using flow objects, connecting objects, and annotations. Note how sequence flow is used in Figure 2 only within the pools, and message flow is used for communication between the two pools.

Relation to Web Services Business Process Execution Language (WS-BPEL)

Web Services Business Process Execution language is a standard for specifying business process behaviour based on Web services (Andrews et al., 2003). Processes that are described by WS-BPEL export and import functionality exclusively by using Web service interfaces, are stored in a directly executable XML-format, and rely on the use of Web Service Description Language (WSDL) and simple object access protocol (SOAP). BPMN was designed with easy translation into WS-BPEL in mind. Because of this, there are only a few terms in BPMN that cannot be translated into WS-BPEL, and vice versa. The BPMN specification (White, 2004) even contains a section describing how to translate a BPD into WS-BPEL.

SEMIOTIC FRAMEWORK FOR EVALUATION OF QUALITY

Lindland, Sindre, and Sølvsberg (1994) present a semiotic framework for understanding and evaluating quality of conceptual models. In Krogstie (2003), this semiotic framework has been extended and also includes a closely related framework for evaluating the quality of conceptual modelling languages. As an example, Krogstie (2003) has evaluated UML using this framework. Krogstie's paper also gives a nice introduction to the semiotic framework.

The semiotic quality evaluation framework specifically distinguishes between goals and means, meaning that it separates what you want to achieve from how you achieve it (Linland et al., 1994). The framework is based on linguistic and semiotic concepts (such as *syntax*, *semantics*, and *pragmatics*) that enable the assertion of quality at different levels, as will be further described. The semiotic framework is based on a constructivistic world-view, meaning that it is recognized that there exists no "absolute truth" in the sense that every participant can always have one common objective agreement on one model. Instead, models are created through dialog as a compromise between the different world views of each participant.

Quality of Conceptual Models

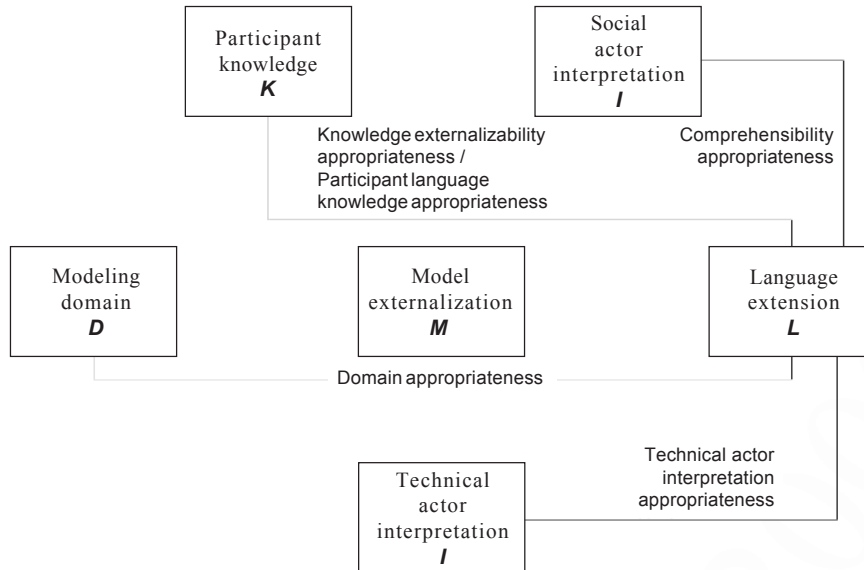
The main concepts of the semiotic framework are model, modeling domain, language extension, participant knowledge, social actor interpretation, and technical actor interpretation (Krogstie, 2003). The relationships between the concepts provide the different quality aspects of the framework. For example, *Syntactic Quality* is based on the relationship between the Model and the modelling language that is used (Language extension).

The seven different relationships represent different aspects of quality: *Physical quality* regards the physical representation of the model and its externalization and internalization. *Empirical quality* regards layout and a presentation that is easy to read and write without mistakes. *Syntactic quality* is about the model being valid and complete with regards to the modeling language being used. *Semantic quality* is about validity and completeness of the model in relation to the domain being modelled. *Perceived semantic quality* of a model is measured like semantic quality above, but in addition it depends on the actor's interpretation of the model and his/her knowledge of the domain. *Pragmatic quality* regards the audience's comprehension of the model. *Social quality* has the definition of actors having agreement (relative or absolute) about their interpretation, knowledge, and model.

Quality of Conceptual Modelling Languages

The semiotic framework for evaluating the quality of conceptual modelling languages is based on the framework for quality of conceptual models (Krogstie, 2003). It is used to evaluate the modeling language's *potential* for making models of high quality. According to Krogstie (2003), one can evaluate two kinds of criteria: criteria for the conceptual basis of a language (e.g., the metamodel for the language), and criteria for the external (graphical) representation of the language. The metamodel for a conceptual

Figure 3. The framework for quality of conceptual modelling languages, as presented in Krogstie (2003)



modelling language can be regarded as a conceptual model in itself, and thus can also be evaluated according to the framework for quality of conceptual models. It may also be useful to evaluate the specification or other documentation of a language according to the semiotic quality framework.

Five aspects are identified for evaluating the quality of conceptual modelling languages: domain appropriateness, participant language knowledge appropriateness, knowledge externalizability appropriateness, comprehensibility appropriateness, and technical actor interpretation appropriateness. The relationships illustrated in Figure 3 represent these five aspects of language quality.

- **Domain appropriateness:** This deals with how suitable a language is for use within different domains. If “there are no statements in the domain that cannot be expressed in the language” (Krogstie & Sølvberg, 2003), then the language has good domain appropriateness.
- **Participant language knowledge appropriateness:** It is a goal here that the participants know the language and are able to use it. They should have explicit knowledge about all the statements in the language-models of the languages they use (Krogstie & Sølvberg, 2003).
- **Knowledge externalizability appropriateness:** This deals with the participants’ ability to express all their relevant knowledge using the modeling language. A language has good knowledge externalizability appropriateness if “there are no statements in the explicit knowledge of the participant that can not be expressed in the language” (Krogstie & Sølvberg, 2003).

- **Comprehensibility appropriateness:** The audience should be able to understand as much of the language as possible. Good comprehensibility appropriateness is achieved if “all the possible statements of the language are understood by the participants in the modeling effort using the language” (Krogstie & Sølvyberg, 2003).
- **Technical actor interpretation appropriateness:** It is important for technical actors that the language is suitable for automatic reasoning. This can be achieved if the language is relatively formally defined and reasoning is efficient and practical to use, for example, by being executable.

ANALYTICAL EVALUATION OF BPMN

Domain Appropriateness

The most central concept in BPMN is the *process*, which is built up from activities. Because of this, the main *perspective* of BPMN is the *functional perspective* (Krogstie, 2003). Data flow diagrams (DFD) and UML activity diagrams are examples of other conceptual modeling languages with a functional perspective. BPMN is well suited to model processes consisting of activities, with simple and advanced rules for the flow of the sequence of activities. BPDs (that are created using BPMN) can also show which actors or roles perform these activities by using Swimlanes.

Because of its functional perspective, however, BPMN has clear limitations to its domain appropriateness. It is not well suited for expressing, for example, models in the object-oriented domain. BPMN lacks concepts like class hierarchies. As stated by White (2004), BPMN is *not* suitable for modelling organizational structures and resources, functional breakdowns, data and information models, strategy, or business rules.

BPMN was created for the main purpose of modelling *business* processes and is hence well suited for modelling the business domain (e.g., B2B processes). However, the BPMN 1.0 specification (White, 2004) and the BPMN metamodel (<http://www.bpmn.org>) do not explicitly limit the usage of the language to business processes. The constructs of the language do not contain any business-specific terms. Because of this, advanced processes can be modelled even if they are not business related. However, BPMN was constructed to support *only* the concepts of modelling that are relevant for business processes. Because of this, some important concepts regarding the specification of processes within other domains are missing from the BPMN language. As an example, BPMN contains no constructs representing valves or pumps for modelling control engineering processes. Those needing to model processes in other domains will, in many cases, prefer and benefit from using other more domain-specific languages. The BPMN specification (White, 2004) does, however, provide possibilities of extending the language to support modelling of different vertical domains, but it is unclear how and to what extent this may be done since the specification is unclear on this point.

Participant Language Knowledge Appropriateness

The graphical elements of BPMN are defined in a clear and concise way to avoid confusion and ease the learning of the language. The language is also made to have similar notation to other languages like flowcharts, UML activity diagrams, event

process chains, Petri nets, and data flow diagrams (DFD). For example, a diamond shape is used in BPMN, UML activity diagrams, and flowcharts to express a decision point, and in both BPMN and activity diagrams to express a merge. BPMN also has a striking resemblance to activity diagrams regarding the notational representation of events (small circles) and activities (rounded rectangles). In addition, the concept of a Swimlane and its graphical representation are very similar in activity diagrams. Sequence flows are represented by arrows with solid lines and solid arrowheads in BPMN, activity diagrams, flowcharts, and Petri nets. These similarities are helpful, at least for IT professionals who are already familiar with the other languages. There are, however, also some graphical elements that are used differently in BPMN compared to other languages. For example, BPMN uses a diamond shape also to represent forks or joins (of parallel activities), but in activity diagrams a thick horizontal line is used for this. Flowcharts use rounded rectangles to represent start- or end-states, not to symbolize activities. These differences make it more difficult to learn BPMN.

It is a goal for BPMN that it should be understandable not only by IT professionals, but also by business analysts and other nontechnical people (White, 2004). However, due to the complexity of the more advanced aspects of BPMN, these authors find it a bit unrealistic that normal business users without training would be able to understand advanced business processes modelled using BPMN. As an example of the complexity of BPMN, there are 23 different predefined diagram elements representing different types of events.

Knowledge Externalizability Appropriateness

This area is highly dependent on the specific knowledge of the actors who are using the language and is, therefore, difficult to evaluate in a general way. We can, however, make assumptions about the typical participants involved in the modelling process. BPMN probably appeals the most to business users, since it was created especially for modelling business processes. The term “business users” is, however, very broad and includes a wide variety of actors. If the actors want to model a process purely within the business domain, BPMN has very good support for this. But if they desire to create models involving other domains as well, this may be difficult (ref. earlier statements in this chapter), and supplements to the BPMN-models may be needed. Thus it may be hard for actors to externalize their relevant knowledge using only business process diagrams (BPDs) if that knowledge goes beyond the domain of business processes. As already mentioned, the BPMN specification (White, 2004) provide possibilities of extending the language to better support modelling of several vertical domains, but it is unclear how and to what extent this may be done.

Comprehensibility Appropriateness

Comprehensibility of a conceptual modelling language can be divided into understanding of language concepts and understanding of notation. Regarding notation, BPMN provides a small number of notational categories so that the readers can easily recognize the basic types of elements that constitute the diagrams (White, 2004). In addition, these basic categories contain variations that may be used when creating more complex BPDs. This categorization helps with the comprehensibility of BPDs. It also helps that the notational categories are easily distinguished from one another and look

partially familiar to other languages like UML activity diagrams. In some cases, the notation is very intuitive, for example, envelopes are used for symbolizing message events and clocks are used for symbolizing timer events. The BPMN specification (White, 2004) gives some helping guidelines on how to create clear and understandable diagrams. But, on the other hand,, it has few strict requirements on how to layout diagram elements and connect flow arrows between them, so the potential for creating BPDs with poor empirical quality (and thus worsened comprehensibility) is present despite the guidelines. Regarding the concepts defined for BPMN, the authors think that the basic concepts used in the language are descriptive, accurate, easily understandable, and well defined in the specification (White, 2004). The more detailed and advanced concepts in BPMN will, however, require user training to fully understand what they mean when used in connection with BPDs.

BPMN supports aggregation by allowing activities that are “collapsed” and contain sub-activities. This helps the user to understand and get an overview over complex models.

Technical Actor Interpretation Appropriateness

BPDs are, with a few exceptions, easily mapped into the WS-BPEL-format. Guidelines for doing this can be found in the BPMN 1.0 Specification (White, 2004), and this relatively easy mapping helps the technical actors who want to implement a BPD into an executable information system (IS). Mappings to other more formally defined languages are not defined, though it is possible to do this. But if the technical actors don’t want to implement the models using WS-BPEL processes and Web services, more work is probably required to convert the BPD into an executable IS.

WS-BPEL requires the use of WSDL and Web services to be executable. Because of this, it is not so easy to perform automated reasoning about processes that are not suitable for implementation using a combination of Web services.

Atomic activities in BPDs are supposed to usually represent a Web service. How to specifically implement these Web services may be difficult to interpret, especially if the activity is vaguely defined using only a short textual description.

Quality of the BPMN Language Model

The BPMN metamodel, and its evaluation, is too complex for a detailed analysis in the scope of this chapter, but its sheer size and complexity suggest that it might have a less than perfect pragmatic quality. This further strengthens our claim that normal business users without training will have difficulties understanding advanced business processes modelled using BPMN.

Bunge-Wand-Weber Ontology for Evaluation

The Bunge-Wand-Weber model (BWW) is an ontological model of information systems that can be used to analyze and evaluate conceptual modeling languages (Wand & Weber, 1993). By comparing the constructs of the BWW ontology to the constructs of the modeling language, one can analyze the meaning of the language constructs to determine if they are appropriate with regards to being well-defined and fitting well together. In Wand and Weber (1993), four “ontological discrepancies” are identified for

evaluating ontological clarity of languages: construct overload, construct redundancy, construct excess, and construct deficit.

Comparing the BWW Ontology to the Semiotic Framework

Both the BWW ontology and the semiotic framework are analytical in that they evaluate languages based on a theoretical framework. The results are lists of language features that correspond to the recommendations of the frameworks, and other features that have room for improvement.

The semiotic framework is well suited for evaluating quality on a relatively general level. The BWW ontology complements this by being more concrete as to evaluating and suggesting which concrete language constructs should be used. The BWW ontology looks at the conceptual basis of modelling languages and cannot be used to evaluate other aspects of a modelling language (such as, for example, the diagram notation). Notation and other aspects can be evaluated using the semiotic framework, giving this technique a broad focus. The BWW ontology, on the other hand, has a narrower focus, but evaluations thus become more thorough. Evaluations using BWW are easily more objective than when using the semiotic framework with its more general concepts. The semiotic framework and the BWW ontology complement each other as methods to analyze conceptual modelling languages.

UML is an example of a modelling language that has been evaluated both using the BWW ontology and the semiotic framework. Opdahl and Henderson-Sellers (2002) evaluated UML using the BWW ontology. They found that many constructs in UML match well with the BWW ontology, but also suggest some concrete improvements based on identified problem areas. Krogstie (2003) evaluated UML using the semiotic framework. He suggests different but useful improvements, based on issues identified in, for example, problems of comprehensibility. Despite their different findings, both these papers have the same basic conclusion — UML is a useful language but one with some weaknesses.

RELATED WORK

The semiotic quality framework has been used to evaluate several conceptual modelling languages. In addition to the mentioned usage of the framework in Krogstie (2003) it was also used by Su and Ilebrekke (2002) to compare the quality of ontology languages and tools. Arnesen and Krogstie (2002) tailored the framework for a concrete organization's needs and used it to evaluate the quality of five enterprise process-modelling languages for use in that organization.

The authors have not been able to find any other published papers evaluating BPMN. However, some evaluations of WS-BPEL have been performed. This is relevant because models created using BPMN, in many cases, can be mapped directly into a corresponding model in BPEL4WS and vice versa. However, WS-BPEL-models are represented in XML and have no graphical notation. Wohed, van der Aalst, Dumas, and ter Hofstede analyzed WS-BPEL using a framework composed of workflow and communication patterns. They concluded that, although being a relatively powerful and flexible language, WS-BPEL is a complex language with partially unclear semantics. A similar

conclusion is reached by van der Aalst (2003). These findings correspond to this chapter's suggestion that diagrams utilizing advanced features might be difficult to comprehend, especially for nontechnical business users.

FUTURE WORK

To further elaborate on the evaluation done in this chapter, the quality of the documentation and tool support for BPMN should be analyzed using the semiotic framework.

Additional evaluation of BPMN should also be performed by comparing the BWW ontology to the BPMN metamodel. The comparison should look for construct overload, redundancy, excess, and deficit (Wand & Weber, 1993). One might, for example, find that the BPMN metamodel lacks some general concepts relevant when modelling outside the business domain. This might be the case because BPMN was created with mainly business processes in mind. An evaluation according to the BWW ontology is useful for finding the ontological discrepancies as described above. While the correctness and completeness of the BWW ontology or any other ontology can always be debated, the use of such an approach to evaluate modelling languages does provide an anchoring point for the discussion and has shown useful application results (Opdahl & Henderson-Sellers, 2002).

In addition to further analytical evaluation, empirical evaluation is needed to validate the results of the analytical investigations. Future work should also include comparative studies of BPMN and several other business process modelling languages.

CONCLUSION

BPMN is a functionally oriented language designed for easily modeling business processes and is well suited for this domain. BPMN has usage limitations within other domains (e.g., the object-oriented domain), but can also be used to model general processes outside the business domain. BPMN has a familiar and easy basic graphical notation, but also includes complex and advanced features that probably require a fair amount of training for nontechnical users to learn. BPDs have relatively good comprehensibility appropriateness due to categorization of the types of graphical elements and support for aggregation of activities. Technical actors may fairly easily map BPDs into the WS-BPEL format, but creation of Web services representing the activities is required to make an executable system in this case.

It has been discussed how BPMN may be evaluated according to the BWW ontology and in what ways this may supplement the evaluation according to the semiotic framework. An evaluation according to the BWW ontology is useful for finding ontological discrepancies, and the semiotic framework is useful for evaluating quality on a relatively general level. The semiotic framework and the BWW ontology complement each other as methods to analyze conceptual modelling languages.

REFERENCES

- Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., et al. (2003). *Specification: Business process execution language for web services, version 1.1*. IBM Corp. [Online]. Retrieved August 1, 2005, from <http://www-128.ibm.com/developerworks/library/specification/ws-bpel>
- Arnesen, S., & Krogstie, J. (2002, May 27-28). Assessing enterprise modelling languages using a generic quality framework. In *Proceedings of the 7.th CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'02)*, Toronto, Canada.
- Gemino, A., & Wand, Y. (2003). Evaluating modelling techniques based on models of learning. *Communications of the ACM*, 46(10), 79-84.
- Krogstie, J. (2003). Evaluating UML using a generic quality framework. In L. Favre (Ed.), *UML and the unified process* (pp. 1-22). Hershey, PA: IRM Press.
- Krogstie, J., & Sølvsberg, A. (2003). *Information systems engineering: Conceptual modelling in a quality perspective*. Trondheim, Norway: Kompendiumforlaget.
- Lindland, O. I., Sindre, G., & Sølvsberg, A. (1994). Understanding quality in conceptual modelling. *IEEE Software*, 11(2), 42-49.
- Opdahl, A. L., & Henderson-Sellers, B. (2002). Ontological evaluation of the UML using the Bunge-Wand-Weber model. *Software and System Modeling*, 1(1), 43-67.
- Su, X., & Ilebrikke, L. (2002, May 27-31). A comparative study of ontology languages and tools. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE'02)*, Toronto, Canada (LNCS 2348, pp. 761-765). Springer-Verlag.
- van der Aalst, W. M. P. (2003). Don't go with the flow: Web services composition standards exposed. *IEEE Intelligent Systems*, 18(1), 72-76.
- Wand, Y., & Weber, R. (1993). On the ontological expressiveness of information systems analysis and design grammars. *Journal of Information Systems*, 3, 217-237.
- White, S. A. (2004). *Introduction to BPMN*. IBM Corporation. [Online]. Retrieved August 1, 2005, from <http://www.bpmn.org/Documents/Introduction to BPMN.pdf>
- White, S. A. (Ed.). (2004). *Business process modelling notation (BPMN) Version 1.0*. BPMI.org. [Online]. Retrieved August 1, 2005, from <http://www.bpmn.org/Documents/BPMN V1-0 May 3 2004.pdf>
- Wohed, P., van der Aalst, W. M. P., Dumas, M., & ter Hofstede, A. (2003). Analysis of Web services composition languages: The case of BPEL4WS. In *Conceptual Modeling (ER 2003)* (LNCS 2813, pp. 200-215).

Chapter VII

Objectification of Relationships

Terry Halpin, Neumont University, USA

ABSTRACT

Some popular information-modeling approaches allow instances of relationships or associations to be treated as entities in their own right. Object-role modeling (ORM) calls this process “objectification” or “nesting.” In the unified modeling language (UML), this modeling technique is called “reification,” and is mediated by means of association classes. While this modeling option is rarely supported by industrial versions of entity-relationship modeling (ER), it is allowed in several academic versions of ER. Objectification is related to the linguistic activity of nominalization, of which two flavors may be distinguished: situational and propositional. In practice, objectification needs to be used judiciously, as its misuse can lead to implementation anomalies, and those modeling approaches that permit objectification often provide incomplete or flawed support for it. This chapter provides an in-depth analysis of objectification, shedding new light on its fundamental nature, and providing practical guidelines on using objectification to model information systems. Because of its richer semantics, the main graphic notation used is that of ORM 2 (the latest generation of ORM); however, the main ideas are relevant to UML and ER as well.

INTRODUCTION

In this chapter, the terms “relationship type,” “association,” and “fact type” all denote relation types that may be identified by typed predicates. For example, “Person plays Sport” and “Sport is played by Person” are alternative readings for the same fact type. In many business domains, it is perfectly natural to think of certain relationship instances as objects about which we wish to talk; for example, Australia’s playing of cricket is rated world class. In *object-role modeling (ORM)* dialects, this process of making an object out of a relationship is called “*objectification*” or “*nesting*” (Bakema, Zwart, & van der Lek, 1994; De Troyer & Meersman, 1995; Halpin, 1998, 2001; ter Hofstede, Proper, & Weide, 1993). In the Unified Modeling Language (UML), this modeling technique is often called “*reification*,” and is mediated by means of association classes (OMG, 2003a, 2003b; Rumbaugh, Jacobson, & Booch, 1999). Although industrial versions of entity-relationship modeling (ER) typically do not support this modeling option (Halpin, 2001, Ch. 8; Halpin, 2004a), in principle they could be extended to do so, and some academic versions of ER do provide limited support for it (e.g., Batini, Ceri, & Navathe, 1992; Chen, 1976). As an example of partial support, some ER versions allow objectified relationships to have attributes but not to play in other relationships.

In practice, objectification needs to be used judiciously, as its misuse can lead to implementation anomalies, and those modeling approaches that do permit objectification often provide only incomplete or even flawed support for it. This chapter provides an in-depth analysis of the modeling activity of objectification, shedding new light on its fundamental nature, and providing practical guidelines on how to use the technique when modeling information systems. Because of its richer semantics, the main graphic notation used is that of ORM 2 (the latest generation of ORM), with some examples being recast in UML; however, the main ideas are also relevant to extended ER.

Objectification is closely related to the linguistic activity of *nominalization*. The next section distinguishes two kinds of nominalization (situational and propositional), and argues that objectification used to model information systems typically corresponds to situational nominalization. The section after that proposes an underlying theory for situational nominalization of binary and longer facts, based on equivalences and composite reference schemes. The subsequent section extends this treatment to unary facts and discusses other issues related to the objectification of unary relationships. Then, we consider what restrictions (if any) should be placed on uniqueness constraints over associations that are to be objectified, and propose a set of rules and heuristics to guide the modeler in making such choices. The subsequent section discusses what kind of modeling support is needed to cater to facts or business rules that involve propositional nominalization or communication acts. The conclusion summarizes the main results, suggests topics for future research, and lists references for further reading.

TWO KINDS OF NOMINALIZATION

In this chapter, we treat nominalization as the recasting of a declarative sentence using a noun phrase that is morphologically related to a corresponding verb in the original sentence. Declarative sentences may be nominalized in different ways. One common way is to use a gerund (verbal noun) derived from the original verb or verb

phrase. For example, “Elvis *sang* the song ‘Heartbreak Hotel’” may be nominalized as “Elvis’ *singing* of the song ‘Heartbreak Hotel’.” Another way is to introduce a pronoun or description to refer back to the original (e.g., “*that* Elvis sang the song ‘Heartbreak Hotel’” or “*the fact that* Elvis sang the song ‘Heartbreak Hotel’”).

In philosophy, it is usual to interpret the resulting nominalizations as naming either corresponding *states of affairs* or corresponding *propositions* (Audi, 1999). In linguistics, further alternatives are sometimes included. For example, states of affairs might be distinguished into events and situations (Gundell, Hegarty, & Borthen). For information modeling purposes, we adopt the philosophical approach, ignoring finer linguistic distinctions, and hence classify all nominalizations into just two categories. A *situational* (or circumstantial) nominalization refers to a state of affairs, situation, or set of circumstances in the world or business domain being modeled. A *propositional* nominalization refers to a proposition. We treat events (instantaneous) and activities (of short or long duration) to be special cases of a state of affairs.

The relationships between states of affairs, propositions, sentences, and communication acts have long been matters of philosophical dispute (Gale, 1967), and no definitive agreement has yet been reached on these issues. At one extreme, states of affairs and propositions are sometimes argued to be identical. Some view logic as essentially concerned with the connection between sentences and states of affairs (*Sachverhalte*) (e.g., Smith, 1989), while others view its focus to be propositions as abstract structures. Our viewpoint on some of these issues is pragmatically motivated by the need to model information systems, and is now summarized.

We define a proposition as that which is asserted when a sentence is uttered or inscribed. A proposition (e.g., Elvis sang “Heartbreak Hotel”) must be true or false (and hence is a truth-bearer). Intuitively it seems wrong to say that a state of affairs (e.g., Elvis’ singing of “Heartbreak Hotel”) is true or false. Rather, a state of affairs is actual (occurs or exists in the actual world, is the case) or not. A state of affairs may be possible or impossible. Some possible states of affairs may be actual (occur in the actual world). States of affairs are thus truth-makers, in that true propositions are about actual states of affairs. In sympathy with the correspondence theory of truth, we thus treat the relationship between propositions and states of affairs as one of *correspondence* rather than identity.

Although natural language may be ambiguous as to what a given usage of a nominalization phrase denotes (a state of affairs or a proposition), the intended meaning can usually be determined from the context in which the nominalization is used (i.e., the logical predicate applied to talk about it). For example:

Elvis sang the song ‘Heartbreak Hotel’. — *original proposition*

Elvis’ *singing* of the song ‘Heartbreak Hotel’ is popular. — *actual state of affairs*

That Elvis sang the song ‘Heartbreak Hotel’ is well known. — *true proposition*

That Elvis sang the song ‘Heartbreak Hotel’ is a false belief. — *false proposition*

It’s snowing outside. — *original proposition*

It’s true *that* it’s snowing outside. — *proposition*

That snowing is beautiful. — *state of affairs*

The first three uses above of the demonstrative pronoun “that” result in propositional nominalization. In the final example, “that” is used in combination with the gerund “snowing” to refer a state of affairs (propositions aren’t beautiful). In the previous two sentences, “snowing” is a present participle, not a gerund. For further examples and discussion of related issues, see Gundell, Hegarty and Borthen, and Hegarty.

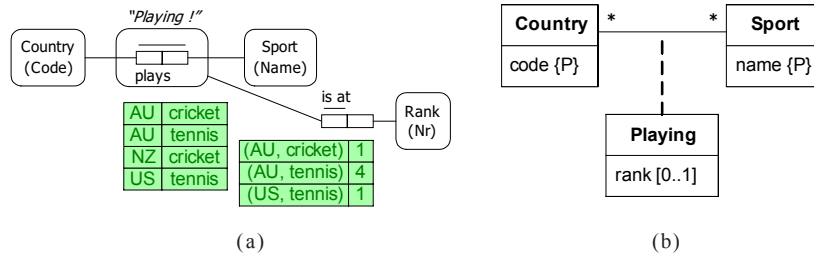
Object-role modeling is sometimes called *fact-oriented modeling*, because it models all the information in the business domain directly as “facts,” using logical predicates rather than introducing attributes as a base construct. For example, the fact that Governor Arnold Schwarzenegger smokes may be declared by applying the unary smokes predicate to the governor, rather than assigning “true” to a Boolean isSmoker attribute of the governor (as in UML). As indicated earlier, states of affairs may be actual or not, and propositions may be true or false. In ordinary speech, the term “fact” is often used to mean a true proposition, but when modeling information in ORM, the term “*fact*” is used in the weaker sense of “*proposition taken to be true*.” A brief explanation for this practice is now given.

Communication within a business domain may involve sentences that express *ground facts* (e.g., **The SecretAgent who** has AgentNr “007” has accrued 10 days Vacation), as well as *business rules* that are either *constraints* on permitted fact populations or transitions (e.g., **Each SecretAgent** accrues **at most** 15 days Vacation per year of employment), or *derivation rules* for deriving some facts from other facts (e.g., **Each SecretAgent who** smokes is cancer-prone). Using $\rightarrow$ for material implication, the derivation rule example may be logically formalized as $\forall x: \text{SecretAgent} (x \text{ smokes} \rightarrow x \text{ is cancer-prone})$. Given the injective (1:1 into) fact type SecretAgent has AgentNr, we may define the individual constant $007 =_{df} \text{SecretAgent who has AgentNr '007'}$, allowing the earlier ground fact to be abbreviated as 007 smokes. Applying Universal Instantiation to the derivation rule yields the conditional: $007 \text{ smokes} \rightarrow 007 \text{ is cancer-prone}$, which in conjunction with the ground fact and the *modus ponens* inference rule $(p, p \rightarrow q / \therefore q)$, enables the following fact to be derived: 007 is cancer-prone.

To prove whether some proposition is actually true is a deep philosophical problem. We live our life by provisionally accepting many propositions to be true even though we are not totally certain about them. The same is true of any business. Looking back on our earlier reasoning, the following propositions have the status of business commitments, rather than propositions that are indisputably established as true: **Each SecretAgent who** smokes is cancer-prone; 007 smokes. Given these propositions, and the definition $007 =_{df} \text{SecretAgent who has AgentNr '007'}$, we still want to determine whether the following proposition is equally acceptable: 007 is cancer-prone. We can do this by establishing a parallel logic to the truth functional logic used earlier, using all the same formulae and inference rules, but reinterpreting the meaning of the truth functional operators as commitment operators, in the sense of *epistemic commitment* (Lyons, 1995, p. 254).

Given any proposition p , we informally define: p is committed to by some business if and only if the business behaves as if it accepted that p (and each of its logical consequences) is true. This still allows the business to violate deontic rules if it so chooses. The business might commit to p because it knows that p (which implies that p is true), or it believes that p , or it feels the chance of p being true is so high that it is prepared to behave as if it believed that p is true, i.e. it treats p as a fact (in the sense of

Figure 1. Objectification of Country plays Sport as Playing in (a) ORM and (b) UML notation



true proposition). The weakest sense of a committed proposition is similar to that of a working assumption. On this analysis, the real meaning (within the context of some business) of $p \rightarrow q$ is: if p is committed to (by the business), then q is committed to (by the business). Note that epistemic commitment does not imply assertion (a consequent may be committed to even if nobody in the business has actually inferred it). Rather than invoking a version of epistemic or doxastic logic, this proposal retains classical truth functional logic, merely providing a different interpretation for the formulae. For example, the truth value “true” now becomes “committed to by the business,” but the same operator definition and inference patterns hold.

In short, such modeling facts (committed propositions) are treated by the business as actual facts (true propositions), even if they might not be known with certainty by the business to be true. In the rest of this chapter, the terms “fact” (i.e., fact instance) and “fact type” should be understood in this sense.

Let us now consider a typical case of objectification in information modeling. Figure 1(a) displays a simple model in the graphic notation of ORM 2 (the latest version of ORM). *Object types* (e.g., Country) are depicted as named, soft rectangles (earlier versions of ORM used ellipses instead). A logical *predicate* is depicted as a named sequence of *role* boxes, each of which is connected by a line segment to the object type whose instances may play that role. The combination of a predicate and its object types is a *fact type*, which is the only data structure in ORM.

If an entity type has a simple, preferred reference scheme, this may be abbreviated by a reference mode in parentheses below the entity type name. In this business domain, for example, countries are identified by country codes, based on the injective (1:1 into) fact type Country has CountryCode, whose explicit display here is suppressed and replaced by the parenthesized reference mode (Code) that simply provides a compact view of the underlying fact type.

Here the fact type Country plays Sport is objectified as the object type Playing, which itself plays in another fact type Playing is at Rank. The latter fact type is said to be nested, as it nests another fact type inside it. The exclamation mark “!” appended to “Playing” indicates that Playing is *independent*, so instances of Playing may exist without participating in other fact types. This is consistent with the optional nature of the first role of Playing is at Rank. Gerunds are often used to verbalize objectifications in both ORM and the KISS method (Kristen, 1994).

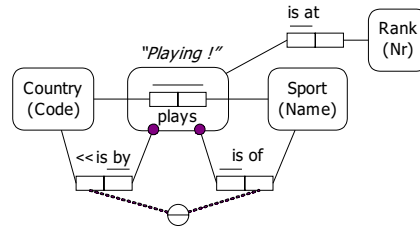
In ORM 2, a bar spanning one or more roles indicates a *uniqueness constraint* over those roles (earlier versions of ORM added arrow tips to the bars). Each role may be populated by a column of object instances, displayed in a *fact table* besides its fact type, as shown in the sample populations. A uniqueness constraint over just a single role ensures that each entry in its fact role column must be unique. For example, in the fact table for Playing is at Rank, the entries for Playing are unique, but some entries for Rank appear more than once, thus illustrating the $n:1$ nature of this fact type. A uniqueness constraint over multiple roles applies only to the combination of those roles. In the fact table for Country plays Sport, the entries for the whole row are unique, but entries for Country and Sport may appear on more than one row. Thus illustrates both the uniqueness over the role pair (the table contains a set of facts, not a bag of facts) and the $m:n$ nature of this fact type.

Figure 1(b) depicts the same example in UML. *Classes* are depicted as named rectangles, and *associations* as optionally named line segments with their *association roles* (association ends) connected to the classes whose object instances may play those roles. By default, association ends have role names the same as their classes (renaming may be required to disambiguate). UML encodes facts using either associations or attributes. The ORM fact type Country plays Sport is modeled here by the association between Country and Sport, which itself is reified into the *association class* Playing. A “*” indicates a *multiplicity* of 0 or more, so the Playing association is $m:n$. UML treats the association class Playing as identical to the association, and permits only one name for it, so linguistic nominalization is excluded. Here the ORM fact type Playing is at Rank is represented instead as an optional attribute (the [0..1] denotes a *multiplicity* of 0 or 1) on the association class Playing.

Now consider the question: *Are the objects resulting from objectification identical to the relationships that they objectify?* In earlier work, we discussed two alternative ORM metamodels, allowing this question to be answered Yes or No (Cuyler & Halpin, 2005). The UML metamodel answers Yes to this question, by treating AssociationClass as a subclass of both Association and Class (OMG, 2003a). Since relationships are typically formalized in terms of propositions, this affirmative choice may be appropriate for propositional nominalization. However, we believe that the objectification process used in modeling information systems is typically situational nominalization, where the object described by the nominalization is a state of affairs rather than a proposition. For situational nominalization, we answer this question in the negative, treating fact instances and the object instances resulting from their objectification as nonidentical. An intuitive argument in support of this position follows, based on the information model in Figure 1.

Consider the relationship instance expressed by the sentence: “Australia plays Cricket.” Clearly this relationship is a proposition, which is either true or false. Now consider the object described by the definite description: “The Playing by Australia of Cricket,” or more strictly “The Playing by the Country that has CountryCode ‘AU’ of the Sport named ‘Cricket’.” Clearly, this Playing object is a state of affairs (e.g., an activity). It makes sense to say that Australia’s playing of cricket is at rank 1, but it makes no sense to say that Australia’s playing of cricket is true or false. So the Playing instance (The Playing by Australia of Cricket) is an object that is ontologically distinct from the fact/relationship that Australia plays Cricket. Our experience is that the same may be said of any typical objectification example that one finds in information system models. In this

Figure 2. Objectification in ORM uses linking fact types for relational navigation



case, so-called “objectified relationships” are in 1:1 correspondence with the relationships that they objectify, but they are not identical to those relationships. Compare this with first order logic, where predicate formulae are often tested for equivalence ($\equiv$) but not identity ($=$). Terms or individuals may be identical, but not equivalent.

In information models, one may sometimes encounter propositional nominalizations where the noun phrase refers to a proposition rather than a state of affairs (e.g., [the fact] that Australia plays cricket is well known; the proposition that Australia plays cricket is true). A related though different case is where the noun phrase refers to a communication act rather than a proposition (e.g., the assertion that Australia plays cricket was made by Don Bradman). We delay discussion of such cases till a later section.

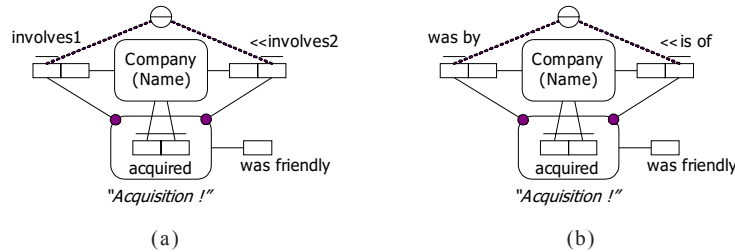
OBJECTIFICATION AND COMPOSITE REFERENCE SCHEMES

Many years ago, we formalized ORM in first-order logic, plus some additional mathematical components (Halpin, 1989). As for situational nominalization, that analysis treated facts as distinct from the objects resulting from objectification, which were formalized in terms of (typically unnamed) ordered pairs. This early formalization assumed that the facts being objectified have an arity of at least 2 (the arity of a predicate is its number of roles), and that each objectified fact type has only the weakest possible uniqueness constraint (spanning all the roles). The improved formalization of objectification outlined in this chapter differs in a number of ways:

- it makes no use of ordered pairs, instead relying on intuitive equivalences that may be visualized graphically;
- it supports objectification of unary predicates;
- it allows objectification of predicates with non-spanning uniqueness constraints; and
- it provides convenient support for navigation between facts and their objectifications.

In this section, we sketch the main ideas, focusing on binary or longer facts with spanning uniqueness constraints. Later sections discuss objectification of facts that either are unary or have non-spanning uniqueness constraints.

Figure 3. Default readings (a) for linking predicates may often be improved (b)



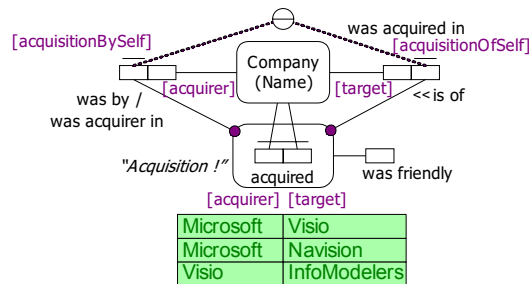
To facilitate high-level declaration of business rules (Halpin, 2004b) and queries (Bloesch & Halpin, 1997) on information models that make use of objectification, we include (implicitly or explicitly) link fact types, that link or relate the objectification result to the objects in the relationship that has been objectified. For example, the definite description “**The Playing that: is by the Country that has CountryCode ‘AU’; and is of the Sport that has SportName ‘Cricket’**” makes use of the linking fact types Playing is by Country and Playing is of Sport (Figure 2). Such descriptions use a formal, textual language for ORM 2, and assume ORM 2’s default algorithms for translating between implicit (e.g., reference mode) and explicit (e.g., fact type) readings. The large dots attached to role links depict mandatory role constraints (each instance of Playing must play both the linking roles). By default, predicates are read left-to-right and top-down; prepending “<<” to a predicate reading reverses the reading order. The external uniqueness constraint depicted as a circled uniqueness bar indicates that each (Country, Sport) pair projected from the attached roles relates to at most one Playing object. The previous ORM notation uses a circled “u” for this kind of constraint. Link fact types have long been used for schema navigation in various ORM dialects, including LISA-D (ter Hofstede, Proper, & Weide, 1993).

If the modeler does not supply readings for the linking predicates, default predicate readings are assigned, such as “involves,” appended by numbers if needed to distinguish linking fact types that link back to the same object type (which plays more than one role in the fact type being objectified). For example, in Figure 3(a), the default predicate readings “involves1” and “involves2” are not very informative. Formally, they may be disambiguated by relating the appended number to a preferred role order (e.g., left-to-right) in reading the objectified predicate. However, it is much easier for humans to understand the intent if the link predicates are renamed as shown in Figure 3(b).

Like any binary fact type, link fact types may be given predicate readings in both directions, and role names. For discussion purposes, Figure 4 adds inverse predicate readings, role names (enclosed in square brackets), and a sample population to the acquisition schema. Display of such model elements on screen and in print may be toggled on/off. Note that the role names (acquirer, target) on the acquisition fact type provide role names for the Company roles in the link fact types — the exact correspondence is derivable if we note the voice (active/passive) of the acquisition predicate reading(s).

ORM schemas may be *navigated* in *relational-style* (using predicate names) or *attribute-style* (using role names), or a *mixture* of both. To navigate from Company to

Figure 4. Adding inverse predicate readings and role names supports full navigation



Acquisition, it matters which link fact type we use. For example, from the company Visio we may navigate via the left link to its acquisition of InfoModelers, or via the right link to its acquisition by Microsoft. Navigating via the left link, the schema path may be verbalized in relational style as “Company **that** was acquirer in Acquisition”; and navigating via the right link we have “Company **that** was acquired in Acquisition”. Here the pronoun “that” performs a conceptual join. Note that each of the above expressions is just a path specification, not a projection on a path. To project on Company and/or Acquisition, we must add a projection indicator (e.g., “√” in the ConQuer language) to the object type occurrence(s) on which we wish to project (Bloesch & Halpin, 1997).

To navigate from Acquisition to Company, it also matters which link we use. Navigating via the links, the schema paths may be verbalized in relational style as: Acquisition **that** was by Company (navigation via left link); and Acquisition **that** is of Company (navigation via right link). Role paths may also be specified in attribute-style, using role names for “attributes.” For example, to navigate from Company to Acquisition, we have the following two different options: Company.acquisitionBySelf (navigation via left link); and Company.acquisitionOfSelf (navigation via right link). To navigate from Acquisition to Company, we have the following two different options: Acquisition.acquirer (navigation via left link); and Acquisition.target (navigation via right link).

Although such expressions could be used to specify projections, here we use them simply to indicate a path obtained by jumping from an object type to one of its far roles. If the dot notation is replaced by the “of-notation,” then the component order is reversed (e.g., instead of “Company.acquisitionBySelf” we have “acquisitionBySelf **of** Company”). In addition to purely relational and purely attribute styles, role paths may be specified using a mix of both styles (e.g., Company.acquisitionBySelf **that** occurred on Date). From an ORM perspective, it is all relational underneath.

The diagram in Figure 2 may be better understood as an abbreviation of the diagram displayed in Figure 5(a). Here, Playing is treated as a normal object type that is related back to the Country and Sport object types using the linking fact types mentioned above. Playing is said to have a *composite reference scheme* since the external uniqueness, internal uniqueness, and mandatory constraints on the link fact types ensure an injection

(1:1-into mapping) from Playing to (Country, Sport) pairs. This is true whether or not we add a simple reference scheme for Playing (e.g., PlayingNr).

When an external uniqueness constraint provides a composite reference scheme, a role sequence obtained by projecting once over each role spanned by that constraint is said to be a *reference projection* for that reference scheme. As with other role projections, the order in which the roles are projected is always recorded, but its display may be toggled on/off as desired. In Figure 5(b), the role sequence annotation (1.1, 1.2) indicates a role projection formed by projecting respectively on the left and right roles of the fact type Country plays Sport; and the annotation (2.1, 2.2) indicates the reference projection for Playing that is formed by projecting respectively on the link roles played by Country and Sport. Display of role sequence annotations visually disambiguates those rare cases where the role sequences are otherwise ambiguous.

The equality constraint depicted by a circled “=” indicates that the (Country, Sport) pairs in the population of the Country plays Sport fact type must be identical to the population of the (Country, Sport) pairs projected from the Country and Sport roles in the join path Playing is by Country **and** is of Sport. In ORM, a set-comparison constraint (subset, equality, or exclusion constraint) applies to two or more arguments, each of which is a sequence of one or more roles. A dotted line connected from a set-comparison constraint to a junction point of two roles includes both the roles in the relevant argument for the constraint. A similar analysis applies to the objectification of ternary and longer facts. For example, we might objectify Country plays Sport in Year as an object type Playing that has a third link fact type Playing is in Year whose year role adds a third component to the composite reference scheme for Playing.

The result of objectifying a binary or longer relationship type may now be viewed as an entity type that has a composite reference scheme whose reference projection bears an equality constraint to the fact type being objectified. This equality constraint may be formalized as an *equivalence*. With respect to our binary example in Figure 5, this equivalence might be introduced to the model in one of three ways:

1. Start with the fact type Country plays Sport, and then objectify it as Playing;
2. Start with the fact types Playing is by Country and Playing is of Sport, then define Country plays Sport as a fully derived fact type in terms of them; or
3. Start with the three fact types Country plays Sport, Playing is by Country, and Playing is of Sport, then assert the equality constraint (corresponding to the equivalence) between them.

Regardless of which way is used, the model fragment is internally stored in terms of the structure shown in Figure 5. If way (1) is used, this is recorded in the metamodel as an instance of the 1:1 meta fact type EntityType objectifies FactType, and the diagram is by default displayed using the compact objectification view (Figure 2) to better reflect the way the modeler conceives of this aspect of the business domain. If way (2) is used, this is recorded in the metamodel as an instance of the meta fact type FactType is derived, and the fact type is flagged (with an asterisk) as derived on the diagram. All three ways employ the same mapping procedure to transform the structure to the chosen implementation (e.g., a relational database schema). Depending on how the equivalence was introduced, it might be formalized in different ways. For example, using $\forall x$ for $\forall x$: Thing, the following three formulations respectively fit naturally with the three ways discussed above.

E1: $\forall x [\text{Playing } x \equiv \exists y:\text{Country } \exists z:\text{Sport } (x \text{ is by } y \ \& \ x \text{ is of } z \ \& \ y \text{ plays } z)]$

E2: $\forall x:\text{Country } \forall y:\text{Sport } [x \text{ plays } y \equiv \exists z:\text{Playing } (z \text{ is by } x \ \& \ z \text{ is of } y)]$

E3: $\forall x:\text{Country } \forall y:\text{Sport } \forall z:\text{Playing } [x \text{ plays } y \equiv (z \text{ is by } x \ \& \ z \text{ is of } y)]$

ORM 2 includes a formal, high-level textual language that enables all its graphical rules as well as additional business rules to be communicated readily with non-technical domain experts. For example, the derivation rule E2 may be rendered as: Country plays Sport **iff some** Playing is by Country **and** is of Sport.

OBJECTIFICATION OF UNARY FACTS

UML provides no direct support for unary relationships, instead requiring their remodeling in terms of attributes or subclasses. ORM supports unary relationships, but in the past typically forbade their objectification. For ORM 2, we removed this restriction by extending the analysis in the previous section to objectified types with simple reference schemes. Consider the unary fact: The President named Abraham Lincoln died. We may objectify this event using the nominalization “that death,” and declare the following additional fact: That death occurred in the Country with country code ‘US’. This natural way of communicating may be supported in a similar way to objectification of non-unary facts. An ORM 2 model for this situation is shown in Figure 6. Small, sample

Figure 5. Explication of the objectification in Figure 2 of Country Plays Sport as Playing

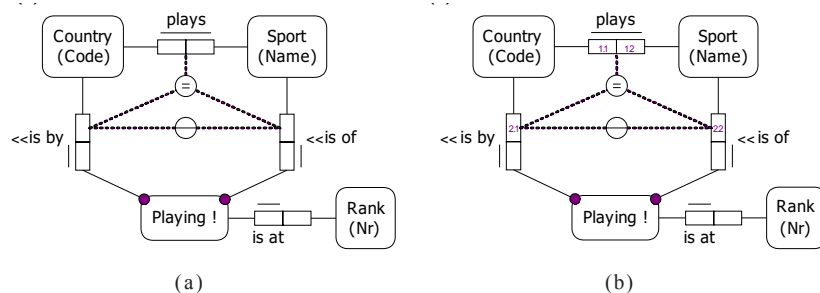


Figure 6. Objectification of unary facts is allowed in ORM 2

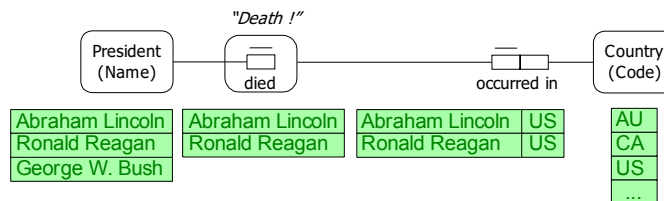


Figure 7. Objectification of unaries may be explicated as shown

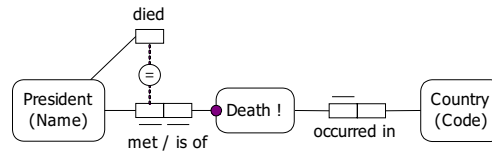


Figure 8. In FCO-IM, non-lexical types are objectifications of roles of lexical object types

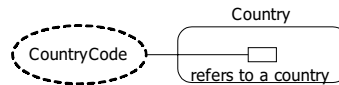
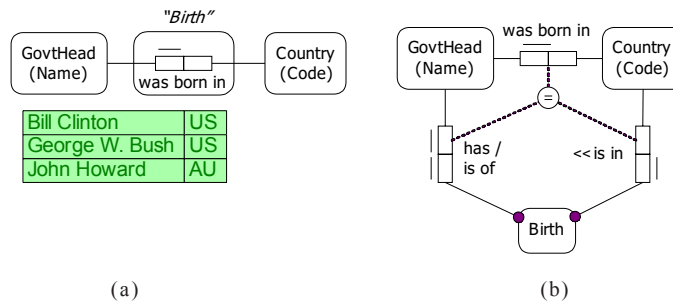


Figure 9. Objectification of an n:1 fact type

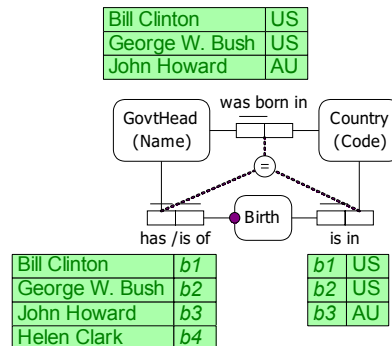


populations are included for the object types and fact types. Here the unary fact type President died is objectified by the object type Death. If desired, the death entries in the fact table for Death occurred in Country may be expanded by prepending “the death of.”

We interpret this case of unary objectification using the expanded schema shown in Figure 7. Here, Death is a normal entity type, with a simple reference scheme provided by its injective relationship to President (e.g., Abraham Lincoln’s death may be referenced by the definite description “**The Death that is of the President who has the PresidentName ‘Abraham Lincoln’**”). Our analysis of objectification of non-unaries may be generalized to include unaries by removing the arity restriction and the composite reference requirement. Hence, *the result of objectifying a relationship type may be viewed as an entity type that has a reference scheme whose reference projection bears an equality constraint to the fact type being objectified.*

This interpretation does not assume that the Death is of President relationship provides the only or even primary way of referring to deaths (e.g., we may introduce a death number to reference deaths, without impacting this analysis). ORM 2 dispensed with the old ORM practice of requiring a primary reference scheme for entity types, since

Figure 10. Because of its optional role, Birth cannot objectify GovtHead was born in Country



it was often an implementation issue. Instead, ORM 2 simply allows a reference scheme to be designated as preferred (not the same as primary) if the business treats it as so.

The ORM version known as Fully Communication Oriented Information Modeling (FCO-IM) also supports objectification of unaries, but in a very different manner (Bakema, Zwart, & van der Lek, 1994). To support existential facts such as “**There is a Country that has the CountryCode ‘AU’**” we introduced to ORM the notion of independent entity types, initially called “lazy” entity types (Halpin, 1993). The FCO-IM approach soon after introduced objectification of unaries to provide an alternative way of supporting existential facts, and to allow models where all base objects are lexical in nature (Bakema, Zwart & van der Lek, 1994). With this approach, an entity (non-lexical object) is an objectification of a role played by a value (lexical object). In Figure 8 for example, the entity type Country is derived by objectifying the unary fact type CountryCode refers to a country. While this approach encourages use of natural reference schemes in modeling and has tool support, we personally find it unintuitive (e.g., it seems to conflate reference with referent) and awkward in dealing with practical modeling issues such as multiple inheritance, context-dependent reference schemes, and changes to reference schemes.

OBJECTIFICATION OF FACT TYPES WITH NON-SPANNING UNIQUENESS CONSTRAINTS

Previous versions of ORM allow an association to be objectified only if either it has just one uniqueness constraint, and this spans all its roles, or it is a binary 1:1 association. This restriction forbids the following two kinds of associations to be objectified:

1. An $n:1$ (or $1:n$) binary association; and
2. A ternary or longer association whose longest uniqueness constraint spans exactly $n-1$ roles.

We exclude from consideration any n -ary association whose longest uniqueness constraint spans fewer than $n-1$ roles, because such an association is compound rather than elementary. Both UML and those versions of ER that support objectification allow cases (1) and (2) to be objectified. The rest of this section briefly summarizes why ORM 2 has been modified to do likewise, though with modeling guidelines. For a discussion of this relaxation, with examples using the ORM 1 and UML notations, see Halpin (2003).

Figure 9(a) depicts in ORM 2 the objectification of the $n:1$ fact type *GovtHead was born in Country as Birth*, together with a sample population. Figure 9(b) shows how the schema is interpreted. The uniqueness constraint on the “has” role of *GovtHead* removes the need for an explicit external uniqueness constraint (because the external constraint is implied by the stronger internal constraint). The equality constraint may be formalized as an equivalence in a similar manner to previous examples. The expanded interpretation avoids denormalization when adding other facts or mapping to implementation structures. For example, adding or mapping the fact type *Birth was on Date* does not require details about birth countries.

Note that mandatory constraints are required on each role played by the objectified type in the link fact types. For example, in Figure 10, it is optional for the birth country to be known for a birth. The model allows the fact type *GovtHead was born in Country* to be defined in terms of the other fact types (by default, a conceptual inner join is performed on the *Birth* roles), but it does not allow *Birth* to be defined in terms of *GovtHead was born in Country* (whose instances always include a country). For example, it is known that Helen Clark is a head of government, but the fact that she is the prime minister of New Zealand is not yet known to the business. In populating the link fact types, surrogate identifiers (italicized) are used here for births; alternatively, we could have referenced the births by the names of the government heads.

As discussed by Halpin (2003), objectification of $n:1$ associations typically portrays the business domain in an unnecessarily complicated way (why introduce birth countries in order to talk about births?) and may add overhead to certain kinds of model changes. However, such objectifications may better depict the semantic affinity between fact types attached to the objectified type, and they simplify model evolution for those cases where the uniqueness constraint on the objectified association changes over time (e.g., from an $n:1$ to an $m:n$ pattern).

The second case to consider is objectification of n -ary associations ($n > 2$) whose longest uniqueness constraint spans $n-1$ roles. Sometimes the n -ary association may have overlapping uniqueness constraints. For example, the ternary fact type *Country in Sport has Rank* may have a uniqueness constraint over its first two roles, and another uniqueness constraint over its last two roles. In such cases, objectifying part of the association based on the roles played by one of the uniqueness constraints typically makes the model harder to understand and may force an arbitrary decision on which uniqueness constraint to use as the basis for a spanning objectification (Halpin, 2003). In rare cases, it is also possible that the uniqueness constraint pattern on the n -ary association may change over time (e.g., to a spanning uniqueness constraint) and, in such cases, semantic stability may be enhanced by allowing nesting of the original association.

The foregoing considerations lead to the following modeling heuristic. *A fact type may be objectified only if: (a) it has only a spanning uniqueness constraint; or (b) its uniqueness constraint pattern is likely to evolve over time (e.g., from $n:1$ to $m:n$, or*

m:n:1 to m:n:p); or (c) it has at least two uniqueness constraints spanning n-1 roles ($n > 1$), and there is no obvious choice as to which of the n-1 role uniqueness constraints is the best basis for a smaller objectification based on a spanning uniqueness constraint; or (d) the objectification significantly improves the display of semantic affinity between fact types attached to the objectified type.

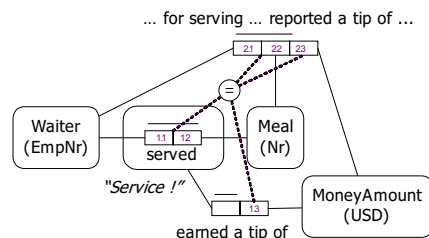
PROPOSITIONAL NOMINALIZATION AND COMMUNICATION ACTS

So far we have discussed objectification in the sense of situational nominalization, where the object being referenced is a state of affairs (event, activity, etc.). Though comparatively rare in information modeling, one may also encounter cases where the object being referenced is either a proposition (resulting from propositional nominalization) or a communication act (e.g., an utterance or inscription act by some speaker/writer). In response to an object management group request for proposal to add a business semantics layer (OMG, 2003c), the Business Rules Team submission specifically included examples of propositional nominalization as instances of practical business rules that need to be declared and enforced. One of its typical rules to illustrate propositional nominalization may be verbalized as follows: If a waiter earns an amount of money as a tip from serving a meal, the waiter must report that fact.

While one may interpret this as a case of propositional nominalization (reporting the fact rather than the act), there is no need to do so for information modeling purposes, as the rule may easily be declared using situational nominalization (reporting the act rather than the fact), as shown in compact form in Figure 11. If the rule is modified to require reporting after the service is performed, a time limit for reporting must be declared to make the rule operational; in this case, the relevant temporal object type may now be added to the model to cater for the extended rule in an obvious way.

Although one could introduce a new syntax to directly support propositional nominalization (e.g., connect to the center of the nominalized predicate) along with a new semantics, such an augmentation seems to add no pragmatic value. For simplicity then, we recommend modeling all propositional nominalizations instead by their corresponding situational nominalizations.

Figure 11. Propositional nominalization may be replaced by situational nominalization



As regards modeling of *communication acts* (Thomas, 1995), when it is of interest to model these acts, they are best modeled directly like any other business domain objects. For example, in a genealogy model, we might be interested in not just descriptions of states of affairs, but assertion acts made by researchers about states of affairs. Such a model might include fact types such as: *AssertionAct reported Proposition*; *AssertionAct was made by Researcher with ConfidenceLevel*; and so forth. These comments relate to the information model only. For modeling communication processes, the information model should be supplemented by other kinds of model (e.g., workflow models) that provide a more intuitive and direct way of understanding essential business processes/services. For some initial discussion of how ORM might be extended in this regard, see Dietz and Halpin (2004) and Halpin and Wagner (2003).

CONCLUSION

This chapter distinguished two kinds of nominalization (situational and propositional), and argued that objectification used to model information systems may be adequately addressed by situational nominalization alone, where the object referenced by the nominalization is a state of affairs. An underlying theory was then presented that interpreted the objectification of facts of any arity (unary, binary, or longer) in terms of normal entity types, their reference schemes, and equality constraints. To cater to objectification over predicates with non-spanning uniqueness constraints, a set of guidelines was proposed to help the modeler decide whether or how to perform the objectification. Finally, it was argued that no additional meta-structures are needed to capture information models for specific business domains that involve propositional nominalization or communication acts.

In previous work, we formalized ORM and worked on the ORM technology currently supported in a Microsoft modeling tool (Halpin, Evans, Hallock, & MacLean, 2003). Currently we are working with a team on the specification of ORM 2 (the next generation of ORM), and an associated open-source modeling tool that supports the refinements to objectification discussed in this chapter, as well as many other extensions being added to ORM 2 (Halpin, 2005).

REFERENCES

- Audi, R. (Ed.). (1999). *The Cambridge dictionary of philosophy* (2nd ed.). Cambridge, UK: Cambridge University Press.
- Bakema, G., Zwart, J., & van der Lek, H. (1994). Fully communication oriented NIAM. In G. M. Nijssen & J. Sharp (Eds.), *NIAM-ISDM 1994 Conference Working Papers* (pp. L1-35). Albuquerque, NM.
- Batini, C., Ceri, S., & Navathe, S. (1992). *Conceptual database design*. Redwood City, CA: Benjamin/Cummings.
- Bloesch, A., & Halpin, T. (1997). Conceptual queries using ConQuer-II. In *Proceedings of ER'97: 16th Int. Conference on Conceptual Modeling* (LNCS 1331, pp. 113-26). Berlin: Springer.
- Chen, P. P. (1976). The entity-relationship model — Towards a unified view of data. *ACM Transactions on Database Systems*, 1(1), 9-36.

- Cuyler, D., & Halpin, T. (2005). Two meta-models for object-role modeling. In J. Krogstie, T. Halpin, & K. Siau (Eds.), *Information modeling methods and methodologies* (pp. 17-42). Hershey, PA: Idea Publishing Group.
- De Troyer, O., & Meersman, R. (1995). A logic framework for a semantics of object oriented data modeling. In *Proceedings of the 14th International ER Conference* (LNCS 1021, pp. 238-249). Gold Coast, Australia: Springer.
- Dietz, J., & Halpin, T. (2004). Using DEMO and ORM in concert: A case study. In K. Siau (Ed.), *Advanced topics in database research* (Vol. 3, pp. 218-36). Hershey, PA: Idea Publishing Group.
- Gale, R. (1967). Propositions, judgments, sentences, and statements. In P. Edwards (Ed.), *The encyclopedia of philosophy* (Vol. 6, pp. 494-505). London: Collier-Macmillan.
- Gundell, J., Hegarty, M., & Borthen, K. *Cognitive status, information structure, and pronominal reference to clausally introduced entities*. Retrieved from <http://www.coli.uni-sb.de/~korbay/essli01-wsh/Jolli/Final/gundel-et-al.pdf>
- Halpin, T. (1989). *A logical analysis of information systems: Static aspects of the data oriented perspective*. Doctoral dissertation, University of Queensland, Australia.
- Halpin, T. (1993). What is an elementary fact? In G. M. Nijssen & J. Sharp (Eds.), *NIAM-ISDM 1994 Conference Working Papers*. Albuquerque, NM. Retrieved from <http://www.orm.net/pdf/ElemFact.pdf>
- Halpin, T. (1998). ORM/NIAM object-role modeling. In P. Bernus, K. Mertins, & G. Schmid (Eds.), *Handbook on information systems architectures* (pp. 81-101). Berlin: Springer-Verlag.
- Halpin, T. (2001). *Information modeling and relational databases*. San Francisco: Morgan Kaufmann.
- Halpin, T. (2003, October). Uniqueness constraints on objectified associations. *Journal of Conceptual Modeling* [Online]. Retrieved from <http://www.orm.net/pdf/JCM2003Oct.pdf>
- Halpin, T. (2004a). Comparing metamodels for ER, ORM and UML data models. In K. Siau (Ed.), *Advanced topics in database research* (Vol. 3, pp. 23-44). Hershey, PA: Idea Publishing Group.
- Halpin, T. (2004b). Business rule verbalization. In A. Doroshenko, T. Halpin, S. Liddle, S., & H. Mayr (Eds.), *Information systems technology and its applications, Proceedings of ISTA-2004* (LNIP-48, pp. 39-52). Salt Lake City, Utah.
- Halpin, T. (2005, in press). ORM 2. In *Proceedings of ORM 2005 Workshop*. Springer-Verlag.
- Halpin, T., Evans, K., Hallock, P., & MacLean, W. (2003). *Database modeling with Microsoft® Visio for enterprise architects*. San Francisco: Morgan Kaufmann.
- Halpin, T., & Wagner, G. (2003). Modeling reactive behavior in ORM. In *Proceedings of the 22nd ER Conference*, Chicago: Springer LNCS.
- Hegarty, M. (n.d.). *Referential properties of factive and interrogative complements indicate their semantics* [abstract]. Retrived from <http://www.linguistics.berkeley.edu/BLS/abstracts/0113.pdf>
- Kristen, G. (1994). *Object orientation — The KISS method: From information architecture to information system*. Reading, MA: Addison Wesley.
- Lyons, J. (1995). *Linguistic semantics: An introduction*. Cambridge, UK: Cambridge University Press.

- OMG (2003a). *UML 2.0 Infrastructure specification*. Retrieved from <http://www.omg.org/uml>
- OMG (2003b). *UML 2.0 Superstructure specification*. Retrieved from <http://www.omg.org/uml>
- OMG (2003c). *Business semantics of business rules RFP*. Retrieved from <http://www.omg.org/cgi-bin/doc?br/2003-6-3>
- Rumbaugh, J., Jacobson, I., & Booch, G. (1999). *The unified language reference manual*. Reading, MA: Addison-Wesley.
- Smith, B. (1989). Logic and the Sachverhalt. *The Monist*, 72(1), 52-69. Retrieved from <http://ontology.buffalo.edu/smith//articles/logsvh.html>
- ter Hofstede, A. H. M., Proper, H. A., & Weide, Th. P. van der (1993). Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 18(7), 489-523.
- Thomas, J. (1995). *Meaning in interaction: An introduction to pragmatics*. London: Longman.

Chapter VIII

A Template-Based Analysis of GRL

Patrick Heymans, University of Namur, Belgium

Germain Saval, University of Namur, Belgium

Gautier Dallons, DECIS SA/NV, Belgium

Isabelle Pollet, SmalS-MvM/Egov, Belgium

ABSTRACT

This chapter applies the template proposed by Opdahl and Henderson-Sellers to the Goal-oriented Requirements Engineering Language (GRL). It proposes a metamodel of GRL that identifies the constructs of the language and the links between them. Each construct is then described through the template in order to extract and formalise detailed syntactic and semantic information. The latter takes the form of a mapping between a construct and its meaning, defined in terms of the Bunge-Wand-Weber ontology. Evaluations of both GRL and the template are provided as well as suggestions for improvements. The purpose of our work is to improve the quality of goal modelling. Indeed, despite the increasing popularity of the goal-oriented paradigm, especially in requirements engineering and enterprise modelling, the central notion of goal remains one of the most controversial. A possible cause might be that researchers have devoted too little attention to studying the ontological foundations of goal-oriented languages. This chapter addresses such issues for GRL.

INTRODUCTION

In 2003, the UEML thematic network (IST Project 2001-34229) started to develop the so-called Unified Enterprise Modelling Language (UEML), a conceptual modelling language designed to be a common ground for representing the various aspects of the enterprise and facilitating the exchange of enterprise models. Reaching these objectives was deemed of utmost importance to improve the development, interoperability, and integration of enterprise information systems.

Indeed, an increasing number of new technologies that strongly rely on enterprise models and ontologies are being introduced into the enterprise: enterprise application integration, domain-specific languages, content and knowledge management systems, semantically enriched agents and Web services, virtual enterprises, and the like. As more and more partial models of the enterprise are created, in different tools and different languages, the risk is that knowledge becomes dispersed and inconsistent. UEML was developed as part of a solution to these problems.

In 2004, UEML 1.0 was delivered. Due to the nature of the UEML project,¹ UEML 1.0 allows mainly the modelling of process aspects but leaves out other aspects (such as static information, functional and non-functional requirements, resources, and goals) and covers only a small set of the identified requirements. It was defined by integrating subsets of three existing enterprise modelling languages (EMLs) — namely, global returnable asset identifier (GRAI) (Doumeingts, 1984), EEML (Jorgensen & Carlsen, 1999) and IEM (Mertins & Jochem, 1999) — by following a methodology inspired by database integration (Petit, 2003). Development of the UEML has since been taken over by the InterOP Network of Excellence (InterOP Project Web site, 2004). It is currently an on-going activity carried out by a consortium made up of the leading practitioners and researchers in the domain of enterprise modelling (EM). The adopted language development approach reconciles scientific rigour and pragmatism. First, it is a requirements document under continuous elaboration that drives the language development process. Second, commonly used EMLs are analysed, each in turn, according to a quality evaluation framework inspired by Krogstie and Sølvsberg (2000) in order to guarantee that not only the most used but also the most appropriate and sound constructs are incorporated into the UEML definition. Third, the integration of these constructs into the UEML is done in such a way that syntactic and semantic problems (widespread in other unified languages, such as UML) do not arise. Examples are synonymous, homonymous, underdefined, ill-defined, overly complex or poorly integrated constructs. Finally, evaluations of the successive versions of the language are performed to provide continuous feedback to the language development requirements and process.

The work reported in this chapter describes some contributions of the authors towards the development of UEML 2.0. We focus on the analysis of existing EMLs and, more specifically, on the analysis of GRL, the goal-modelling language standardized by the International Telecommunication Union (ITU, 2003a). GRL is presented in section “GRL in a Nutshell.” The future integration of a goal-modelling language into the UEML is expected to improve it by adding the goal dimension that is currently missing. GRL is one of the candidates. In contrast with the previous version of the UEML, for which the semantic aspects of the EMLs were not taken into account in a systematic manner, this time, for UEML 2.0, we decided to adopt the template-based approach (or template, for short) defined by Opdahl and Henderson-Sellers (2004). The template was designed as

a simple and easy tool to provide a semantic definition to a modelling language in a systematic way. This makes the template well suited for working in a distributed environment like the InterOP NoE, because definitions done by different researchers can be compared easily. The template is introduced in the section entitled “Template-based Analysis of Modelling Languages.” It is meant to be applied to every construct pertaining to a language’s abstract syntax (or metamodel). Since GRL does not have a proper metamodel, we had to define it ourselves (see the section, “A Metamodel for GRL”). The template-based analysis of GRL’s constructs follows in the section entitled, “Template-based Analysis of GRL Constructs,” and then undergoes discussion in the section that follows. Because of the space limitations of this chapter, it was impossible to reproduce and discuss the analyses of all the GRL constructs that we performed. Instead, we develop our most important results and summarize the others. The companion technical report (Dallons, Heymans, & Pollet, 2005) provides the full details. We conclude with a summary and an outlook towards future work.

GRL IN A NUTSHELL

GRL stands for Goal-oriented Requirements Language. It results from the integration of the *i** goal-modelling language (Yu, 1997) and the NFR framework (Mylopoulos, Chung, & Nixon, 1992). The latter consists of a language and method designed to represent and reason about non-functional requirements (NFRs). Both languages originate from research performed at the University of Toronto and were among the first ones to consider goals as first-class citizens. However, each of them has a different focus. Indeed, NFR, which is a few years older than *i**, had for primary concern the modelling of NFRs and the various types of relationships between them (and- and or-decomposition, positive and negative contributions, etc.). NFR comes with goal decomposition strategies as well as propagation algorithms to estimate the satisfaction of higher level goals given the (more measurable) attainment or non-attainment of lower level ones. *i**, on the other hand, focuses on modelling the intentions of, and strategic dependencies between, actors. Dependencies between actors concern goals, softgoals, resources, and tasks. It is the key concept of goal that makes the link between the two notations.

GRL is now in its third version and is a component of the unified requirements notation (URN) standardized by the ITU. URN has two main components: use case maps (URN-UCM, see ITU, 2003b) and GRL aka URN-NFR. Note that two other well-known and widely used languages are standardized by the ITU, namely MSCs and SDL.

Analysis of GRL was given a high-priority by the UEML development team because (1) the lack of goal and NFR modelling is currently a major drawback of UEML; (2) the GRL notation is expected to gain popularity through its development by an international standardization body such as the ITU; (3) GRL is already the result of the integration of two complementary pioneering goal-oriented languages; and (4) a relatively precise definition of GRL’s syntax is public, which we have not found to be the case for other goal-oriented languages.

Details on the constructs of the language will be given below where we report on their analysis. First, we just give an overview by quoting the International Telecommunication Union (ITU, 2003a):

The URN-NFR language specified here is GRL [...], which is a language for supporting goal-oriented modelling and reasoning about requirements, especially non-functional requirements. It provides constructs for expressing various types of concepts that appear during the requirement process. There are four main categories of concepts: actors, intentional elements, non-intentional elements, and links. The intentional elements in GRL are goal, task, softgoal, resource and belief. They are intentional because they are used for models that allow answering questions such as why particular behaviours, informational and structural aspects were chosen to be included in the system requirements, what alternatives were considered, what criteria were used to deliberate among alternative options, and what the reasons were for choosing one alternative over the other. Actors are holders of intentions, they are the active entities in the environment or the system, who want goals to be achieved, tasks to be performed, resources to be available and softgoal to be satisfied. Links are used to connect isolated elements in the requirement model. Different types of link depict different intentional relationships. Non-intentional elements are equipped as a mechanism to refer to objects outside GRL model.

Figure 1. A model of a centralized insurance claim handling process

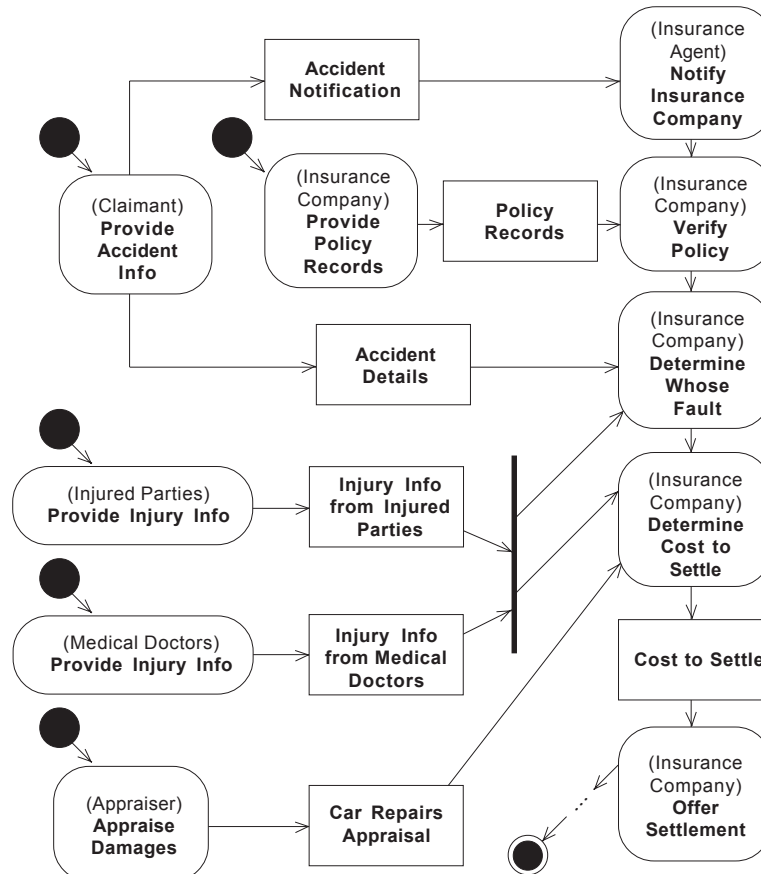
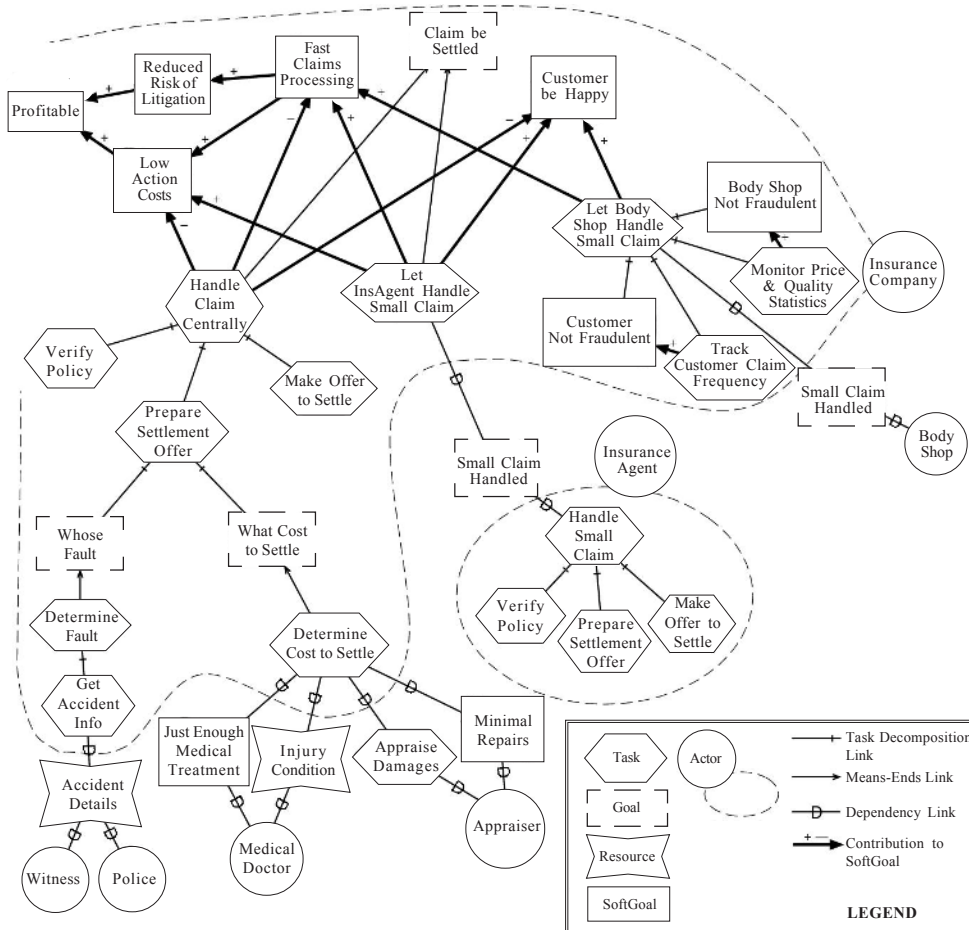


Figure 2. A GRL model of claim handling with alternatives²

To introduce GRL and illustrate the complementarity of goal modelling with respect to traditional process modelling, we take the example of an automobile insurance company described by Yu and Mylopoulos (1997) and Yu (2001). Let us assume that the company currently processes the claims centrally. A typical process for insurance claims is described in Figure 1. The claimant notifies the insurance company, which checks the accident details and policy records of the claimant. The injured parties and doctors give injury information while the appraiser provides an estimate of the cost of repair. Then, the insurance company determines the cost to settle and makes an offer. In such a process, different actors have different and sometimes conflicting goals that here remain unstated. Nevertheless, ideally, an adequate solution (process) must emerge from a negotiation and from priorities based on the goals of the various parties involved.

Basic goals are, for example:

- the car owner wants his car fixed quickly;
- the insurance company wants to minimize claims pay-out;
- the car owner wants fair appraisal of repairs; and
- insurance company wants to maintain good customer relations.

These goals (as well as potentially many others not mentioned here) and their relative importance might push towards the adoption of a different process than the one currently in place. For example, if good customer relations become a top priority for the company, it might be decided to move from the current centralized claim processing to a more decentralized one where body shops can handle small claims directly, thereby returning a repaired car more quickly to the customer. Using goal models gives more flexibility because such models open the way for a variety of choices (processes) that might be overlooked otherwise. It can also reveal conflicting goals that could impede the future solution and that have to be negotiated and mitigated before going further.

Figure 2, adapted from Yu and Mylopoulos (1997), provides a GRL model for the same example.³ We start commenting the example from within the boundary of the `InsuranceCompany Actor`, represented by the dashed curve. The `InsuranceCompany` has one central goal: `settle the claim`. In GRL, a `Goal` can be refined into subgoals and/or subtasks. `Tasks` are linked to `Goals` through `Means-Ends` links indicating they are a means to achieve such `Goals`. For instance, each of the `Tasks` `HandleClaimCentrally`, `LetInsAgentHandleSmallClaim`, and `LetBodyShopHandleSmallClaim` is an alternative way to achieve the `ClaimBeSettled Goal`.

When handling claims centrally, the insurance company has to achieve three subtasks: verify the policy, prepare a settlement offer, and make the offer. A `Task A` linked to another `Task B` by a `Decomposition` link indicates that `A` participates to the achievement of `B`. Graphically, a segment cuts the link on the compound `Task`'s side. For example, the `Task HandleClaimCentrally` has three subtasks — `VerifyPolicy`, `PrepareSettlementOffer` and `MakeOfferToSettle` — that must be performed to achieve it.

In order to prepare a settlement offer, the insurance company must determine whose fault it is and what cost to settle. The `Task PrepareSettlementOffer` is first decomposed into `Goals` which then, in turn, are refined into `Tasks` which can be decomposed again. Finally, we get to the `Task GetAccidentInfo`. To achieve this `Task`, the insurance company must get accident information from `Actors Police` and `Witness`. In GRL, we model this through a `Dependency` link between the `Actors`⁴. To achieve the `Task GetAccidentInfo`, `InsuranceCompany`, hence, depends on `Police` and `Witness` to provide the required informational `Resource AccidentDetails`. In GRL, the latter is called the `Dependum` of the `Dependency`. `Goals`, `SoftGoals` (see below) and `Tasks` can also be used as `Dependums` as appears in other `Dependencies` in the figure.

The `Actor MedicalDoctor` is in charge of providing information on injuries to determine the cost to settle and the `Actor Appraiser` is asked to assess the cost of the damage. They both have `SoftGoals` pushing them to minimize their costs by limiting themselves to adequate medical treatment and minimal repairs, respectively.

Finally, `Tasks` contribute to the achievement of `SoftGoals` by `Contribution` links, positive or negative.^{5,6} `SoftGoals` differ from `Goals` in that it is subjective to

tell whether they are achieved or not, but one can still try to push towards or against their satisfaction. The `SoftGoal LowAdminCosts`, for example, is favored by the `LetInsAgentHandleSmallClaim Task` but, on the contrary, potentially harmed by `HandleClaimCentrally`. In turn, `LowAdminCosts` favors `Profitability`.

The above explanation is by no means a detailed discussion of the example, nor a proper introduction to the GRL language. It is just meant to provide some intuition in the event that the reader is not familiar with the language. For a more comprehensive introduction to the language, the reader is referred to the standard (ITU, 2003a). The content of the latter is as follows:

- a general introduction (from which the paragraph cited earlier in this section was taken);
- three concrete syntaxes for GRL: a textual syntax (expressed in BNF), a graphical syntax (expressed in BNF augmented with topological information), and an XML syntax (expressed in an XML Document Type Definition (DTD));
- informal semantic definitions of the constructs;
- examples of GRL models; and
- a tutorial.

No abstract syntax is provided⁷ in the standard. Thus, we provided our own that we elaborated from the concrete syntaxes listed above. Doing this, we encountered inconsistencies, ambiguities, and underdefinitions. Some will be mentioned as we go along, together with the explanation of how we chose to resolve them.

A METAMODEL FOR GRL

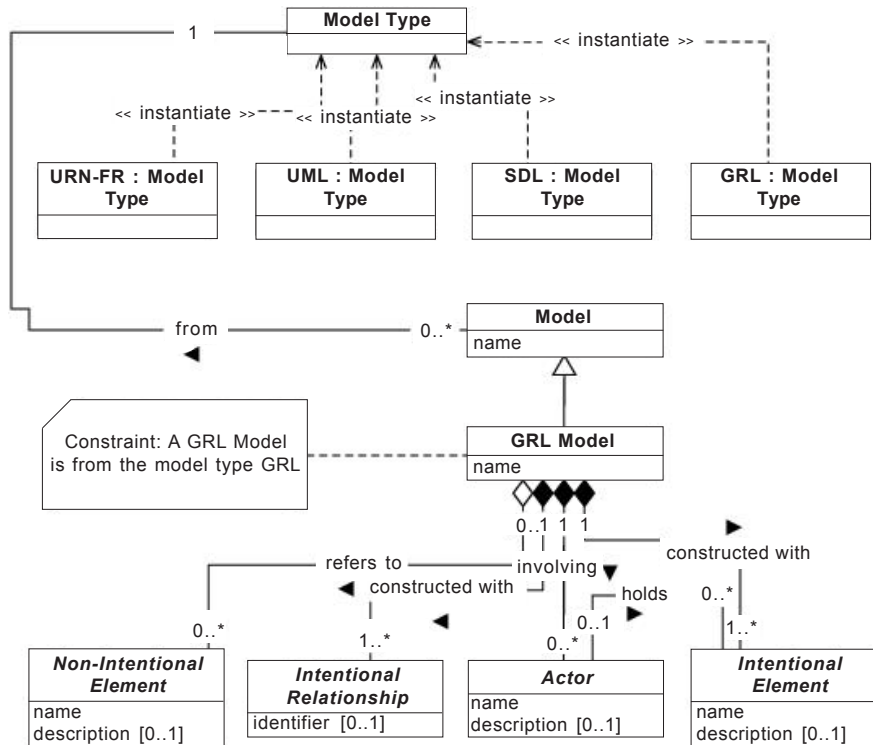
As a metamodeling language, we use standard UML Class Diagrams. The metamodel discussed in the sequel and shown in Figures 3 to 5 is a simplified version of the complete metamodel we have made. The one we show here is focused on the topics discussed in the chapter. The reader interested by the complete metamodel will find it in the companion technical report (Dallons et al., 2005).

Figure 3 gives the top-level view of the metamodel. `ExceptModel` and `ModelType`, which are generic to all the languages we plan to examine, all the classes in this and subsequent diagrams are specific to GRL. We have located them in the GRL package in order to avoid confusion in a multilanguage context.

In Figure 3, the four main types of elements mentioned in the quoted paragraph in the previous section appear immediately: `Actor`, `IntentionalElement`, `NonIntentionalElements`, and `link` (which was renamed `IntentionalRelationship` to be compliant with the GRL syntax definition). Instances of `NonIntentionalElements` are “proxies” used by GRL to refer to constructs in different models. This provides traceability of GRL models with other types of models used during a project. We skip this issue in the chapter.

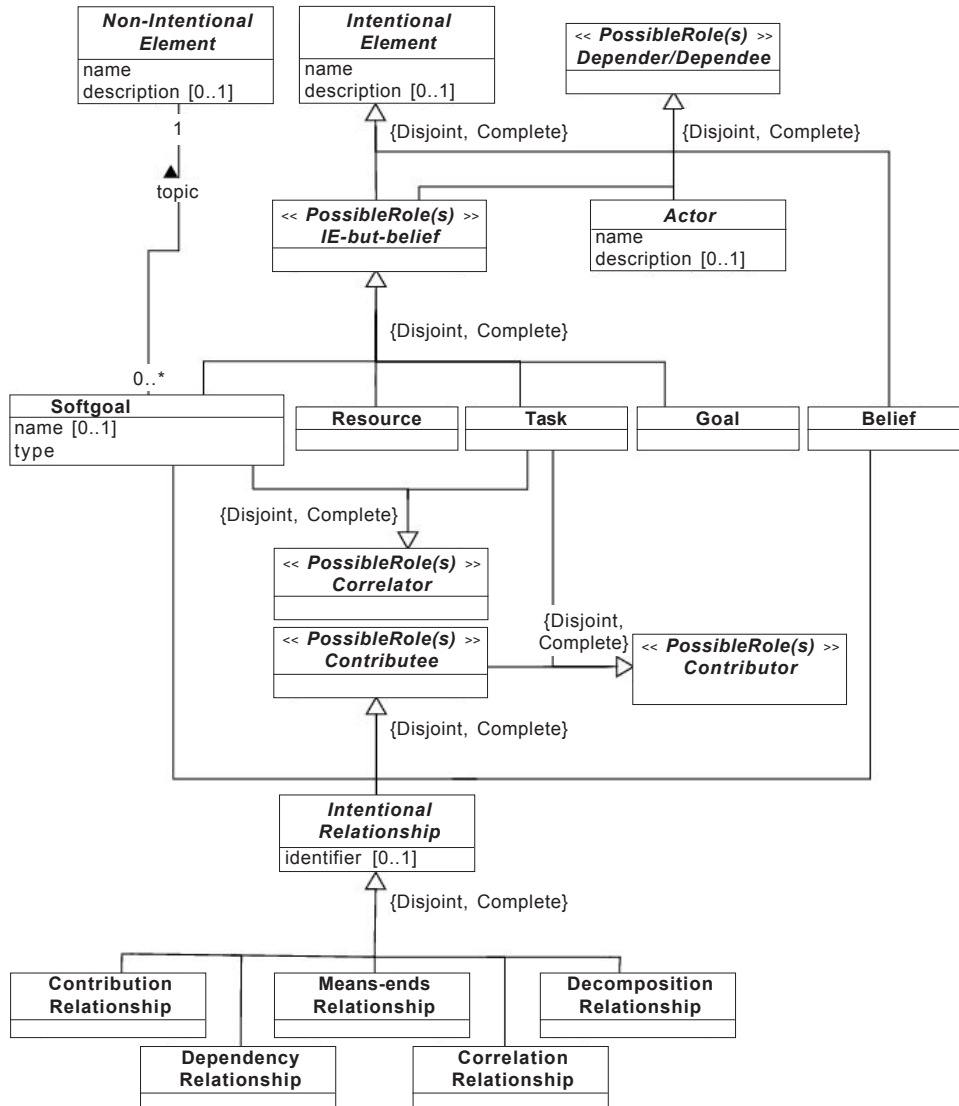
Figure 4 details the five kinds of intentional elements: `SoftGoal`, `Resource`, `Task`, `Goal` and `Belief`⁸. Figure 4 also introduces the various kinds of GRL relationship types as subclasses of `IntentionalRelationship`. How these interact with other classes is detailed in Figure 5.

Figure 3. Top-level view of the GRL metamodel



In Figure 4, various abstract classes appear: *IEButBelief* (denoting all intentional elements except beliefs), *Correlator*, *Contributee*, *Contributor*, and *Depender/Dependee*. These latter classes are somewhat artificial. We introduced them for the sole purpose of showing graphically and succinctly the groups of classes that are likely to play a role in an intentional relationship. For example, the GRL syntax requires that the contributee in a contribution relationship is either a belief, a softgoal, or an intentional relationship. Therefore, we have introduced an abstract superclass *Contributee* generalizing *Belief*, *SoftGoal*, and *IntentionalRelationship*. This way, modelling GRL relationship types becomes much easier. We can observe that in Figure 5. For example, we see that representing contribution relationships only requires one class (*ContributionRelationship*) and two associations (*contributor* and *contributee*), both pointing to abstract classes (respectively, *Contributor* and *Contributee*). If we had not introduced these abstract classes, the associations would have been multiplied (one for each type of contributor and one for each type of contributee), and Object Constraint Language

Figure 4. GRL metamodel: Zoom on intentional elements



(OCL) constraints would have been necessary to exclude the possibility that a contribution relationship has more than one contributee. To distinguish them from the other more obvious classes, all such abstract superclasses have been given the stereotype `<<PossibleRole(s)>>` and were named after their corresponding roles (except **IEButBelief**, for brevity).

Figure 5. GRL metamodel: Zoom on intentional relationships

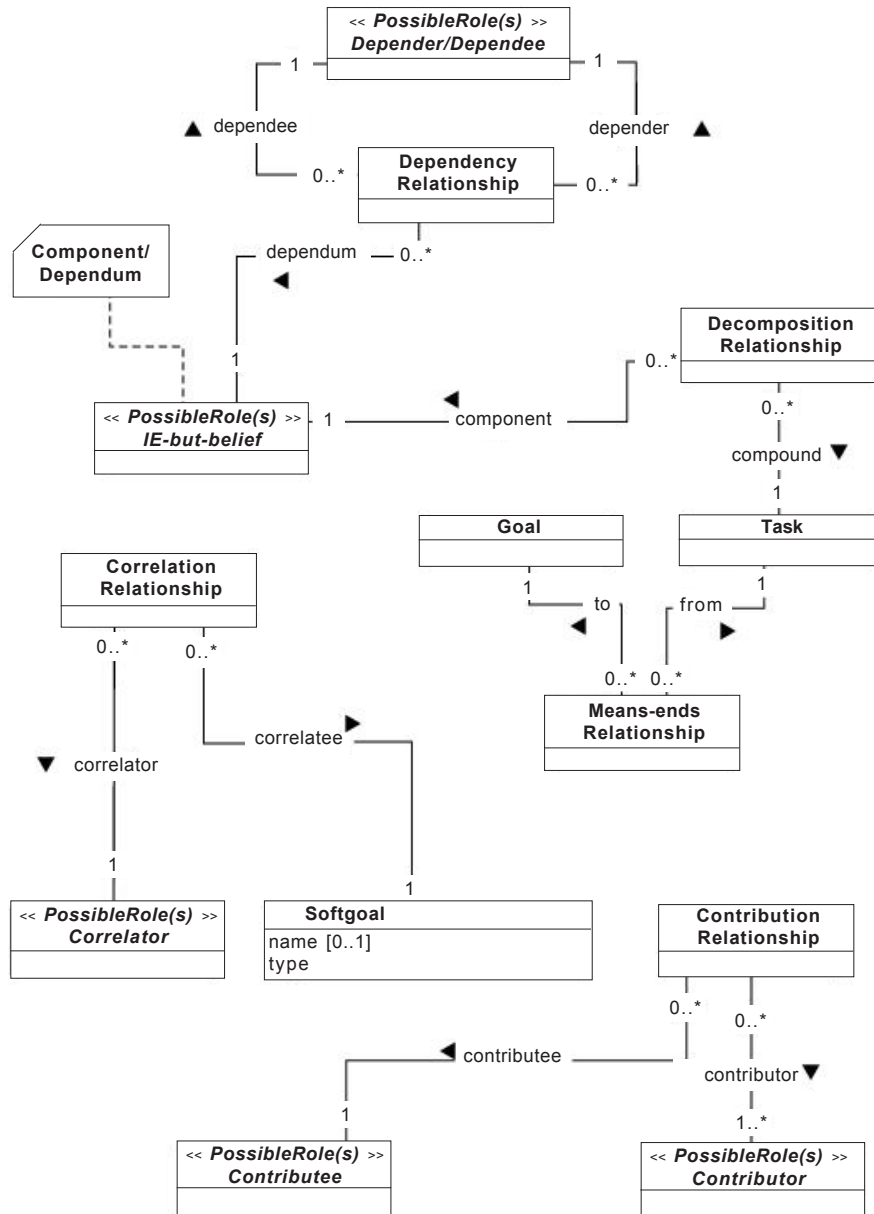
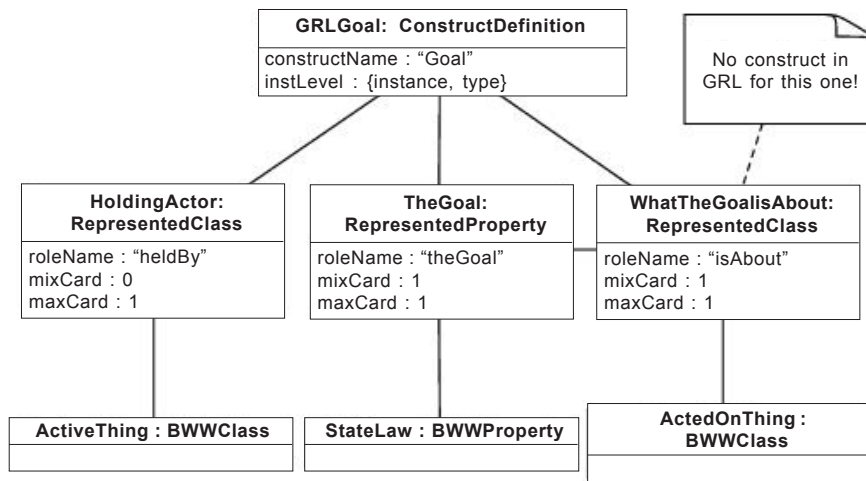


Figure 6. Semantic mapping for the Goal construct



TEMPLATE-BASED ANALYSIS OF MODELLING LANGUAGES

In this section, we briefly present the template that we have used to analyze the GRL constructs. The template was proposed by Opdahl and Henderson-Sellers (2004) as a means to systematize the description of EML constructs. It can be used for various purposes like comparing and integrating EML constructs or, simply, to better understand them. Translation between EMLs is another possible use.

Opdahl and Henderson-Sellers (2004) describe the template as follows:

By “template” we mean a standard way of defining modelling constructs by filling in standard sets of “entries,” some of which are complex and some of which are interrelated. [...] The main idea is to provide a standard way of defining modelling constructs in terms of the BWW model [see further in this section], in order to make the definitions cohesive and, thus, learnable, understandable, and as directly comparable to one another as possible. Another important idea is to provide a way of defining modelling constructs not only generally, in terms of whether they represent “classes,” “properties,” or other ontological categories, but also in terms of which classes and/or properties they represent, in order to make the definitions more clearly and precisely related to the enterprise.

In version 1.1 of the template (the latest at the time of writing), each construct is defined by filling in the following sections:

1. **Preamble:** General issues are specified here, namely, construct, diagram type, language name and version, acronyms, and external resources.

2. **Presentation:** Such issues as lexical information (icons, line styles), syntax, and layout conventions are specified here.
3. **Semantics:** This section is the most important as well as the most complex. It requires the analyst to answer the following questions:
 - Is the construct at the *instance* or *type* level?
 - Which *class(es)* of things in the world does the construct represent?
 - Which *property(-ies)* of those things does the construct represent?
 - Which *segment of the lifetimes* of those property(-ies) and things does the constructs represent? This question is only relevant for constructs denoting *behavioural* properties.
 - What is the *modality* of the assertions made using the construct? Is it that *something is the case* (regular assertion)? Is it that *somebody wants something to be the case*? Is it that *somebody knows that something is the case*? And so forth.
4. **Open issues:** All the issues that the template failed to address should be mentioned here.

The section of the template devoted to semantics is by far the most important. It provides a standard way of expressing what a construct *represents*. It is based on the Bunge-Wand-Weber (BWW) ontology (see, e.g., Wand & Weber [1988, 1993, 1995]), also called the BWW model. The BWW model is an adaptation to the information systems field of Mario Bunge's philosophical ontology (a theory about the nature of things in general) (Bunge, 1977, 1979). The BWW model is now a widespread reference for the semantic definition and evaluation of information system concepts. How it was constructed over time, how it has been used previously to define and evaluate modelling languages, and what are its advantages over alternative frameworks is described elsewhere, for example, in Opdahl and Henderson-Sellers (2004). Here, we will just quickly go through the basics, in an extremely simplified manner, and explain more advanced elements further, as they are needed. The following explanation is based on Opdahl (2005), which summarizes various sources where the BWW model is defined, including those cited above.

The basic assumption of Bunge and the BWW model is that the world (independent from the human observers) consists of *things* and *properties*. All the other concepts derive from these two central concepts. BWW things are concrete things, for example, "atoms, fields, persons, artifacts and social systems" (Bunge, 1999). Things possess properties. Properties cannot themselves have properties. One can talk about *particular properties* of an individual thing, like "my bike is red," or *general properties* possessed by many things, like "red bikes are nice." A collection of things that all possess the same general property is called a *class*. Such a property is called the *characteristic property* of the class. All the things that possess it belong to the class and, conversely, all the things that do not have this property do not belong to the class. The main structuring mechanism for classes is *generalisation/specialisation*. The generalisation/specialisation relationship parallels the *precedence* relationship that operates on properties. A property *p* precedes another property *q* if all things that possess *q* also possess *p*. For instance, "being alive" precedes "being a mammal." *Q* is a subclass of *P* if each

characteristic property of P precedes a characteristic property of Q. Many classes and properties are already “predefined” and structured in their respective hierarchies in the BWV model. We do not have space to start presenting them here. Generalisation/specialisation and precedence are central because they are the main structuring mechanisms through which a common ontology of the enterprise domain will be built in the UEML “on top of” the BWV model.

The BWV model, despite recent efforts to formalize it and make it more accessible (see, for example, Rosemann & Green [2002]), remains complex, sometimes ambiguous, and not so well known by EM experts. One of the main advantages of the template is that it does not require its users to be BWV or ontology experts. It helps relate EML constructs to the abstract categories of BWV by asking simple questions, giving practical recommendations, and providing concrete examples. It is systematic, and this makes the definitions of modelling constructs made by different persons directly comparable, which is what we were looking for in our distributed research context.

TEMPLATE-BASED ANALYSIS OF GRL CONSTRUCTS

We have defined 11 GRL constructs through the template. Due to space limitations, we cannot provide the detailed analysis of each construct in this chapter. We will just give and comment on the analysis of the Goal construct and summarize the results obtained for the other constructs. The interested reader will find all the detailed construct analyses in the technical report (Dallons et al., 2005).

As a preliminary remark, we would like to insist on the following: we do not claim that this semantic mapping is better than any other. We have tried to be as faithful as we could to the GRL and BWV definitions but, in the end, the template remains the product of our subjectivity. It is exposed here for the purpose of being discussed with peers and hopefully improved.

The filled-in template for Goal can be found in the Appendix. The information gathered in the *Preamble* and the *Presentation* sections is pretty straightforward. In particular, the content of subsections *Builds on*, *Built on by*, *User-definable attributes*, and *Relationships to other constructs* could have been derived almost automatically from the GRL metamodel (or another syntax). The most interesting section is the one devoted to *Semantics*. Complementary to the textual version of the template, Opdahl and Henderson-Sellers (2004) have also defined a class diagram describing the semantic information required by the template. Hence, the analysis of a construct can also be represented as an instantiation of this class diagram. This is what we provide in Figure 6, for the Goal construct.

The top level of the diagram contains the instance of the *ConstructDefinition* class at hand: Goal, in this case. On the bottom level, we find the instances of *BWVClass* and *BWVProperty* that the construct represents. Since the same BWV-class can be represented by more than one construct or be represented several times by the same construct, Opdahl and Henderson-Sellers (2004) introduced an intermediate level where the mapping between the modelling constructs and the BWV elements is made explicit. An instance of *RepresentedClass* or *RepresentedProperty* is therefore specific to a single construct.

After going through the informal semantics provided in the GRL specification, we understood that Goal primarily represents a *state law* property. BWV defines a *state law* as “a law that constrains the values that other properties can have for individual states the thing can be in, that is, state laws are structural/static” (Opdahl, 2005). Indeed, in GRL, as usual in goal modelling languages, goals are used to express constraints on the possible states in which a thing can be. This thing is usually the proposed system, the entire organization, or a particular actor. The problem is that there is no construct in GRL to indicate what this thing is, that is, what the goal is about.⁹ On the other hand, in the BWV model, *state law* is a property which, as with all BWV properties, must be possessed by some BWV class. The most general BWV class we found that could fulfill the role *isAbout is*, according to us, *acted on thing* because it seems that one can reasonably only want to constrain the state of things that can be acted upon. The BWV model says, “One thing *acts on* another thing if and only if the history of the second thing would have been different had the first thing not existed” and “[...] one thing is *acted on* by another thing if and only if the second thing acts on the first” (Opdahl, 2005).

These definitions raise a second issue: what is the thing acting on the acted-on thing, then? To answer this question, one should recall that Goals are eventually reduced to Tasks through Means-end relationships. It is the holders of those Tasks that are responsible for acting on the acted-on thing. Hence, the semantic mappings of these latter two constructs will have to provide the answers.

At this stage, we continue with Goal by noticing that a goal might be contained within an Actor Boundary. But standalone goals can also exist in GRL. Thus, in Figure 6, the represented class instance HoldingActor has a minimum cardinality of 0. The Actor construct is mapped to the BWV class *active thing*. We also find this to be a problem. This time, it is not due to constraints in the BWV model itself but to the *Modality* entry of the template (see Appendix), something that the BWV model does not yet account for. Indeed, a Goal is not an assertion that is always true, but rather one that an Actor (the holding actor) wants to be true at some stage. So, the question is, if the Goal is not held by an Actor, then how do we identify who wants it? GRL does not force the modeller to answer this question. Since the notion of modality is not really part of the BWV model right now, we cannot say that a constraint in the BWV model is violated, but we find it disturbing that GRL allows the Actor holding the goal to be omitted by the modeler.¹⁰ A related question is: do Actors have to be humans or can they be other things (e.g., computer systems, organisational systems, specific individuals, classes, roles,...)? We have found no answer in the GRL specification, so, we decided to remain as general as possible: the Actor construct was mapped to the BWV class *active thing*.

For the rest of the constructs, our analysis is summarized in Table 1. The left column lists the GRL constructs. The upper part of the table lists the intentional elements, while the lower part is devoted to intentional relationships. If a construct is primarily mapped to a BWV class, this is indicated in the middle column. If it is primarily mapped to a BWV property, this is indicated in the right column.

The Resource construct is mapped to the *acted on thing* class. The Goal, SoftGoal, and Belief constructs are all mapped to a *state law* property. We have already given the explanation for Goal. The same explanation holds for SoftGoal. A Belief is also a *state law*, but it has a different modality than goals. Here, the modality is that the holder “thinks” that the assertion is true, whereas for goals, he “wishes” or

Table 1. Summary of the GRL template analysis

GRL construct	Class entry	Property entry
Actor	<i>Active Things</i>	—
Goal	—	<i>State Law</i>
Task	—	<i>Transformation Law</i>
SoftGoal	—	<i>State Law</i>
Resource	<i>Acted On Things</i>	—
Belief	—	<i>State Law</i>
Means-ends	—	<i>Transformation Law</i>
Dependency	—	<i>State Law</i>
Decomposition	—	<i>Transformation Law</i>
Contribution	—	<i>Mutual</i>
Correlation	—	<i>Mutual</i>

“wants” it to become true. Just as for Goal, the issues of who is the holder and what is the *state law* about hold for SoftGoal and Belief, too.

Task is mapped to a *transformation law*. “A *transformation law* is a law that constrains the values that the other properties can have across multiple states, that is, transformation laws are behavioural/dynamic” (Opdahl, 2005). Indeed, a Task will have an impact on some *thing* and will hopefully result in a change of the state of world. However, as for Goal, only the holder of the Task can be specified in GRL; the target cannot.¹¹ If the target could be specified, consistency between what the fulfilled¹² Goal is about and what the Task targets could be verified, but at the moment, all this remains tacit knowledge in GRL.

The Means-End relationship is also mapped to a *transformation law*. The end is a Goal and the means (Task) is the way to achieve it. So, Means-End defines a transformation from the current state of what the Goal is about to a next state closer to satisfying the *state law* expressed by the Goal. Again, tacit knowledge in GRL about the object of the Goal prevents being more precise in the semantic mapping and doing more accurate verification of models. We understand Decomposition in a similar way; it is also a *transformation law*. For example, a system with only one task evolves towards a system with several tasks that are the subtasks of the former. We understand Means-End and Decomposition as two kinds of system refinements; however, what the system is is never defined in GRL.

The Dependency relationship is mapped to a *state law*. This relationship denotes the dependency of an actor on another with respect to an object of dependency, called the Dependendum. The state of the Dependendum is constrained by the Dependency between Actors. Hence, a *state law*.

Finally, *Contribution* and *Correlation* are mapped to *mutual property*. A *contribution* is a shared property between two coupled objects. In BWW terms, this amounts to a mutual property. A *correlation* is similar, except that it does not happen by design but as a side-effect.

DISCUSSION

Assessment of GRL

First, we recall the subjectivity of the results exposed in the previous section. This is actually reinforced by the fact that the GRL specification is quite imprecise. Indeed, in the specification, we found only very broad semantic definitions, and the tutorial does not help us make them more precise. Most of the time, our interpretation has played a key role in the understanding of GRL constructs. This could be seen as a force since, this way, the GRL application domain remains vast. From our point of view, this is also a weakness because we were left with many questions, which could be a major impediment if one has to make a concrete GRL model, check its consistency, or transform an existing GRL model into another notation. In particular, we think that all the issues mentioned in the previous section about the central concepts of *Goal*, *SoftGoal*, *Task*, and *Actor* are quite serious. On the other hand, we are pleased to observe that researchers are now busy investigating semantic issues related to goals. For example, Regev and Wegmann (2005) also highlight the problem we uncovered, that is, how to make the object of goals explicit (although their approach has little to do with ours because they use an approach based on General Systems Thinking). The problem does not seem to be bound to GRL, however. Rather, it seems to be widespread among all goal-modelling languages.

Another problem we encountered is the existence of contradictions between the concrete syntaxes from which we had to build the metamodel. We had to make choices that do not necessarily represent the intentions of the GRL authors. For example, the textual syntax sometimes allows so-called short-hand forms that we doubt would be in compliance with the informal semantics. An example is decomposition. In the text, only tasks are said to be decomposable. However, the syntax allows a shortcut where a goal can be decomposed. In this case, we have decided to stick to the text and ignore the syntax definition. The metamodel presented in this chapter was constructed in this spirit.

Finally, we think the textual syntax could be improved especially with respect to the chosen keywords which are not always intuitive. For example, the syntax of decomposition is defined by the following rule:

DECOMPOSITION *Optional Identifier* **FROM** *sub-element* **TO** *Decomposed Element*

We think the **FROM** and **TO** keywords are quite misleading in this order. A more intuitive definition could be: **DECOMPOSITION** *Optional Identifier* **OF** *Decomposed Element* **INTO** *sub-element*.

Assessment of the Template-Based Approach

For our purpose, the template was found to be very useful. It helped to raise a number of important issues about the analysed language that have been exposed in the previous sections. Many of these problems are of a semantic nature and cannot usually be identified without some kind of formalisation of a language's semantics. Providing a formal semantics to a language can be a very complex task (Harel & Rumpe, 2000). What we appreciated in the template is its simple approach, which boils down to enrich a language's metamodel (usually describing only its abstract syntax) with semantic information (as seen in Figure 6). We found it fairly easy to use, even for those who, like us, are not BWV experts. Some familiarity was quickly gained by browsing through the many available examples, especially the analyses of well-known UML constructs. Still, we think that some improvements could be brought to the template.

First, it appeared that the predefined BWV concepts were quite broad. They were more or less sufficient because the semantic description of GRL is itself quite vague and wide-embracing. However, we encountered some problems. For example, a goal and a dependency are both BWV state laws, although they are very different concepts. Hence, it is important to refer not only to represented classes and properties (see Fig. 6), but sometimes also to the initial less formal definitions in order to understand the differences. In some cases, it would become quite dangerous to directly compare two concepts mapped to a single BWV class without looking at the original definitions, unless new, more specific, BWV classes could be defined by the user. This was not attempted in this first use of the template. Similarly, although we did not detail the various contribution and correlation subtypes at this stage, we foresee the same problem to occur here.

Another point is the modality field. Currently, it only asks whether the assertion is modal or not and gives a few examples to help the user describe the modality of the construct. That was sufficient because GRL is not very precise here either, but a finer grained list of modalities could be provided. This could be useful, for example, if we had to capture categories of goals such as those proposed by Kavakli (2002) or Letier (2001). Finally, we think that tool support could be of great help to fill in the various entries. It could give more guidance (by restricting the possible values), allow safer reuse than copy and paste (which was heavily used throughout the analysis and was the source of many mistakes), and directly create the links between BWV and metamodel elements (which would facilitate other automated treatments and visualizations).

SUMMARY AND FUTURE WORK

In this chapter, we have reported on the experimental analysis of the GRL language through the template-based approach defined by Opdahl and Henderson-Sellers (2004). Despite its simplicity and its discussed limitations, the template allowed us to identify a number of important issues in the current GRL specification. We think that the template is likely to scale up to be a solid basis for the analysis and comparison of enterprise modelling languages needed for the elaboration of UEML 2.0. However, due to the amount of subjectivity that we had to put into the analysis, we first need to discuss our results with peers before we can reach a stable consensus. This also includes validating the metamodel of GRL that we had to define in the process.

In the future, we plan to improve the analysis with the feedback obtained and go deeper into the exploration of constructs that necessitate the creation of custom BWW definitions. Other enterprise modelling languages will be analysed. Then, we will proceed to the selective integration of the analysed languages and constructs into UEML 2.0. For this larger scale application, tool support is deemed crucial and will be investigated readily.

NOTE

The reported work is supported by the Commission of the European Communities — InterOP Network of Excellence, C508011 (InterOp Project Web site, 2004).

ACKNOWLEDGMENT

We thank Andreas Opdahl for the time he spent sharing his knowledge, reviewing our analysis, and giving precious comments.

REFERENCES

- Bunge, M. (1977). Ontology I: The Furniture of the World. In M. Bunge, *Treatise on Basic Philosophy*, vol.3. Boston: Reidel.
- Bunge, M. (1979). Ontology II: A world of systems. In M. Bunge, *Treatise on Basic Philosophy*, vol. 4. Boston: Reidel.
- Bunge, M. (1999). *Dictionary of Philosophy*. Amherst, NY: Prometheus Books.
- Dallons, G., Heymans, P., & Pollet, I. (2005). *A template-based analysis of GRL*. Technical Report, University of Namur, Namur, Belgium.
- Doumeings, G. (1984). *GRAI: Méthode de Conception des Systèmes en Productique*. PhD thesis, University of Bordeaux, France [in French].
- Harel, D., & Rumpe, B. (2000). *Modeling languages: Syntax, semantics and all that stuff, Part I: The basic stuff*. Technical Report MCS00-16, Faculty of Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot, Israel.
- InterOP Project Web site. (2004). Retrieved from <http://www.interop-noe.org>
- ITU (2003a). Recommendation Z.151 (GRL)— Version 3.0. International Telecommunication Union, Geneva, SWI.
- ITU (2003b). Recommendation Z.152 (UCM)— Version 3.0. International Telecommunication Union, Geneva, SWI.
- Jorgensen, H., & Carlsen, S. (1999). Emergent workflow: Integrated planning and performance of process instances. In *Proceedings of Workflow Management '99*, Münster, Germany.
- Kavakli, E. (2002). Goal-oriented requirements engineering: A unifying framework. *Requirements Engineering Journal*, 6(4), 237-251.
- Krogstie, J., & Sølberg, A. (2000). Information systems engineering: Conceptual modeling in a quality perspective. Technical report, NTNU, Trondheim, Norway.
- Letier, E. (2001). *Reasoning about agents in goal-oriented requirements engineering*. PhD thesis, Université Catholique de Louvain, Belgium.

- Mertins, K., & Jochem, R. (1999). *Quality-oriented design of business processes*. Boston; Dordrecht; London: Kluwer Academic Publishers.
- Mylopoulos, J., Chung, L., & Nixon, B. (1992). Representing and using nonfunctional requirements: A process-oriented approach. *IEEE Trans. Softw. Eng.*, 18(6), 488-497.
- Opdahl, A. L. (2005). *Introduction to the BWW representation model and Bunge's ontology*. Technical report, InterOP Network of Excellence. Retrieved from <http://interop-noe.org/>
- Opdahl, A. L., & Henderson-Sellers, B. (2004). A template for defining enterprise modeling constructs. *Journal of Database Management*, 15(2), 39-73.
- Petit, M. (2003). Some methodological clues for defining a Unified enterprise Modelling Language. In K. Kosanke, R. Jochem, J. G. Nell, & A. O. Bas (Eds.), *Enterprise inter- and intra-organisational integration — Building an international consensus*. Norwell, MA: Kluwer Academic Publishers.
- Regev, G., & Wegmann, A. (2005). Where do goals come from?: The underlying principles of goal-oriented requirements engineering. In *Proceedings of Requirements Engineering Conference, 2005*, Paris.
- Rosemann, M., & Green, P. (2002). Developing a meta model for the Bunge-Wand-Weber ontological constructs. *Information Systems*, 27(2), 75-91.
- Wand, Y., & Weber, R. (1988). An ontological analysis of some fundamental information systems concepts. In J. I. DeGross & M. H. Olson (Eds.), *Proceedings of the Ninth International Conference on Information Systems* (pp. 213-225).
- Wand, Y., & Weber, R. (1993). On the ontological expressiveness of information systems analysis and design grammars. *Journal of Information Systems*, 3, 217-237.
- Wand, Y., & Weber, R. (1995). On the deep structure of information systems. *Journal of Information Systems*, 5, 203-223.
- Yu, E. (2001). Strategic actor relationships modelling with *i**. Lecture slides.
- Yu, E., & Mylopoulos, J. (1997). Modelling organizational issues for enterprise integration. In *Proceedings of International Conference on Enterprise Integration and Modelling Technology*.
- Yu, E. S. K. (1997). Towards modeling and reasoning support for early-phase requirements engineering. In *RE '97: Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE '97)* (p. 226). Los Alamitos, CA: IEEE Computer Society.

ENDNOTES

- ¹ The UEMML project lasted only 15 months and its objectives were confined to demonstrate the feasibility of using UEMML for exchanging models among three enterprise modelling software environments.
- ² Adapted from Yu and Milopoulos (1997)
- ³ Actually, this is an *i** model, but it is also a GRL model. Indeed, the syntaxes of both languages largely overlap.

- ⁴ This `Dependency` link is actually shorthand for two `Dependency` links: one between the `InsuranceCompany` and the `Police` and one between the `InsuranceCompany` and the `Witness`.
- ⁵ There are actually more types of `Contribution` links that are not discussed here.
- ⁶ `Correlation` links — not appearing in the example — are similar to `Contribution` links, except that they indicate side effects rather than effects looked for by design.
- ⁷ However, the standard indicates that this is foreseen.
- ⁸ The concept of `Belief` was not introduced in the example. In practice, it appears to be used less frequently than others. Basically, a `Belief` is an assertion used to motivate some claim (typically a `Contribution`) and hence attached to it.
- ⁹ One might argue that the general-purpose attribute that GRL offers for most constructs can be used, but the issue seems so central that we do not think it would be an appropriate solution. We believe that a first-class, built-in, mandatory construct would be needed. Furthermore, the general-purpose attribute only exists in the textual versions of GRL, not in the graphical syntax.
- ¹⁰ Unless this is just a view of a more complete specification, of course. But views (concrete syntax) should be separated from abstract syntax.
- ¹¹ Except, again, with the general-purpose attribute, but we have already argued against this solution.
- ¹² Through `Means-End` links. See discussion of next construct.

APPENDIX: TEMPLATE-BASED ANALYSIS OF **Goal**

Preamble

Construct Name

- Goal

Alternative Construct Names

- Condition to achieve
- State of affairs to achieve
- Objective

Related, but Distinct Construct Names

- SoftGoal

Related Terms

- **Intentional Element:** a Goal is an Intentional Element. Intentional Element is the set comprising SoftGoal, Resource, Task, Goal, and Belief.
- **Sub-element:** this is the role played by a Goal that is decomposed in a Decomposition.
- **Dependum:** that is the role played by a Goal, a SoftGoal, a Resource, or a Task depended upon in a Dependency.
- **End:** this is the role played by a Goal that is the objective achieved using Task in a Means-Ends link.

Comments: Sometimes a Goal plays the role of a Depender or a Dependee in a Dependency Relationship (if this Goal is held by an Actor). In this analysis, we have ignored this case.

Language

- Recommendation Z.151 (GRL) - Version 3.0, Sept. 2003
http://www.usecasemaps.org/urn/z_151-ver3_0.zip. Also called: GRL or URN-NFR (Last accessed May, 2005)

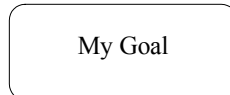
Diagram Type

- GRL Model (the only diagram type in GRL)

Presentation

Icon

- A Goal is represented by an oval with the name inside and attributes between square brackets. Here is an example :



Builds On

- None

Built On By

- A Dependency can have a Goal as a Dependium
- A Decomposition can have a Goal as a decomposed element
- A Means-Ends can have a Goal as an end element

Comments: Sometimes a Goal plays the role of a Depender or a Depende in a Dependency Relationship (if this Goal is held by an Actor). In this analysis, we ignored this case.

User-Definable Attributes

- **Name:** the name of the Goal
- **Description:** an optional textual description of the Goal
- Any other attribute that the user wishes to add.

Relationship to Other Constructs

- Belongs to 1..1 GRL Model
- Can have 0..n Attribute
- Can be held by 0..1 Actor
- **Can play the role of:**
 - a Dependium in 0..n Dependency links
 - a sub-element in 0..n Decomposition link
 - an end element in 0..n Means-Ends link

Layout Conventions

- Nothing particular

Semantics

Instantiation Level

- Both type and instance level

Classes of Things

- **ActiveThing:** representing the Actor holding the Goal.
 - *Cardinality:* 0-1
 - *Role name:* “heldBy”
- **ActedOnThing:** representing what the Goal is about.
 - *Cardinality:* 1-1
 - *Role name:* “isAbout”

Properties (and Relationships)

- **State Law:** representing the Goal
 - *Cardinality:* 1-1
 - *Role name:* “theGoal”
- **AnyRegularProperty:** representing the attributes of the Goal.
 - *Cardinality:* 0-n
 - *Role name:* “hasAttribute”
- **State Law:** representing the dependencies the Goal is a Dependendum of
 - *Cardinality:* 0-n
 - *Role name:* “isDependumOf”
- **Transformation Law:** representing the Tasks that are means for the Goal (the end)
 - *Cardinality:* 0-n
 - *Role name:* “isEndIn”
- **Transformation Law:** representing the Tasks that are decomposed into the Goal
 - *Cardinality:* 0-n
 - *Role name:* “isSubElementIn”

Behavior

- Lifetime.

Modality (permission, recommendation, etc.)

- The holding Actor (if any) wishes the state law represented by the Goal to become true.

Open Issues

- **For the second iteration:** a Goal could be a Depender or a Dependee. Not yet dealt with at this stage.

Section II: Database Designs and Applications

Chapter IX

Externalisation and Adaptation of Multi-Agent System Behaviour

Liang Xiao, Queen's University Belfast, UK

Des Greer, Queen's University Belfast, UK

ABSTRACT

This chapter proposes the adaptive agent model (AAM) for agent-oriented system development. In AAM, requirements can be transformed into externalised business rules. These rules represent agent behaviours, and collaboration between agents using the rules can be modelled using extended UML diagrams. Specifically, a UML structural model and a behavioural model are employed. XML is used to further specify the rules. The XML-based rules are subsequently translated by the agents. The UML diagrams and XML specification can both be edited at any time, the newly specified behaviours being available to the agent system immediately. An illustrative example is used to show how AAM is deployed, demonstrating adaptation of inter-agent collaboration, intra-agent behaviours, and agent ontologies. With AAM, there is no need to recode and regenerate the agent system when change occurs. Rather, the system model is easily configured by users and agents will always get up-to-date rules to execute at run-time.

INTRODUCTION

Agent-oriented systems differ from object-oriented systems in that agents are active, while objects are passive. Thus, agents have the goal of having dynamic behaviours. Therefore, agent systems should be easily adaptable, being easily changed by engineers. Better still would be that they were adaptive, where systems change their behaviours according to their context (Lieberherr, 1995).

Although many tools and techniques are available for agent-oriented systems development, there is no unified and mature way to do it. What is more, existing agent platforms, like Java Agent Development (JADE) (Bellifemine, Caire, Poggi, & Rimassa, 2003), require designers and developers to code agent behaviours in fixed methods and the way to write them varies from one platform to another. This lack of uniformity of approach means that maintaining agent systems is potentially expensive. Being able to automatically generate agent systems and adapt their behaviours with changing requirements would alleviate this maintenance burden.

The objective of the chapter is to find a way to externalise agent behaviours in a repository. The configuration of the agent behaviours can be made at run-time by changing the repository, supported by tools. Therefore, new requirements can be continually reflected in the agent systems. We call this repository a requirements database and our approach adaptive agent model. The requirements database is in XML format, and the stored agent behaviours are represented as business rules.

BACKGROUND

In this section, we will first briefly introduce agent systems in general, and discuss how such systems are currently developed. We then present some existing approaches towards system adaptivity. After that, business rules, being able to capture system behaviours, are presented as a means to achieve more flexible agent behaviour code. Following the demonstration of possible implementation of rules as agent behaviours, we come back to the design aspect and describe the addition of the rule in two existing extended UML notation systems, their usefulness, and insufficiencies. Finally, our perspective and the main idea of our approach are given.

Agent Systems and Platforms

Software agents are defined as follows: “An agent is an encapsulated computer system that is situated in some environment, and that is capable of flexible, autonomous action in that environment in order to meet its design objectives” (Jennings, 2000, p. 280). Sending and receiving messages are the two main activities of agents. Various agent system development platforms are available, the JADE framework being one of them. JADE is aimed at developing multi-agent systems and applications conforming to Foundation for Intelligent Physical Agents (FIPA) (2005) standards. With JADE, an agent is able to carry out several concurrent tasks in response to different external events. To date, developers have, generally, been required to write repetitive and tedious code for the behaviour of every agent manually.

Existing Approaches

Current approaches to agent-oriented system design and implementation are fundamentally based on the identification of agent interaction protocols, message routing, and the precise specification of the ontology. This need for complete upfront design makes it difficult to manage agent conversations flexibly and to reuse agent behaviour (Griss, Fonseca, Cowan, & Kessler, 2002). Using agent patterns (Cossentino, Burrafato, Lombardo, & Sabatucci, 2002) is one way for better code encapsulation and reuse. In support of agent patterns, it is argued in Cossentino et al. (2002) that much research work such as Gaia (Wooldridge, Jennings, & Kinny, 2000), MaSE (DeLoach, Wood, & Sparkman, 2001), and Tropos (Castro, Kolp, & Mylopoulos, 2002) emphasise only the design of basic elements like goals, communications, roles, and so on, whereas the reuse of patterns, which are observed as recurring agent tasks appearing in similar agent communications, can reduce repetitive code. However, the chance that a pattern can be reused without change is low, and reuse of patterns in different context is not straightforward. In addition, this approach is not adaptive since system requirements change means that models need to be changed, patterns need to be re-written, and agent classes re-generated.

State machines have also been suggested for agent behaviour modelling (Arai & Stolzenburg, 2002) and the Extensible Agent Behaviour Specification Language (XABSL) has been specified (Lotzsch, Bach, Burkhard, & Jungel, 2004) to replace native programming language and to support behaviour modules design. Intermediate code can be generated from XABSL documents and an agent engine has been developed to execute this code. The language is good at specifying individual agent behaviours, but cannot express behaviours that involve inter-agent collaboration. Moreover, although agent behaviours are modelled in XABSL, they must be compiled before being executed by the agent engine. Thus, changing the XABSL document always requires recompilation.

Agent behaviours are modelled as workflow processes in (Laleci et al., 2004) and a behaviour type design tool is described for constructing behaviours. This approach provides a convenient way to compose agent behaviours visually. However, its use of Agent Behaviour Representation Language (ABRL) to describe agent interaction scenarios and “guard expressions” to control the behaviour execution order does not facilitate the modelling of systems as a whole. Further, the approach does not offer an agent system generation solution.

Business Rules and Agent Behaviours

A business rule is a compact statement about some aspect of a business. It is a constraint in the sense that a business rule lays down what must or must not be the case (Morgan, 2002). Often, business rules are hard-coded into programs, but keeping business rules distinct from code has many advantages, including the possibility that they can remain highly understandable and accessible to non-programmers. XML-based rules have been used in the IBM San Francisco Framework (Bohrer, 1998) as templates to specify the contents and structures for code that is to be generated. With this approach, changing XML rule templates allows mappings to new object structures. Figure 1 shows an example where a generic XML rule has been converted to a specific Java method, `getDiscount()` in this case.

Figure 1. Example of code generation using rules

```

<Rule>
  <Target> Attributes </Target>
  <Condition> scope = public </Condition>
  public &type; get&u.name;() {
    return iv&u.name;;
  }
</Rule>

If the name of one of the public attributes for an "Order" class was
"discount", and its type "Double", then this template would generate:

  public Double getDiscount() {
    return ivDiscount;
  }

```

Because agent behaviours represent actual system requirements and are subject to change, the application of business rules to the agent world should offer similar advantages as in the object world.

Agent-Oriented UML

Agent UML (AUML) by FIPA (FIPA, 2005) extends UML diagrams to cover needs for agent-oriented system design. In the context of agents and multi-agent systems, AUML class diagrams and interaction diagrams introduce new concepts, like agent, role, organization, message, protocol, and so on, with their corresponding notations. Interaction protocols (IPs) between agents are defined to describe various inter-agent activities in a pre-agreed message exchange style. Agents intending to participate in any IP must adhere to the AUML specification. Levelling is used for refinement of the interaction processes.

Agent-object-relationship (AOR) models (Wagner, 2003) show social interaction processes in organizational information systems in the form of interaction pattern diagrams. These model agents, ordinary objects, events, actions, claims, commitments, and reaction rules that dictate behaviours. AOR can be viewed as an extension of UML for agent systems and is capable of capturing the semantics of business domains.

Although AOR introduces an additional element of rule over the AUML notation system for modelling agent behaviours, the construction and editing of rules are not in its scope. Moreover, how agents, objects, and rules work together is not described adequately. However, it provides an appropriate notation system for the agent world, and we later adapt and use it for our conceptual modelling of agents, rules, and their interactions.

Our Approach

In response to the weaknesses of the existing modelling patterns and coding approaches, we propose that the agent interaction models, represented in the form of

UML, are related with the agent behaviour specification, represented as XML-based business rules. The combination of them is used to transform to the agent behavioural model for the agent systems at run-time.

The central component of the approach is rules. They capture customer requirements, participate as a behavioural element in the design models, are specified in XML, and are interpreted by agents as behavioural guidelines while the system is running. The transformation of rules turns requirements database into executable systems. While such systems are running, rules have business classes available to act upon. They govern agent behaviours, make decisions for agents in various contexts, have control over the invocation of business classes, and are adaptive. Each agent reacts to external events according to the XML-based rules at run-time. Rule definitions are easy to adapt, therefore, different business classes can be invoked by agents to achieve dynamic effect.

SOLUTION APPROACH: ADAPTIVE AGENT MODEL

We propose an approach, the adaptive agent model (AAM). In this, we emphasise the integration of UML diagrams, which model inter-agent relationship, and XML-based rule definitions, each of which describes an individual agent behaviour. UML model information will become part of the XML definitions and enable agents to understand their communication with the outside world. The transformation of a piece of requirement to rule descriptions, then a rule element in UML, after that XML specification, and finally, an interpreted agent behaviour is systematically demonstrated using our case study.

Case Study

To illustrate our approach and to use in our discussion later, we introduce an e-commerce case study. Suppose a retailer runs an online shop. The retailer has an association with customers and also with various supplier companies, who may or may not serve the retailer, depending on different policies that different companies would make in different sale seasons. If the requested order is profitable to the supplier company, it proposes a deal, including the price and delivery time, and so forth, for the order. The retailer accepts the proposal if it is satisfied with the deal. Overall, the relationship between customers, the retailer, and supplier companies can change at any time. The business vocabulary is also changeable and the decision-making process for each company, retailer, and customer is unpredictable.

Requirements Analysis

Functional requirements can be identified according to the described case study. They are organised according to the actors that use them. Obviously one actor may have multiple requirements related to other actors. These are uniformly documented in tables. Each table contains the information about a function owned by an actor, describing the function name, informative description of the function, the cause of the function, information used by the function, its outputs, required effects, and, finally, an identifier. Table 1 is such an example.

Table 1. Sample requirement table

Name	saleProcessing
Description	To handle order request from the <i>retailer</i> .
Cause	Receipt of a call for sale proposal from a <i>retailer</i> . The received business information is provided in the form of a combination of <i>retailer information</i> , <i>order identity</i> , and <i>ordered goods details</i> .
Information Used	Retailer information and order information.
Outputs	A sale proposal, to the <i>retailer</i> .
Required Effect	If the order is evaluated as <i>attractive</i> , then a new sale proposal is created from the request and sent to the retailer; otherwise, the request is rejected or renegotiated.
Identifier	Company. saleProcessing

We will use this specific requirement about the behaviour of the company in processing the request from the retailer for customer order throughout the remainder of the chapter.

Rule Model

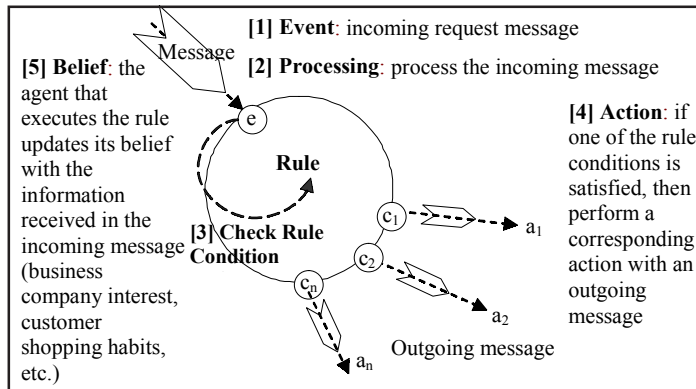
In the object-oriented (OO) world, for each functional requirement table, a method in a class could be written. However, because the relationship between the communicating parties may change, a fixed method would not be suitable to the described scenario. Moreover, each supplying company may change its sale policy, which means the way of evaluating order request and creating sale proposal varies with company policies and sale seasons. Thus, it is desirable to choose a configurable behavioural element for the executing component.

We use a business rule to represent a configurable behaviour for a runnable agent. One rule makes use of stable business classes and tells an agent how to collaborate with other agents by receiving/sending messages.

The following diagram is a generic rule model. In such a model, events cause agents to execute rules and if certain conditions are satisfied, some actions are triggered which in turn include generated events for other agents. An agent processes a rule using the following steps, in accordance with what is shown in Figure 2. A rule definition is made up of the steps that an agent takes to execute the rule.

1. **Check event:** Find out if the rule is applicable to deal with the perceived event.
2. **Do processing:** Decode the incoming message, including the construction of business objects to be used in later phases.
3. **Check condition:** Find out if the {condition c_i } is satisfied.
4. **Take an action:** If c_i is satisfied, then do the corresponding {action a_i } that is related to {condition i } as defined by the rule. Then, send a result message to another agent (possibly the triggering one). If c_i is not satisfied, and this is not the last condition, then go back to Step 3 and check the condition c_{i+1} .
5. **Update beliefs.** Using the information obtained from the message just received, the knowledge of the agent of the outside world is updated.

Figure 2. The rule model



The actors that can be identified in the case study reveal the participating agents, which we call “CompanyAgent,” “RetailerAgent,” and “CustomerAgent.” Two types of classes can be identified, which we name “Order” and “Proposal.” The requirement table (Table 1) states a required behaviour of the “CompanyAgent,” which makes use of the two classes. The transformation of a requirement table to a rule is straightforward.

1. The “Cause” section is used to make the rule “event”;
2. The sections of “Information Used” and “Required Effect” are used to make the rule “processing”; and
3. The sections of “Required Effect” and “Outputs” are used to make the rule “condition” and “action.”

A rule does not necessarily have multiple {condition, action} couplets. The requirement of “saleProcessing” in Table 1 turns to a rule with the following specification, concentrating only on the case that the deal will be done.

In the requirements transformation process, concepts like “Order,” “Proposal,” and “attractiveness” are expressed explicitly. However, it is the designers that designate classes and methods for these, later on.

By means of the rule model, the requirement in table 1 is modelled as a dedicated rule that will be used by an agent for a specific task and uses business classes to realise that purpose. This is in contrast with the traditional model that a class method or function call has a fixed body and input/output, designed for a particular type of object. In our model, what and how classes are to be invoked can be specified in rules and these are configurable. The mutable requirements on component collaboration can be externalised in rules and reflected in agent knowledge in terms of their collaboration partners, events processing, and the response messages. Different actions can be set in rules as reactions to different conditions, in an order of user preferences/priorities. Business rules, as we specify here, make agents another abstraction over classes and, therefore, superior to classes.

Rationality of Using Rule Model for Adaptive Agent System

Agent systems always require interactions among many agents, modelled as message passing, such that the message sender requests a service from the message receiver. The message receiver uses its internal business objects for the computation required to fulfil the request and then, possibly, takes a further action. Different situations will arise and these are modelled as rules that agents should obey. Thus, a rule is responsible for the behaviour of an agent in dealing with a particular situation. Multiple rules can be defined to let the agent collaborate with other agents to achieve different goals.

Such a model, by using the communication natural of agent system and defining rules for the communication, can help to achieve adaptivity. A rule specifies an agent interface and describes the functionality the agent provides. An agent interface is a contract that is made between an incoming event and an outgoing action, both involving an external agent.

An interface specifically dedicated for the description of system interactions can bring adaptivity. message-oriented middleware (MOM) (Mahmoud, 2004) is a middleware infrastructure that offers distributed messaging communication similar to a postal service. MOM has an architectural style well suited to support applications that must react to changes in the environment. It provides an independent layer as an intermediary for the exchange of messages between senders and receivers. This allows source and target systems to link without having to adapt them to each other (Mahmoud, 2004).

Having a more loosely coupled architecture, the AAM not only provides the independence of the interface layer between all participants, but also the functionalities of each of them is collectively centralised in a rule base. Thus, each agent in our system is adaptive externally and internally.

Design Models

Once rules are collectively transformed from the requirements tables, they must be related to the agents that will use them and the classes that will be used by them in design diagrams. For example, Figure 3 indicates that the rule “saleProcessing” will be executed by “CompanyAgent” when the agent receives a call for sale proposal message from the “RetailerAgent.” An “Order” class and a “Proposal” class may be invoked by the rule to assist this operation. Traditional UML models need to be extended to accommodate not only the concept of class, but also agent and rule, and more importantly, their relationship. Two main models have been designed for the design of systems using our AAM approach.

Structural Model: Agent Diagram

Structural Models are built through *Agent Diagrams*, and show agents, business rules, business classes, and their relationship. Agents manage rules and rules manage the invocation of business classes. Such models are used for agent identification, agent relationship identification, and eventually building an agent/rule/class hierarchy. They are later the basis for the behavioural models.

Figure 3. Transformed requirements as a rule, called “saleProcessing” for the case study

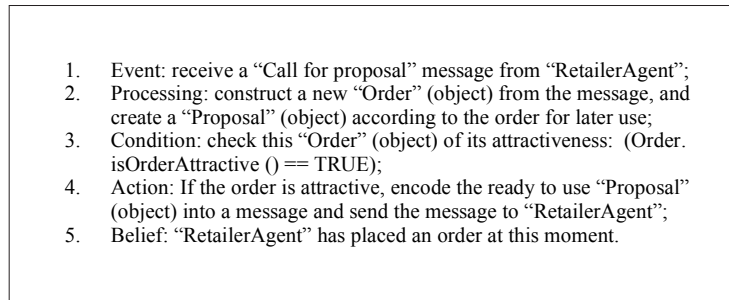
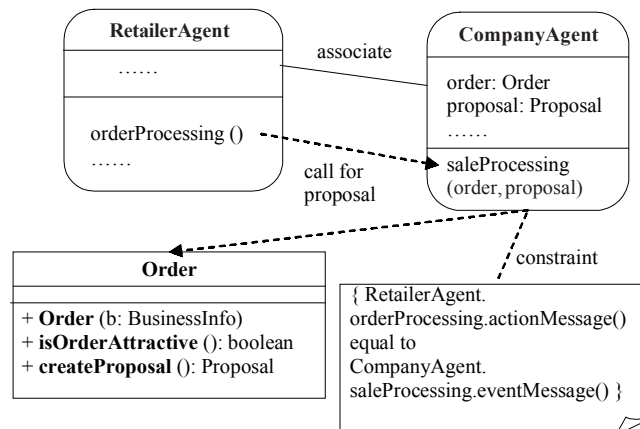


Figure 4. The agent diagram for the case study



Agents are identified to represent distinct conceptual domains. Agent diagram has the class diagram, the backbone of UML (Fowler, 2004), as its counterpart in the object-oriented models. In our AAM approach, agents are regarded as superior to classes. Each rounded- cornered box represents an agent and is divided into three compartments. The top compartment holds the name of the agent, the middle compartment holds the classes managed by the agent along with their instantiation, and the bottom compartment holds the rules that govern the functions of the agent. This construct resembles a class name, an attribute list, and an operation list constituting a class diagram in the OO world.

In Figure 4, two identified agents, “RetailerAgent” and “CompanyAgent,” for our case study are shown. “RetailerAgent” has a rule “orderProcessing” that will construct an object with type “BusinessInfo,” package it into a “Call for proposal” message, and

send the resulting message to “CompanyAgent.” To respond to such requests, “CompanyAgent” will offer a deal, if the order is attractive, using the rule “saleProcessing.” Thus, we have an association relationship between the two agents involved and a constraint for them. They resemble an association between two classes and a constraint for classes in the OO world. During the processing of rule “saleProcessing,” an “Order” object will be constructed from the received “BusinessInfo” structure and the constructed object should pass an “isOrderAttractive” check before “CompanyAgent” proceeds to offer a deal, “Proposal,” for the order. Thus, such a business class of “Order” is related with “CompanyAgent” via “saleProcessing,” and it has at least three methods that will be invoked by the agent rule.

The diagram of Figure 4 structurally documents the system model, highlighting “saleProcessing” rule, which makes use of “Order” and “Proposal” classes and will be used by “CompanyAgent.” This rule-centred diagram is constructed from the requirements rule in Figure 3 using the following steps:

1. In the descriptions of “Event” and “Action,” find out the message-passing pattern between participating agents, and identify the agent to which the rule belongs. Draw the agent boxes and passing messages in the diagram.
2. Analyse “Processing” and extract all the business classes that are used by the rule. Relate the classes to the agent/rule and update the middle and bottom compartments of the agent box. Draw the class boxes and connect them with the rule in the bottom compartment of the agent box.
3. Consider the possible methods that the recognised classes may have by examining “Processing” and “Condition.” From these, respectively, at least a constructor method and a method with return-type Boolean should be identified. Add class methods to the class boxes in the diagram.

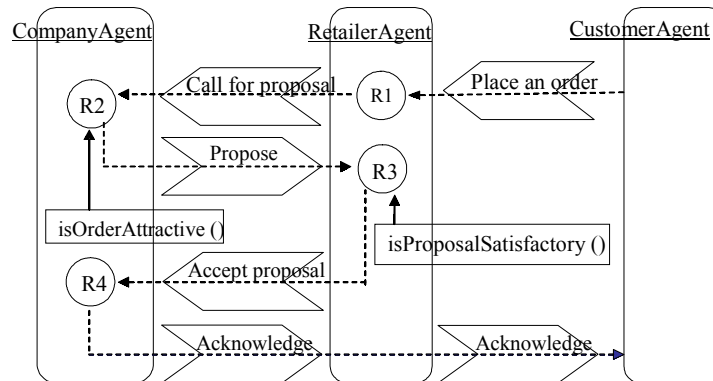
Behavioural Model: Agent Communication Diagram

Agent Diagrams capture the static relationship between different entities and depict the whole system. Agent communication diagrams are used to model the interaction of agents. Such behavioural models organise agents, rules, and messages around business processes. For every business process, all participating agents will appear in the diagram, with message passing between them to accomplish certain business goals.

Software architecture refers to the communication structures for system entities. In traditional object-oriented systems, objects are aware of which other objects they will pass messages to, but are unaware of which objects will pass messages to them. Full architecture independence requires that the detail of where objects will send messages should also be hidden (Hogg, 2003). In agent-oriented systems, business processes are implemented by the collaboration of agents. The management of this collaboration requires the agent architecture to be well modelled. In order to generate agent systems and be able to adapt them afterwards without re-generation and re-compilation, full architecture independence (two-way encapsulation) is required, and the interaction information should not be hard-coded so that agents can adapt their collaboration in communication according to changing requirements.

In our approach, an extended UML diagram, as shown in Figure 5, is used to model agent collaboration, describing how message passing among coordinated agents can

Figure 5. An agent communication diagram describing a business process



accomplish business tasks. These diagrams provide a blueprint for involved business rules, the composition elements of our diagrams. Each rule governs an individual agent behaviour in the participating collaboration. Rules are connected to form a flow of decision making, process-by-process, one decision being made at each connection point. As such, the model visualises the actual system function in a sequence of agent actions dictated by rules. User specified agent collaboration in the diagrams is used to generate the inter-agent part of the rules definition in XML format. It is through these rules that agent systems are adapted, both in collaboration and internally without re-code or re-generation, since we let agents get appropriate rules to execute only at run-time, and rules get configured continuously through supporting tools we provide.

The diagram used for the design of multi-agent behaviours is the agent communication diagram. It has been developed based on the agent diagram and used for the generation of agent systems. Figure 5 describes the process for the case study, where a customer orders products from a business company through a retailer. Business classes are not shown on the diagram, but the invocations of their methods are, such as the one for condition check. R2 has been shown previously as “saleProcessing” in the bottom compartment of “CompanyAgent” in Figure 4.

A similar transformation can construct the behavioural model from requirements rules, just as has been done to make the structural model. Alternatively, the model can be built based on the previous one.

1. Draw all the agents and, within them, the appropriate rules.
2. Draw all the passing messages between associated agents in their rules. Wherever a message goes from one rule to another, the “Action” of the former rule and the “Event” of the later one describes the same message and they match in the recipient and sender.
3. Draw only the {condition, action} couplets that eventually contribute to the successful result.

A structural diagram concentrates on one rule, while a behavioural diagram ignores structural and low-level details and puts agent actions logically in a sequence for a complete business process. Only the main route, which directs to the successful result, is shown in the diagram. The route has many divisions, each of which has a message passing between two rules in two agents. Figure 5 describes our case study with all participating agents and their rules collectively connected. This conforms and extends the rule model for multiple agent/rule collaboration. The model has a different view from the previous model of the same system.

Rule Implementation

The traditional software system development process can be viewed as a series of transformations through the form of requirements documents, design models, and implemented code. The performance of the final product precisely reflects the desired behaviour in the required system. The initially captured knowledge, usually documented in UML diagrams, is essential to the system implementation. However, these models rapidly lose their value as, in practice, changes are often done at the code level only.

In our approach, rules serve as a requirements database. Then, after being transformed into a UML element, they represent agent behaviours. After that, they are specified more accurately in XML. Finally, they are interpreted by the running agent software.

UML-style diagrams are good at showing collaboration among agents, while XML specification is good at precise definition of agent behaviours, an aspect that UML diagrams lack (Fowler, 2004). The use of rules allows designers to use the combination of UML and XML, one complementing another, where the former models the system blueprint and the latter models the behavioural details. Because the UML and XML models are combined for interpretation as agent behaviours at run-time, the design and implementation of the system are seamlessly integrated. Changes are done and can only be done at the model level. This is less error prone and safer than direct change of code.

According to the rule model, we encode the diagrammatic rule in the models in XML, with the structure {event, processing, condition, action, priority}. The computer-readable Java-style code specifies on receipt of an event, how an agent should act if the condition of the rule is satisfied. Rules are considered for execution by agents according to priorities set by users. The XML representation for rule R2 is given in Figure 6.

The construction of rule in XML from the models takes the following steps:

1. Each rule definition has a root element of <business-rule>.
2. An element of <name> and <owner-agent> defines the name of the rule and the agent that owns it respectively. These have been given in the structural model already.
3. An element of <global-variable> defines all the business classes with their instantiation that will be used by the rule. These classes can be found in the structural model, where they are related to the rule.
4. An element of <event> and another element of <action> define the message flow, and the structure of the messages. An element of <from> and another element of <to> in the <message> structure defines where the incoming message comes from and where the outgoing message goes to. These can be found in the behavioural model, where each rule is connected with two messages, one in and one out. An

Figure 6. The XML definition for rule “saleProcessing” owned by “CompanyAgent”

```

- <business-rule>
  <name>saleProcessing</name>
  <business-process>retailer business</business-process>
  <owner-agent>CompanyAgent</owner-agent>
- <global-variable>
  - <var>
    <name>order</name>
    <type>Order</type>
  </var>
  - <var>
    <name>proposal</name>
    <type>Proposal</type>
  </var>
</global-variable>
- <event>
  <type>receipt of message</type>
  - <message>
    <from>RetailerAgent.orderProcessing</from>
    <to>CompanyAgent.saleProcessing</to>
    <title>Call for proposal</title>
    - <content>
      - <businessInfo>
        - <retailer> ... </retailer>
        - <order>
          <id>10010001</id>
          - <product>
            <classification>book</classification>
            ...
          </product>
        </order>
      </businessInfo>
    </content>
  </message>
</event>
<processing>
  order = new Order (businessInfo)
  proposal = order.createProposal ()
</processing>
<condition>
  order.isOrderAttractive() == true
</condition>
- <action>
  <type>send a message</type>
  - <message>
    <from>CompanyAgent.saleProcessing</from>
    <to>RetailerAgent.proposalProcessing</to>
    <title>Propose</title>
    - <content>
      - <proposal>
        <id>10011101</id>
        <businessInfo> ... </businessInfo>
      </proposal>
    </content>
  </message>
</action>
<priority>5</priority>
</business-rule>

```


- element of `<content>` in the `<message>` structure defines the encoded objects in each message. They are not given in the models but can be found in the requirements rules. The objects defined in the `<global-variable>` are usually involved;
5. An element of `<processing>` and another element of `<condition>` define the construction of business objects from the event message and invocation of methods on them. All the methods can be found in the structural model, where classes are related to the rule. The evaluation method for the `<condition>` can be found in the behavioural model.

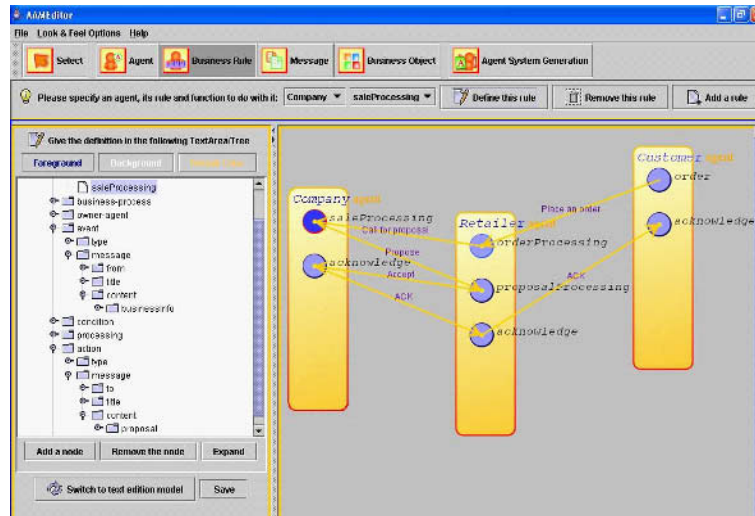
In the diagram of Figure 5, “CompanyAgent” reacts to the “Call for proposal” message from “RetailerAgent” by executing the above specifically defined rule “saleProcessing” in XML, shown in Figure 6. In general, each agent executes a rule in the following way:

1. Get a list of its managed rules from a rules document according to the `<owner-agent>` section.
2. Filter these rules and retain those that are applicable to the current business process according to the `<business-process>` section.
3. Get the rule that currently has the highest priority according to the `<priority>` section.
4. Check the applicability of this selected rule; that is, check if the `<event>` section matches the event that has occurred. In other words, check if the agent that triggers the received message is the same as that given in the `<from>` section of the `<message>` in `<event>`, and the received message format is also as specified in the `<message>`. If that is not the case, go to Step 9.
5. Decode the message received and build business objects from it following the `<processing>` instructions. Constructor methods of existing classes will be involved. Global variables declared in the `<global-variable>` section will be used to save the results.
6. Check if the current condition specified in the rule is satisfied according to the `<condition>` section. Constructed business objects will be involved, and their methods will be invoked upon to assist the rule to function. If the condition is not satisfied and it is not the last condition, move to the next condition and repeat Step 6; otherwise go to Step 9.
7. Execute the corresponding `<action>` section. This involves encoding constructed business objects that refer to `<global-variable>` into a message. Send the message to the agent that is specified in the `<to>` section of the `<message>` in `<action>`.
8. Analyse the business objects that have been decoded from the message received and update the agent’s beliefs with the new information available.
9. Remove this selected rule from the rules set obtained in Step 2 and if not the last rule, go to Step 3.
10. Wait for the next event.

Supporting Tool and Agent System Implementation

A CASE tool has been developed to enable the specification of the agent collaboration, rule definitions and message flows. Figure 7 captures a window from this tool showing the construction of an Agent Communication Diagram in its main panel. Rules

Figure 7. AAM supporting tool



can be defined either in XML text or using a more user-friendly tree structure as shown in the left panel. The tree is structured using the same schema that constructs the document in Figure 6. Business classes can be registered using the tool and after that selected for the specification of messages passing between agents. For example, Figure 7 shows that “businessInfo” has been chosen as the content of the event message and “proposal” as the content of the action message for the “saleProcessing” rule, conforming to the specification in Figure 6. The section of `<from>` and `<to>` for event and action messages can be generated when the direction of the messages are set up visually in the main panel of the tool. Existing class methods can be selected for `<processing>` and `<condition>`, and a number for `<priority>`. XML code is eventually generated from the completed tree structure and saved in a rules document.

Our supporting tool uses a business rules document as the database. Once business processes are specified graphically in the tool, agent interaction models, rule reaction patterns and message flows are established accordingly. The agent system framework is automatically generated such that each rule maps to an agent behaviour. Program code is not generated at this moment. Instead, XML-based rules are plugged in and are subsequently translated by agents at run-time. Figure 8 shows the pseudo code that “CompanyAgent” will interpret from the “saleProcessing” rule to execute as one of its behaviours.

The system runs on the JADE platform and can be in a distributed network. All agents access the central XML-based rules document via a parsing package. By using this package, agents can do the comparison to check the applicability of rules (① in Figure 8) and run pre-defined statements embedded in the XML tags of the rules (②, ③, ④ in Figure 8). These are interpreted from the XML specification in Figure 6. While the system is running, the rule specification can be continually changed through the tool. This

Figure 8. Pseudo code for behaviour of “CompanyAgent”, mapping to its “saleProcessing” rule

```

thisAgent. addBehaviour (Rule thisRule) {
    thisBehaviour. setPriority (thisRule. getPriority ());
    Order order;
    Proposal proposal;
    Message m = thisAgent. receiveMessage ();
    while (m != null)
    {
        Agent fromAgent = m. getSenderAgent ();
        if (fromAgent. equals
            (thisRule. getEvent (). getMessage (). getFromAgent ())) // ①
        {
            /* the rule is applicable to the received message */
            BusinessInfo businessInfo = (BusinessInfo) m. getContentObject ();
            order = new Order (businessInfo); // ②
            if (order. isOrderAttractive ()) // ③
            {
                /* the condition of the rule is satisfied */
                proposal = order. createProposal (); // ④
                Message m2 = new Message ();
                m2. setContentObject (proposal);
                Agent toAgent =
                    thisRule. getAction (). getMessage (). getToAgent ();
                m2. addReceiverAgent (toAgent);
                thisAgent. send (m2);
                /* update this agent's beliefs */
                thisAgent. addBelief (System. getcurrentTime (), fromAgent, m);
            }
        }
        m = thisAgent. receiveMessage ();
    }
}

```

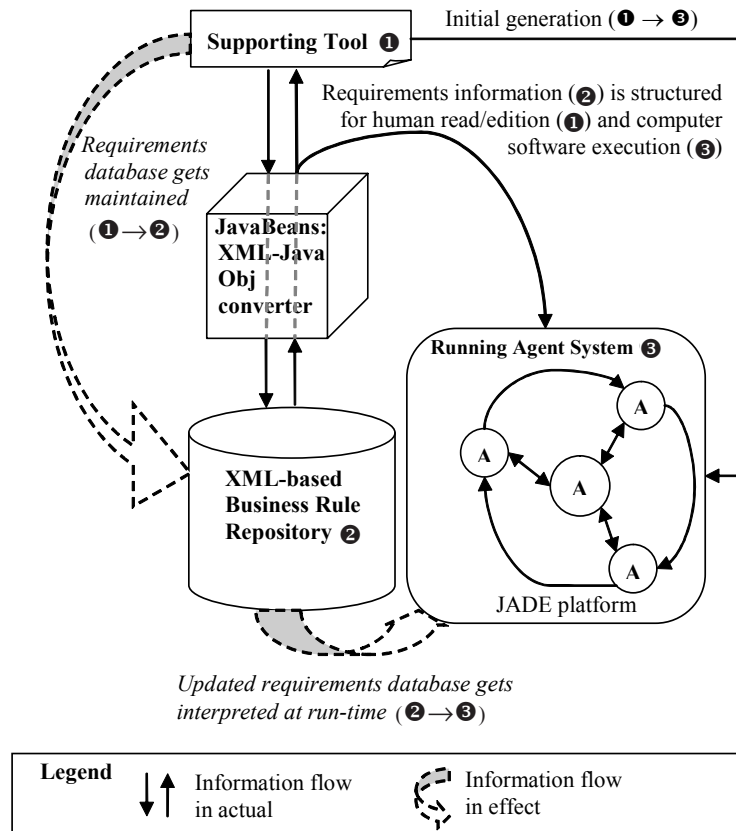
allows dynamic adjustment of agent communication structure and therefore the software architecture of the system.

A shared module in the XML parsing package called “Rule”, being able to access the XML definition of rules and assemble corresponding objects, is used by all agents. The methods `getPriority()`, `getEvent()`, and `getAction()` are provided by “Rule”.

Deployment

The deployment of an implemented system using the proposed approach is shown in Figure 9. The actual agent system (③ in Figure 9), running on the JADE platform in a distributed network, is initially generated from the supporting tool (① in Figure 9). A central XML-based business rule repository (② in Figure 9) is deployed in the network, containing the rule definitions and the registered business classes that are used by the rules. The XML parsing package is implemented as a JavaBeans component, responsible for parsing the XML format of business rules and presenting the parsed business knowledge in the tool. The tool is continuously used by business people to maintain requirements (① → ②). The edition through the tool for the requirements change is saved in the XML repository using the same JavaBeans.

Figure 9. Deployment of the system



All agents access the repository via the JavaBeans as well, in order to obtain the most up-to-date knowledge in an easy to operate format. In the beginning, each agent has the knowledge of whom and how they will collaborate with, dictated by the initial rules. While the system is running, the business requirements model can be continuously under maintenance through the tool. With the assistance of the JavaBeans, each agent in the generated agent system interprets the updated requirements knowledge for action/reaction (② → ③). Eventually agents can always get the desired behaviours as soon as they have been specified through the tool, and can be continuously updated.

Adaptation

Modelled as business rules, the requirements database in our system can be adapted in three aspects for three purposes, from the perspective of the running agents. They are respectively: the collaboration between agents; the internal behaviours of each individual agent; and the classes that agents can make use of.

Adapting Inter-Agent Collaboration

Being able to adapt the collaboration between agents at run-time, AAM achieves two-way encapsulation. Agent behaviours are guided by rules so that they do not need to know who they will contact in advance. To reflect business process change, the Behavioural Models can easily be changed visually with the tool. These changes are automatically reflected in the XML definitions of corresponding agent rules, for example, in their `<event>/<message>/<from>` and `<action>/<message>/<to>` sections. This enables agents in the running system to have their partners changed in order to accomplish the updated business processes. On receipt of any message, an agent reads the most recent rules, analyses them and finds out the appropriate agents to send messages to. In the case study, we may wish to re-configure the rule “saleProcessing”, and let the “CompanyAgent” take a new action in a condition originally not predicted.

Suppose we wish to introduce a new occasion where if the current “CompanyAgent” does not evaluate the received order request to be “attractive” or can not fulfil the order request, it forwards the order to another “CompanyAgent”. This new requirement can be specified, implemented and deployed by agents automatically by configuring the Agent Communication Diagrams using the tool. The achievement of this dynamic collaboration is through painless model adjustment rather than expensive code change. Further, we achieve the model-driven communication architecture.

Adapting Intra-Agent Behaviours

The behaviours of agents in processing the event, checking the condition, and taking the action are externalised in business rules. This means that they can be configured dynamically. In fact, by changing the `<event>`, `<processing>`, `<condition>`, and `<action>` fields in appropriate rules, alternative methods of the managed business objects can be selected for invocation. In the case study, we can re-configure the rule “saleProcessing” to invoke a new evaluation method of the “Order” class or even a method of a new “Order” class to check the attractiveness of the order. In addition, we can configure two couplets of `<condition>` and `<action>`, so that for ordinary customers and company customers, different means to generate sale proposals can be used. All this can be carried out at run-time.

Adapting Ontologies

Only business concepts registered through the tool and saved in the rules document may appear in agent messages. When a new business concept is required, it can be registered with its properties, and a new business class with attributes will be generated by the tool. New vocabularies thus can become available for the specification of agent rules through the tree structure on the left panel of the tool (Figure 7). Also at run-time new classes with new methods thus can become available for invocation by the running agent system. Eventually, all agents will be able to understand the new vocabularies the other agents in the system are using even those registered after the system has been running for a while. Hence, ontologies are always updatable. For the case study, suppose that an additional attribute of the “BusinessInfo” business class is required and added while the system is running, the updated class becomes available to all agents and they start to use the new concept immediately.

EVALUATION: THE MODIFIABLE ARCHITECTURE OF AAM

AAM achieves the quality of modifiability (Bass, Clements, & Kazman, 2003) in its architecture in terms of its prevention of ripple effects and deferment of binding time.

Prevention of Ripple Effects

A ripple effect from a modification is the necessity of making changes to modules not directly affected by it (Bass et al., 2003). The introduced Agent/Rule/Class hierarchy, having a higher level abstraction of agents over classes and a rule interface between agents helps prevent ripple effects, so reducing the time and cost to implement changes.

Semantically, agents are considered more meaningful communication entities than objects. They are actors that have the rule interface publicly and use a combination of multiple concrete objects privately. This conforms to the idea of information hiding, where changes are isolated within one module, usually a private one, and changes propagating to others, usually public ones are prevented. Rules specified for agents serve as the descriptions of agent responsibilities. They separate the interactions between agents and the use of objects by agents. The change of one agent in its use of objects is kept private and has no influence on the agent that uses the result, as long as the interaction pattern of the two agents is unchanged. For example, no matter what has been changed in the <processing> or <condition> section of the “saleProcessing” rule in Figure 6, the “RetailerAgent” would not be affected in its action, although the “CompanyAgent” starts to use a different means to generate proposal or evaluate order attractiveness. Nevertheless, the “RetailerAgent” expects a proposal as a result from the “CompanyAgent”, as it usually does. In rules, the <processing> and <condition> sections are private to agents and <event> and <action> sections are the public interface.

The supporting tool for AAM always validates and ensures that the action message of A is syntactically equal to the event message of B, if A sends a message to B. This prevents the changes required by syntactic dependencies from propagating: for B to compile/execute correctly, the type of the data that is produced by A and consumed by B must be consistent with the type of the data assumed by B (Bass et al., 2003). For example, if the structure of the “proposal” in the action message of the “CompanyAgent” is required to change, the “RetailerAgent” would expect the new structure automatically when the change has been made (the *Adapting Ontologies* section described the details of this).

Deferment of Binding Time

The mechanism of AAM lets agents interpret changeable rules at run-time and so the binding of the actual software agents and their function specification is deferred until then. This helps to control the time and cost to test and deploy changes.

When a modification is made by the developer, there is usually a testing and distribution process that determines the time lag between the making of the change and the availability of that change to the end user. Binding at run-time means that the system has been prepared for that binding and all of the testing and distribution steps have been completed. Deferring binding time also supports allowing the end user or system administrator to make settings or provide input that affects behaviour (Bass et al., 2003).

In AAM, rules are constructed with the supporting tool and ensured of their validity. No matter how they are changed, they are free of testing for their syntax. Changing rules does not cause any necessary change to the deployment of agents. In addition, the tool is simple enough for use by non-developers to make changes that will be reflected at run-time. The achieved benefit from the deferred binding time is at the cost of additional interpretation time while the system is running.

FUTURE WORK AND CONCLUSION

Agent behaviours reflect functional requirements. These behaviours are modelled and externalised as rules in the adaptive agent model. The rules are, in effect, executable requirements. In the design models they are present in extended UML diagrams. In the implementation models they are centrally managed and easily changed through their XML-based definitions. Because rules are easy to edit, and agents always get the most recent rules for interpretation, deploying new requirements requires minimal effort.

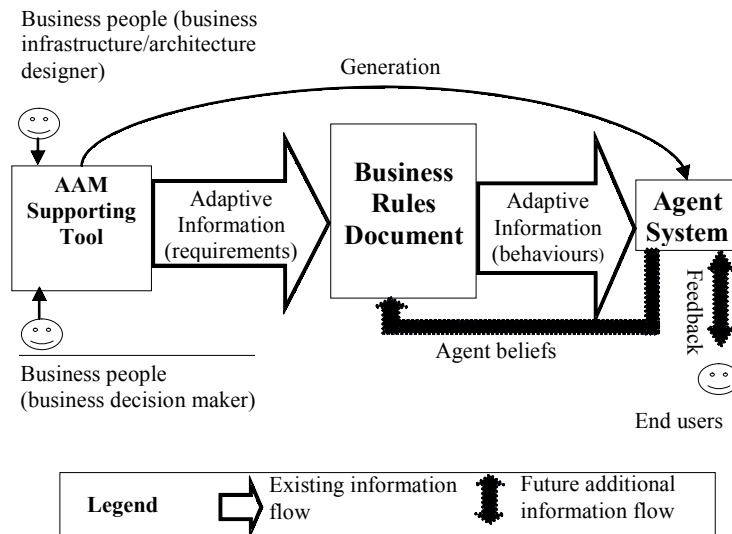
The XML specification of the rules, related with the corresponding UML elements, makes our models which combine UML with XML reusable. The models are continuously reused, not only for the regular revision by users, but also for constant interpretation by software agents. The maintenance of the AAM models is, in fact, equivalent to the maintenance of the final software system.

One weakness of AAM is that the framework's externalisation of agent behaviours in XML-based rules will degrade the performance of such systems. Every time an agent acts and reacts to events, it will read the rules document, test rules' applicability, find the one with the highest priority, and execute it. Therefore, there is a trade-off between ease of adaptation and performance. Resolution of this issue remains an aspect of future work.

Ultimately, we expect to achieve self-adaptivity in the AAM where, as agents interact with end users they perceive their behaviours and preferences. As shown in Figure 10, this allows agents to update their beliefs, and so deduce rules that can be added to the central rules document. These inferred rules can be shared and executed by all agents and are subject to amendment. After some time, a mature and reliable rule set, independent of those acquired through the tool can be established.

Further, we plan to develop the reflective and adaptive agent model (RAAM), the logical follower of AAM. Usually, a reflective system will have a number of interceptors and system monitors that can be used to examine the state of a system, reporting system information such as its performance, workload, or current resource usage (Mahmoud, 2004). RAAM will build on AAM and provide an improved service for its user needs by supporting advanced adaptation features. It allows for the automated self-examination of system capabilities and adjusts and optimises those capabilities automatically. The proposed new feature of auto-adaptivity in RAAM is a natural add-on to the current approach. This characteristic is in contrast with the adaptivity already achieved in AAM, where the system adjusts itself automatically whenever there is a non-functional need, rather than changes according to functional requirements only. Using dedicated agents to examine and react to the undesirable results from the execution of ordinary rules by ordinary agents is a straightforward means to realise auto-adaptivity and hence build quality into the system. These agents would specifically explore inappropriate decisions

Figure 10. Future adaptive agent model



made by human beings, negative impact to the overall performance caused by carrying out certain rules, insecure operations caused by certain agents, and respond by suggesting amendment and enhancement. For example, new agents may be created and assigned tasks when degrading system performance is detected or original agents fail. Higher level rules might be specified for these dedicated agents for their examination and reaction.

AAM would be useful for those domains that have frequently changing requirements where re-development would otherwise be costly. Particularly, AAM should work well when there is collaboration between many different entities and where this collaboration may be subject to adjustment, as a result of changing business processes. AAM is also suitable where the business environment is frequently changing with emerging concepts and behaviours.

Other future work will include the development of richer business rules. The adaptive agent model will be made more powerful and more flexible, but work so far indicates that it is highly relevant and useful to the development and evolution needs of multi-agent systems.

REFERENCES

- Arai, T., & Stolzenburg, F. (2002, July 15-19). Multiagent systems specification by UML statecharts aiming at intelligent manufacturing. *Proceedings of the First International Conference on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy (pp. 11-18). New York: ACM Press.

- Bass, L., Clements, P., & Kazman, R. (2003). *Software architecture in practice* (2nd ed.). Boston: Addison-Wesley.
- Bellifemine, F., Caire, G., Poggi, A., & Rimassa, G. (2003, September). JADE — A white paper [Electronic version]. Retrieved July 26, 2005, from <http://jade.tilab.com/papers/WhitePaperJADEEXP.pdf>
- Bohrer, K. A. (1998). Architecture of the San Francisco Frameworks. *IBM Systems Journal*, 37(2), 156-169.
- Castro, J., Kolp, M., & Mylopoulos, J. (2002). Towards requirements-driven information systems engineering: The Tropos Project. *Information Systems*, 27(6), 365-389. Amsterdam, The Netherlands: Elsevier.
- Cossentino, M., Burrafato, P., Lombardo, S., & Sabatucci, L. (2002). Introducing pattern reuse in the design of multi-agent systems. In R. Kowalczyk, J. Muller, H. Tianfield, and R. Unland (Eds.), *Agent Technologies, Infrastructures, Tools, and Applications for E-Services (AITA'02 Workshop at NODE02)* (LNAI 2592, pp. 107-120). Berlin: Springer-Verlag.
- DeLoach, S. A., Wood, M. F., & Sparkman, C. H. (2001). Multiagent systems engineering. *International Journal on Software Engineering and Knowledge Engineering*, 11(3), 231-258.
- Foundation for Intelligent Physical Agents (FIPA). (2005). *FIPA specifications*. Retrieved July 26, 2005, from <http://www.fipa.org/specifications/>
- Fowler, M. (2004). *UML distilled* (3rd ed.). Boston: Addison-Wesley.
- Griss, M., Fonseca, S., Cowan, D., & Kessler, R. (2002). *Smartagent: Extending the JADE agent behavior model* (Tech. Rep. No. HPL-2002-18). School of University of Utah.
- Hogg, J. (2003, October). *Applying UML 2 to model-driven architecture* [Electronic version]. Retrieved July 26, 2005, from http://www.omg.org/news/meetings/workshops/MDA_2003-2_Manual/5-1_Hogg.pdf
- Jennings, N. R. (2000). On agent-based software engineering. *Artificial Intelligence*, 117(2), 277-296.
- Laleci, G. B., Kabak, Y., Dogac, A., Cingil, I., Kirbas, S., Yildiz, A., et al. (2004). A platform for agent behavior design and multi agent orchestration. In *Agent-Oriented Software Engineering V: 5th International Workshop (AOSE 2004)* (LNCS 3382, pp. 205-220). Springer.
- Lieberherr, K. (1995, October 15-19). Workshop on adaptable and adaptive software. In *Proceedings of the Tenth Conference on Object Oriented Programming Systems Languages and Applications*, Austin, TX (pp. 149-154). New York: ACM Press.
- Lotzsch, M., Bach, J., Burkhard, H.-D. & Jungel, M. (2004). Designing agent behavior with the extensible agent behavior specification language XABSL. In *RoboCup 2003: Robot Soccer World Cup VII*, (LNAI 3020, pp. 114-124). Springer.
- Mahmoud, Q. H. (Eds.). (2004). *Middleware for communications*. Chichester, UK: John Wiley & Sons.
- Morgan, T. (2002). *Business rules and information systems*. Boston: Addison-Wesley.
- Wagner, G. (2003). The agent-object-relationship metamodel: Towards a unified view of state and behavior. *Information Systems*, 28(5), 475-504.
- Wooldridge, M., Jennings, N. R. & Kinny, D. (2000). The gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3), 285-312.

Chapter X

Reuse of a Repository of Conceptual Schemas in a Large Scale Project

Carlo Batini, University of Milano Bicocca, Italy

Manuel F. Garasi, Italy

Riccardo Grosso, CSI-Piemonte, Italy

ABSTRACT

This chapter describes a methodology and a tool for the reuse of a repository of conceptual schemas. Large amounts of data are managed by organizations, with heterogeneous representations and meanings. Since data are a fundamental resource for organizations, a comprehensive and integrated view is needed for it. The concept of data repository fulfils these requirements, since it contains the description of all types of data produced, retrieved, and exchanged in an organization. Data descriptions should be organized in a repository to enable all the users of the information system to understand the meaning of data and the relationships among them. The methodology described in the chapter is applied in a project where an existing repository of conceptual schemas, representing information of interest for central public administration, is used in order to produce the corresponding repository of the administrations located in a region. Several heuristics are described and experiments are reported.

INTRODUCTION

The goal of this chapter is to describe a methodology and a tool for the reuse of a repository of conceptual schemas. The methodology is applied in a large scale project related to the Italian Public Administration (PA); the goal of the project is to use the repository of conceptual schemas of the most relevant databases of the Italian central PA, developed several years ago, in order to build the corresponding repository of the local public administrations located in one of the 21 regions of Italy. Due to the limited amount of available resources, the methodology conceives and applies several approximate techniques, which allows for the rapid prototyping of the local repository. This is to be refined by domain expert, which results in a resource consumption one order of magnitude lower than by using a traditional process. We initially provide some details about the context in which the methodology has been investigated and developed.

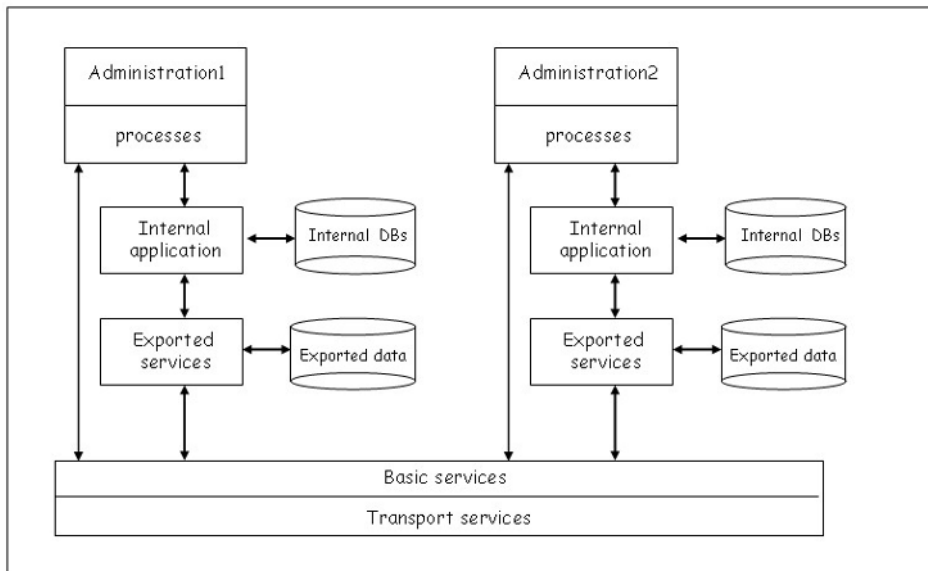
In all countries, in the past few years, many projects have been set up to effectively use information and communication technologies (ICT) to improve the quality of services for citizens, by gradually improving on the services that are provided by information systems and databases of their administrations. In the following section, we focus in particular on the Italian experience.

In the past, the lack of cooperation between the administrations led to the establishment of heterogeneous and isolated systems. As a result, two main problems have arisen, namely, duplicated and inconsistent information and difficult data access. Moreover, the government efficiency depends on the sharing of information between administrations, due to the fact that many of them are often involved in the same procedures, but they are using different, overlapped, and heterogeneous databases.

Therefore, in the long term, a crucial aspect for the overall project is to design a cooperation architecture that allows both the central and the local administrations to share information in order to provide services to citizens and businesses on the basis of the “one-stop shopping” paradigm. A crucial aspect of such cooperation architecture is the data architecture: data have to be interchanged with an interoperable format; all the administrations have to assign the same meaning to the same data, achieving database integration in the long term. The database integration will provide for the spread of information within the government branches and will result in a more easily accessible working environment, in an increased quality of information management, and in an improved statewide decision-making process.

The long term goal of database integration has to be achieved in the complex organizational scenario of the Public Administration (PA). The structure of the Public Administration in Italy consists of central and local agencies that together offer a suite of services designed to help citizens and businesses to fulfill their obligations towards the PA. Central PAs are of two types: ministries, such as Ministry of the Interiors and Ministry of Revenues; and other central agencies, such as Social Security Agency, Accident Insurance Agency, and Chambers of Commerce. The main types of local administrations correspond to Regions (21), Provinces (about 100), and Municipalities (about 8,000).

To address this problem, the approach to cooperation among administrations followed in Italy is based on the concept of Cooperative Information Systems (CIS), that is, systems capable of interacting by exchanging services with each other. The general cooperative architecture for the Nationwide CIS network of the Italian PA is shown in Figure 1.

Figure 1. The structure of the cooperative architecture

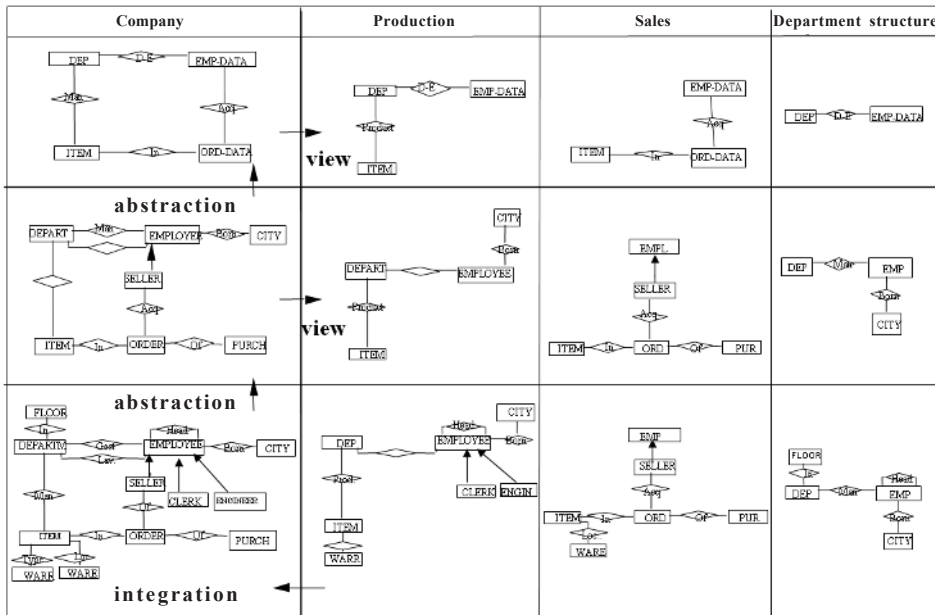
One of the first activities performed in the last decade, with the final goal of designing suitable data architecture, has been the project of building an inventory of existing information systems operating within the central PA in Italy. The activity was performed on about 500 databases in which logical schemas were translated into Entity Relationship schemas through reverse engineering activities. In order to provide a structure to such a large amount of schemas, the methodology for building repositories of conceptual schemas described in Batini, Di Battista, and Santucci (1993) was used. We describe briefly this methodology in the next section.

In order to achieve cooperation among central and local administrations, it is necessary to design a data architecture that covers both types of administrations, and, consequently, a similar repository has to be developed for local administrations. For this reason, several regional administrations are now designing their own data architecture.

The most advanced organizational context among local administrations in a region occurs when they are coordinated by a regional agency that provides services to all or at least to the majority of them. This is the situation of the administrations of the Piedmont region, where such a central agency exists, CSI Piemonte. But also in such a fortunate context, only logical relational schemas are available as input to the process of the construction of the local repository. So, a methodology and tools are needed which let the approximate production of conceptual schemas be arranged in a repository. In this chapter, we describe this methodology and the experience we achieved so far in applying this to the context of the Piedmont Public Administrations.

The chapter is organized as follows. In the next section, we provide the background on primitives that are used to structure repositories in our approach, the original

Figure 2. An example of repository



methodology for repository construction where only loose restrictions on resources existed, and we sketch the methodology for reuse, discussing related work at the end. We then describe in detail the methodology for reuse. Future trends in the area of repository reuse are discussed in the subsequent chapter, followed by our conclusions.

BACKGROUND: HOW TO STRUCTURE AND BUILD A REPOSITORY OF SCHEMAS AND GUIDELINES ON ITS REUSE

The Structure of a Repository of Conceptual Schemas

A repository, in the context of this chapter, can be defined as a set of conceptual schemas, each one describing all the information managed by an organisational area within the information system considered, organized in such a way as to highlight their conceptual relationships and common concepts. In particular, the repositories referenced in this chapter use the entity relationship model to represent conceptual schemas. However, a simple collection of schemas does not display the relationships among schemas of different areas; the repository has to be organised in a more complex structure through the use of suitable structuring primitives.

Figure 3. A fragment of repository

IS12345678											
	IS123				IS456				IS178		
		S1	S2	S3		S4	S5	S6		S7	S8

The primitives used in our approach are: abstraction, view, and integration. Abstractions allow the description of the same reality at different refinement levels. This mechanism is fundamental for a data repository, since it helps the user to perceive a complex reality step-by-step, going from a more abstract level to a more detailed one (or vice versa). Views are descriptions of fragments of a schema. They allow users to focus their attention on the part of a complex reality of interest to them. Integration is the mechanism by which it is possible to build a global description of data managed by an organisational area starting from local schemas. By jointly using these structuring primitives, we obtain a repository of schemas. Each column of the repository represents an organisational unit, while each row stands for a different abstraction level. The left column contains the schemes resulting from the integration of all the other schemes belonging to the same row (views of the integrated schema). In Figure 2, we show an example of repository, where the production, sales, and department schemas are represented at different refinement levels respectively in the second, third, and fourth column, while the company schema in the first column is the result of their integration.

In practice, when the repository is populated at the bottom level by hundreds of schemas, as in the cases that we will examine in the following section, it is unfeasible to manage the three structuring primitives, and the view primitive is sacrificed. Furthermore, the integration/abstraction structuring mechanism is iterated, producing a sparsely populated repository such as the one symbolically represented in Figure 3, where, for instance, schema S123 results from the integration/abstraction of schemas S1, S2, and S3.

The repository structure described previously has been adopted for representing the conceptual content of a wide amount of conceptual schemas related to the most relevant databases of Italian central PA in an integrated structure.

A Methodology for Building a Repository of Schemas

In order to build the whole repository, an initial methodology has been designed. It is described in detail in Batini, Di Battista, and Santucci (1993), and Batini, Castano, De Antonellis, Fugini, and Pernici (1996), and is briefly described here. The methodology is made up of three steps:

1. **Schema production:** Starting from logical relational schemas or requirement collection activities, traditional methodologies for schema design have been used (e.g., see Batini, Ceri, and Navathe [1991]) that lead to the production of about 500 basic schemas, representing the information content of the most relevant databases used in the central public administration at the conceptual level.

Figure 4. The repository of schemas of central public administration

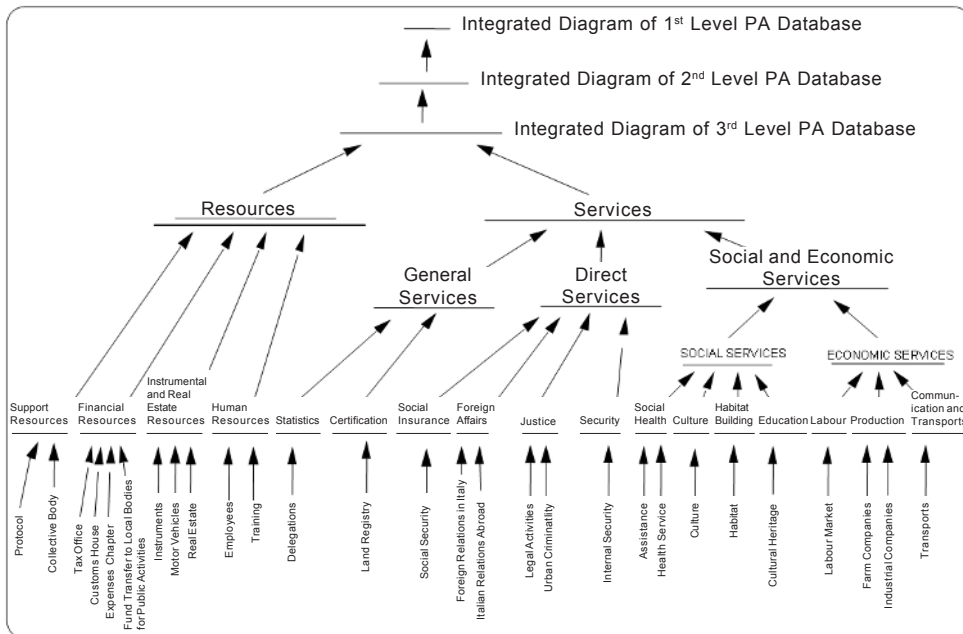
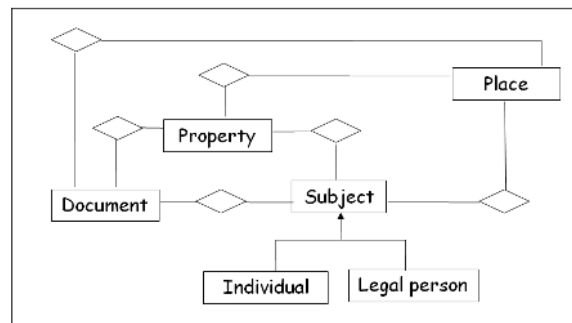
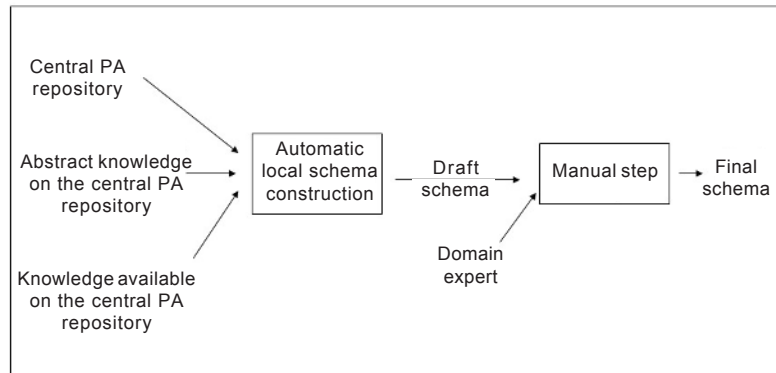


Figure 5. The schema at the top level of the repository



2. **Schema clustering:** First, conceptual schemas representing the different organization areas are grouped in terms of homogeneous classes, corresponding to meaningful administrative areas of interest in central public administration. 27 different areas have been defined; examples of areas are social security, finance, cultural heritage, and education. As we said, at the bottom level of the repository, we have about 500 schemas, corresponding to the logical schemas of the databases of the 21 most relevant central PAs in Italy, with approximately 5,000 entities and

Figure 6. The two steps of the reuse methodology

a similar number of relationships. We denote in the following basic schemas the conceptual schemas defined at the bottom of the repository.

3. **Iterative integration/abstraction:** Each group of basic schemas is integrated and, at the same time, abstracted, resulting in a unique schema for each area that populates the second level of the repository, resulting in 27 second-level abstract schemas. In Figure 4, the different levels of the repository are represented, starting from the second level; for instance, the Internal security second-level schema results from the integration/abstraction process, performed over six schemas corresponding to 130 concepts.

About 200 person months were needed to produce the 500 basic conceptual schemas of the repository in the schema production step, while about 24 person months were needed to produce the 55 abstract schemas of the upper part of the repository (approximately two weeks per schema, both for basic and for abstract schemas). In Figure 5, the schema at the top level of the repository is shown.

Assumptions and Basic Choices for a Methodology for Repository Reuse

In the project related to the production of the repository for local PA, available resources were one order of magnitude lower. For this reason, we were forced to reuse the Repository developed for the central PA and adapt the methodology to the new context, by conceiving new heuristic techniques.

To do so, as we will describe in detail in the next section, we propose a methodology for reuse in a different domain based on the following guidelines:

1. While basic schemas of the central PA repository and the local PA repository may probably differ due to the different functions between central and local administrations, our first assumption holds that the similarity should be much higher between the abstract schemas of the central PA repository and the more relevant concepts of basic + abstract schemas of the local PA repository.

2. In order to reduce human intervention as much as possible, the methodology (its high-level structure is shown in Figure 6) first performs an automatic activity, where several heuristics are applied, that use abstract knowledge of the central PA repository, producing a first draft version of the basic schemas. This version is then analyzed by the domain expert that may add or modify concepts, thus producing the final schema.

Literature Review

The literature on the application of ICT technologies in e-government is vast; see Mecella and Batini (2001) for an introductory discussion and a description of the Italian experience. Repositories of conceptual schemas are proposed in several application areas; see, for example, in biosciences the Taxonomic Database Working Group (2004).

The literature on repositories of conceptual schemas can be organized into two different areas: a) primitives for repository organization and methodologies for repository production, and b) new knowledge representation models for repositories.

Concerning primitives and methodologies, using a descriptive model based on words and concepts, Mirbel (1997) proposes primitives for integration of object-oriented schemas that generate abstract concepts as a result of the integration process. As a consequence, the primitives of Mirbel (1997) are similar to ours, but no evidence is provided to prove the effectiveness of the approach on a large-scale project.

Castano, De Antonellis, and Pernici (1998) and Castano and De Antonellis (1997) propose criteria and techniques to support the establishment of a semantic dictionary for database interoperability, where similarity-based criteria are used to evaluate concept closeness and, consequently, to generate concept hierarchies. Experimentation of the techniques in the public administration domain is discussed.

Shoval, Danoch, and Balabam (2004) introduce the concept of conceptual schema package as an abstraction mechanism in the entity relationship model. Several effective techniques are proposed to group entities and relationships in packages, such as dominance grouping, accumulation and abstraction absorbing. While the Shoval et al. package primitive is more powerful than our abstraction primitive, it does not address the integration issue.

Perez, Ramos, Cubel, Dominguez, Boronat, and Carsi (2002) present a solution and methodology for reverse engineering of legacy databases using formal method-based techniques.

Concerning new knowledge representation models, repositories of ontologies are proposed in several papers. The alignment and integration of ontologies is investigated by Wang and Gasser, (2002), Di Leo, Jacobs, Pand, and De Loach (2002), and Fanquhar, Fikes, Pratt, and Rice (1995), where information integration is enabled by having a precisely defined common terminology. A set of tools and services is proposed to support the process of achieving consensus on such commonly shared ontologies by geographically distributed groups. Users can quickly assemble a new ontology from a library of modules.

In Pan, Cranfield, and Carter (2003), multi-agent systems rely on shared ontologies to enable unambiguous communication between agents. An ontology defines the terms or vocabularies used within encoded messages, using an agent communication language. In order for ontologies to be shared and reused, ontology repositories are needed.

Slota et al. (2003) propose a repository of ontologies for public sector organizations. The repository is used in a system supporting organizational activity by formalizing, sharing, and preserving operational experience and knowledge for future use.

In our approach, as regards to the above-mentioned contributions, the following aspects are new:

- a. the abstraction/integration primitive adopted for structuring the repository;
- b. the attention devoted to feasibility aspects and resource constraints;
- c. the consequent heuristic methodology for reuse; and
- d. the experiments conducted (reported later in the chapter) provide evidence of the effectiveness of the approach.

On the other hand, conceptual models are less powerful than ontology-based models, while being more manageable in practical cases.

A METHODOLOGY FOR REPOSITORY REUSE

Knowledge Available in the New Domain

In this section, we describe in more detail the knowledge available for the design of the local PA repository, and we describe the assumptions that have been made in the activity.

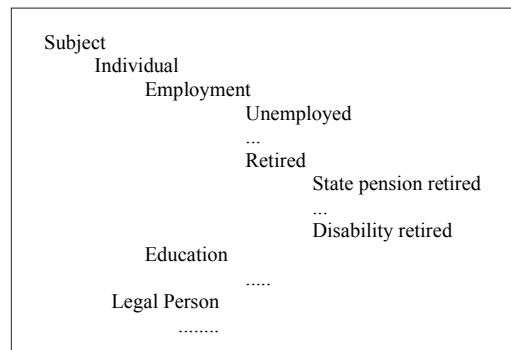
A first relevant input available for the process is the central PA repository of schemas, made of basic and abstract schemas. A second input concerns local databases. The Piedmont local PA is centrally served by a unique consortium, CSI Piemonte, that created approximately 450 databases of 12 main local administrations in the last few years, whose logical schemas are documented in terms of: relational database schemas, tables (approximately 17.000), textual descriptions of tables, referential integrity constraints defined among tables, attributes, definitions of attributes, and primary keys.

The basic sources of knowledge available for the production of the local PA repository, as results from the above discussion, are very rich, but characterized by two significant heterogeneities: the conceptual documentation concerns central administrations, while for local Piedmont administrations, the prevalent documentation concerns logical schemas.

A second relevant condition of our activity has concerned budget constraints; for the first year of the project, we had only one person year available, which was less than one-tenth of the resources that were available for the construction of the central repository. So, in conceiving the methodology for the local PA repository production, we used heuristics and approximate reasoning in order to reduce human intervention as much as possible.

As a consequence of resource constraints and the assumption discussed in the previous section, we decided to use in some steps of the methodology a more manageable knowledge base than the 500 central basic schemas + the 50 abstract schemas. Such schemas can be represented in terms of a much more dense conceptual structure that

Figure 7. A fragment of the Subject generalization hierarchy



corresponds to the four generalization hierarchies that have the entities defined in the schema of Figure 5 at their top level. At lower levels, they have the concepts present in more refined abstract schemas and basic schemas, which were obtained applying the refinements top-down along the integration/abstraction hierarchy. We show in Figure 7 a fragment of one of the hierarchies, namely, the one referring to subjects.

So, as a further choice, we decided to use, in addition to the basic schemas and the abstract schemas, the four generalization hierarchies of subject (individual + legal person), property, document, and place.

As a consequence of the above assumptions, constraints and choices, the inputs to the methodological process, shown in figure 8, have been:

1. The central PA Repository of 550 basic + abstract schemas;
2. The four central PA Generalization hierarchies; and
3. The logical schemas of the 450 local PA databases.

The Methodology

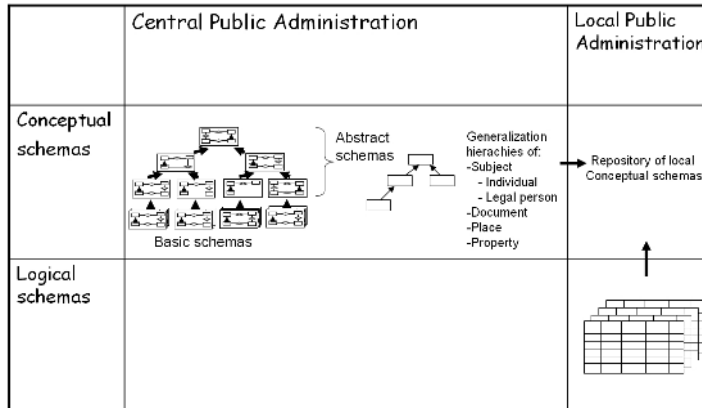
In this section, we present the methodology for building the basic schemas (its extension to abstract schemas is briefly discussed in the *Future Trends Section*). The methodology is composed of five steps. Each step is described with a common documentation frame, providing the inputs to the step, the procedure, and, in some cases when relevant, the outputs of the step. An example is provided, related to a logical schema concerning grant monitoring of industrial business activities.

Step 1. Extract Entities

Inputs: Central PA generalization hierarchies, one local PA logical schema.

Names of entities in hierarchies are compared with names and descriptions of each table, the set of names of the attributes and the descriptions of the attributes in the logical schema. The comparison function presently makes use of a simple distance function among the different strings. The entities and corresponding frequency of matching are

Figure 8. Input knowledge for the production of the Repository of local conceptual schemas



sorted, and a threshold is fixed; all the entities with frequency over the threshold are selected, resulting in a first draft schema made only of entities.

The output is a draft schema made up of disconnected entities.

Step 2. Add Generalizations

Inputs: The draft schema obtained in the previous step and the four central PA generalization hierarchies.

Visit the generalization hierarchies and add to the draft schema subset relationships present in hierarchies, defined among the entities in the draft schema.

Step 3. Extract Relationships

Inputs: The draft schema + all the basic schemas in the central PA repository

Entities of the draft schema are pair wise compared with all the basic schemas in the central PA repository. For each pair of entities E1 and E2, several types of relationships are extracted by the basic schemas:

1. relationships defined exactly on E1 and E2;
2. relationships corresponding to chains of relationships defined among pairs E1-Ei; Ei-Ei+1; ...; Ei+j-E2; and
3. relationships defined among entities E1* and E2* corresponding to ancestors of E1 and E2 in the four generalization hierarchies; they are to be added due to the inheritance property of the generalization hierarchies.

Relationships collected in the first and third step are sorted according to the frequency of names. Here we have two possibilities:

1. The most frequent name is chosen as the name of the relationship; and
2. The name is assigned by the domain expert.

Step 4. Check the Schema with Referential Integrity Constraints Defined Among Logical Tables

Input: The draft schema + constraints defined in tables

An integrity constraint between two tables, T1 and T2, is an indication of the presence of a possible relationship between the entities corresponding to T1 and T2 in the ER schema. For each referential integrity constraint defined among two tables, T1 and T2, in the logical schema, it is controlled as to whether T1 and/or T2 have been already selected as entities in the draft schema, and in case they are not selected, they are added as new entities. Furthermore, it is controlled as to whether a relationship is defined among the entities, and if not, it is added. The type of relationship (e.g., one to many) in the present version of the methodology is chosen by the domain expert in Step 5. Since particular cases of referential integrity constraints exist that do not give rise to ER relationships (e.g., key/foreign key relationships corresponding to IS-A hierarchies), all the ER relationships generated in this step are controlled by the domain expert.

Step 5. Domain Expert Check of the Draft Schema and Construction of the Final Schema

Input: The draft schema

In this step the schema produced by the semi-automated process is examined by the knowledge domain expert that may add new concepts, cancel existing concepts, or else modify some concepts.

Since Step 5 is performed after the addition of relationships and entities resulting from referential integrity constraints, it may occur that too many concepts have been added, and the manual check of the domain expert leads to deleting some concepts. Sometimes, new concepts are added, resulting in an enriched schema in which the kernel is the initial schema. Frequently, schemas obtained after the integrity constraints check step and after the domain expert check step coincide.

Output: the: final schema

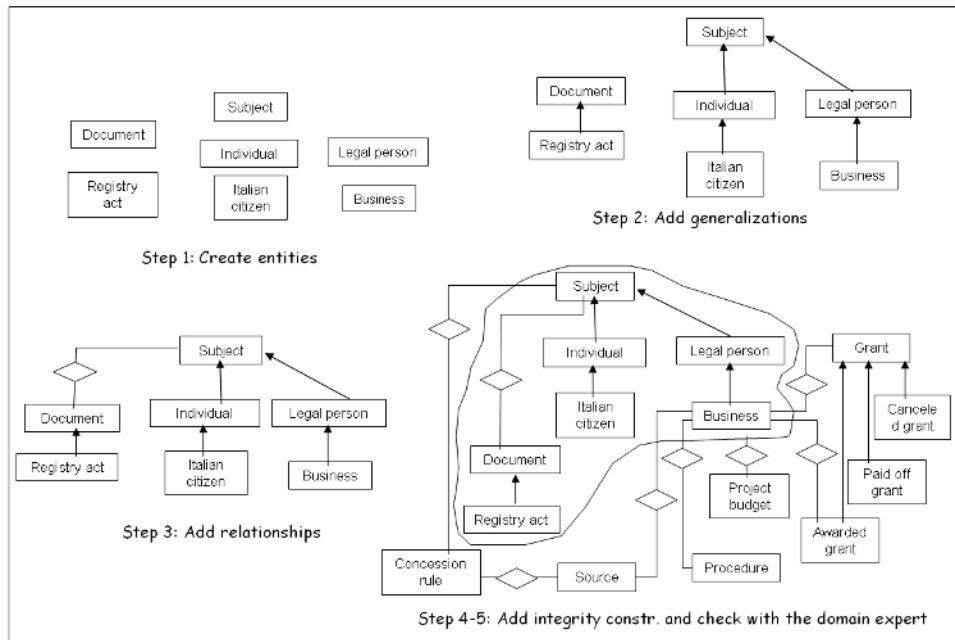
We show in Figure 9 the schemas obtained as a result of the execution of Steps 1 to 5 of the methodology in our case study. In this case, schemas obtained after the integrity constraints check step and after the domain expert check step coincide and, consequently, are not distinguished in the figure.

Experiments and Improvements

We have experimented with the above methodology in three different areas — businesses, health care, and regional territory — and nine related fields. The total number of tables of the nine databases is approximately 550, corresponding to 3% of the total. We were interested in measuring two relevant qualities of the process:

1. The correctness of the conceptual schema with respect to the “true” one, that is, the schema that could be obtained directly by the domain expert through a traditional analysis or else a reverse engineering activity. Correctness is measured with an approximate indirect metrics, corresponding to the percentage of new/deleted concepts in the schema produced by the expert at the end of step 5 with respect to concepts produced in the semi- automatic steps 1-4.

Figure 9. Schemas obtained after Steps 1-5



2. The completeness of the conceptual schema with respect to the corresponding reengineered logical schema. Completeness is measured by the percentage of tables that are extracted in steps 1-5, in comparison with the total number of tables, after excluding tables not carrying relevant information, such as redundant tables, tables of codes, and so forth.

Table 1 summarizes the main results of experiments. Concerning correctness, in general, the schemas obtained after the check with integrity constraints step and after the domain expert check step are very similar; that is, domain experts tend to confirm and consider complete entities and relationships added in the previous step. The overall figure for the nine experiments results in more than 80% of concepts common to the two types of schemas. We see also that the add constraints step introduces approximately 30% of new concepts in comparison with the extract entities step. Consequently, the joint application of the central PA knowledge and local PA knowledge is shown to be effective. These are encouraging results, considering the highly heuristic nature of the methodology.

Concerning completeness, in the first experiments of the methodology, results have been less reassuring. On average, only 50% of the tables are extracted. This value changes significantly in the different areas. Furthermore, as was to be expected, completeness decreases significantly when the referential integrity constraints are not

Table 1. Experiments results

Step	# of tables extracted	% of tables extracted
Create entities	172	30
Add constraints	219	41
Domain expert check	275	51

documented or partially documented, resulting in lower quality (completeness) conceptual schemas. Apart from the quality of the documentation, another cause of reduced completeness is the static nature of generalization hierarchies used in Step 1, and the unequal semantic richness in representing related top-level concepts. For instance, in the initial Subject hierarchy, 20 concepts represent individuals, while only three represent legal persons.

An improvement we have made concerns the incremental enrichment of the generalization hierarchies with new concepts, possibly generated in Step 5. Such enriched hierarchies have been progressively reconciled and made similar to hierarchies characteristic of local administrations, resulting in a corresponding and more effective selection mechanism. We performed a new experiment in which we used an enriched subject hierarchy, with legal persons represented by 20 concepts, that resulted in an increment of tables extracted after the create entities step from 30% to 35%, and tables extracted after the add constraints step from 51% to 73%.

A final comment on resources. The amount of resources spent in the experiments has been, on the whole, 30 person/days, corresponding to three person/day per schema. About 30% of the time has been spent in steps 1-4, and 60% of the time has been spent in the manual check. So, the domain expert has been engaged for two days per schema; we have to add a fixed cost of a 3-day course to this variable cost. We may expect greater efficiency as long as the activity proceeds, and estimate in one person/day the average final effort, significantly lower than the 2-3 person/weeks needed for design of one schema in the central PA repository.

The Tool

A prototype has been implemented, which results in a tool that can fully automate the first four steps of the reuse methodology, and can document the decisions of the domain expert made in the fifth step. The output of each step is represented as a text file that describes the schema both in an internal XML format and in a semi-natural language. The XML format can be provided to a design tool, e.g., Erwin, to produce a graphic schema; the semi natural language is used as a user friendly description of schemas.

The prototype is presently implemented in Visual Basic 6.0 and uses an Access DBMS. We are currently moving to a Visual Basic.Net version and Oracle DBMS.

In Figure 10, we show an example of a screenshot produced by the tool which shows the result of the execution of the add entity step to a specific database.

Figure 10. A screenshot produced by the tool

Reports

Data Base: AAEP Business Registry [All Steps]

Add-Entity | Add-Generalization | Add-Relationship | Inter-Constraints

Action

F1					
ID	nbd	nt	dt	nc	dc
814910	AAEP Business Registry	AAEP_BUSINESS_ID	Contains the	CC_YEAR	Year of registration
814911	AAEP Business Registry	AAEP_BUSINESS_ID	Contains the	CC_REG_ID	Registration Id
814912	AAEP Business Registry	AAEP_BUSINESS_ID	Contains the	CC_PROV	Province Id
814913	AAEP Business Registry	AAEP_BUSINESS_ID	Contains the	CC_TRIBUNAL	Tribunal Identification
814914	AAEP Business Registry	AAEP_BUSINESS_ID	Contains the	D_EXP_VAL	Expiration Date of validity
814915	AAEP Business Registry	AAEP_BUSINESS_ID	Contains the	D_START_VAL	Starting Date of validity
814916	AAEP Business Registry	AAEP_BUSINESS_ID	Contains the	ID_BUSINESS	Business Id

F2				
ID	Cod	Level	Name	Rule
97	P	1	B01_PROPERTY	%prop
98	P	2	B02_REAL_ESTATE	%realEstate%
99	P	3	B03_RESIDENCE	%resid%
100	P	3	B03_BUILDING	%build%
101	P	3	B03_LAND	%land%
102	P	2	B02_PERSONAL_PROPERTY	%pers%prop%
103	P	3	B03_CAR	%car%

FA				
Hierarchv	Table	TableDesc	Field	FieldDesc
T04_REGION	TT_MUNICIPALITY	Contains the		
T05_PROVINCE	TT_MUNICIPALITY	Contains the		
T07_MUNICIPALITY	TT_MUNICIPALITY	Contains the		
T04_REGION	TT_MUNICIPALITY	Contains the		
T05_PROVINCE	TT_MUNICIPALITY	Contains the		

FUTURE TRENDS

We are now analyzing lessons learned and we are improving on the methodology.

First, we are extending the methodology to the production of abstract schemas in the repository. This step may effectively use the results of previous steps 1-5. In fact, the initial schema obtained after steps 1-3 inherits high-level abstract knowledge from the central PA Repository and basic knowledge from the local PA logical schemas, while the enriched schema obtained in steps 4-5 encapsulates basic knowledge from the local PA logical schemas. We may conjecture that the initial schema is a candidate for abstract schema for the upper levels of the local PA repository, while the enriched schema — being a more detailed description representing a logical schema — populates the basic level of the repository. So, we can conceive two possible strategies for the repository update step.

In the first strategy, starting from the initial schema and the enriched schema, we first complete the “local” repository of abstract schemas corresponding to the enriched schema; we then integrate the local repository with the actual one. It may occur that we have to update, due to similarities between concepts, the abstract schemas of the actual repository, or else add new schemas, autonomous with respect to the previous ones.

In the second strategy, the new repository is obtained through abstraction/integration activities on the actual local PA repository and the initial and refined schemas.

The first strategy is probably more effective when the actual local PA repository and the new schema represent very different knowledge, while the second strategy has the advantage of natively using the structuring paradigm of the repository, the abstraction/integration operation.

We are currently experimenting with the two strategies and other possible strategies, such as building small homogeneous repositories and then integrating them to obtain a larger repository.

We are also investigating new techniques that use more complex similarity measures in matching between generalization hierarchies and logical schemas. Furthermore, since some of the local PA schemas (and corresponding hierarchies) have been independently developed, especially in the regional territory area, we are using such schemas as training examples to tune semi-automatic steps of the methodology and similarity measures which have been adopted.

CONCLUSION

In this chapter, we have investigated methodologies for conceptual schema repository construction and reuse in complex organizations such as, in particular, public administration. We have shown how accurate methodologies, which can be used when large amounts of resources are available, have to be modified into approximate methodologies when we want to reuse previous knowledge and when available resources are limited. We have compared the proposed approach with existing literature in the area, and we made several experiments that provide evidence of the effectiveness of the approach and of the incremental improvements that can be achieved.

NOTE

This work has been fully supported by CSI Piemonte and partially supported by the Italian MIUR FIRB Project MAIS.

REFERENCES

- Batini, C., Castano, S., De Antonellis, V., Fugini, M. G., & Pernici, B. (1996). Analysis of an inventory of information systems in the public administration. *Requirements Engineering*, 1(1), 47-62.
- Batini, C., Castano, S., & Pernici, B. (1997). *Tutorial of inventories of information system in the public administration*. Paper presented at the 17th Entity Relational Conference, Cottbus, Germany.
- Batini, C., Ceri, S., & Navathe, S. B. (1991). *Logical database design using the entity relationship mode*. Palo Alto, CA: Benjamin and Cummings/Addison Wesley.
- Batini, C., Di Battista, G., & Santucci G. (1993). Structuring primitives for a dictionary of entity relationship data schemas. *IEEE Transactions on Software Engineering*, 19(4), 344-365.

- Castano, S., & De Antonellis, V. (1997). Semantic dictionary design for database interoperability. In *Proceedings of the 13th International Conference on Data Engineering*, University of Birmingham, UK (pp. 43-54).
- Castano, S., De Antonellis, V., & Pernici, B. (1998). Conceptual schema analysis: Techniques and applications. *ACM Transactions on Database Systems*, 23(3), 286-332.
- DiLeo, J., Jacobs, T., & DeLoach, V. (2002). Integrating ontologies into multiagent systems engineering. In *Proceedings of the Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems*, Bologna, Italy.
- Farquhar, A., Fikes, R., Pratt, W., & Rice, J. (1995). *Collaborative ontology construction for information integration* (Tech. Rep. No. KSL-95-10). Knowledge Systems Laboratory, Department of Computer Science Stanford University, CA.
- Fonseca, F., Davis, C., & Camara, G. (2003). Bridging ontologies and conceptual schemas in geographic information systems. *Geoinformatica*, 7(4), 355-378.
- Mecella, M., & Batini, C. (2001). Enabling Italian e-government through a cooperative architecture. In A. K. Elmagarmid, & W. J. McIver Jr. (Eds.), Special Issue on Digital Government. *IEEE Computer*, 34(2), 40-45.
- Mirbel, I. (1997). Semantic integration of conceptual schemas. *Data and Knowledge Engineering*, 21(2), 183-195.
- Pan, J., Cranefield, S., & Carter, D. (2003). A lightweight ontology repository. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, Melbourne, Australia (pp. 632-638). New York: ACM Press.
- Perez, J., Ramos, I., Cubel, J., Dominguez, F., Boronat, A., & Carsi, J. (2002). Data reverse engineering of legacy databases to object oriented conceptual schemas. *Electronic Notes in Theoretical Computer Science*, 74(4).
- Shoval, P., Danoch, R., & Balabam, M. (2004). Hierarchical entity-relationship diagrams: The model, method of creation and experimental evaluation. *Requirements Engineering*, 9(4), 217-228.
- Slota, R., Majewska, M., Dziewierz, M., Krawczyk, K., Laclavik, M., Balogh, Z., et al. (2003, Sept. 7-10). Ontology-assisted access to document repositories for public sector organizations. In *Proceedings of Parallel Processing and Applied Mathematics: 5th International Conference (PPAM)*, Czestochowa, Poland (LNCS 3019, pp. 700-705). Berlin: Springer-Verlag.
- Taxonomic Databases Working Group on Biodiversity Informatics (2004). *Proceedings of the Taxonomic Databases Working Group Annual Meeting*, University of Canterbury, Christchurch, New Zealand. Retrieved from <http://www.tdwg.org>
- Wang, J., & Gasser, L. (2002). Mutual online ontology alignment. In *Proceedings of the AAMAS 2002 Workshop on Ontologies for Agent Systems* (CEUR Workshop Proceedings Vol. 66).

Chapter XI

The MAIS Approach to Web Service Design

Marzia Adorni, Francesca Arcelli, Carlo Batini, Marco Comerio,
Flavio De Paoli, Simone Grega, Paolo Losi,
Andrea Maurino, Claudia Raibulet, Francesco Tisato,
Università di Milano Bicocca, Italy

Danilo Ardagna, Luciano Baresi, Cinzia Cappiello, Marco Comuzzi,
Chiara Francalanci, Stefano Modafferi, Barbara Pernici,
Politecnico di Milano, Italy

ABSTRACT

This chapter presents a first attempt to realize a methodological framework supporting the most relevant phases of the design of a value-added service. A value-added service is defined as a functionality of an adaptive and multichannel information system obtained by composing services offered by different providers. The framework has been developed as part of the multichannel adaptive information systems (MAIS) project. The MAIS framework focuses on the following phases of service life cycle: requirements analysis, design, deployment, and run-time use and negotiation. In the first phase, the designer elicits, validates, and negotiates service requirements according to social and business goals. The design phase is in charge of modeling services with an enhanced version of UML, augmented with new features developed within the MAIS project. The deployment phase considers the network infrastructure and, in particular, provides an approach to implement and coordinate the execution of services from different providers. In the run-time use and negotiation phase, the MAIS methodology provides support to the optimal selection and quality renegotiation of services and to the dynamic evaluation of management costs. The chapter describes the MAIS methodological tools available for different phases of service life cycle and discusses the main guidelines driving the implementation of a service management architecture called reflective architecture that complies with the MAIS methodological approach.

INTRODUCTION

The design and implementation of multichannel and mobile information systems presents cross-disciplinary research problems. The information system should support adaptivity, since the execution environment is characterized by continuous change, particularly in mobile and ubiquitous systems where it is highly distributed and characterized by a high heterogeneity in both technological platforms and user requirements. Therefore, concepts such as stratification and information hiding turn out to be inadequate, since it is almost impossible to identify and implement optimal built-in strategies. Moreover, non-functional requirements (performance, reliability, security, cost, and, more generally, quality of service) become more and more relevant, and the management of the resources of the system can no longer be hidden, but instead has to be visible and controllable at the application level.

The goal of the multichannel adaptive information systems (MAIS) project is the development of models, methods, and tools that allow the implementation of multichannel adaptive information systems. The information system functionalities are provided as services on different types of networks and access devices and are the result of the composition of services offered by different providers to build a value-added service.

This chapter presents a proposal to realize a methodological framework supporting the most relevant phases of the design of a value-added service. In particular, we focus on the support of creation (e.g., analysis, design, and development) of a service as an abstract service and on its use as an orchestration of a set of existing component services.

Within the framework presented, the design of value-added services is restricted to the abstract definition of their functional and non functional features. Thus, the MAIS framework does not pay attention to specific implementation details, such as service location and service access protocols, or the component service actually selected during a specific information system use, since the selected service may change quickly in a loosely coupled information system. The framework is also focused on the design of deployment alternatives and on the monitoring and control of quality of service during execution.

The goal of the MAIS framework is to provide a first integrated view of design aspects which are not considered in an integrated methodological framework in the literature. In particular, the objective is to focus on the service selection phase and on the representation of quality requirements at a system level.

The chapter is organized as follows: the next section provides a survey of methodologies already proposed in the literature to deal with web service design and quality of service representation; then, we present the MAIS methodological framework covering the most relevant phases of the Web service life cycle; the subsequent four sections describe in depth each component of the methodological framework; and the last section draws conclusions and outlines future work.

RELATED WORK

Several approaches have been proposed in the literature for the design of Web services as composed services and of cooperative information systems based on a service-oriented approach.

Some approaches focus on the selection of component services with a goal-based approach, at a conceptual level. In Kaabi, Souveyet, and Rolland (2004), cooperative processes are built on the basis of intentions and strategies in virtual organizations, and in Colombo, Francalanci, and Pernici (2004), a goal-based approach considering also non-functional requirements to identify resources and constraints is proposed. Other approaches propose to dynamically select and adapt services in a process based on meta-level descriptions (Casati & Shan, 2001) or to compose them based on planning and monitoring techniques (Lazovik, Aiello, & Papazoglou, 2004). Mecella, Parisi-Presicce, and Pernici (2002) consider process control and responsibility to design processes that involve several organizations cooperating on a service oriented approach. In Baña, Benatallah, Casati, and Toumani (2004), the same problem is tackled by modeling the interaction between participating organizations focusing on the evolution state of a process. Other approaches, for example, in Web design literature, are more focused on the interaction design, but they are out of the scope of this chapter.

The Model Driven Architecture (MDA) has the purpose of separating the specification of the operation of a system from the details of the way the system uses the capabilities of a specific platform. The Object Management Group proposes, in Siegel and the OMG Staff Strategy Group (2001), its MDA to support the application development process. Such architecture provides an approach for: (a) specifying a system independently of the platform that supports it; (b) specifying platforms; (c) choosing a particular platform for the system; and (d) transforming the system specification into one for a particular platform. The development process proposed by OMG is divided into three steps. In the first step, the platform independent model (PIM) is created, expressed in UML. This model describes business rules and functionalities of the application and it exhibits a specified degree of platform independence. In the second step, a platform specific model (PSM) is produced by mapping the PIM into a particular platform. In the last step, there is the generation of the application. The “service modeling” phase presented in this chapter has the same goal as the first step of the process proposed by OMG. In fact, for the creation of the PIM, the technological aspects are considered in an abstract way. Moreover, in the application-development process proposed by OMG, a user customization phase that takes into account the user profile is missing. Grønmo and Solheim (2004) and Skogan, Grønmo, and Solheim (2004) present an MDA strategy to develop Web services. Their approach uses a platform independent model (PIM) such as UML to model Web service, then, by means of a translator tool, it produces both Web Service Definition Language (WSDL) and Business Process Executive Language for Web Services (BPEL4WS) descriptions. However, these authors neither enrich the Web service description with the quality of service (QoS) specification nor use ontologies to support the designer.

QoS, Ontology, and design of Web services have been investigated in the last few years, but no one explored them together. Quality of service aspects, in particular, are being considered more and more in the service orientation literature, for example, Menasce (2004) and Abhijit et al. (2004). Their focus, however, is more on the representation and monitoring of quality of service aspects rather than on design for quality services. Ulbrich, Weis, and Geihs, (2003), Sun et al. (2002), and Jaeger, Rojec-Goldmann, and Muhl (2004) face the problem of evaluating QoS dimensions in composite Web services. They define a set of composition rules able to evaluate the global value of a QoS

dimension according the specific workflow patterns used. However, they do not consider how QoS dimensions are designed and assigned to each Web service component; moreover, in these approaches, QoS dimensions are fixed and the authors do not consider an open and flexible approach in the definition of QoS dimensions.

Cardoso, Sheth, Miller, Arnold, and Kochut (2004) present a fixed QoS model (time, cost, reliability) that makes it possible to compute the quality of service for workflows automatically based on atomic task QoS attributes. Such a QoS model is then implemented on top of the METEOR workflow system. The reduced number of QoS dimensions considered reduces the possibility to adopt this approach in several application domains.

Mylopoulos and Lau (2004) propose to design Web services by means of Tropos, an agent-oriented software development technique. Tropos supports the early and late requirements analysis, as well as architectural and detailed design, but it does not offer ontology support for QoS description and does not provide an automatic tool to generate WSDL descriptions from Tropos schema.

Ontology-driven design frameworks are mainly investigated in the context of Semantic Web. Gomez-Perez, Gonzalez-Cabero, and Lama (2004) and Pahl and Casey (2003) propose an MDA compliant, ontology-based framework for describing semantic Web services. Ontologies are used to describe functional descriptions and a set of axioms representing composition rules, but they do not consider how to use ontologies for supporting the definition of QoS. Finally, several contributions consider QoS to improve the result of Web Services discovery (Shuping, 2003), but none of them considered the problem of designing QoS-enabled Web services.

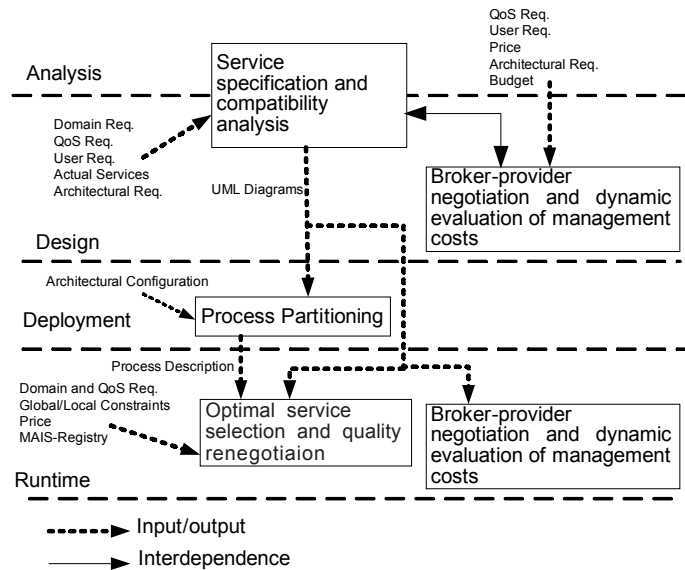
THE MAIS METHODOLOGICAL FRAMEWORK

The life cycle of Web services, both simple and complex, is composed of a series of methodological phases, from requirements analysis to service monitoring at run time. Figure 1 reports the phases of the MAIS methodological framework:

- requirements analysis;
- design;
- deployment; and
- run-time use and negotiation.

In the first phase, the designer elicits, validates, and negotiates Web service requirements according to social and business goals. Services are supposed to be provided to users through different distribution channels. The inputs of this phase are domain requirements, QoS requirements, user profiles, and architectural requirements for different distribution channels. The output of this phase is a set of functional and non-functional requirements, which is taken as input by the subsequent design phase (see Figure 1). The MAIS methodological framework described in this chapter assumes that an informal description of functional and non-functional requirements is available and provides support starting from the design phase.

Figure 1. MAIS contributions to the Web service life cycle



The design phase is in charge of modeling services with an enhanced version of Unified Modeling Language (UML), augmented with new features developed within the MAIS project, for example, Abstract Interaction Unit presented in Bertini and Santucci (2004). At this stage of the methodology, the designer is interested in defining a high-level description of the whole system. Therefore, starting from functional and non-functional requirements, the designer identifies the information and the operating services that will be supplied in a multichannel fashion and the corresponding distribution channels. The result of this phase is a set of MAIS-UML diagrams that will be used in the following phases.

Design is also supported by the evaluation of the management costs of services with a varying level of QoS. This evaluation allows the analysis of different service scenarios and for the selection of the most profitable approach of service management for the MAIS brokering architecture.

The deployment phase considers the network infrastructure. The MAIS methodology provides an approach to implement and coordinate the execution of complex services built from multiple services of different providers. The input of this phase is a MAIS-BPEL description, which is a BPEL4WS specification describing a composition of abstract services augmented with QoS and coordination definitions automatically derived from the MAIS-UML diagrams. The output is a set of MAIS-BPEL specifications. The MAIS execution environment provides the possibility to split the execution of a MAIS-BPEL specification on several coordinators, while preserving the execution flow of the process specification. The decentralization of control is supported, for instance, by loosely

coupled networks such as Mobile Ad-hoc Network (MANET) or autonomous interacting organizations with their own BPEL engines.

In the run-time use and negotiation phase, the MAIS project proposes two different tools supporting the adaptive and context-aware use of Web services. The first one is based on the optimal selection and quality renegotiation of services and relative QoS based on a set of abstract descriptions of services and QoS requirements. The second one is in charge of supporting the negotiation and dynamic evaluation of management costs allowing for the maximization of MAIS brokering profits.

The first tool allows workflow engines to invoke the best service satisfying a set of QoS requirements and abstract service descriptions according to the specific execution context and end-user profile. The concepts of abstract services and concrete services are distinguished. An abstract service is a non-invokable service specifying the functional interface of the service and its QoS requirements. A concrete service is a completely described service, that is, an invocable service, inheriting the functional interface and QoS requirements of a corresponding abstract service, but specifying additional implementation details (e.g., access protocol). This distinction allows the designer to define a generic description of Web services at design time without paying attention to implementation problems. Thus, at run-time, an optimization module is in charge of selecting the set of concrete services that satisfies the constraints defined globally on the entire workflow. Implementation problems can be solved at run time, when the right (and optimal) selection and invocation of Web services is realized.

The second tool evaluates the returns of the MAIS brokering service for each concrete service. Indeed, MAIS provides brokering functionalities that the designer may exploit to leverage the available services QoS. QoS improvements are considered at design time when a set of services that satisfies global constraints is not found by the optimization module. The evaluation supports run-time decisions on the most profitable degree of QoS improvement that the MAIS brokering architecture can implement to meet user requirements. The MAIS architecture can improve QoS in several ways. For example, it can improve the quality of a data set requested by a user by complementing the information provided by the supplier of the concrete service with higher quality information from additional sources. These improvements increase QoS, but also involve additional costs. Profits are maximized when the returns from higher QoS are greater than QoS improvement costs.

The MAIS project has also proposed a reflective architecture to support the run-time selection of services and QoS negotiation. The term *reflective* indicates the ability of a system to dynamically adapt to user requirements by using appropriate metadata. The chapter presents the main guidelines driving the implementation of a reflective architecture to show how it is possible to design and realize a reflective middleware, even in a fully distributed environment.

Figure 1 summarizes how the MAIS approach supports the Web service life cycle. In the following sections, the MAIS contribution for each methodological phase is reported. The reader will be referred to MAIS papers and reports to find more detailed descriptions of the methodological components described in the next sections.

SERVICE SPECIFICATION AND COMPATIBILITY ANALYSIS

Research work on service design started from the definition of a methodology for the redesign of existing services, described in Comerio et al. (2004a, 2004b). This redesign methodology is based on existing specifications of services and information on new requirements. The service redesign methodology considers several aspects of the information on new requirements, including communication channels and technologies, user profiles, and quality of service (QoS). Information on communication channels and technologies is necessary to allow redesigned services to provide the same functionalities through a broader set of channels (e.g., PDA, PC, and Mobile phones). The redesign methodology has also reconsidered traditional development processes to take into account new requirements. The output of the methodology is enhanced UML diagrams that describe services in terms of functional and non functional properties.

Recently, a revised version of the methodology that considers design in addition to redesign has been proposed. In order to design new services from scratch, a comprehensive requirements elicitation and specification approach is needed. The revised methodology is composed by three macro-phases: functional-service modeling, high-level redesign, and context adaptation. The functional-service modeling phase aims at modelling functional service requirements as a set of UML diagrams. These diagrams highlight the logical and operational structure of services. The main objective of the second phase, high-level redesign, is to redesign existing services according to new requirements. QoS requirements are modelled by means of appropriate quality dimensions and metrics extracted from the MAIS QoS registry, which provides a structured list of QoS dimensions and corresponding metrics (Cappiello, Missier, Pernici, Plebani, & Batini, 2004). QoS requirements are then quantified with B_k values that represent the quality level that the service must provide for the k^{th} quality dimension. Finally, QoS constraints are modeled by using an extension of UML proposed by OMG. The enhanced UML diagrams that are the output of this phase define services at an abstract level, that is, without considering specific technologies or user characteristics.

The context adaptation phase takes into account the actual target environment in order to evaluate technological and user requirements. An abstract QoS requirement is verified if contextual technical characteristics (for example, the actual device or the network connection) provide quality values greater than or equal to threshold B_k . Quality thresholds can be fixed a priori in the high-level redesign phase by the domain expert or can be set by evaluating end-user profiles.

The value of each quality dimension can be quantified by considering ideal quality values associated with the profile of the requesting user. Therefore, a comparison between the level B_k of each quality dimension defined in the high-level redesign phase (service quality request) and the ideal level B_u associated with the profile of the requesting user (user quality request) allows the compatibility analysis between user requirements and service characteristics. An overall evaluation of compatibility on multiple quality dimensions can be performed by using QoS trees, where each node represents a quality dimension. The MAIS methodology provides a bottom-up quality evaluation approach based on the simple additive weighting technique (see Comerio et

al., [2004b] and Lum and Lau [2003]), which associates a set of weights and quality composition rules to each node. If the evaluation of compatibility identifies mismatches between user requirements and service characteristics, the methodology recommends the identification of the most violated constraints that can be used to select a different set of services satisfying requirements (Glover & Kochenberger, 2003). Instead, if design assumptions are compatible with quality requirements, the context adaptation phase is completed. The output is a set of UML diagrams that model the multichannel service along with its quality characteristics. Such a model will be exploited to actually implement and deploy the service in subsequent methodological phases.

BROKER-PROVIDER NEGOTIATION AND DYNAMIC EVOLUTION OF MANAGEMENT COSTS

The MAIS methodology assumes the existence of a broker between providers and users. The broker has two conflicting goals: to maximize the satisfaction of user requirements and to achieve maximum possible returns from its brokering role. The broker is supposed to be paid by each provider every time a service of that provider is supplied to a user. Payment is quantified as a percentage of price. The value of this percentage is the output of a negotiation process between the broker and the provider occurring when the provider subscribes to the brokering service. The broker can also increase the quality of a service offered by a provider by complementing the service in several ways; however, discussion of this is out of the scope of this chapter.

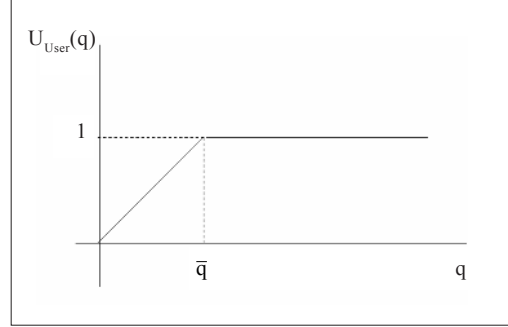
The aim of the service provider i of the service j and the broker in the preliminary negotiation phase is to set the value of a triple $\langle p_{ij}, perc_{ij}, q_{ij} \rangle$ where p_{ij} is the price paid by the user for the service, $perc_{ij}$ is the percentage on the price due by the service provider to the broker, comprised between 0 and 1, and q_{ij} is the aggregate value of QoS (see Section 3) with which the service will be provided ($0 \leq q_{ij} \leq 1$). In Jennings, Lomuscio, Parsons, Sierra, and Wooldridge (2001), an automated negotiation process is defined by the negotiation protocol, the negotiation objectives, and the participants' decision model. These arguments are translated as follows within the brokering negotiation phase:

- **Negotiation protocol:** a bilateral bargaining protocol is adopted;
- **Negotiation objectives:** the preliminary negotiation is a typical multi-attribute problem, since a triple of attributes has to be negotiated, $\langle p_{ij}, perc_{ij}, q_{ij} \rangle$;
- **Decision model:** a trade-off based strategy is adopted to model the participants' behaviour (Jennings, Luo, & Shadbot, 2003).

A utility function V is defined, evaluating how much an offer is worth to a participant. Such utility function is:

- $V = \frac{p_{ij}}{perc_{ij} \cdot q_{ij}}$ for the provider that is interested in maximizing its revenue; and
- $V = p_{ij} \cdot q_{ij} \cdot perc_{ij}$ for the broker that is interested in maximizing both its revenue and users' satisfaction.

Figure 2. Sample user utility function



The broker can increase the service quality level q_{ij} to a quality level q_{ij}^* . In order to provide an example, let us consider a user that requires a data quality level equal to Q_j . If the service provider can offer a quality $q_{ij} < Q_j$, the broker can increase the quality level by improving the data provided with other data retrieved from certified external sources. The quality improvement operation involves a cost that is composed of two different factors:

- C_{acq} : the acquisition cost of certified information; and
- C_e : the processing cost associated with the integration between provider data and external data.

In general, in order to increase the quality level of a service, the broker will incur an extra cost $c^*(q_{ij}^*)$, but can also provide the service to the customer at a higher price $p^*(q_{ij}^*)$. Formally, the goal of the broker is to maximize the function:

$$W_{Broker} * U_{Broker}(q) + W_{User} * U_{User}(q);$$

where U_{Broker} and U_{User} indicate the broker and user utility functions, while W_{Broker} and W_{User} are two weights such that $W_{Broker} + W_{User} = 1$, which establishes the relative importance of broker returns and user satisfaction. Figure 2 shows a sample user utility function. If the quality level provided by the MAIS platform equals the quality level $\bar{q}$ required by the end-user, then the user utility function reaches its maximum. For the sake of simplicity, the figure shows a linear dependency between U_{User} and the aggregated quality level, but non-linear and discontinuous utility functions are also considered.

Conversely, the broker's utility function is expressed as the net revenue from service provisioning, which includes the percentage obtained from the service provider, the actual price of service to the end-user, and the extra cost paid in order to increase quality, $U_{Broker}(q) = p^*(q) - p + p * perc - c^*(q)$. As will be discussed in Section 6, the maximization problem is NP-hard if the platform has to guarantee global constraints for the execution of complex services built from simple services from multiple providers.

PROCESS PARTITIONING

The execution of a complex service in a mobile environment, with different devices connected through different network technologies, needs new strategies with respect to the traditional solutions adopted for centralized workflows. These solutions rely on a single engine that knows and controls all system resources, while mobility demands a decentralized execution carried out by a federation of heterogeneous devices. These requirements lead to a new strategy that stresses the independency among actors, to minimize interaction and knowledge sharing, and thus increases reliability.

The MAIS methodology proposes a set of formal partitioning rules that transform a unique workflow into a set of federated workflows that can be executed by different engines. This is the typical scenario where different devices contribute to the enactment of the whole process by executing a fragment of process and synchronizing with other devices. Our partitioning approach is based on graph transformation systems, where a typed graph defines the types of nodes and edges that can be used to create graphs and transformation rules manipulate these graphs. The left-hand side L of a rule defines the pre-conditions that must hold on the graph to enable the rule, while the right-hand side R describes the post-conditions, that is, the modifications on the graph after applying the rule.

From a methodological point of view, the output of the design phase represents input for this one; nevertheless, the output of design is a set of UML models, while processes are described with MAIS-BPEL. This is only a problem of using the right format; in fact, as described in Gardner, Griffin, and Iyengar (2003), it is possible to translate UML models into BPEL specifications.

The rules read a MAIS-BPEL specification of the original workflow, along with the description of the topology of the network infrastructure (i.e., the list of available engines). The result is a set of MAIS-BPEL specifications that represent the local processes (views) of each engine. This is what each engine is supposed to execute.

The partitioning framework is implemented as a Web service called Partitioner, based on attributed graph grammar (AGG), an existing general-purpose graph transformation tool. This module receives a Graph eXchange Language (GXL) file, representing the original MAIS-BPEL description, and produces a set of GXL files representing the local views for the orchestrators. Consequently, we first translate the original MAIS-BPEL description into GXL by means of XSL technology, and then we re-translate GXL files into a MAIS-BPEL description.

The feasibility of our transformation depends on the assumptions that partitioning rules define a graph transformation system that exposes a functional behavior that is confluent and terminates. Moreover, the execution flow of the original workflow has to be preserved. The first assumption is mandatory to ensure that the actual transformation does not depend on the order in which we apply rules (confluence) and does not enter infinite loops (termination). The second assumption is needed to preserve the original "behaviour," even if we move from centralized to decentralized execution. We can check the first hypothesis by exploiting the critical pair analysis capabilities supplied by AGG. The set of critical pairs precisely represents all potential conflicts. There exists a critical pair if and only if $p1$ may disable $p2$, or $p2$ may disable $p1$ (Hausmann, Heckel, & Taentzer, 2002). Our rules have no conflicts such as the ones described before; thus, our graph transformation system has a functional behaviour (Baresi, Maurino, & Modafferi, 2004).

We are conducting experiments with formal models that allow us to analyze the execution traces in the two cases (i.e., centralized and decentralized execution), but currently our proof is based on the observation that partitioning rules only add activities to synchronize the different sub-workflows, which do not alter the execution flow.

OPTIMAL SERVICE SELECTION AND QUALITY RENEGOTIATION

The goal of this phase is to select, from a service registry, a set of services satisfying requirements from a registry of available services at run time. Usually, a set of functional equivalent services can be selected, that is, services that implement the same functionality but differ in their quality parameters (Bianchini, DeAntonellis, Pernici, & Plebani, 2006). Therefore, service selection introduces an optimization problem. In the work presented by Zeng, Benatallah, Dumas, Kalagnanam, and Chang (2004), two main approaches have been proposed: local and global optimization. The former selects the best candidate service at run time that supports the execution of a running high-level activity. The latter identifies the set of candidate services that satisfy end-user preferences for an entire application. The two approaches allow the specification of quality of service (QoS) constraints at a local and global level, respectively. A local constraint allows the selection of a service according to a required characteristic. For example, a service can be selected so that its price or its execution time is lower than a given threshold. Global constraints are constraints on the overall execution of a set of services constituting an application, that is, constraints such as “the overall execution time of the application has to be less than 3 seconds” or “the total price has to be less than \$2.”

Note that the end-user is mainly interested in global constraints. For example, he is typically concerned with the total execution time of the application rather than the execution time of individual activities. Furthermore, service composition could be transparent to the end-user (i.e., he cannot distinguish between simple and complex services); hence, global constraints must be supported and guaranteed. In the MAIS methodology, we have implemented a global approach for service selection and optimization. The problem of service composition with QoS constraints has been modeled as a mixed integer linear programming (MILP) problem. The problem is NP-hard, since it is equivalent to a multiple choice multiple dimension knapsack problem (see Ardagna & Pernici [2005]; Ardagna et al. [2004]; Wolsey, [1998]). The optimization problem is solved by formulating a mixed integer linear programming model that is solved with CPLEX, a state of the art commercial solver, that implements a branch-and-cut technique.

The quality values negotiated with a provider are the parameters of the optimization problem; parameters are subject to variability, and this is the main issue for the fulfillment of global constraints. The variability of quality values is mainly due to the high variability of the workload of Internet applications, which implies a variability of service performance.

In the MAIS methodology, negotiation, service selection, optimization, and service execution are interleaved. Re-optimization is performed periodically, if the end-user changes the service channel, and if a service invocation fails. The re-optimization period is adapted dynamically to environmental changes (i.e., the time interval decreases if new

candidate services become available). Furthermore, if the end-user changes its channel, for example, switches from a PC to a PDA, the re-optimization is performed since he can expect a higher delay from a wireless connection with restricted bandwidth and could be more focused on the price of the service than on its performance. Finally, if a service fails, a different service will be invoked at run time to replace the faulty one; this may lead to a global constraint violation and, hence, the re-optimization has to be performed. It has to be noticed that in specific cases the optimization problem may become unfeasible, that is, a set of concrete services that meet user requirements, expressed in terms of global constraints, may not be found. In these cases, the MAIS brokering functionalities described in the previous section become useful, since the MAIS designer may obtain services with improved QoS that, if chosen in the optimization process, make the complex service satisfy both local and global constraints. Services QoS may also be negotiated directly by the MAIS architecture at run time, without the need for the designer to exploit MAIS brokering functionalities. The strategies that characterize the MAIS Reflective Architecture, described in the next section, are a typical example of how services QoS can be negotiated at run time directly by the MAIS architecture.

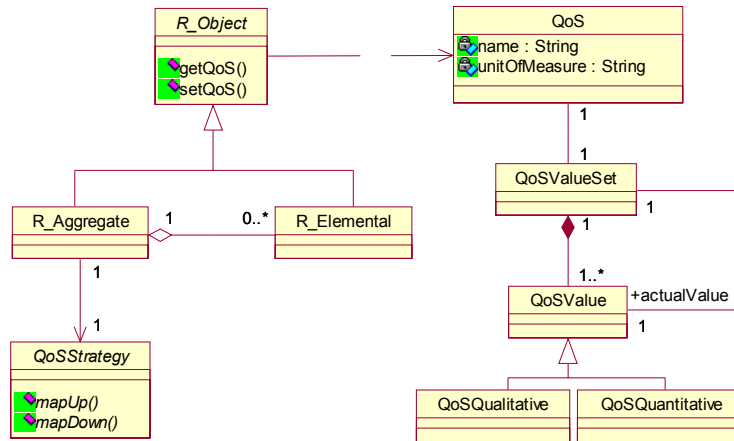
In order to evaluate the effectiveness of our approach, we have compared our solutions with the solutions provided by a local optimization approach proposed by Zeng et al. (2004). For every test case, we first run the local optimization algorithm. Then we perform our global optimization including, as global constraints, the values of the quality dimensions obtained from the local optimization. Ardagna and Pernici (2005) have shown that the global optimization provides better results, since bounds for quality dimensions can be always guaranteed and the value of the quality dimensions can be improved by 10 to 70%.

IMPLEMENTATION GUIDELINES FOR A QoS-ORIENTED REFLECTIVE ARCHITECTURE

The methodological framework for the definition of adaptive services introduced in the previous sections is supported by an underlying reflective architecture. Generally, services rely on a logical layer (e.g., OS and middleware) exploiting functional features of the system components (e.g., devices and network services). Architectural reflection (see Cazzola, Savigni, Sosio, & Tisato [1999] and Maes [1987]) introduces a reflective layer allowing applications to observe and control at non-functional features of the system components execution time, thus supporting adaptability. A reflective layer is causally connected to the physical layer. In Adorni, Arcelli, Raibulet, Sarini, and Tisato (2004), Arcelli, Raibulet, Tisato, and Adorni, (2004), and Tisato et al. (2004), it is reported how the reflective architecture models — via reflective objects (*R_Objects*) — the quality of services (QoS) of the system components (see Chalmers & Sloman, 1999).

Neither components nor their QoS can be defined in an absolute way. For example, an application may observe only the maximum screen resolution of an end-user channel in terms of qualitative, domain-dependent QoS (e.g., low, medium, high). Another application may observe and/or control both specific devices (e.g., a desktop monitor, a wall

Figure 3. QoS extension pattern



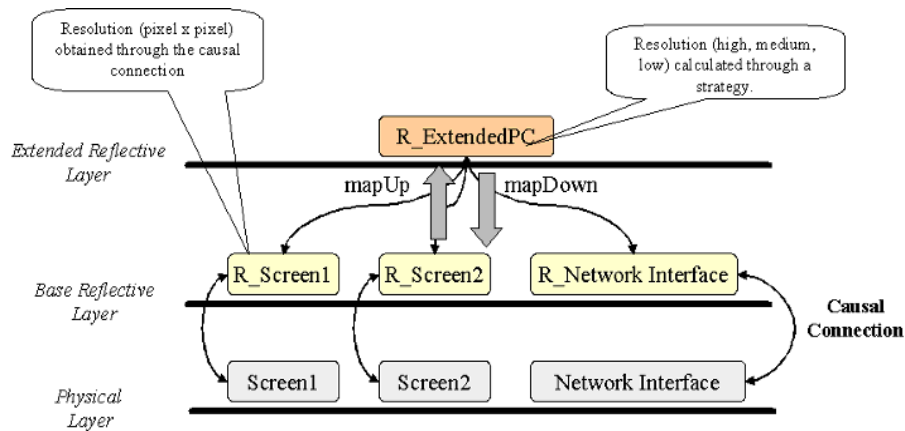
screen, and a projector) and their *pixel x pixel* resolution. Therefore, a general mechanism for defining *R_Objects* and their QoS according to domain requirements is needed.

The QoS extension pattern in Figure 3 highlights that an *R_Aggregate* is a reflective object whose QoS is causally connected via a *QoSStrategy* to the QoS of a collection of *R_Elemental* reflected objects. The *mapUp()* method of the *QoSStrategy* defines how QoS of an aggregate is obtained by exploiting the QoS of its elemental components. The *mapDown()* method defines how the QoS of an aggregate is mapped onto the QoS of its elemental components. Figure 4 shows how the general extension pattern fits into the reflective architecture. *R_Objects* at the *Base Reflective Layer* are causally connected to the physical layer components. They expose measurable QoS values that can be observed and/or controlled via platform-dependent mechanisms.

R_Objects at the *Extended Reflective Layer* model higher level, domain-oriented abstractions. For example, the maximum resolution of a laptop is computed as the maximum resolution of all the display components to which it is connected (e.g., wall monitor, desktop, hands-on device monitor). The bandwidth of the extended network service can be controlled by selecting one of several service providers.

The QoS of an aggregate can be expressed in the same measurement unit as the QoS of its elementals (for instance, the resolution of the laptop is still expressed in terms of *pixel x pixel*). Alternatively, the QoS of an aggregate can be expressed at a higher abstraction level (for instance, it is expressed as low, medium, high) according to domain-specific requirements. In both cases, the general extension pattern is exploited via QoS strategies that define the mapping among QoS with different semantics. For example, the bandwidth of an aggregate network service is computed as the average of the bandwidths of several elemental network services over a given period of time. Though the measurement unit (e.g., *bit·s*) is the same, the semantics are different.

Figure 4. MAIS reflective layers exploiting QoS strategies



The role of QoS strategies is twofold. From the methodology point of view, they allow abstract QoS to be operationally specified in terms of more concrete QoS according to specific domain requirements. From the implementation point of view, strategies turn into pieces of software (i.e., classes) that can be plugged into the system to reify the QoS abstractions.

CONCLUSION

This chapter discusses the MAIS methodological framework supporting the most relevant phases of the design of a value-added service that is a functionality of an adaptive and multichannel information system obtained by composing services offered by different providers. The discussion has focused on the following phases of the life cycle of Web services: requirements analysis, design, deployment, run-time use and negotiation. Current work is focusing on the use of specific requirement techniques to elicit user requirements and usage scenario (Bolchini & Mylopoulos, 2003) and to extend our proposal to include other contributions of the MAIS project, such as design and deployment of context-aware data-intensive web applications (Ceri, Fraternali, & Bongio, 2000), techniques for evaluating the usability of interfaces (Bertini et al., 2005), and tools for adaptive interfaces (Torlone & Ciaccia, 2003).

Future work will consider more complex negotiation scenarios, where multiple providers publish the same type of service simultaneously and require multiparty and multi-attribute negotiation protocols to be modelled, such as combinatorial auctions. In future work, the information obtained from the service selection optimization problem solution will be exploited, also in a run-time negotiation process, in order to further improve the broker's revenue and the end-user's satisfaction. Future work on the

deployment phase will focus on the complete demonstration that our partitioning rules do not alter the execution flow. We are also working on analyzing the transactional behaviour of partitioned sub-processes.

ACKNOWLEDGMENT

This work has been supported by the Italian MIUR-FIRB Project MAIS. The authors acknowledge the contribution of all MAIS participants to this work in many discussions at project meetings.

REFERENCES

- Abhijit, A., Patil, A., Swapna, A., Oundhakar, A., Sheth, A. P., & Verma, K. (2004, May 17-20). Meteor-s Web service annotation framework. In S. I. Feldman, M. Uretsky, M. Najork, & C. E. Wills (Eds.), *Proceedings of the 13th International Conference on World Wide Web, WWW 2004*, New York (pp. 553-562). New York: ACM Press.
- Adorni, M., Arcelli, F., Raibulet, C., Sarini, M., & Tisato, F. (2004, June 21-24). Designing an architecture for multichannel adaptive information systems. In H. R. Arabnia & H. Reza (Eds.), *Proceedings of the International Conference on Software Engineering Research and Practice (SERP '04)*, Las Vegas, NV (Vol. 2, pp. 652-658). Las Vegas, NV: CSREA Press.
- Arcelli, F., Raibulet, C., Tisato, F., & Adorni, M. (2004, June 20-24). Architectural reflection in adaptive systems. In F. Maurer & G. Ruhe (Eds.), *Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering (SEKE2004)*, Banff, Alberta, Canada (pp. 74-79).
- Ardagna, D., Batini, C., Comerio, M., Comuzzi, M., De Paoli, F., Grega, S., et al. (2004). Negotiation protocols definition (Tech. Rep. No. R2.2.2.). *MAIS*. Retrieved July 1, 2005, from <http://www.mais-project.it>
- Ardagna, D., & Pernici, B. (2005, September 5). Global and local QoS guarantee in Web service selection. In C. Bussler & A. Haller (Eds.), *Business Process Management Workshops: BPM 2005 International Workshops, BPI, BPD, ENEI, BPRM, WSCOBPM, BPS*, Nancy, France (Revised selected papers, pp. 32-46). Berlin: Springer.
- Baïna, K., Benattallah, B., Casati, F., & Toumani, F. (2004, June 7-11). Model-driven Web service development. In A. Persson & J. Stirna (Eds.), *Advanced Information Systems Engineering, Proceedings of the 16th International Conference, CAiSE 2004*, Riga, Latvia (LNCS 3084, pp. 290-306). Berlin: Springer.
- Baresi, L., & Heckel, R. (2002, October 7-12). Tutorial introduction to graph transformation: A software engineering perspective. In A. Corradini, H. Ehrig, H. Kreowski, & G. Rozenberg (Eds.), *Graph Transformation, Proceedings of First International Conference, ICGT 2002*, Barcelona, Spain (LNCS 2505, pp. 202-229). Berlin: Springer.
- Baresi, L., Maurino, A., & Modafferi, S. (2004, September 15-17). Workflow partitioning in mobile information systems. In E. Lawrence, B. Pernici, & J. Krogstie (Eds.), *Mobile*

- Information Systems, Proceedings of IFIP TC 8 Working Conference on Mobile Information Systems (MOBIS)*, Oslo, Norway (pp. 93-106). Laxenburg, AU: IFIP.
- Bertini, E., Billi, M., Burzagli, L., Catarci, T., Gabbanini, F., Graziani, P., et al. (2005). Evaluation of the usability and accessibility channels, devices, and users (Tech. Rep. No. R7.3.5.). *MAIS*. Retrieved July 1, 2005, from <http://www.mais-project.it>
- Bertini, E., & Santucci, G. (2004, May 25-28). Modeling Internet-based applications for designing multi-device adaptive interfaces. In M. F. Costabile (Ed.), *Proceedings of the Working Conference on Advanced Visual Interfaces, AVI 2004*, Gallipoli, Italy (pp. 252-256). New York: ACM Press.
- Bianchini, D., De Antonellis, V., Pernici, B., & Plebani P. (2006). Ontology-based methodology for e-Service discovery. *Information Systems*, 31(4-5), 361-380.
- Bolchini, D., & Mylopoulos, J. (2003, December 10-12). From task-oriented to goal-oriented Web requirements analysis. In *Proceedings of the 4th International Conference on Web Information Systems Engineering (WISE 2003)*, Rome, Italy (pp. 166-175). Los Alamitos, CA: IEEE Computer Society.
- Cardoso, J., Sheth, A., Miller, J., Arnold, J., & Kochut, K. (2004). Modeling quality of service for workflows and Web service processes. *Web Semantics Journal*, 1(3), 281-308.
- Cappiello, C., Missier, P., Pernici, B., Plebani, P., & Batini, C. (2004, July 28-30). QoS in multichannel IS: The MAIS approach. In M. Matera, & S. Comai (Eds.), *Engineering Advanced Web Applications: Proceedings of Workshops in connection with the 4th International Conference on Web Engineering (ICWE 2004)*, Munich, Germany (pp. 255-268). Princeton, NJ: Printon Press.
- Casati, F., & Shan, M. (2001). Dynamic and adaptive composition of e-services. *Information Systems*, 6(3), 143-162.
- Cazzola, W., Savigni, A., Sosio, A., & Tisato, F. (1999, October 12-15). Rule-based strategic reflection: Observing and modifying behaviour at the architectural level. In *Proceedings of the 14th IEEE International Conference on Automated Software Engineering (ASE'99)*, Cocoa Beach, FL (pp. 263-266). Los Alamitos, CA: IEEE Computer Society.
- Ceri, S., Fraternali, P., & Bongio, A. (2000, May 15-19). Web modeling language (WebML): A modeling language for designing Web sites. In *Proceedings of the 9th International World Wide Web Conference, WWW 2000*, Amsterdam, The Netherlands. Retrieved July 1, 2005, from <http://www9.org/w9cdrom/177/177.html>
- Chalmers, D., & Sloman, M. (1999). A survey of quality of service in mobile computing environments. *IEEE Communications Surveys*, 2(2), 2-10.
- Colombo, E., Francalanci, C., & Pernici, B. (2004). Modeling cooperation in virtual districts: A methodology for e-service design. *International Journal of Cooperative Information Systems*, 13(4), 369-411.
- Comerio, M., De Paoli, F., De Francesco, C., Di Pasquale, A., Grega, S., & Batini, C. (2004a, June 21-23). A re-design methodology for multi-channel applications in the zootechnical domain. In M. Agosti, N. Dessì, & F. A. Schreiber (Eds.), *Proceedings of the 12th Italian Symposium on Advanced Database Systems, SEBD 2004*, S. Margherita di Pula, Cagliari, Italy (pp. 178-189). Italy: Rubettino Editore.
- Comerio, M., De Paoli, F., Grega, S., Batini, C., Di Francesco, C., & Di Pasquale, A. (2004b, November 15-19). A service re-design methodology for multi-channel adaptation.

- In M. Aiello, M. Aoyama, F. Curbera, & M. P. Papazoglou (Eds.), *Service-oriented computing - Proceedings of ICSOC 2004, 2nd International Conference*, New York (pp. 11-20). New York: ACM Press.
- Gardner, T., Griffin, C., & Iyengar, S. (2003). Draft UML 1.4 profile for automated business processes with a mapping to the BPEL 1.0. *IBM alphaWorks*. Retrieved July 1, 2005, from <http://www-128.ibm.com/developerworks/rational/library/4593.html>
- Glover, F. W., & Kochenberger, G. A. (2003). *Handbook of metaheuristics*. Heidelberg: Springer-Verlag.
- Gomez-Perez, A., Gonzalez-Cabero, R., & Lama, M. (2004, March 22-24). *A framework for design and composition of semantic Web services*. Paper presented at the AAAI Spring Symposia Series 2004. Stanford University, Palo Alto, CA. Retrieved July 1, 2005, from <http://www.daml.ecs.soton.ac.uk/SSS-SWS04/44.pdf>
- Grønmo, R., & Solheim, I. (2004). Towards modelling Web service composition in UML. In S. Bevinakoppa & J. Hu (Eds.), *Web services: Modeling, architecture and infrastructure — Proceedings of the 2nd International Workshop on Web Services: Modeling, Architecture and Infrastructure (WSMAI 2004), in conjunction with ICEIS 2004*. Porto, Portugal (pp. 72-86). Setubal, Portugal: INSTICC Press.
- Hausmann, J. H., Heckel, R., & Taentzer, G. (2002, May 19-25). Detection of conflicting functional requirements in a use case-driven approach: A static analysis technique based on graph transformation. In *Proceedings of the 22nd International Conference on Software Engineering, ICSE 2002*, Orlando, FL (pp. 105-115). New York: ACM Press.
- Jaeger, M. C., Rojec-Goldmann, G., & Muhl, G. (2004, September 20-24). QoS aggregation for Web service composition using workflow patterns. In *Proceedings of 8th International Enterprise Distributed Object Computing Conference (EDOC 2004)*, Monterey, CA (pp. 149-159). Los Alamitos, CA: IEEE Computer Society.
- Jennings, N. R., Lomuscio, A. R., Parsons, S., Sierra, C., & Wooldridge, M. (2001). Automated negotiation: Prospects, methods, and challenges. *Group Decision and Negotiation*, 10(2), 199-215.
- Jennings, N. R., Luo, X., & Shadbot, N. (2003). Knowledge-based acquisition of trade-off preferences for negotiating agents. In *Proceedings of the 5th International Conference on Electronic Commerce, ICEC'03*, Pittsburgh, PA (pp. 138-144). New York: ACM Press.
- Kaabi, R. S., Souveyet, C., & Rolland, C. (2004, November 15-19). Eliciting service composition in a goal driven manner. In M. Aiello, M. Aoyama, F. Curbera, & M. P. Papazoglou (Eds.), *Service-oriented computing — Proceedings of ICSOC 2004, 2nd International Conference*, New York (pp. 308-315). New York: ACM Press.
- Lazovik, A., Aiello, M., & Papazoglou, M.P. (2003, December 15-18). Planning and monitoring the execution of Web service requests. In M.E. Orlowska, S. Weerawarana, M. P. Papazoglou, & J. Yang (Eds.), *Service-Oriented Computing, Proceedings of ICSOC 2003, 1st International Conference*, Trento, Italy (LNCS 2910, pp. 335-350). Berlin: Springer.
- Lum, W. Y., & Lau, F. C. M. (2003). User-centric content negotiation for effective adaptation service in mobile computing. *IEEE Transaction on Software Engineering*, 29(12), 1000-1111.
- Maes, P. (1987, October 4-8). Concepts and experiments in computational reflection. In N. K. Meyrowitz (Ed.), *Proceedings of Conference on Object-Oriented Program-*

- ming Systems, Languages, and Applications (OOPSLA '87)*, Orlando, FL. *SIGPLAN Notices*, 22(12), 147-155.
- Mecella, M., Parisi-Presicce, F., & Pernici, B. (2002, August 23-24). Modeling e-service orchestration through Petri nets. In A. P. Buchmann, F. Casati, L. Fiege, M. Hsu, & M. Shan (Eds.), *Technologies for E-Services, Proceedings of the 3rd International Workshop (TES 2002)*, Hong Kong, China (LNCS 2444, pp. 38-47). Berlin: Springer.
- Menasce, D. (2004). Composing Web services: A QoS view. *IEEE Internet Computing*, 8(6), 80-90.
- Mylopoulos, J., & Lau, D. (2004, June 6-9). Designing Web services with Tropos. In *Proceedings of the IEEE International Conference on Web Services (ICWS '04)*, San Diego, CA (pp. 306-316). Los Alamitos, CA: IEEE Computer Society.
- Pahl, C., & Casey, M. (2003, September 1-5). Ontology support for Web service processes. In *Proceedings of the 11th ACM SIGSOFT Symposium on Foundations of Software Engineering 2003 held jointly with 9th European Software Engineering Conference, ESEC/FSE 2003*, Helsinki, Finland (pp. 208-216). New York: ACM Press.
- Shuping, R. (2003, June 23-26). A framework for discovering Web services with desired quality of services attributes. In L. Zhang (Ed.), *Proceedings of the International Conference on Web Services (ICWS '03)*, Las Vegas, NV (pp. 208-213). Las Vegas, NV: CSREA Press.
- Siegel, J., & The OMG Staff Strategy Group (2001). Developing in OMG's model-driven architecture. *OMG document*.
- Skogan, D., Grønmo, R., & Solheim, I. (2004, September 20-24). Web service composition in UML. In *Proceedings of the 8th International Enterprise Distributed Object Computing Conference (EDOC 2004)*, Monterey, CA (pp. 47-57). Los Alamitos, CA: IEEE Computer Society.
- Sun, C., Raje, R. R., Olson, A. M., Bryant, B. R., Burt, C., Huang, Z., & Auguston, M. (2002). Composition and decomposition of quality of service parameters in distributed component-based systems. In *Algorithms and Architectures for Parallel Processing Proceedings of Ica3Pp 2002, 5th International Conference* (pp. 273-277). Los Alamitos, CA: IEEE Computer Society.
- Tisato, F., Adorni, M., Arcelli, F., Campanini, S., Limonta, A., Melen, R., Raibulet, C., & Simeoni, M. (2004). The MAIS reflective architecture (Tech. Rep. No. R3.1.1.). *MAIS*. Retrieved July 1, 2005, from <http://www.mais-project.it>
- Torlone, R., & Ciaccia, P. (2003). Management of user preferences in data intensive applications. In S. Flesca, S. Greco, D. Saccà, & E. Zuppano (Eds.), *Proceedings of the 11th Italian Symposium on Advanced Database Systems, SEBD 2003*, Cetraro (CS), Italy (pp. 257-268). Soveria Mannelli, Italy: Rubettino Editore.
- Ulbrich, A., Weis, T., & Geihs, K. (2003, May 19-22). QoS mechanism composition at design-time and runtime. In *Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops (ICDCS 2003 Workshops)*, Providence, RI (pp. 118-126). Los Alamitos, CA: IEEE Computer Society.
- Wolsey, L. (1998). *Integer programming*. New York: John Wiley & Sons.
- Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., & Chang H. (2004). QoS-aware middleware for Web services composition. *IEEE Transactions on Software Engineering*, 30(11), 315-327.

Chapter XII

Toward Autonomic DBMSs: A Self-Configuring Algorithm for DBMS Buffer Pools

Patrick Martin, Queen's University, Canada

Wendy Powley, Queen's University, Canada

Min Zheng, Queen's University, Canada

ABSTRACT

This chapter introduces autonomic computing as a means to automate the complex tuning, configuration, and optimization tasks that are currently the responsibility of the database administrator. We describe an algorithm called the dynamic reconfiguration algorithm (DRF) that can be implemented as part of an autonomic database management system (DBMS) to manage the DBMS buffer pools, which are a key resource in a DBMS. DRF is an iterative algorithm that uses greedy heuristics to find a reallocation that benefits a target transaction class. DRF uses the principle of goal-oriented resource management. We define and motivate the cost-estimate equations used in the algorithm and present the results of a set of experiments to investigate the performance of the algorithm.

INTRODUCTION

The explosion of the Internet and electronic commerce in recent years has database management system (DBMS) vendors scrambling to cope not only with the ever-increasing volumes of data to be managed, but also with the unique requirements precipitated by diverse data and unpredictable, often “bursty” access patterns. The addition of new features and functionality to DBMSs to address these issues has led to increased system complexity. The management of DBMSs has traditionally been left to the experts — the database administrators — who monitor, analyze, and tweak the system for optimal performance. Given the increased complexity of DBMSs and the diverse and integrated environments in which they currently function, manual maintenance and tuning has become impractical, if not impossible.

DBMS parameter tuning is just one facet of tuning a database system, yet even this task has become a burden due to its complexity. Commercial database management systems typically provide upwards of 100 parameters that can be manually tuned. These parameters are often interconnected, so tuning one parameter may require an adjustment of one or more dependent resources. Determining optimal settings for tuning parameters requires knowledge of the characteristics of the system, the data, the workload, and of the interrelationships between them. These optimal settings often deteriorate over time, as the database characteristics change, or periodically, as the workload changes. With the varying and unpredictable patterns of electronic commerce workloads, changes in the workload tend to be more frequent and more extreme than those observed in traditional business environments. It is impractical for a database administrator to constantly monitor and tune the DBMS to adapt to these dynamic workloads. Instead, the system itself should be able to recognize or, where possible, predict workload changes, evaluate the benefit of reconfiguration, and independently take appropriate action.

Autonomic computing is an initiative spawned by IBM in 2001 to address the management problems associated with complex systems (Ganek & Corbi, 2003). IBM’s use of the term “autonomic” is a direct analogy to the autonomic nervous system of the human body. The autonomic nervous system unconsciously regulates the body’s low-level vital functions such as heart rate and breathing. The vision is to create computer systems that function much like the autonomic nervous system, that is, the low-level functionality and management of the system is attended to without conscious effort or human intervention. The goal is that system management will become the sole responsibility of the system itself.

In this chapter, we illustrate how the concept of autonomic computing can be applied to automate one aspect of DBMS tuning, that is, the tuning of the DBMS buffer pools. We present a self-tuning algorithm called the dynamic reconfiguration algorithm (DRF). This algorithm is based on the concept of goal-oriented resource management, which allows administrators to specify their expectations or goals for performance, while leaving it up to the system to decide how to achieve those goals (Nikolaou, Ferguson, & Constantopoulos, 1992). The database administrator (DBA) specifies average response-time goals for each transaction class in the workload. If one or more classes are not meeting their goals, then DRF chooses a reallocation of buffer pages to buffer pools that will improve the performance of the classes so that their goals can be met.

DRF is an iterative algorithm. It uses greedy heuristics to find a reallocation that benefits a target transaction class, namely, the class with the worst performance. Each iteration of DRF reallocates a number of pages from one buffer pool to another. The source and target buffer pools of a reallocation are chosen such that the benefit to the target transaction class is maximized. The benefit of a reallocation to a transaction class is the estimated effect that a shift of pages from the source buffer pool to the target buffer pool has on the average response time of that class. Adding pages to a buffer pool can increase the hit rate of the buffer pool, which is the proportion of times that block requests are satisfied by pages in the buffer pool. The increased hit rate in turn reduces the response time of transactions using that buffer pool, since there are, on average, fewer accesses to the disk. DRF has been implemented and tested with DB2 Universal Database (DB2) (IBM, 2004). We show the results of a set of experiments using DRF to tune the buffer pools in DB2 for a workload consisting of the TPC-C benchmark (Leutenegger & Dias, 1993; Transaction Processing Performance Council, 2004).

BACKGROUND

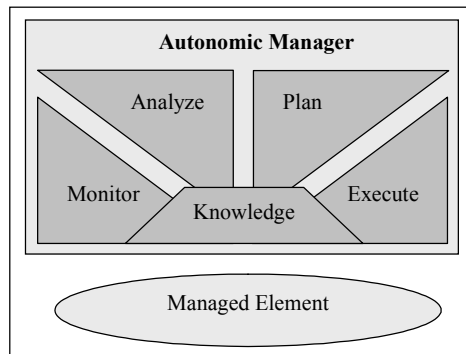
The work in this chapter relates to research in two main areas, namely, autonomic computing and buffer pool tuning. We outline the main concepts in Autonomic Computing and then examine previous work in the area of buffer pool tuning.

Autonomic Computing

Autonomic computing systems are intelligent systems that are capable of adapting to a changing environment. Ganek and Corbi (2003) identify the following four fundamental features of autonomic systems:

- **Self-configuring:** new features, software and hardware, can be dynamically added to the infrastructure with no disruption of service. A system should not only be able to configure itself on the fly, but it should also be able to configure itself to adapt to a new environment into which it is introduced.
- **Self-healing:** the system must be able to minimize outages, thus predicting and avoiding failures and/or recovering quickly from unavoidable failures.
- **Self-optimizing:** the system must be able to efficiently maximize resource utilization to meet end-user requirements without human intervention.
- **Self-protecting:** autonomic systems must be able to protect against unauthorized access, to detect and protect against intrusions, and to provide secure backup and recovery capabilities.

According to Ganek and Corbi (2003), the implementation of autonomic features into computing systems will be a gradual process, progressing from systems that are manually managed to fully autonomic integrated components with IT management driven by business policies. During this evolution, features and functionality will be added to systems to gradually shift the responsibility for management from the human expert to the system itself resulting in a self-managing system that requires little to no human intervention. They identify the following five levels in the evolution of autonomic computing:

Figure 1. Autonomic element

- **Basic:** manual management.
- **Managed:** tools and technologies are provided that are used to collect and consolidate information from various sources, thus reducing this burden for the system administrator.
- **Predictive:** the system itself begins to recognize patterns, predict the optimal configuration, and provide advice to the system administrator who then uses this information to determine the best course of action.
- **Adaptive:** the system begins to independently take corrective action.
- **Autonomic:** systems are governed by business policies and objectives and are solely responsible for all management aspects.

Current DBMSs fall in the managed level of autonomic computing capabilities. Most vendors supply sophisticated tools to assist the DBA in monitoring and analyzing the system performance, but they provide little in the way of autonomic management (Elnaffar, Powley, Benoit, & Martin, 2003). Although there is some degree of automation, the vast majority of tuning and optimization decisions still require expert knowledge and human intervention.

Autonomic features are typically implemented as a feedback control loop controlled by an autonomic manager, as shown in Figure 1 (Kephart & Chess, 2003). The autonomic manager oversees the monitoring of the system, and by analyzing the collected statistics in light of known policies and/or goals, it determines whether or not the performance is adequate. If necessary, a plan for reconfiguration is generated and executed.

The idea of self-tuning DBMSs commenced prior to IBM's autonomic computing initiative. Self-tuning and adaptive techniques have been applied to several aspects of the management problem, including index selection (Chaudhuri, Christensen, Graeffe, Narasayya, & Zwillig, 1999; Schiefer & Valentin, 1999), materialized view selection (Agrawal, Chaudhuri, & Narasayya, 2000), distributed join optimization (Arcangeli,

Hameurlain, Migeon, & Morvan, 2004; Khan, McLeod, & Shahabi, 2001), and memory management (Brown, Carey, & Livny, 1993, 1996; Chung, Ferguson, Wang, Nikolaou, & Teng, 1995; Sinnwell & König, 1999).

Buffer Pool Tuning

The buffer area used by a DBMS is particularly important to system performance because effective use of the buffers can reduce the number of disk accesses performed by a transaction. Current DBMSs, such as DB2 Universal Database (IBM, 2004), divide the buffer area into a number of independent buffer pools, and database objects (tables and indices) are assigned to a specific buffer pool. The size of each buffer pool is set by configuration parameters, and page replacement is local to each buffer pool. Tuning the size of the buffer pools to a workload is therefore crucial to achieving good performance.

For example, suppose we initially configure a database to have four buffer pools of equal size where indices are assigned to one buffer pool, and tables are assigned to the other three buffer pools. If the index buffer pool is too small to hold at least all the non-leaf pages of the active indices, then there will be excessive swapping on the index buffer pool and performance will be poor for many of the queries. A more appropriate use of buffer pages is to give more pages to the index buffer pool so that index pages can remain in memory, and to take away pages from the table buffer pools where pages are reused less often.

Past research in the area of DBMS caches has focused on buffer management techniques (Effelsberg, & Härder, 1984; Chou & DeWitt, 1985) and page replacement algorithms (Faloutsos, Ng, & Sellis, 1991; O'Neil, O'Neil, & Weikum, 1993) to optimize the performance of the buffer cache.

To the best of our knowledge, three previous goal-oriented buffer tuning algorithms have appeared in the literature: dynamic tuning (Chung et al., 1995), fragment fencing (Brown et al., 1993), and class fencing (Brown et al., 1996). These algorithms can be compared based on a number of factors including performance goals, the hit-rate estimator, support of data sharing, the underlying buffer pool model, and the validation method.

Performance Goals

Dynamic tuning differs from the other goal-oriented algorithms with respect to how its response time goals are defined. Fragment fencing, class fencing, and DRF all define average response-time goals for overall transaction response time. Dynamic tuning, on the other hand, assumes that a transaction class is mapped to a single buffer, which means that transaction class response times are directly proportional to buffer access times. Thus, dynamic tuning can specify goals for low-level read/write requests for each buffer pool.

Fragment fencing, class fencing, and DRF assume that response times are directly proportional to the miss rate, and all use equations based on the miss rate to estimate transaction response time. The equation we use in DRF tries to use more information than the other approaches in producing its estimate. Specifically, we try to account for the proportion of dirty pages in the buffer at any one time and the effect of asynchronous reads and writes.

Hit-Rate Estimator

Dynamic tuning uses an equation from Belady's virtual memory study (Belady, 1966) that models hit rate as a function of memory allocation. The computation is specific to a workload and requires two observation points. Brown et al. (1996) observe that Belady's equation is not a good fit to every hit-rate curve.

Fragment fencing's goal is to determine the minimum number of pages required for each fragment to ensure that a class meets its response-time goal, which is called the fragment's target residency. Brown et al. (1996) define a fragment to be a set of pages with the same access frequency. The hit-rate estimator determines a target residency for each fragment referenced by a class.

Class fencing uses the concept of hit-rate concavity as its hit-rate estimator. The concavity theorem states that the slope of the hit-rate curve never increases as more memory is added to the optimal buffer replacement policy. This enables a simple straight-line approximation to be used to predict the memory required for a particular hit rate. Two observation points are required to produce the straight-line approximation. Brown et al. (1996) claim that hit-rate concavity provides a more general solution than Belady's equation, and allows class fencing's hit-rate estimator to aggressively allocate memory in large increments because there is no danger of "overshooting" hit-rate targets.

Data Sharing

Dynamic tuning assigns a buffer pool to each transaction class. Allocation decisions are based on the value of a performance index for each buffer pool, which is defined as the ratio of the actual response time to the goal response time. A performance index value greater than one implies that a class is not meeting its goal. It attempts to minimize the maximum performance index and balance the performance index values of the buffer pools. Buffer pages are taken from those buffer pools with the minimal performance indices and given to the buffer pool with the maximum performance index. A shortcoming of this approach is that it does not consider classes that share data pages.

In fragment fencing, when a performance goal is violated and the hit rate of a class needs to be increased, the algorithm sorts the fragments referenced by a class in order of decreasing class temperature, which is the size-normalized access frequency for a class. The target residencies for the "hottest" classes are increased until the desired hit rate is achieved. Fragment fencing uses a passive allocation method, which does not explicitly transfer pages from one buffer pool to another but only prevents their ejection from the pool by the DBMS replacement mechanism. This passive method will converge more slowly to the desired buffer allocations than approaches with an active allocation method, such as our approach. Fragment fencing, like dynamic tuning, does not consider classes that share pages.

Class fencing allocates memory by building a single fence around all pages used by a class, regardless of the fragment to which they belong. There is a local buffer manager for each class and a global buffer manager for pages of classes without a goal and any less valuable "unfenced" pages of classes with a goal. Classes remain under the control of the global buffer manager as long as they can achieve their goals. Once a class violates its goal, it is given its own buffer pool and local buffer manager. The goal of class

Table 1. Comparison of buffer pool tuning algorithms

Algorithm	Performance Goal	Hit Rate Estimator	Data Sharing	Buffer Pool Model	Validation
Dynamic Tuning	Average Read/write Times	Belady's Equation	No	Transaction Oriented	Simulation
Fragment Fencing	Average Response Time	Target Residency	No	Transaction Oriented	Simulation
Class Fencing	Average Response Time	Concavity Theorem	Yes	Transaction Oriented	Simulation
DRF	Average Response Time	Least Squares Approximation	Yes	Data Oriented	Experimental

fencing is to determine a buffer pool size so that a class can meet its goal. If the buffer pool size for a violating class is increased, then, on a buffer miss, the memory allocation mechanism takes a free page from the global buffer and assigns it to the violating class. Class fencing, like our approach, considers data-page sharing between classes.

Buffer Pool Model

An important difference between DRF and the three previous approaches is the assumed model of buffer pool organization. The previous approaches all use a transaction-oriented model, that is, they assume that buffer pools are organized based on workload classes. DRF, on the other hand, uses a data-oriented model that assumes that buffer pools are organized based on database objects. In our model, the buffer pool pages used by a transaction class are not likely to be in a single buffer pool but instead spread out over several buffer pools. DB2, for example, uses a data-oriented model.

Validation

Another difference between our work and the other research is how the approaches are validated. The other algorithms are analyzed using simulation studies. In this chapter, we present an experimental evaluation of an implementation of DRF for DB2.

Our comparison of the buffer pool tuning algorithms is summarized in Table 1. We conclude that DRF improves upon previous algorithms in several ways:

- DRF uses a more sophisticated response-time estimator that accounts for the effects of dirty buffer pages and asynchronous reads and writes performed by system processes.
- DRF accounts for classes that share data pages. The dynamic tuning and fragment fencing algorithms do not consider shared data pages.
- DRF uses a data-oriented model of buffer organization. Previous algorithms all use a transaction class-oriented model.

SELF-TUNING APPROACH

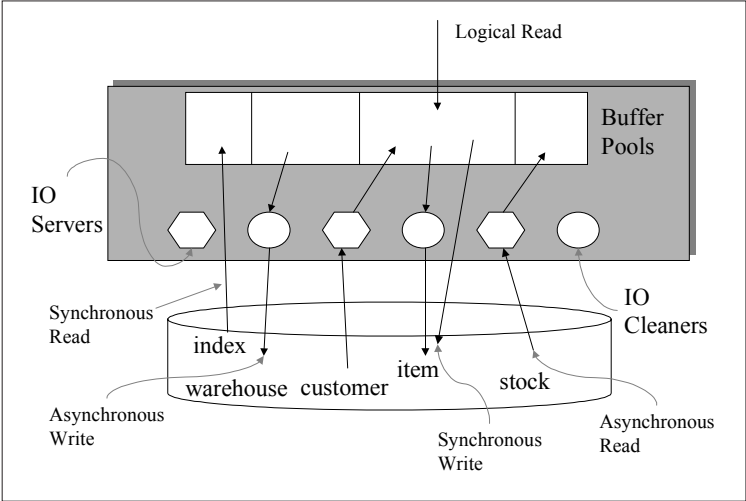
We adopt the feedback control loop approach to autonomic computing as previously shown in Figure 1 (Kephart & Chess, 2003) for our buffer pool tuning algorithm. The feedback loop is composed of three phases — monitor, assess, and reallocate. A DBMS must constantly monitor itself to detect if performance metrics exceed specified thresholds. When a problem is detected, the system must assess the various resource adjustments that can be made to solve the problem. Finally, the DBMS must reallocate its resources to solve the problem.

In explaining our approach to managing the buffer pools, we first provide a model of the buffer pool that is used by our algorithm. We next describe the DRF algorithm, which fits into the assessment phase of the feedback loop. We present the cost models used by DRF to assess the impact of potential reallocations of pages among buffer pools on the average response time of the transaction classes that make up the systems’ workload. The thresholds used in this assessment are response-time goals provided by the DBA.

Buffer Pool Model

We assume the buffer pool model shown in Figure 2. The model is similar to the buffer pool organization used in DB2 (IBM, 2004). Buffer memory is partitioned into a number of independent buffer pools, and database objects (tables and indices) are assigned to specific buffer pools when the system is configured. For example, in Figure 2, indices are assigned to the first buffer pool, the “warehouse” table is assigned to the second buffer pool, the “customer” and “item” tables are assigned to the third buffer pool, and the “stock” table is assigned to the fourth buffer pool. An object’s pages are

Figure 2. Buffer pool model



moved between disk and its designated buffer pool. The size of each buffer pool is set by configuration parameters and page replacement is local to each buffer pool.

An access to a buffer pool by a transaction is called a logical read. A read access to a disk is called a physical read and a write access to a disk is called a physical write. If the page required by the logical read is already in the buffer pool, then the DBMS can satisfy the request immediately. If the required page is not in the buffer pool, then it must be retrieved from the disk, which is called a synchronous read. The proportion of logical reads that require a disk access is called the buffer pool's miss rate. The buffer pool's hit rate is $(1 - \text{miss rate})$. If there are no free clean pages to hold the new page, then a dirty page, that is, a page with updates, must be selected for replacement and written back to disk in order to make room for the new page. This write is called a synchronous write. The DBMS may use background tasks to enhance buffer pool performance by performing asynchronous I/O, which is system-initiated data transfer between disk and the buffer pools. I/O servers are background tasks that prefetch pages into the buffer pools. We say that I/O servers perform asynchronous reads. I/O cleaners are background tasks that write dirty pages back to disk. We say that I/O cleaners perform asynchronous writes.

Dynamic Reconfiguration Algorithm

Our dynamic reconfiguration algorithm aims to provide an allocation of buffer pages to buffer pools such that the performance goals of the transaction classes using the database are met. The DBA provides an average response-time goal for each of the transaction classes in the system workload. We assume that a transaction class is a collection of transactions with the same requirement; that is, they access the same set of data objects and have the same performance goals. For example, in the TPC-C benchmark (Leutenegger & Dias, 1993; Transaction Processing Performance Council, 2004), which typifies an on-line transaction processing (OLTP) application, we can represent each type of transaction with its own class.

DRF compares current performance measurements with the performance goals of each transaction class. If one or more of the transaction classes are not meeting their goals, then DRF attempts to find a reallocation of buffer pages to buffer pools such that all the classes meet their goals. The performance of the DBMS relative to the transaction classes' goals is measured by the Achievement Index for each transaction class T_i , which is given by

$$AI_i = \frac{\text{Goal Average Response Time for } T_i}{\text{Actual Average Response Time for } T_i} \quad (1)$$

If $AI_i < 1$ then class T_i is not achieving its goal. If $AI_i \geq 1$ then class T_i is meeting or exceeding its goal. DRF tries to converge to a situation where each AI_i is close to 1. DRF determines a reallocation of buffer pool pages in favour of the transaction class with the smallest AI . We call this class the target transaction class for the tuning session.

An iteration of DRF reallocates a fixed number of pages from one buffer pool to another. Adding pages to a buffer pool can increase the hit rate of the buffer pool, which in turn reduces the response time of transactions using that buffer pool since there are,

on average, fewer accesses to the disk. The effect of a reallocation is estimated using the cost-estimate equations described below.

The number of pages shifted in an iteration of the algorithm is a parameter to DRF. If a large number of pages are reallocated each time, then the algorithm will converge quickly to a point where the goal is met. We run the risk, however, of overshooting the goal and reallocating too many pages, which can detract unnecessarily from the performance of other classes. If, on the other hand, a small number of pages are reallocated each time, then the algorithm will not overshoot the goal but will converge more slowly to an appropriate configuration. A reallocation unit of 500 pages is used in the experiments described later in the chapter. It is also possible to gradually decrease the number of pages shifted as a means of gaining the benefits of both large and small reallocation sizes, but this is not investigated in the chapter.

The target buffer pool for a reallocation is the buffer pool that, when given more pages, provides the largest performance improvement to the target class. The source buffer pool for a reallocation is the buffer pool that, when relieved of pages, has the smallest negative impact on the performance of the target class. DRF repeats the

Table 2. Symbols used in cost-estimation equations

Symbol	Meaning
TC	Set of transaction classes in a workload.
T_i	Transaction class i .
BP_i	Set of buffer pools used by transaction class T_i .
B_j	Buffer pool $B_j \in BP_i$.
$cpuLR$	Processing cost for a logical read.
$miss_j(m)$	Miss rate for buffer pool j with memory size m pages.
$costLR_j(m)$	Cost of a logical read of buffer pool j with memory size m pages.
$costAW_j(m)$	Amortized cost of asynchronous writes to buffer pool j .
$costAR_j(m)$	Amortized cost of asynchronous reads to buffer pool j .
$pAR_j(m)$	Proportion of asynchronous reads to buffer pool j with memory size m pages.
$pAW_j(m)$	Proportion of asynchronous writes to buffer pool j with memory size m pages.
$pD_j(m)$	Proportion of synchronous writes to buffer pool j with memory size m pages.
$L_i(B_j)$	Number of logical reads of buffer pool j by transaction class T_i .
C_i	Average response time of transaction class T_i .

reallocation exercise until all transaction classes meet their goals or no further improvement in performance can be achieved.

A problem with this approach of selecting target and source buffer pools is the potential for the algorithm to thrash, that is, repeatedly move pages back and forth among the same set of buffer pool in an attempt to meet different goals. We avoid this problem by ensuring that DRF will not choose a source buffer pool that was a target buffer pool in a recent reallocation.

Cost-Estimate Equations

The symbols used in our cost-estimate equations are summarized in Tables 2 and 3. A number of the cost estimates described below use the least squares approximation (Isaacson & Keller, 1966) curve fitting technique to calculate values for arbitrary memory sizes. In these cases, we must collect the performance statistics in Table 2 from the DBMS for three different buffer pool sizes.

An application using the DBMS is characterized by a set of transaction classes $\mathbf{TC} = \{T_1, T_2, \dots, T_n\}$. Instances of a particular transaction class, $T_i \in \mathbf{TC}$, use a subset of the buffer pools, say $\mathbf{BP}_i = \{B_1, B_2, \dots, B_b\}$. The elements of $\mathbf{BP}_i$ are determined by the set of database objects that are used by instances of T_i . The average number of logical reads per instance of T_i on buffer pool $B_j \in \mathbf{BP}_i$ is represented as $L_i(B_j)$.

We assume that the average response time for a transaction class T_i is directly proportional to the average data-access time for instances of the class. The data-access time for a transaction depends upon the number of logical reads issued by that transaction. So an estimate of the average response time per instance of transaction class T_i is given by:

$$C_i = \sum_{j=1}^b L_i(B_j) \times \text{costLR}_j(m) \quad (2)$$

Table 3. Performance statistics collected at data points

Symbol	Meaning
$noLR_j$	Number of logical reads on buffer pool j of size m pages
$noPR_j(m)$	Number of physical reads into buffer pool j of size m pages
$noAR_j(m)$	Number asynchronous reads into buffer pool j of size m pages
$noPW_j(m)$	Number of physical writes from buffer pool j of size m pages
$noAW_j(m)$	Number of asynchronous writes from buffer pool j of size m pages
$costPR_j$	Average cost of physical read into buffer pool j
$costPW_j$	Average cost of physical write into buffer pool j

where $costLR_j(m)$ is the average cost of a logical read from buffer pool B_j of size m pages. We observed in our experiments that, as expected, many of the cost components of a logical read depend upon the size of the buffer pool in use.

We estimate the cost (response time) of a logical read on buffer pool j with m memory pages as follows:

$$costLR_j(m) = cpuLR + costAR_j(m) + costAW_j(m) + (1 - pAR_j(m)) \times miss_j(m) \times (costPR_j + ((1 - pAW_j(m)) \times pD_j(m) \times costPW_j)) \quad (3)$$

The cost of a logical read ($costLR_j(m)$), as indicated by the equation, contains several components. The first component is the processing cost associated with a logical read ($cpuLR$). In this chapter, we assume that the processing cost is not significant and can be set to zero.

The second component of the cost of a logical read is the delay added by IO servers performing asynchronous reads. We estimate the impact of the IO servers by amortizing the cost of all asynchronous reads to a buffer pool across all logical reads ($noLR_j$) as follows:

$$costAR_j(m) = \frac{pAR_j(m) \times noPR_j(m) \times costPR_j}{noLR_j} \quad (4)$$

The cost of a physical read from buffer pool j ($costPR_j$) is estimated as the average of the physical read costs calculated at the initial data collection points. The number of asynchronous reads is dependent upon the buffer pool size and is calculated as a portion of the total number of physical reads ($noPR_j(m)$). The proportion of asynchronous reads to buffer pool j at memory size m is:

$$pAR_j(m) = \frac{noAR_j(m)}{noPR_j(m)} \quad (5)$$

and the ratio is approximated at all values of m by a first-order polynomial and least squares approximation. The number of physical reads of buffer pool j at memory size m is approximated by:

$$noPR_j(m) = miss_j(m) \times noLR_j \quad (6)$$

where the miss rate for buffer pool j at memory size m is:

$$miss_j(m) = \frac{noPR_j(m)}{noLR_j} \quad (7)$$

The miss rate is approximated at all values of m by a second-order polynomial and least squares approximation. We first used Belady's equation to approximate the hit rate, but we found that our current method gives better approximations to miss rate curves in a wide variety of circumstances. Our method requires three observation points, that is, observed miss rates at three different memory sizes. Belady's equation and the concavity theorem used in class fencing each require two observation points. Once a self-tuning algorithm like DRF is integrated into the DBMS, the observation points can be collected as part of the tuning process without significant additional costs.

The third component of the cost of a logical read is the delay caused by IO cleaners performing asynchronous writes. As with the IO servers, we estimate the impact of the IO cleaners by amortizing the cost of all asynchronous writes across all logical reads as follows:

$$\text{costAW}_j(m) = \frac{pAW_j(m) \times \text{noPW}_j(m) \times \text{costPW}_j}{\text{noLR}_j} \quad (8)$$

The cost of a physical write from buffer pool j (costPW_j) is estimated as the average of the physical write costs calculated at the initial data collection points. The number of physical writes of buffer pool j at all memory sizes m ($\text{noPW}_j(m)$) is approximated by a second-order polynomial and least-squares estimation. The number of asynchronous writes is dependent upon the buffer pool size and is calculated as a portion of the total number of physical writes. The proportion of asynchronous writes to buffer pool j at memory size m is:

$$pAW_j(m) = \frac{\text{noAW}_j(m)}{\text{noPW}_j(m)} \quad (9)$$

and the ratio is approximated at all values of m by a first-order polynomial and least squares approximation.

The fourth component of the cost of a logical read, which is given by the factor:

$$(1 - pAR_j(m)) \times \text{miss}_j(m) \times \text{costPR} \quad (10)$$

is the percentage of logical reads that result in a physical read. This percentage is determined by the miss rate of the buffer pool. The IO servers also affect the miss rate of the buffer pool since they prefetch pages into the buffer pool.

The fifth component of the cost of a logical read, which is given by the factor:

$$(1 - pAR_j(m)) \times \text{miss}_j(m) \times ((1 - pAW_j(m)) \times pD_j(m) \times \text{costPW}_j) \quad (11)$$

is the percentage of all logical reads that involve a physical write. In these cases, there are no clean buffer pages available for replacement so a dirty page must be written to disk before the new page can be read.

The possibility of having to write a dirty page from buffer pool j of size m is:

$$pD_j(m) = \frac{noSW_j(m)}{noPR_j(m)} \quad (12)$$

and the ratio is approximated at all values of m by a first-order polynomial and least-squares approximation. IO cleaners increase the probability that a free page is found by asynchronously writing dirty pages back to disk, which is captured by the factor $(1 - pAW_j(m))$ in the equation.

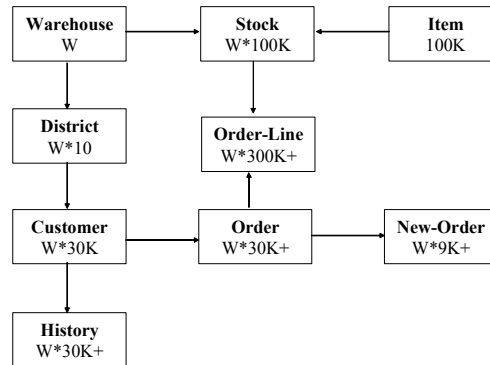
EXPERIMENTS

We now present the results of a set of experiments that used DRF to tune the sizes of the buffer pools of a commercial DBMS running a typical OLTP workload. The objective of the experiments is to show the accuracy and the robustness of DRF under a variety of initial conditions and under a changing workload. The conditions considered in the experiments include the number of transactions classes initially not meeting their goals (one or more than one) and the initial allocation of pages to buffer pools (uniform or skewed).

Experimental Workload

The workload used in the experiments is the TPC-C OLTP benchmark, which simulates a typical order-entry application (Transaction Processing Performance Council, 2004). The database schema from the TPC-C benchmark, which is shown in Figure 3, is composed of nine relations. The size of the database depends primarily on the number of warehouses (W). Each warehouse stocks 100,000 items and services 10 districts. Each district has 3000 customers. Each customer generates at least one order, and each order has 10 to 15 items. Our experimental database consists of 75 warehouses and is approximately 7.5 GBs.

Figure 3. TPC-C entity-relationship schema



TPC-C simulates the activities of a wholesale supplier and includes five order-entry type transactions. The transactions include entering new orders (New Order), delivering orders (Delivery), checking the status of an order (Order Status), recording payments (Payment), and monitoring the level of stock at the warehouses (StockLevel). In our experiments, 40 clients — or simulated “terminal operators” — issue the transactions against the database. The relative frequencies and the operations involved in the different transactions, which are shown in Table 3, are as specified in the benchmark. Each transaction type is considered as a separate class with its own performance goal.

Experimental System

The experiments were run using DB2 Version 5.2 under Windows NT on an IBM PowerServer 704. The machine was configured with one 200 MHz Pentium Pro processor, 1 GB of RAM, and 16 8.47 GB SCSI disks. DB2 was configured with a disk page size of 4K bytes, 16 I/O cleaners, one I/O server, and three buffer pools, which we identify as BP_D1, BP_D2, and BP_X. The buffer pools are allocated 400 MB of memory (100,000 4K pages), and the database objects are assigned to buffer pools as follows:

- All data tables (with the exception of Warehouse, District, and Item tables) are assigned to BP_D1.
- Warehouse, District and Item tables are assigned to BP_D2.
- All indices are assigned to BP_X.

Database objects are spread out among the 16 disks to maximize performance.

Experimental Method

DB2 Version 5 does not support dynamic adjustment of the buffer pool sizes, so we had to carry out the following steps for each experiment:

- Run TPC-C workload on DB2 with the initial buffer pool configuration and collect performance measurements.
- Execute DRF to determine the new buffer pool configuration.
- Stop DB2 and reset the buffer pool configuration.
- Run TPC-C workload on DB2 with the new buffer pool configuration and collect performance measurements.

The workload was run against the system for 20 minutes each time. We allow the application to run for 10 minutes in order to stabilize performance, and then take the average of the performance statistics over the next 5 minutes. All DB2 performance measures are collected using the system’s monitoring API (IBM, 2004). The application was always the only user on the machine.

The response time and hit-rate estimators used in DRF require that statistics be collected at two different buffer pool allocations as well as at the current configuration. The initial two points are static as long as the workload remains the same. We arrive at estimates of the number of logical reads to each buffer pool by transactions of each class by first independently running each class. A reallocation unit of 500 pages was used in DRF.

We evaluate the accuracy of DRF based on the percentage difference between the goal average response time and the real average response time achieved with the

configuration suggested by DRF. We evaluate the robustness of DRF based on its ability to achieve reasonable accuracy under a variety of conditions. We consider situations where one and then two transaction classes are not meeting their performance goals. For each of these situations we start with different initial buffer pool allocations, namely a uniform allocation, where pages are spread evenly among the buffer pools and a skewed allocation, where pages are assigned unevenly.

We do not report the computation times for DRF in the following discussions. We found that, in all cases, the computation time is not significant. The time will vary depending on the initial conditions and the reallocation unit (that is, the number of pages moved in each iteration of the algorithm). For the experiments reported here, the computation time for DRF was in the range 0.5 to 0.9 seconds.

Experimental Results

We present results for experiments with three buffer pools. We achieved similar results with four buffer pools so they are not shown here. The following sets of initial conditions were used in the experiments:

- **Case 1 — One target transaction class and a skewed initial buffer pool allocation:** The Stock transaction class is not meeting its goal, and BP_D1, BP_X, and BP_D2 are allocated 99000, 500, and 500 pages, respectively.
- **Case 2 — One target transaction class and a uniform initial buffer pool allocation:** The Delivery transaction class is not meeting its goal, and BP_D1, BP_X, and BP_D2 are allocated 33333, 33333, and 33334 pages, respectively.
- **Case 3 — Two target transaction classes and a skewed initial buffer pool allocation:** The New Order and the Delivery transaction classes are not meeting their goals, and BP_D1, BP_X, and BP_D2 are allocated 5000, 90000, and 5000 pages, respectively.
- **Case 4 — Two target transaction classes and a uniform initial buffer pool allocation:** The New Order and Delivery transaction classes are not meeting their goals, and BP_D1, BP_X, and BP_D2 are allocated 33333, 33333, and 33334 pages, respectively.
- **Case 5 — Workload shift:** The typical TPC-C workload consists of 45% New Order transactions, 43% Payment transactions, 4% each of Order Status, Delivery, and Stock Level transactions. In this experiment, we simulate a shift in workload. Our new workload consists of 90% New Order transactions, 4% Payment transactions, and 2% each of Order Status, Delivery, and Stock Level transactions. With the shift in workload, the Delivery and the New Order transaction classes are in violation of their goals. Initially, BP_D1, BP_X, and BP_D2 are allocated 5000, 90000, and 5000 pages, respectively.

The results of the first two sets of experiments, where there is a single class not meeting its goal, are shown in Table 4. In all cases, DRF converges to a reallocation of pages such that the target transaction class's real average response time is within 9% of goal. In most cases, the real response time is below the goal, but in one case it is slightly above.

Table 4. Average frequencies and operations in TPC-C

Transaction	Frequency	Selects	Updates	Inserts	Deletes	Non-unique Selects	Joins
New Order	43	23	11	12	0	0	0
Payment	44	4.2	3	1	0	0.6	0
Order Status	4	11.4	0	0	0	0.6	0
Delivery	5	130	120	0	10	0	0
Stock Level	4	0	0	0	0	0	1

In case 1, we begin with a skewed buffer pool allocation with 90% of the available pages allocated to BP_D1, which services most of the data tables. The stock transaction class is in violation of its goal. In this case, DRF suggests moving pages from BP_D1 to the index buffer pool (BP_X) to improve the performance of the stock transaction. This is logical because the stock transaction class uses the stock index heavily. When the size of the index buffer pool is increased, the response time of the stock transaction decreases drastically from 20.1 seconds to less than 1 second.

In case 2, we begin with a uniform buffer pool allocation, and the delivery transaction class is in violation of its goal. In order to achieve a goal of 1.8 sec or 1.6 sec for Delivery, DRF suggests moving pages from the index buffer pool (BP_X) or BP_D2 (the buffer pool for the warehouse, district, and item tables) to BP_D1. By doing so, the defined goals for delivery are achieved. To reach a lower goal of 1.5 sec, DRF suggests increasing both BP_D1 and BP_X by taking pages from BP_D2. Since the delivery transaction class uses both the data tables and the index tables, it is logical that an increase in both these buffer pools will result in improved performance for this class. The decrease in pages from BP_D2 does not have a negative effect, as the tables using this buffer pool are small.

In case 2, two different strategies were used by DRF to achieve the goals. In the first, pages from the index buffer pool were sacrificed to increase the size of the data buffer pool, BP_D1. This provided the necessary gain in performance when the goal response time was relatively high. When we lower the goal, the algorithm takes a different approach and instead uses the pages from BP_D2. DRF's goal is not to optimize performance, but to achieve a buffer pool configuration that will allow the user-defined goals to be met. Therefore, depending on the goals set, the algorithm will take different approaches to solving the problem.

The results of the experiments where there are two classes — new order and delivery — that are not meeting their goals (cases 3 & 4) are shown in Table 5. In all cases, DRF converges to allocations such that both transactions' real average response times are within 11% of their goals. In all but two of these cases, the goal response time is achieved.

In case 3, when we begin with a skewed buffer pool allocation having 90% of the pages allocated to the buffer pool holding the indices (BP_X), DRF suggests increasing

Table 5. Cases 1 and 2: One class violating goal

Initial Configuration	Goal (sec)	Real (sec)	% Diff	Final Configuration (BP_D1, BP_X, BP_D2)
Skewed Allocation (99000,500,500) Stock Level in violation (20.1 sec)	1.0	0.91	9	78000, 21500, 500
	0.8	0.80	0	77000, 22500, 500
	0.7	0.73	3	76000, 23500, 500
Uniform Allocation (33333, 33333, 33334) Delivery class in violation (2.5 sec)	1.8	1.72	8	34333, 32333, 33334
	1.6	1.53	7	42333, 33333, 24334
	1.5	1.41	9	53333, 36333, 10334

the size of the primary data buffer pool, BP_D1, by moving pages from both the index buffer pool, BP_X and BP_D2. This improves the performance of both new order and delivery.

In case 4, we begin with a uniform buffer pool allocation. In this case, the algorithm performs the same as for case 2. When the goals are higher, the algorithm suggests moving pages from BP_D2 to BP_D1. As the goals are lowered, the algorithm suggests increasing the sizes of both BP_D1 and the index buffer pool (BP_X). In all cases, the goals for new order and delivery are achieved by implementing the configurations suggested by DRF.

The results for the last set of experiments (case 5) are shown in Table 6. In this case, the system is presented with a change in workload. The transactions remain the same, but the relative frequencies of the transactions are different. In this case, we want to show that if the performance of a transaction class changes due to a shift in workload, DRF can be used to restore the performance of the transaction class to (or close to) its original state.

The original buffer pool configuration for case 5 is 5000 pages for BP_D1, 90000 pages for BP_X, and 5000 pages for BP_D2. For the workload shift, we increase the percentage of new order transactions from 45% to 90%. Under these circumstances, the new order transaction class and the delivery transaction class do not perform as well as they did under the original TPC-C workload mix. Before the workload shift, new order's average response time was 2.31 seconds. After the workload shift, it increases to 2.87. Delivery's response time increased from 2.04 seconds to 3.35 seconds in response to the workload shift.

Using the original response times as guidelines for goals (that is, we wish to have these transaction classes perform as well as they were before), we run DRF to find a suggested buffer pool configuration that will allow these goals to be met. The suggested allocation is 42500 pages for BP_D1, 57000 for BP_X, and 500 pages for BP_D2, thus increasing the size of the data buffer pool, BP_D1, by taking pages from the other two buffer pools. Table 7 shows that this new buffer pool configuration allows the new order

Table 6. Cases 3 and 4: Two classes violating goals

Initial Config	New Order (2.5 sec)			Delivery (2.5 sec)			Final Configuration BP_D1, BP_X, BP_D2
	Goal (sec)	Real (sec)	% Diff	Goal (sec)	Real (sec)	% Diff	
Skewed (5000, 90000, 5000)	2.0	1.95	5	1.5	1.48	2	25000, 74500, 500
	1.9	1.85	5	1.4	1.33	7	28000, 71500, 500
	1.8	1.81	1	1.3	1.34	4	33500, 66000, 500
Uniform (33333, 33333, 33334)	1.8	1.74	6	1.7	1.67	3	34333, 33333, 32334
	1.7	1.62	8	1.6	1.54	6	42333, 33333, 24334
	1.6	1.49	11	1.5	1.41	9	53333, 36333, 10334

Table 7. Case 5: Shifting workload

	Original TPC-C (sec)	Shifted Workload (sec)	Goal (sec)	Response Time After Reallocation (sec)
New Order	2.31	2.87	2.3	1.84
Delivery	2.04	3.35	1.9	1.99

transaction class to overshoot its goal and delivery to come very close to reaching its goal.

In our results, we have illustrated the change in response times for the transaction classes that were in violation of their goals, but we have not shown what effect the new buffer pool configuration has on those transaction classes that were already achieving their goals. DRF, if possible, produces a configuration that satisfies the goals of all transaction classes. While improving the performance of the target transaction class(es), the performance of the other transaction classes sometimes improves (thus benefiting from the new allocation) and sometimes degrades (because of the loss of pages from a buffer pool key to this class's performance). In all cases in our experiments, all transaction classes continued to perform within their goals, although response times may increase or decrease slightly.

FUTURE TRENDS

Despite the challenges of building autonomic systems, platform vendors and developers of complex software systems are committed to the autonomic computing initiative. The need for self-management becomes more obvious with the growth of the Internet, the explosion of electronic commerce, and the increased complexity of the

software systems, such as DBMSs, that participate in this domain. Management of this complex, heterogeneous, interconnected environment is mandatory.

Due to the complexity of today's DBMSs, the implementation of autonomic features is a daunting task, one that led Chaudhuri and Weikum (2000) to propose that the requirement for automatic systems management may be better met by restructuring the DBMS architecture. They propose a reduced instruction set computer (RISC), style architecture with functionally restricted components and specialized data managers. It is suggested that this type of architecture will be more amenable to automatic tuning. It is possible that current software systems may undergo structural changes to facilitate the addition of autonomic features or that we may see a trend towards more RISC-style architectures for new software products.

To elevate DBMSs from the adaptive level of autonomic computing to the final stage of autonomic computing, the autonomic level, we will see a shift towards policy-based tuning. In this approach, the user specifies high-level policies that will govern how the system manages itself. This will require the development of policy languages and the specification of mappings between the policies and the low level system performance.

Although significant progress has been made towards autonomic computing, in most cases, developers still have a long way to go before truly autonomic systems are realized. Alan Ganek, IBM vice president of autonomic computing, states, "We expect that 10 to 15 years from now, most companies will have achieved many of the objectives of autonomic computing; systems that significantly reduce the complexity of managing technology by automatically tuning themselves, sensing and responding to changes, preventing and recovering from outages, and — perhaps the most important of all — systems that are responsive, productive, and resilient" (Preimesberge, 2004).

CONCLUSION

An autonomic DBMS is able to automatically reallocate its resources to maintain acceptable performance in the face of changing conditions. Such a DBMS requires self-tuning algorithms for its resources that analyze the performance of the system and suggest new resource allocations to improve the system's performance. In this chapter, we described such an algorithm for the buffer area, which we call dynamic reconfiguration or DRF. It is a general algorithm that can be used with any relational DBMS that uses multiple buffer pools.

We presented the buffer pool model and cost estimate equations used by DRF. Using an implementation of DRF for DB2, we explored the performance of DRF under a variety of workloads and initial memory configurations. The experiments used an OLTP workload from the TPC-C benchmark.

DRF improves previous self-tuning approaches to buffer pool sizing in three ways. First, DRF provides a more sophisticated response-time estimator than other algorithms that accounts for the effects of dirty buffer pages and asynchronous reads and writes performed by system processes. Second, DRF uses a data-oriented model of buffer pool allocation rather than a transaction-oriented one, which better represents how systems like DB2 actually manage their buffer pools. Third, DRF's data-oriented model accounts for the sharing of data pages among transaction classes, which was not captured by previous transaction-oriented approaches.

We conclude, based on the results of our experiments, that DRF is accurate and robust for OLTP workloads. We claim that DRF is accurate because, in all the experiments, DRF converged to buffer pool sizes that yield actual response times within 11% of the stated goals. We claim that DRF is robust because it was able to converge to satisfactory buffer pool allocations for cases with both one and two transaction classes violating their goals and when different initial page allocations were used. We also showed that DRF was also able to bring a system back to its original performance goals when the frequency characteristics of the workload changed. The experiments demonstrate the usefulness of goal-oriented resource management, generally, and dynamic buffer management, specifically, in a realistic DBMS environment.

ACKNOWLEDGMENTS

We thank IBM Canada Ltd., National Science and Engineering Research Council (NSERC) and Communications and Information Technology Ontario (CITO) for their support of this research.

REFERENCES

- Agrawal, S., Chaudhuri, S., & Narasayya, V. (2000, September 10-14). Automated selection of materialized views and indexes. In *Proceedings of the 26th International Conference on Very Large Databases*, Cairo, Egypt (pp. 496-505). Morgan Kaufmann.
- Arcangeli, J.-P., Hameurlain, A., Migeon, F., & Morvan, F. (2004). Mobile agent based self-adaptive join for wide-area distributed query processing. *Journal of Database Management*, 15(4), 25-45.
- Belady, L. (1966). A study of replacement algorithms for a virtual-storage computer. *IBM Systems Journal*, 5(2), 78-101.
- Brown, K., Carey, M., & Livny, M. (1993, August 24-27). Managing memory to meet multi-class workload response time goals. In *Proceedings of the 19th International Conference on Very Large Databases*, Dublin, Ireland (pp. 328-341). Morgan Kaufmann.
- Brown, K., Carey, M., & Livny, M. (1996, June 4-6). Goal-oriented buffer management revisited. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, Montreal, Canada (pp. 353-364). ACM Press.
- Chaudhuri, S., Christensen, E., Graefe, G., Narasayya, V., & Zwilling, M. (1999). Self-tuning technology in Microsoft SQL server. *IEEE Data Engineering Bulletin* 22(2), 20-26.
- Chaudhuri, S., & Weikum, G. (2000, September 10-14). Rethinking database system architecture: Towards a self-tuning RISC-style database architecture. In *Proceedings of the 26th International Conference on Very Large Databases*, Cairo, Egypt (pp. 1-10). Morgan Kaufmann.
- Chou, H., & DeWitt, D. (1985, August 21-23). An evaluation of buffer management strategies for relational database systems. In *Proceedings of the 11th International Conference on Very Large Databases*, Stockholm, Sweden (pp. 127-141). Morgan Kaufmann.

- Chung, J.-Y., Ferguson, D., Wang, G., Nikolaou, C., & Teng, J. (1995, November 6-10). Goal-oriented dynamic buffer pool management for database systems. In *Proceedings of the International Conference on Engineering of Complex Systems (ICECCS'95)*, Southern Florida (pp. 191-198). IEEE Computer Society Press.
- Effelsberg, W., & Härder, T. (1984). Principles of database buffer management. *ACM Transactions on Database Systems*, 9(4), 560-595.
- Elnaffar, S., Powley, W., Benoit, D., & Martin, P. (2003, September 1-5). Today's DBMSs: How autonomic are they? In *Proceedings of the 1st International Workshop on Autonomic Computing Systems (DEXA 03)*, Prague, Czech Republic (pp. 651-659). IEEE Computer Society Press.
- Faloutsos, C., Ng, R., & Sellis, T. (1991, September 3-6). Predictive load control for flexible buffer allocation. In *Proceedings of the 17th International Conference on Very Large Databases*, Barcelona, Catalonia, Spain (pp. 265-274). Morgan Kaufmann.
- Ganek, A. G., & Corbi, T. A. (2003). The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1), 5-19.
- IBM (2004). *DB2 Universal Database*. [Online]. Retrieved June 23, 2004, from <http://www.software.ibm.com/data/db2/udb>
- Isaacson, E., & Keller, H. (1966). *Analysis of numerical methods*. New York: John Wiley & Sons Inc.
- Kephart, J.O., & Chess, D.M. (2003). The vision of autonomic computing. *Computer*, 36(1), 41-50.
- Khan, L., McLeod, D., & Shahabi, C. (2001). An adaptive probe-based technique to optimize join queries in distributed Internet databases. *Journal of Database Management*, 12(4), 3-14.
- Leutenegger, S.T., & Dias, D. (1993, May 26-28). A modeling study of the TPC-C benchmark. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, DC (pp. 22-31). ACM Press.
- Nikolaou, C., Ferguson, D., & Constantopoulos, P. (1992, April). *Towards goal-oriented resource management*, IBM Research Report RC17919. IBM Press.
- O'Neil, E. J., O'Neil, P. E., & Weikum, G. (1993). The LRU-K page replacement algorithm for database disk buffering. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, DC (pp. 297-306). ACM Press.
- Preimesberge, C. (2004, February 25). Why IBM is hot on autonomic computing. *IT Manager's Journal*. (Online). Retrieved March 27, 2004, from <http://www.itmanagersjournal.com/software/04/02/24/2114246.shtml>
- Schiefer, B., & Valentin, G. (1999). DB2 universal database performance tuning. *IEEE Data Engineering Bulletin*, 22(2), 12-19.
- Transaction Processing Performance Council (2004). *Benchmark specifications*. [Online] Retrieved March 27, 2004, from <http://www.tpc.org>

Chapter XIII

Clustering Similar Schema Elements Across Heterogeneous Databases: A First Step in Database Integration

Huimin Zhao, University of Wisconsin-Milwaukee, USA

Sudha Ram, University of Arizona, USA

ABSTRACT

Interschema relationship identification (IRI), that is, determining the relationships among schema elements in heterogeneous data sources, is an important first step in integrating the data sources. This chapter proposes a cluster analysis-based approach to semi-automating the IRI process, which is typically very time-consuming and requires extensive human interaction. We apply multiple clustering techniques, including K-means, hierarchical clustering, and self-organizing map (SOM) neural network, to identify similar schema elements from heterogeneous data sources, based on multiple types of features, such as naming similarity, document similarity, schema specification, data patterns, and usage patterns. We describe an SOM prototype we have developed that provides users with a visualization tool for displaying clustering results and for incremental evaluation of potentially similar elements. We also report on some empirical results demonstrating the utility of the proposed approach.

INTRODUCTION

In today's technological environment, organizations and users are constantly faced with the challenge of integrating heterogeneous data sources. Most organizations have developed a variety of information systems for operational purposes over time. Having an integrated data source, however, is a prerequisite for decision-support applications, such as On-Line Analytical Processing (OLAP) and data mining, which require simultaneous and transparent access to data from the underlying operational systems. Business mergers and acquisitions further amplify the emergence of heterogeneous data environments and the need for data integration. Cooperating enterprises and business partners also need to share or exchange data across system boundaries for applications such as supply chain management.

The information systems that need to be integrated are typically heterogeneous in several aspects, such as hardware, operating systems, data models, database management systems (DBMS), application programming languages, structural formats, and data semantics. Many technologies are already available for bridging the syntactic differences across heterogeneous information systems. Some examples are heterogeneous DBMS, connectivity middleware (e.g., open database connectivity [ODBC], object linking and embedding for databases [OLE DB], and Java database connectivity [JDBC]), and the emerging Web services technology (Hansen, Madnick, & Siegel, 2002). However, resolving the heterogeneities in data semantics across systems is still a resource-consuming process and demands automated support.

A particularly critical step in semantic integration of heterogeneous data sources is to identify semantically corresponding schema elements, that is, tables that represent the same entity type in the real world and attributes that represent the same property of an entity type, from the data sources (Seligman, Rosenthal, Lehner, & Smith, 2002). This problem has been referred to as interschema relationship identification (IRI) (Ram & Venkataraman, 1999). IRI has been shown to be a very complex and time-consuming task in integrating large data sources due to various kinds of semantic heterogeneities among the data sources. For example, Clifton, Houseman, and Rosenthal (1997) reported on a project performed by the MITRE Corporation over a period of several years to integrate the information systems that had been developed semi-independently over decades for the U.S. Air Force. They found that tremendous effort was required from the investigator, local database administrators (DBAs), and domain experts to determine attribute correspondences across systems.

While completely automating the IRI process is generally infeasible, it is possible to semi-automate the process using techniques to reduce the amount of human interaction. We propose a cluster analysis-based approach to semi-automating the IRI process. We apply multiple clustering techniques, including K-means, hierarchical clustering, and self-organizing map (SOM) neural network, to identify similar schema elements from heterogeneous data sources, based on multiple types of features, such as naming similarity, document similarity, schema specification, data patterns, and usage patterns. An SOM prototype we have developed provides a visualization tool for users to display clustering results and for incremental evaluation of candidate solutions. We have empirically evaluated our approach using real-world heterogeneous data sources and report on some encouraging results in this chapter.

The chapter is organized as follows. First, we briefly review some related work in IRI, identifying the shortcomings of previous approaches. We then present a cluster analysis-based approach to IRI, discussing applicable cluster analysis techniques and potential semantic features about schema elements that can be used in cluster analysis. Next, we report on some empirical evaluation using real-world heterogeneous data sources. Finally, we summarize the contributions of this work and discuss future research directions.

RELATED WORK

Several approaches to detecting schema correspondences across heterogeneous data sources have been proposed in the past. Linguistic techniques, such as fuzzy thesaurus (Mirbel, 1997), semantic dictionary, taxonomy (Bright, Hurson, & Pakzad, 1994; Song, Johannesson, & Bubenko, 1996), conceptual graph, case grammar (Ambrosio, Métais, & Meunier, 1997), and speech act theory (Johannesson, 1997) have been used to determine the degree of similarity between schema elements, based on the names of the elements. Giunchiglia and Yatskevich (2004) used the lexical reference system WordNet and string matching methods, such as edit distance, in comparing element names. An assumption of these approaches is that schema elements are named using reliable terms, which describe the meanings of the elements appropriately. In many legacy systems, however, schema elements are frequently poorly named, using ad-hoc acronyms and phrases. When the schema element names are “opaque” or very difficult to interpret, such techniques for comparing element names may not even apply (Kang & Naughton, 2003).

Heuristic formulae have been designed to compute the degree of similarity between schema elements, based on the names and structures of the elements (Hayne & Ram, 1990; Madhavan, Bernstein, & Rahm, 2001; Masood & Eaglestone, 1998; Palopoli, Sacca, Terracina, & Ursino, 2000, 2003; Rodríguez, Egenhofer, & Rugg, 1999). These formulae often have been derived based on experiments and experiences from particular integration projects, giving rise to concern about the generalizability of the heuristic formulae over different settings.

Information-retrieval techniques have been used to compute the degree of similarity between text documents of schema elements (Benkley, Fandozzi, Housman, & Woodhouse, 1995). In many legacy systems, however, design documents are outdated, imprecise, incomplete, ambiguous, or simply missing.

Kang and Naughton (2003) used mutual information to measure the attribute dependencies within each database and compared the dependency patterns across databases, identifying attributes with similar dependency patterns as potentially corresponding attributes. However, attributes with similar dependency patterns may not be related at all. For example, the degree of dependency between “city” and “state” and that between “car model” and “car manufacturer” are likely to be quite similar, but the two pairs of attributes are not related.

Statistical analysis techniques, such as correlation and regression, have been used to analyze the relationships among numeric attributes based on actual data, assuming that some matching records across the data sources are available (Fan, Lu, Madnick, &

Cheung, 2001, 2002; Lu, Fan, Goh, Madnick, & Cheung, 1997). However, they require data from heterogeneous databases to be integrated in some manner (e.g., based on a common key) first.

Cluster-analysis techniques have been used to group similar schema elements (Duwairi 2004; Ellmer, Huemer, Merkl, & Pernul, 1996; Srinivasan, Ngu, & Gedeon, 2000). Since these techniques are “unsupervised,” relatively less human intervention is involved. SemInt (Li & Clifton, 2000) uses both cluster-analysis and classification techniques to identify potential similar attributes. The attributes in one database are first grouped into several clusters. A neural network classifier is trained using the clustered attributes as training examples and classifies attributes in other databases into the clusters of attributes in the first database. Although both cluster-analysis and classification techniques are used, the pure effect of SemInt is of a clustering nature; attributes of heterogeneous databases are clustered into groups based on similarity. When the attributes of the first database are clustered, it is difficult to estimate the accuracy of the classifier built later to classify other attributes into the clusters. The clustering step needs to be rather conservative; few clusters, each containing a large number of attributes, are generated to prevent attributes in other databases from being classified into wrong clusters. Consequently, large amount of human evaluation is still needed to identify the truly corresponding attributes from the large clusters.

Rahm and Bernstein (2001) provided a survey of various approaches to schema matching. Do, Melnik, and Rahm (2002) provided a survey of evaluation of some of the approaches. Interested readers may refer to these surveys for more comprehensive coverage of this area.

CLUSTER ANALYSIS-BASED APPROACH

We use cluster analysis techniques to find groups of similar schema elements from heterogeneous databases. In this work, we have attempted to overcome several shortcomings in previous approaches.

1. Previous approaches have been committed to a particular technique (Ellmer et al., 1996; Srinivasan et al., 2000). We apply multiple techniques to *cross-validate* clustering results.
2. Previous approaches (Ellmer et al. 1996; Li and Clifton, 2000; Srinivasan et al., 2000) also require users to specify the number of clusters prior to cluster analysis. We *visualize* clustering results and allow users to *incrementally* evaluate candidate similar elements.
3. Previous approaches have used some particular features about schema elements for cluster analysis. We use multiple types of available semantic features about schema elements to deal with different situations and to improve clustering accuracy.

Cluster-Analysis Techniques

Cluster-analysis techniques group objects drawn from some problem domain into unknown groups, called clusters, such that objects within the same cluster are similar to each other (i.e., internal cohesion), while objects across clusters are dissimilar to each

other (i.e., external isolation). The objects to be clustered are represented as vectors of features, or variables. When there are many features, other analyses, such as principal component analysis and factor analysis (Afifi & Clark, 1996), can be performed prior to cluster analysis to reduce the dimensionality of the input vectors. The degree of similarity between two objects is measured using some distance function (e.g., Euclidean, Mahalanobis, Cosine, etc.). The features may be weighted empirically, based on the analyst's subjective judgment, to reflect their importance in discriminating the objects. However, since it is often difficult for the analyst to determine these weights, equal weights are often given to all the features after they have been normalized or standardized.

Many techniques for cluster analysis have been developed in multivariate statistical analysis and artificial neural networks. The most widely used statistical clustering methods fall into two categories: hierarchical and nonhierarchical (Everitt, Landau, & Leese, 2001). K-means is a popular nonhierarchical clustering method. It requires users to specify the number of clusters, K , prior to a cluster analysis. Hierarchical methods cluster objects on a series of levels, from very fine to very coarse partitions. Kohonen's self-organizing map (SOM) (Kohonen, 2001), an unsupervised neural network, has recently received much attention as an alternative to traditional clustering techniques. SOM usually projects multi-dimensional data onto a two-dimensional map, roughly indicating the proximities among the objects in the input data.

Statistical clustering methods are available in many statistical packages, such as SAS and SPSS. We have implemented an SOM prototype. The prototype uses the U-matrix method (Costa & de Andrade Netto, 1999) to present SOM results. On a two-dimensional map consisting of output network nodes, each input object corresponds with a best-matching node called "response." The responses of similar input objects are located close to each other. The prototype uses gray levels to indicate relative distances between neighboring output nodes and, therefore, boundaries between clusters. We have further designed a slider that allows users to vary the similarity threshold and obtain clustering results on different similarity levels interactively (see examples later).

Cluster analysis is highly empirical; different methods often produce different clusters (Afifi & Clark, 1996). The result of a cluster analysis should be carefully evaluated and interpreted in the context of the problem. It is also recommended that different techniques be tried to compare the results. Mangiameli, Chen, and West's (1996) empirical evaluation found that SOM is superior to seven hierarchical clustering methods. However, Petersohn's (1998) empirical comparison of various clustering methods, including K-means, seven hierarchical clustering methods, and SOM, did not find any method that was consistently the best for every problem. Many other empirical studies have also concluded that there is no universally superior method (Everitt et al., 2001). In our approach, we apply multiple clustering methods in the identification of similar schema elements to cross-validate clustering results. If multiple methods agree on some clusters, it gives users more confidence about the validity of these clusters. Otherwise, users should pay more attention to the conflicting parts.

Semantic Features about Schema Elements

The choice of input features has an obvious impact on the performance of cluster analysis. Missing relevant features and/or including noisy ones can lead to performance

degradation. We classify the semantic information about schema elements that might be used as input features for cluster analysis and discuss related technical issues in the following paragraphs.

Naming Similarity

A general principle in database design is that tables and attributes should be named to reflect their meanings in the real world. Linguistic techniques, such as fuzzy thesaurus (Mirbel, 1997), semantic dictionary, taxonomy (Bright et al., 1994; Song et al., 1996), conceptual graph, case grammar (Ambrosio et al., 1997), and speech act theory (Johannesson, 1997), can be used to determine the degree of semantic similarity between schema element names. String matching methods, such as edit distance (Stephen, 1994), can also be used to determine the degree of syntactic similarity between schema element names.

However, there are various problems associated with schema element names. First, they usually cannot completely capture the semantics of the elements. Second, they are often “opaque” or very difficult to interpret (Kang & Naughton, 2003); phrases and ad-hoc acronyms rather than single words are commonly used to name schema elements. Third, in some regions where pictographic languages are used officially, it is a frequent practice that pronunciation notations (e.g., Pingying for Chinese), which are easier to map to English characters than the actual pictographic characters, are used to name database objects. The same pronunciation may mean multiple and totally different things. Fourth, the meaning of a schema element changes as the associated business processes evolve. The name originally given to a schema element may not reflect its current meaning appropriately. It is also possible, especially in canned legacy systems, that some schema elements are reserved for future extension and initially given meaningless names. The semantics of these reserved elements are customized by the end-users or business processes. For example, a reserved “comment” attribute might be used to store critical data.

Document Similarity

Database design documents usually contain descriptions of schema elements. Sometimes these documents are stored in database dictionaries or metadata repositories and are associated with schema elements. If this information is available, it may convey more semantics than names. An information retrieval tool called DELTA has been used to look for potential attribute relationships based on descriptions about attributes (Benkley et al., 1995). DELTA can find relationships when attribute names are very different but the descriptions are similar. However, as has been normal in software engineering practice, this information is often outdated, incomplete, incorrect, ambiguous, or simply not available.

Schema Specification

Schema elements representing similar real-world concepts should be modeled similarly and therefore should have similar structures (Ellmer et al., 1996; Li & Clifton, 2000; Srinivasan et al., 2000). In other words, structure and semantics are correlated. Schema specifications about attributes (e.g., data type, length, and constraints) and those about tables (e.g., foreign keys in relational databases and superclass/subclass

relationships in object-oriented databases) (Duwairi 2004) are usually stored in the system catalog of a DBMS.

However, semantically similar concepts could often be modeled using different structures while semantically different concepts could have similar structures. In addition, schema specifications extracted from different DBMSs or different data models may be incompatible. Even worse, this information may not be available in some cases, such as legacy systems that use flat files.

Data Patterns

Semantics are also embedded in the actual data stored in the databases. Some patterns, or summary statistics, about the actual data or data samples can be used as features for cluster analysis. Patterns of an attribute value include: the length of a value, the percentage of digits within a string (a numeric value can readily be converted into a string), the percentage of alphanumeric characters within a string, and the percentage of special characters within a string. Patterns of an attribute include summary statistics (central tendency and variability) of the patterns of its values, the ratio of the number of distinct values to the number of records, the percentage of missing (or non-missing) values, and the dependencies between the attribute and other attributes (Kang & Naughton, 2003). The patterns of all attributes of a table can further be summarized to generate patterns of the table.

The problems associated with using data patterns as semantic features are often similar to those associated with using schema specifications, in that structures restrict the possible data values that can be stored. Data patterns are often correlated more with structures than with semantics. Categorical data values can be coded differently. For example, “gender” can be defined as a numeric attribute and coded as 1 for male and 2 for female in one database, while it is defined as a character attribute and coded as “M” for male and “F” for female in another database. The aggregate of several attributes in one database may correspond with a single attribute in another database (e.g., student last name and first name vs. student name). The same attribute value may be measured in different units (e.g., sales in dollars vs. thousands of dollars). Ram, Park, Kim, and Hwang (1999) proposed a comprehensive framework for classifying semantic conflicts. Nevertheless, data patterns are the only features that can readily be computed based on the actual data or data samples. They are the least that is available for cluster analysis of schema elements in extremely “dirty” situations.

Usage Patterns

Usage patterns, such as update frequency and number of users or user groups, have been considered in clustering entities (Srinivasan et al., 2000). An assumption is that the same entity should be accessed in similar manners (e.g., in terms of access frequency and group of users) in different systems. Usage data may be extracted from the audit trail of a modern DBMS but may not be available in legacy systems.

Business Rules and Integrity Constraints

Many complex business rules and integrity constraints are often implemented using assertions, procedures, triggers, and application programs. In general, semantics embedded in code is hard to extract. However, if some constraints are specified in the schemas

declaratively, documented in the database design specifications, or provided by designers or domain experts, they can be used to provide deep semantics about the underlying databases and reflect the real world state of the underlying databases more accurately. Another possibility is that if these business rules or integrity constraints are specified in database design specifications, they can be dumped into text documents and compared using information retrieval tools such as DELTA (Benkley et al., 1995).

Users' Mind and Business Processes

While some semantics can be extracted from metadata, actual data contents, usage catalogs, or even application programs, others may be defined only by the user or the business process. Semantics that reside in users' minds or business processes can only be explored via interaction with users themselves.

From the above discussion, we have the following observations. First, completely automating the IRI process is generally infeasible. Human intervention is necessary to capture the last two, and arguably the most reliable and important, categories of information. A useful tool should provide interactive interfaces to capture the domain knowledge of users. Second, unlike some other clustering problems, where there are features that naturally discriminate input objects, no optimal set of features exists for describing the semantics of schema elements, due to the problems stated earlier. Features must be carefully evaluated and selected in each particular case. Such feature selection is often subjective because no objective measures of goodness can be defined. Third, while names and documents directly describe the meanings of schema elements, schema specification, data patterns, and usage patterns reflect the semantics only indirectly. We posit that direct semantic features are more discriminating than indirect ones in semantic clustering. When there are no quality direct semantic features in some real-world hard cases, the performance of cluster analysis will inevitably degenerate. In our approach, we incorporate all available semantic information to achieve the best possible clustering results.

EMPIRICAL EVALUATION

We have evaluated our approach using two cases of real-world heterogeneous data sources. The two cases may not be representative of all possible real-world heterogeneous databases, as there are a large variety of possible situations, with different degrees of heterogeneities and data quality. While it is infeasible to enumerate all possible situations, we have selected a relatively "clean" example and a "dirty" one to illustrate the best and the worst possible performance of the techniques. Meanwhile, we are continually looking for opportunities to apply and validate our approach in more real-world data integration projects. The first case is relatively "clean," where the schemas of the two data sources largely overlap and schema elements are well-named (some names are manually assigned), so that both indirect and direct semantic features can be used for cluster analysis. We use this case to demonstrate the best result that our approach can generate in relatively "clean" situations. The second case is extremely "dirty." Two legacy databases have been independently developed by different operational departments for different purposes. Only small portions of the two databases overlap. Data

Table 1. Sample data from Bookstore A

ISBN	Authors	Title	List Price	Our Price	Cover	Pages	...
0072127309	Greg Bucek	Instant ASP Scripts	49.99	39.99	Paperback	928	...
0130132942	Guy Harrison	Oracle Desk Reference	34.99	27.99	Paperback	520	...
047139288X	Oracle Corp	Oracle 8i		15.65	Hardcover		...
...	...	...	...	...	...	...	...

Table 2. Sample data from Bookstore B

ISBN	Author	Title	OurPrice	RetailPrice	Cover Format	Pages	...
1928994040	Syngress Media Inc	DBA Linux Handbook	59.95		Paperback	656	...
1891762494	Kevin A. Siegel	RoboHelp HTML 2000	45.00		Paperback	260	...
1861003439	Frank Boumphrey	Beginning XHTML	31.99	39.99	Paperback	400	...
...	...	...	...	...	...	...	...

patterns are the only comparable features available for cluster analysis. We use this case to demonstrate that our approach can help in extremely “dirty” situations.

Case 1: E-Catalog Integration

The rapid growth of the Internet continuously creates new requirements and opportunities for data integration. A particular example is the need to integrate electronic product catalogs (E-catalogs) of different vendors, driven by business-to-customer (B2C) online malls, business-to-business (B2B) exchanges, and mergers and acquisitions (Navathe, Thomas, Satitsamitpong, & Data, 2001). In one empirical study, we evaluated book catalogs extracted from two leading online bookstores. One catalog (Catalog A) contained the following 16 fields (i.e., attributes) about books on the Web: ISBN, authors, title, series, list price, our price, cover, type, edition, month, day, year, publisher, pages, average rating, and sales rank. The other (Catalog B) contained 14 similar fields, including ISBN, title, author, retail price, our price, cover format, edition, pages, publisher, pubmonth, pubyear, editiondesc, salesrank, and rating. We manually copy-pasted 737 and 722 records from the Web sites of the two stores, respectively (Tables 1 and 2 show some examples).

The Web sites did not display the names of some fields; therefore we assigned names to the fields, based on our understanding of the fields. Even the displayed field names might be different from the attribute names actually used in the back-end databases. Since we did not have direct access to the back-end databases, we could only use the displayed or manually assigned field names in our analysis. Similar tasks are faced by emerging online shopbots (or shopping agents). They usually do not have direct access to the back-end databases of online shops, but try to reason about the data structures indicated by the front-end Web pages and build wrappers to extract data from the databases.

We used the K-means and hierarchical clustering methods included in SPSS and our SOM prototype to cluster the attributes (i.e., fields) of the two catalogs. The same techniques can be used to cluster tables as well, if there are many tables to compare. In this case, however, there was only one table from each catalog.

Table 3. Similarity between some attribute names

	A.ISBN	A.AUTHORS	A.TITLE	...	B.ISBN	B.TITLE	B.AUTHOR	...
A.ISBN	1.000	0.000	0.200	...	1.000	0.200	0.000	...
A.AUTHORS	0.000	1.000	0.143	...	0.000	0.143	0.857	...
A.TITLE	0.200	0.143	1.000	...	0.200	1.000	0.167	...
...	...	...	...	...	...	...	...	...
B.ISBN	1.000	0.000	0.200	...	1.000	0.200	0.000	...
B.TITLE	0.200	0.143	1.000	...	0.200	1.000	0.167	...
B.AUTHOR	0.000	0.857	0.167	...	0.000	0.167	1.000	...

We evaluated and selected some features about the attributes. Since we extracted the data from the Web sites, we did not have any documentation, schema definition, usage pattern, or business rules. We had field names displayed on the Web pages or manually assigned and used a similarity measure based on the string edit distance (Stephen, 1994) to measure the similarity between two attribute names (Table 3 shows some examples). This similarity measure between two strings was defined as one minus the ratio between the minimum number of characters that needs to be inserted into or deleted from one string to transform it into another string and the length of the longer string. We estimated some statistics about data patterns of each attribute, based on the sample. These included summary statistics (i.e., mean, standard deviation, max, and min) on the lengths of values, summary statistics on the percentages of digits in the values, summary statistics on the percentages of alphanumeric characters in the values, the percentage of values that are not missing, and the ratio of the number of distinct values to the number of records. There were 14 such features about data patterns (Table 4 shows some examples).

We preprocessed the features, including the naming similarity based on the string edit distance and statistics about data patterns, prior to cluster analysis. First, we linearly normalized each of the features into the range of [0,1]. We then performed principal component analysis on the features to obtain a set of orthogonal components with a reduced dimensionality. The number of features based on data patterns does not increase when there are more attributes to be compared. However, the number of features based on comparing names is proportional to the number of attributes to be compared and poses a dimensionality problem when the number of attributes is large. There are 30 features related to degree of similarity between attribute names and 14 features related to data patterns. We extracted ten components from the 44 features using principal component analysis, using the default extraction threshold (i.e., eigenvalues greater than 1) of SPSS. The ten components explain 89.3% of the variance in the original features. The input data set for the cluster analysis of attributes is a 30 (attributes) $\times$ 10 (components) matrix.

We ran three cluster analysis techniques, K-means, hierarchical clustering (using the centroid method), and SOM, on the input data set about attributes using the Euclidean distance function. A hierarchical clustering result allows users to start from very similar elements and incrementally evaluate less similar ones. We ran K-means several times, using different Ks, to simulate a hierarchical clustering effect. Figures 1-3 show some results generated by the three techniques. For example, in the result

Table 4. Data patterns of some attributes

Feature	A.ISBN	A.Authors	A.Title	...	B.ISBN	B.Title	B.Author	...
%(Non-missing Values)	1.00	0.98	1.00	...	0.98	0.98	0.97	...
%(Unique Values)	1.00	0.85	0.96	...	1.00	0.95	0.87	...
Mean(Length)	10.00	26.63	41.43	...	10.00	39.48	25.76	...
StdDev(Length)	0.00	21.72	25.82	...	0.04	23.39	17.34	...
Max(Length)	10.00	199.00	199.00	...	10.00	187.00	199.00	...
Min(Length)	10.00	3.00	4.00	...	9.00	4.00	5.00	...
Mean%(Digits))	0.99	0.00	0.02	...	0.99	0.02	0.00	...
StdDev%(Digits))	0.03	0.02	0.03	...	0.03	0.04	0.00	...
Max%(Digits))	1.00	0.45	0.21	...	1.00	0.25	0.00	...
Min%(Digits))	0.90	0.00	0.00	...	0.90	0.00	0.00	...
Mean%(Alphanumeric))	1.00	0.86	0.85	...	1.00	0.86	0.86	...
StdDev%(Alphanumeric))	0.00	0.06	0.05	...	0.00	0.04	0.06	...
Max%(Alphanumeric))	1.00	1.00	1.00	...	1.00	1.00	1.00	...
Min%(Alphanumeric))	1.00	0.64	0.57	...	1.00	0.70	0.64	...

generated by K-means using K=10, A.ISBN and B.ISBN, are grouped into a cluster; A.Our_Price, A.List_price, B.Retailprice, and B.Ourprice are grouped into a cluster. In the result generated by hierarchical clustering, A.Edition and B.Edition are grouped into a cluster on a low-distance level; A.Edition, B.Edition, and A.Editiondesc are grouped into a cluster on a higher distance level; all attributes are grouped into a single cluster on the highest distance level. On a map generated by SOM, similar attributes are located close to each other; gray levels indicate relative distances between neighboring attributes. For example, in Figure 3(a), A.ISBN and B.ISBN appear to be very similar, and A.Title and B.Title appear to be very similar. But there is a dark boundary between the two groups, indicating that the two groups are quite dissimilar. The clustering results generated by the three techniques are quite similar, providing users some confidence in the validity of the results.

Although we did not find significant differences among the three methods in terms of accuracy, SOM does appear better than K-means and hierarchical clustering in visualizing clustering results. Using the SOM tool, users can vary the similarity threshold on a slider and obtain clustering results on different similarity levels interactively — see Figure 3 (b)(c)(d). The higher the similarity threshold, the tighter the clusters. The SOM tool provides users with a visualization tool for displaying clustering results and for incremental evaluation of candidate solutions. Users can begin with the most similar attributes and gradually examine less similar ones.

Our experiments also show that features such as names, which directly reflect the semantics of schema elements, have more discriminating power than those such as schema specification and data patterns, which indirectly reflect the semantics of schema elements. Figure 4 shows a clustering result generated by SOM using only indirect semantic features, similar to those used in SemInt (Li & Clifton, 2000). The boundaries between clusters become very vague. At a medium similarity level, the attributes are roughly clustered into two big groups: numeric and character. When used in a real database integration project, SemInt encountered similar problems and generated relatively big clusters (the average cluster size was about 30) (Clifton et al., 1997).

Figure 1. Some K-means results for the e-catalog example

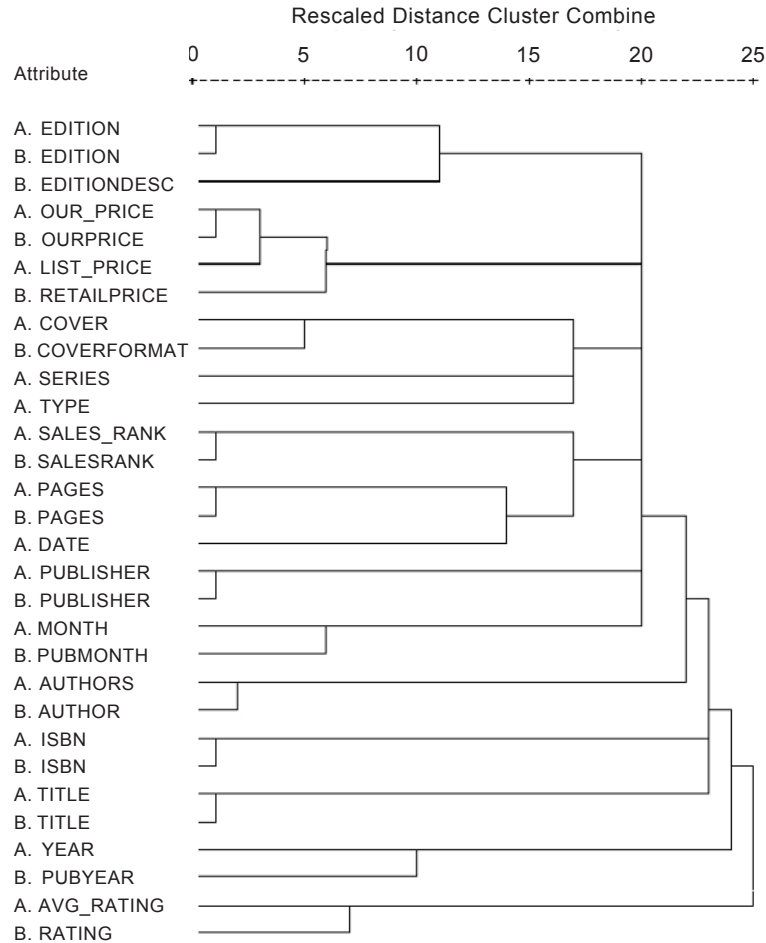
Cluster	Attribute	Cluster	Attribute
1	A.ISBN	1	A.ISBN
	B.ISBN		B.ISBN
2	A.AUTHORS	2	A.AUTHORS
	B.AUTHOR		B.AUTHOR
3	A.TITLE	3	A.TITLE
	A.TYPE		B.TITLE
	B.TITLE	4	A.SERIES
4	A.PAGES	5	A.OUR_PRICE
	A.SALES_RANK		A.LIST_PRICE
	A.DATE		B.RETAILPRICE
	B.PAGES		B.OURPRICE
	B.SALESRANK	6	A.COVER
5	A.OUR_PRICE	6	B.COVERFORMAT
	A.LIST_PRICE		A.TYPE
	B.RETAILPRICE	7	A.EDITION
	B.OURPRICE		B.EDITION
6	A.COVER		B.EDITIONDESC
	B.COVERFORMAT	9	A.MONTH
7	A.AVG_RATING	9	B.PUBMONTH
	B.RATING	10	A.YEAR
8	A.SERIES	10	B.PUBYEAR
	A.EDITION		A.PUBLISHER
	A.PUBLISHER	11	B.PUBLISHER
	B.EDITION		A.AVG_RATING
	B.PUBLISHER	12	B.RATING
9	A.MONTH		A.PAGES
	B.PUBMONTH	13	B.PAGES
10	A.YEAR		A.SALES_RANK
	B.PUBYEAR	14	B.SALESRANK
			A.DATE
		15	A.DATE

(a) k=10

(b) k=15

Figure 5 shows the clustering result generated by SOM using only direct semantic features (i.e., degrees of similarity between attribute names). The clusters are much tighter than those in Figure 4. There are problems, however, when similar attributes are named very differently. For example, attributes A.Type and B.Editiondesc are named very differently although they describe the same property (i.e., whether a book contains a CD, Disk, etc.). They are located far away from each other on the map. The clusters reflect naming similarities. When both direct and indirect semantic features are used, cluster analysis takes both into account. Even if two semantically dissimilar attributes may have very similar structures and data patterns, their dissimilar names help to differentiate them. Conversely, even if two semantically similar attributes may have very dissimilar names, their similar structures and data patterns can help to bring them somewhat closer. We

Figure 2. A hierarchical clustering result for the e-catalog example (Dendrogram using the Centroid Method)

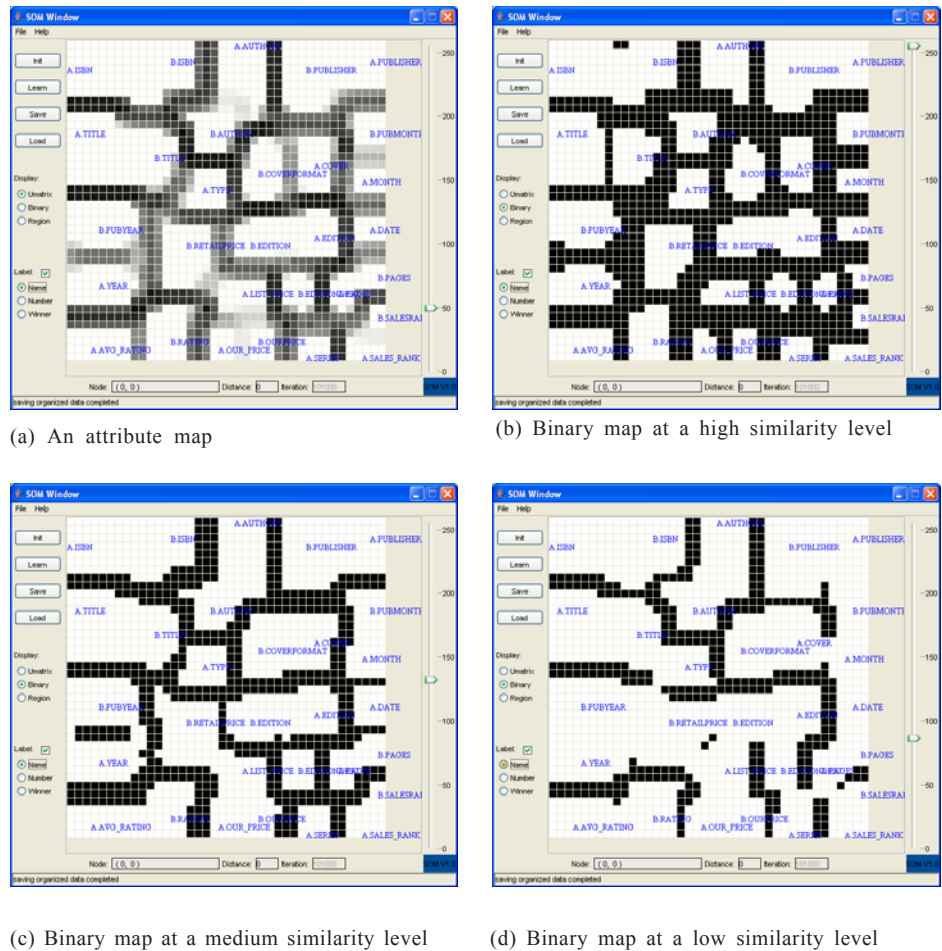


therefore recommend using both direct and indirect semantic features when they are available and meaningful.

Case 2: Legacy Database Integration

Modern organizations often rely on many heterogeneous data sources, including legacy systems, operational databases, departmental data marts, and Web sites, to accomplish their daily business operations and need to integrate these data sources for analytical purposes. We have evaluated our approach using the databases of the property management department and the surplus property office of a large public

Figure 3. An SOM result for the e-catalog example



university. The property management department manages all property assets owned by departments of the university. When a unit wants to dispose an item, the item is delivered to the surplus property office, where it is sold to another unit or a public customer. The database maintained by the property management department, named FFX, is managed by IBM IDMS. The surplus database is managed by Foxpro.

An initial evaluation revealed parts of the two databases that overlap. There are nine tables in FFX and three tables in surplus. In surplus, data stored in two tables are generated locally and are not closely related to data of FFX. The INVMSTR table in surplus corresponds closely with the FFX_ASSET table in FFX; both tables store one record for each property item. Three additional tables in FFX, FFX_ACCOUNT, FFX_CLASS_CODE, and FFX_MFG_CODE, also contain data that correspond with data

Figure 4. An SOM result for the e-catalog example using only indirect semantic features

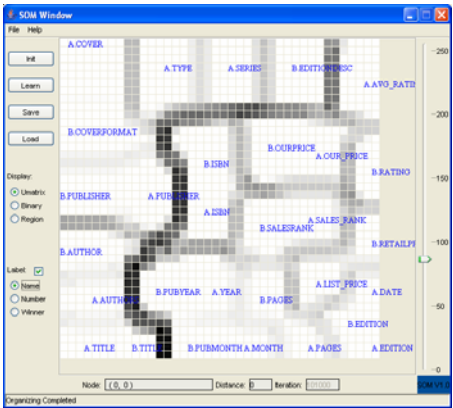
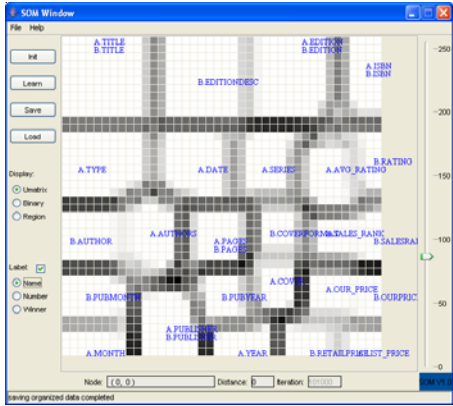


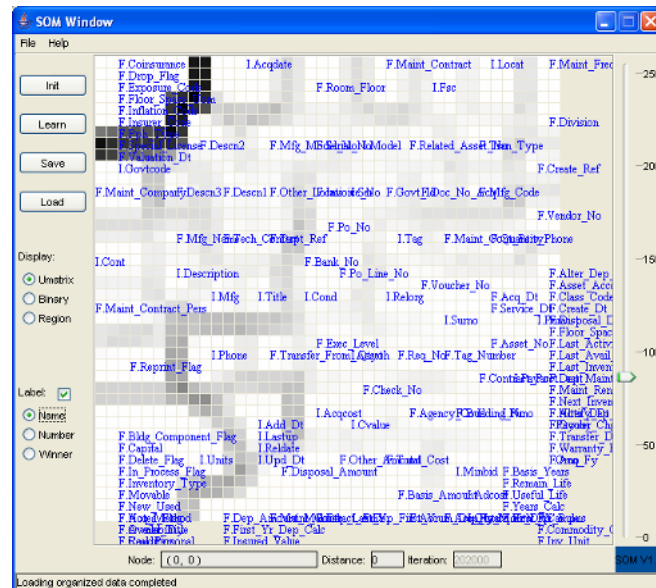
Figure 5. An SOM result for the e-catalog example using only direct semantic features



in INVMSTR. INVMSTR therefore corresponds with the join of FFX_ASSET, FFX_ACCOUNT, FFX_CLASS_CODE, and FFX_MFG_CODE. We denote the INVMSTR table I and the join of the four FFX tables F.

Based on our evaluation, it appears that only naming similarity and data patterns are easily available for clustering attributes in the two databases. Other features, such as document similarity, schema specification, and usage patterns, are hardly comparable. FFX has an online dictionary that contains a text description of several lines for each attribute. However, there is no counterpart on the surplus side. A single person, the expert in the surplus office, is regarded as the authority in interpreting the meaning of every attribute. The two databases are designed for different systems, IBM IDMS and Foxpro. The data types are incompatible between the two systems. Keys or any other

Figure 6. An SOM result for the property example using only indirect semantic features



types of constraints are not specified on either database declaratively, but rather are embedded in application programs or even manually enforced. The lengths of attributes are usually much longer in surplus than in FFX, probably because Foxpro supports variable-length character attributes. Neither of the two databases maintains an active audit trail.

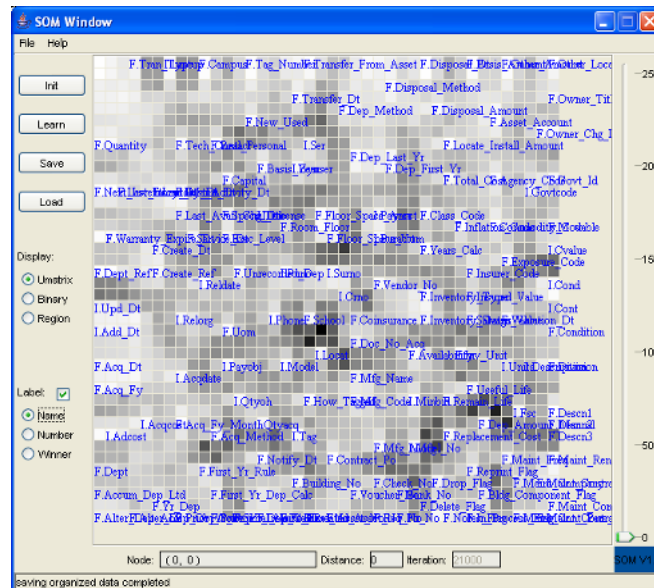
We used the similarity measure based on the string edit distance again to measure the similarity between attribute names. However, there are serious problems with this similarity measure, as almost all attributes are named using abbreviations of phrases and are abbreviated very differently in the two databases.

No matter how different the two databases are in terms of all other characteristics, the patterns of data stored in the databases are much more comparable. Of course, there are variations too. For example, “acquisition date” is specified as character attributes in both databases but the formats are very different. We selected the same 14 features based on data patterns as in the e-catalog integration case and linearly normalized each of the features into the range of [0,1].

We ran cluster analysis using only data patterns, only naming similarity, and both data patterns and naming similarity, respectively. When naming similarity was included, we used principal component analysis to reduce the dimensionality of the input data first. Figures 6-8 show some results generated by SOM.

The results show that the naming similarity measure based on the string edit distance cannot adequately reflect the similarity between the attributes and is not useful in this extremely dirty case. When naming similarity was included in the input features (Figures 7 and 8), the attributes were clustered into numerous small clusters, as most of

Figure 7. An SOM result for the property example using only direct semantic features

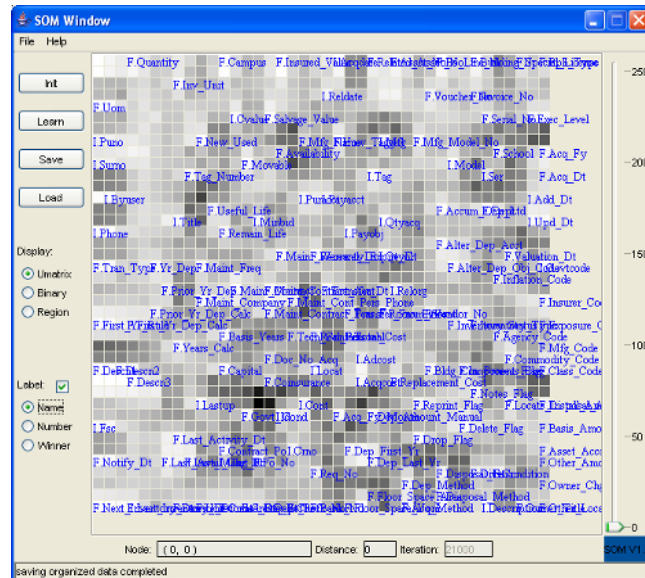


the attribute names are very different no matter whether the attributes are indeed semantically similar or not.

When we used only data patterns, which are considered “indirect” semantic features, we also expected that the accuracy of the cluster analysis would be much lower than the accuracy of the analysis we performed over the e-catalog example, where we used both “direct” and “indirect” semantic features. With such limited informative input, the results of K-means and hierarchical clustering are hardly useful. SOM results (e.g., Figure 6) still visualize the relative structural similarity among attributes.

Now, the question is — with such low-accuracy results, is automated support still useful to users for detecting schema correspondences from heterogeneous databases? In this particular case, SOM results based on data patterns help users in several ways. First, SOM results reveal several groups of very similar attributes; the attributes in a group are located at the same node on a map. In Figure 6, one group at the upper-left corner consists of 10 attributes, including F.Coinsurance, all of which are unused; that is, the values of these attributes are all missing; they have been designed, but never used and therefore can be totally ignored in the subsequent analyses. One group on the right-hand side consists of 16 attributes, including F.Create_Dt, all of which are system-generated dates. Another group consists of 10 attributes, including F.Bldg_Component_Flag, all of which are binary (True/False) flags. Over 50% of all the attributes are included in groups of this kind. Such groups help users to categorize attributes. Second, some attributes that are common to the two databases are indeed located close to each other. Five out of 12 such common attribute pairs, including model (I.Model and F.Mfg_Model_No), manufacturer (I.Mfg and F.Mfg_Name), serial number (I.Ser and

Figure 8. An SOM result for the property example using both direct and indirect semantic features



F.Serial_No), acquisition cost (I.Acqcost and F.Total_Cost), and description (I.Desc and F.Descn1), can be identified from the SOM result at a medium similarity threshold. However, the usefulness of the cluster-analysis results is limited in this extremely “dirty” case. The boundaries between clusters are vague. The clusters reflect structural rather than semantic similarity. Many attributes with similar data patterns are semantically dissimilar, while many (7 out of 12) common attribute pairs can not be identified.

CONCLUSION AND FUTURE RESEARCH

We have described a cluster analysis-based approach to semi-automating the IRI process and presented some empirical findings. We argue that no optimal set of features exists for IRI, and therefore feature evaluation and selection must be performed depending on particular applications. We use multiple techniques to cross-validate clustering results and incorporate a more complete set of semantic features than past approaches. While our initial experiments did not find significant difference among various cluster analysis methods in terms of accuracy, our SOM tool provides additional benefits of offering visualization and incremental evaluation. Field studies and designed experiments can be conducted in the future to validate the usability of the tool.

Our approach alleviates some of the shortcomings of past approaches for IRI. We have classified potential features for clustering schema elements into several categories, including naming similarity, documentation similarity, schema specification, data patterns, and usage patterns. We advocate using multiple categories of such features

whenever they are available and meaningful, rather than relying on a particular type of feature, as past approaches did. Our approach continues to provide useful support even in extremely “dirty” situations, where schema elements are poorly named and there is no documentation to consult, although with reduced quality, as our second case study shows. Previous approaches relying on linguistic techniques (Ambrosio et al., 1997; Bright et al., 1994; Johannesson, 1997; Mirbel, 1997; Song et al., 1996) or information retrieval techniques (Benkley et al., 1995) simply cannot be applied in such situations. Our approach does not rely on any heuristics and is free of the generalizability problem of heuristic-based approaches (Hayne & Ram, 1990; Madhavan et al., 2001; Masood & Eaglestone, 1998; Palopoli et al., 2000, 2003; Rodríguez et al., 1999). Our approach allows the user to incrementally evaluate hierarchical clustering results, rather than fixing the number of clusters prior to analysis (Ellmer et al., 1996; Li & Clifton, 2000; Srinivasan et al., 2000).

Our experiments indicate that direct semantic features such as names of schema elements are more discriminating than indirect semantic features such as those used by SemInt (Clifton et al., 1997). However, in real-world heterogeneous databases, comparison of names is not always feasible due to the problems we have discussed. When attribute names are extremely “opaque,” including naming similarity measures in the analysis can even hurt the performance. In such cases the accuracy of semantic cluster analysis can degenerate seriously. We recommend the use of cluster analysis results as a reference in an early stage of IRI so that users can quickly discover similar schema elements and reduce the search space. Good tools do help to reduce the amount of interaction between domain experts and analysts, even in extremely “dirty” situations such as the second case we described. The analysts must bear in mind, however, that any automated tool can provide only limited support and should not replace careful evaluation conducted under close collaboration with domain experts, especially when direct semantic features are unavailable for the automated analysis, as even human analysts cannot get all the semantic correspondences right in such “hard” situations without collaborating with domain experts.

The techniques we have described in this chapter are useful for detecting schema correspondences across data sources. Another related problem in heterogeneous database integration is identification of instance correspondences (i.e., records that represent the same entity in the real world) (Zhao & Ram, 2005). After some instance correspondences have been identified and data from heterogeneous databases linked or integrated, statistical analysis techniques, such as correlation and regression, can be used to evaluate schema correspondences more accurately (Fan et al., 2001, 2002; Lu et al., 1997). Correspondences previously identified in cluster analysis can be verified. Other possible combinations of attributes can be explored to detect missed potential correspondences. Furthermore, improved understanding of schema correspondences can then trigger another iteration of detecting instance correspondences, followed by analysis of schema correspondences, thus forming an iterative procedure (Ram & Zhao, 2001; Zhao, 2005), in which correspondences on the schema level and the instance level are identified alternately and incrementally. Such an iterative procedure needs to be further investigated.

When there are many databases that need to be compared, the proposed method can be combined with machine-learning techniques (Berlin & Motro, 2002; Doan,

Domingos, & Halevy, 2003) to improve efficiency and scalability. The proposed method can be used first to identify attribute correspondences across several databases. These identified correspondences can then be used as training examples to train various classifiers, which are then applied on the remaining databases.

NOTE

An earlier version of the material in this chapter appeared in Zhao and Ram (2004).

REFERENCES

- Afifi, A. A., & Clark, V. (1996). *Computer-aided multivariate analysis* (3rd ed.). New York: Chapman & Hall.
- Ambrosio, A. P., Métais, E., & Meunier, J. (1997). The linguistic level: Contribution for conceptual design, view integration, reuse and documentation. *Data & Knowledge Engineering*, 21(2), 111-129.
- Benkley, S. S., Fandozzi, J. F., Housman, E. M., & Woodhouse, G. M. (1995). *Data Element Tool-based Analysis (DELTA)* (Tech. Rep. No. MTR 95B0000147). Bedford, MA: The MITRE Corporation.
- Berlin, J., & Motro, A. (2002, May 27-31). Database schema matching using machine learning with feature selection. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering*, Toronto, Canada (LNCS 2348, pp. 452-466). Berlin; Heidelberg, Germany: Springer.
- Bright, M. W., Hurson, A. R., & Pakzad, S. H. (1994). Automated resolution of semantic heterogeneity in multidatabases. *ACM Transactions on Database Systems*, 19(2), 212-253.
- Clifton, C., Housman, E., & Rosenthal, A. (1997, October 7-10). Experience with a combined approach to attribute-matching across heterogeneous databases. In *Proceedings of the 7th IFIP 2.6 Working Conference on Data Semantics (DS-7)*, Leysin, Switzerland (pp. 429-451). London: Chapman and Hall.
- Costa, J. A. F., & de Andrade Netto, M. L. (1999). Estimating the number of clusters in multivariate data by self-organizing maps. *International Journal of Neural Systems*, 9(3), 195-202.
- Do, H., Melnik, S., & Rahm, E. (2002, October 7-10). Comparison of schema matching evaluations. In *Proceedings of the 2nd International Workshop on Web Databases (German Informatics Society)*, Erfurt, Germany (LNCS 2593, pp. 221-237). London: Springer.
- Doan, A., Domingos, P., & Halevy, A. (2003). Learning to match the schemas of databases: A multistrategy approach. *Machine Learning*, 50(3), 279-301.
- Duwairi, R. M. (2004). Clustering semantically related classes in a heterogeneous multidatabase system. *Information Sciences*, 162(3-4), 193-210.
- Ellmer, E., Huemer, C., Merkl, D., & Pernul, G. (1996, September 9-13). Automatic classification of semantic concepts in view specifications. In *Proceedings of the 7th International Conference on Database and Expert Systems Applications*, Zurich, Switzerland (LNCS 1134, pp. 824-833). New York: Springer-Verlag.

- Everitt, B. S., Landau, S., & Leese, M. (2001). *Cluster analysis* (4th ed.). Arnold; London: Oxford University Press.
- Fan, W., Lu, H., Madnick, S. E., & Cheung, D. W. (2001). Discovering and reconciling value conflicts for numerical data integration. *Information Systems*, 26(8), 635-656.
- Fan, W., Lu, H., Madnick, S. E., & Cheung, D. W. (2002). DIRECT: A system for mining data value conversion rules from disparate sources. *Decision Support Systems*, 34(1), 19-39.
- Giunchiglia, F., & Yatskevich, M. (2004, November 8). Element level semantic matching. In *Proceedings of Meaning Coordination and Negotiation Workshop at ISWC*, Hiroshima, Japan (pp. 37-48).
- Hansen, M., Madnick, S., & Siegel, M. (2002, May 28). Data integration using web services. In *Proceedings of International Workshop on Data Integration over the Web*, Toronto, Canada (pp. 3-16). Toronto, Canada: University of Toronto Press.
- Hayne, S., & Ram, S. (1990, February 5-9). Multi-user view integration system (MUVIS): An expert system for view integration. In *Proceedings of the Sixth International Conference on Data Engineering*, Los Angeles, CA (pp. 402-410). Los Alamitos, CA: IEEE Computer Society Press.
- Johannesson, P. (1997). Supporting schema integration by linguistic instruments. *Data & Knowledge Engineering*, 21(2), 165-182.
- Kang, J., & Naughton, J. F. (2003, June 9-12). On schema matching with opaque column names and data values. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, San Diego, CA (p. 205-216). New York: ACM Press.
- Kohonen, T. (2001). *Self-organizing maps* (3rd ed.). Berlin: Springer.
- Li, W. S., & Clifton, C. (2000). SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data & Knowledge Engineering*, 33(1), 49-84.
- Lu, H., Fan, W., Goh, C. H., Madnick, S. E., & Cheng, D. W. (1997, October 7-10). Discovering and reconciling semantic conflicts: A data mining perspective. In *Proceedings of the 7th IFIP 2.6 Working Conference on Data Semantics (DS-7)*, Leysin, Switzerland (pp. 410-427). London: Chapman and Hall.
- Madhavan, J., Bernstein, P. A., & Rahm, E. (2001, September 11-14). Generic schema matching with Cupid. In *Proceedings of the 27th International Conferences on Very Large Databases*, Roma, Italy (pp. 49-58). San Francisco: Morgan Kaufmann.
- Mangiameli, P., Chen, S. K., & West, D. (1996). A comparison of SOM neural network and hierarchical clustering methods. *European Journal of Operational Research*, 93(2), 402-417.
- Masood, N., & Eaglestone, B. (1998, August 24-28). Semantics based schema analysis. In *Proceedings of the 9th International Conference on Database and Expert Systems Applications*, Vienna, Austria (pp. 80-89). London: Springer-Verlag.
- Mirbel, I. (1997). Semantic integration of conceptual schemas. *Data & Knowledge Engineering*, 21(2), 183-195.
- Navathe, S., Thomas, H., Satitsamitpong, M., & Datta, A. (2001, April 25-28). A model to support e-catalog integration. In *Proceedings of the 9th IFIP 2.6 Working Conference on Database Semantics (DS-9)*, Hong Kong (pp. 247-261). Deventer, The Netherlands: Kluwer Academic Publisher.

- Palopoli, L., Pontieri, L., Terracina, G., & Ursino, D. (2000). Intentional and extensional integration and abstraction of heterogeneous databases. *Data & Knowledge Engineering*, 35(3), 201-237.
- Palopoli, L., Sacca, D., Terracina, G., & Ursino, D. (2003). Uniform techniques for deriving similarities of objects and subschemes in heterogeneous databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(2), 271-294.
- Petersohn, H. (1998). Assessment of cluster analysis and self-organizing maps. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2), 136-149.
- Rahm, E., & Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, 10, 334-350.
- Ram, S., Park, J., Kim, K., & Hwang, Y. (1999, December 11-12). A comprehensive framework for classifying data- and schema-level semantic conflicts in geographic and non-geographic databases. In *Proceedings of the 9th Annual Workshop on Information Technologies and Systems*, Charlotte, NC (pp. 185-190).
- Ram, S., & Venkataraman, R. (1999). Schema integration: past, present and future. In: A. Elmagarmid, M. Rusinkiewicz, & A. Sheth (Eds.), *Management of heterogeneous and autonomous database system* (pp. 119-156). San Francisco: Morgan Kaufmann.
- Ram, S., & Zhao, H. (2001, December 15-16). Detecting both schema-level and instance-level correspondences for the integration of e-catalogs. In *Proceedings of the 11th Annual Workshop on Information Technology and Systems*, New Orleans, LA (pp. 193-198).
- Rodríguez, M. A., Egenhofer, M. J., & Rugg, R. D. (1999, March 10-12). Assessing semantic similarities among geospatial feature class definitions. In *Proceedings of the 2nd International Conference on Interoperating Geographic Information Systems*, Zürich, Switzerland (pp. 189-202). New York: Springer-Verlag.
- Seligman, L., Rosenthal, A., Lehner, P., & Smith, A. (2002). Data integration: Where does the time go? *IEEE Data Engineering Bulletin*, 25(3), 3-10.
- Song, W. W., Johannesson, P., & Bubenko, J. A. (1996). Semantic similarity relations and computation in schema integration. *Data & Knowledge Engineering*, 19(1), 65-97.
- Srinivasan, U., Ngu, A. H. H., & Gedeon, T. (2000). Managing heterogeneous information systems through discovery and retrieval of generic concepts. *Journal of the American Society for Information Science*, 51(8), 707-723.
- Stephen, G. A. (1994). *String searching algorithms*. Singapore: World Scientific Publishing Co. Pte. Ltd.
- Zhao, H. (2005). Semantic matching across heterogeneous data sources. *Communications of the ACM*, forthcoming.
- Zhao, H., & Ram, S. (2004). Clustering schema elements for semantic integration of heterogeneous data sources. *Journal of Database Management*, 15(4), 88-106.
- Zhao, H., & Ram, S. (2005). Entity identification for heterogeneous database integration — A multiple classifier system approach and empirical evaluation. *Information Systems*, 30(2), 119-132.

Chapter XIV

An Efficient Concurrency Control Algorithm for High-Dimensional Index Structures

Seok Il Song, Chungju National University, Korea

Jae Soo Yoo, Chungbuk National University, Korea

ABSTRACT

This chapter introduces a concurrency control algorithm based on link-technique for high-dimensional index structures. In high-dimensional index structures, search operations are generally more frequent than insert or delete operations and need to access many more nodes than those in other index structures, such as B^+ -tree, B-tree, hashing techniques, and so on, due to the properties of queries. This chapter proposed an algorithm that minimizes the delay of search operations in all cases. The proposed algorithm also supports concurrency control on reinsert operations for the high-dimensional index structures employing reinsert operations to improve their performance. The authors hope that this chapter will give helpful information for studying multidimensional index structures and their concurrency control problems to researchers.

INTRODUCTION

In the past couple of decades, multi-dimensional index structures have become the crucial component of multi-dimensional feature vectors-based similarity search systems

such as GIS, content-based image retrieval systems, multimedia database systems, moving object database systems, and so on. To satisfy the requirements of the modern database applications, various multi-dimensional index structures have been proposed. There are space-partitioning methods like Grid-file (Nievergelt, Hinterberger, & Sevcik, 1984), K-D-B-tree (Robinson, 1981), and Quad-tree (Finkel & Bentley, 1974) that divide the data space along predefined or predetermined lines regardless of data distributions. On the other hand, R-tree (Guttman, 1984), R+-tree (Sellis, Roussopoulos, & Faloutsos, 1987), R*-tree (Beckmann, Kornacker, Schneider, & Seeger, 1990), X-tree (Berchtold, Keim, & Kriegel, 1996), SR-tree (Katayama & Satoh, 1997), M-tree (Ciaccia, Patella, & Zezula, 1997), TV-tree (Lin, Jagadish, & Faloutsos, 1994), and CIR-tree (Yoo et al., 1998) are data-partitioning index structures that divide the data space according to the distribution of data objects inserted or loaded into the tree. In addition, Hybrid-tree (Chakrabarti & Mehrotra, 1999a) is a hybrid approach of data-partitioning and space-partitioning methods; VA-file (Weber, Schek, & Blott, 1998) uses flat-file structure, and that described by Indyk and Motwani (1998) uses hashing techniques.

In order for the multi-dimensional index structures to support the modern database applications, they should be integrated into existing database systems. Even though the integration is an important and practical issue, not much previous work on it exists. To integrate an access method into a data base management system (DBMS), we must consider two problems, namely, concurrency control and recovery. The concurrency control mechanism contains two independent problems. First, techniques must be developed to ensure the consistency of the data structure in the presence of concurrent insertions, deletions, and updates. Several methods that use lock-coupling techniques and link techniques have been proposed for multi-dimensional index structures (Chen & Huang, 1997; Kornacker & Banks, 1995; Kornacker, Mohan & Hellerstein, 1997; Ng & Kamada, 1993; Ravi, Kanth, Serena & Singh, 1998; Song, Kim, & Yoo, 2004). Second, phantom protection methods that protect searchers' predicates from subsequent insertions, and the rollbacks of deletions before the searchers commit must be developed (Chakrabarti & Mehrotra, 1998; Chakrabarti & Mehrotra, 1999b; Kornacker, Mohan, & Hellerstein, 1997).

In this chapter, we propose a concurrency control method that ensures the consistency of the data structure in presence of multiple running transactions. Concurrency control methods for multi-dimensional index structures should consider the different properties of multi-dimensional index structures from B+-tree or B-tree. Usually, multi-dimensional index structures used as access methods in the similarity search system have the following properties:

- First, search operations are generally more frequent than insert or delete operations.
- Second, when processing the search operations, they need to access many more nodes than other index structures, such as B⁺-Tree, B-Tree, hashing techniques, and so on, due to the characteristics of queries (Range Search, K-NN Search).
- Finally, some of them employ forced reinsert operations to reorganize index structures efficiently and to gain high search performance.

We need to add the above properties to the design requirements of the concurrency control algorithm of multi-dimensional index structures.

We propose a concurrency control algorithm for multi-dimensional index structures that considers reinsert operations and focuses on minimizing the delay of search operations at all cases. Also, we apply the algorithm to the CIR-Tree and implement it on MiDAS-b!, which is the storage system of a multimedia DBMS, called BADA-3 (Chae et al., 1998). It is shown through experiments that our proposed method outperforms the existing concurrency control algorithm for GiST(CGiST) (Kornacker, Mohan, & Hellerstein, 1997).

This chapter is organized as follows. In the next section, we describe related works. Then, we describe the proposed concurrency control algorithm. An evaluation of the performance of the proposed method and the concurrency control algorithm of the CGiST through experiments is then presented. Finally, we describe our conclusions.

RELATED WORK AND MOTIVATION

Multi-dimensional index structures as mentioned are in the R-tree family. They are height-balanced trees similar to the B-tree. In those index structures, leaf nodes contain index records of the form (BR, OID), where OID uniquely determines an object in the database and BR determines a bounding (hyper) rectangle of the indexed spatial object. Non-leaf nodes contain entries of the form (MBR, child-pointer), where child-pointer refers to the address of a lower node in the R-tree and MBR is the minimum bounding rectangle that contains the MBR of all of its children nodes. Before going further, we need to mention the concepts of latches and locks and their compatibility matrix to make the following explanation easy. Even though latches and locks are used to maintain consistency of index trees, they are slightly different. Both of them are used to control access to shared information. Latches are like semaphores. Generally, latches are used to guarantee physical consistency of data, while locks are used to assure the logical consistency of data. Latches are usually held for a much shorter period than locks are. Also, the deadlock detector cannot recognize latch waits, so it is impossible to detect deadlocks involving latches alone, or those involving latches and locks. Two lock and latch modes — shared mode and exclusive mode — are used in existing methods and our proposed algorithm. Table 1 and Table 2 show the compatible matrix of locks and latches referred to in this chapter, respectively.

In the following, we describe the existing concurrency control algorithms to maintain the physical consistency of multi-dimensional index structures and phantom protection methods.

Table 1. Latch compatibility matrix

	Shared latch (s-latch)	Exclusive latch (x-latch)
Shared latch (s-latch)	o	-
Exclusive latch (x-latch)	-	-

Table 2. Lock compatibility matrix

	Shared lock(s-lock)	Intention Shared lock (is-lock)	Intention Exclusive lock (ix-lock)	Exclusive lock (x-lock)
Shared lock(s-lock)	o	o	-	-
Intention Shared lock (is-lock)	o	o	o	-
Intention Exclusive lock (ix-lock)	-	o	o	-
Exclusive lock (x-lock)	-	-	-	-

Concurrency Control Algorithms to Maintain the Physical Consistency of Multi-Dimensional Index Structures

Several concurrency control algorithms for multi-dimensional index structures (Chen & Huang, 1997; Kornacker & Banks, 1995; Kornacker, Mohan, & Hellerstein, 1997; Ng & Kamada, 1993; Ravi, Kanth, Serena, & Singh, 1998; Song, Kim, & Yoo, 2004) have been proposed. They can be classified simply into link-based and lock coupling-based algorithms. The lock coupling-based algorithms (Chen & Huang, 1997; Ng & Kamada, 1993) release the lock on the current node when the lock on the next node to traverse is granted while processing search operations. While processing node splits and MBR updates, the scheme holds multiple locks simultaneously that significantly degrade concurrency.

On the other hand, the link-based algorithms (Kornacker & Banks, 1995; Kornacker, Mohan, & Hellerstein, 1997; Ravi, Kanth, Serena, & Singh, 1998; Song, Kim, & Yoo, 2004) were presented to solve the problems of lock coupling-based concurrency control algorithms. They need not perform lock coupling during traversing an index but just hold one lock at the time. However, while backing up trees for node splits and MBR updates, they employ lock-coupling, that is, they keep the child node write-locked until a write-lock on the parent is obtained.

The link technique, proposed by Lehmann and Yao (1981), was originally for B-tree. The tree structure is modified so that all nodes at the same level are chained together through a right-link on each node, which is a pointer to its right sibling node. When a node is split into two nodes, appropriate right links are assigned to both. All nodes in a right link chain on the same level are ordered by their highest keys. When a search process visits a node that was split and not yet propagated to the parent node, it detects that the highest key on that node is lower than the key it is looking for and correctly concludes that a split must have taken place. This guarantees that at most one lock is needed at any case, so insert operations can be performed without blocking search processes.

Unfortunately, in multi-dimensional index structures there is no such an ordering between nodes at the same level. For that reason, the algorithm proposed by Kornacker and Banks (1995) assigns a logical sequence number (LSN) at each node in addition to right links, and an entry associated with a node has the LSN of the node. The ordering of LSNs is used to compensate for a missed split. However, while ascending the trees to perform node splits and MBR updates, this algorithm employs lock coupling, that is, it keeps the child node write-locked until a write-lock on the parent is obtained. The lock on the child node may be kept during I/O time in certain cases. It degrades the concurrency degree of the index trees. Also in this algorithm, each entry of internal nodes has extra information to keep the LSNs of associated child nodes. This extra information reduces storage utilization.

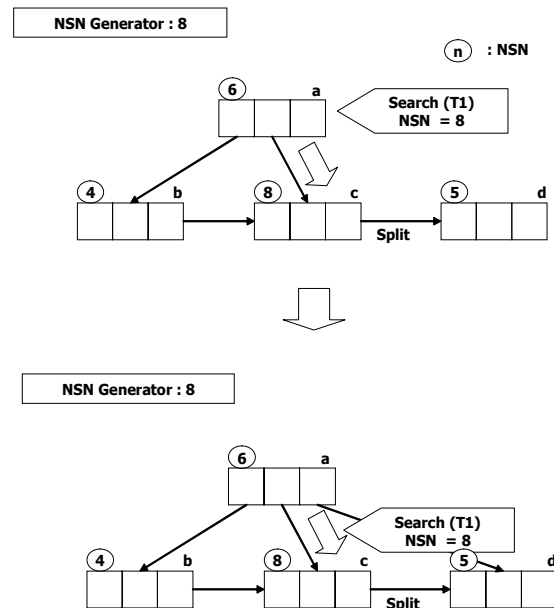
Another link technique-based concurrency control algorithm for multi-dimensional index structures was proposed by Kornacker, Mohan, and Hellerstein (1997). One of the major shortcomings of R^{Link} -tree is the additional information produced by adding the LSN to each entry in internal nodes. Kornacker, Mohan, and Hellerstein (1997) also assign a node sequence number (NSN), which is same as the LSN of Kornacker and Banks (1995), to every node and chains nodes on the same level with right links to detect missed splits. However, it eliminates the space overhead caused by LSN in internal entries. The NSN of Kornacker, Mohan, and Hellerstein (1997) is taken from a tree-global, monotonically increasing counter variable. During a node split, this counter is incremented and its new value assigned to the original node. The new sibling node receives the original node's prior NSN and right link. A traverser can detect a missed split by memorizing the global counter value when reading the parent entry and comparing it with the NSN of the current node. If the latter is higher, the node must have been split, so the traverser follows right links until it reaches a node with an NSN less than or equal to the NSN that was originally memorized.

However, the introduced global counter of Kornacker, Mohan, and Hellerstein (1997) has some side effects. In order for the algorithm to work correctly, when splitting a node, an inserter must acquire an x-lock on its parent node first, split the node, assign the NSN, increment global counter, and release the x-lock. Therefore, while the processing node splits, the inserters keep multiple locks on two levels. This affects search operations and explicitly increases the blocking time of searchers. Also, due to the recovery scheme proposed in Kornacker, Mohan, and Hellerstein (1997), x-latches are kept on nodes that are involved in splits or minimum bounding region (MBR) updates until the whole operation ends.

Figure 1 shows why CGiST must keep the x-latch on the parent node during a node split. If the x-latch is not placed on the parent node during split, searchers cannot detect the split. The following scenario is a simple example that illustrates this.

1. An inserter splits the node c into node c' and d without acquiring an x-latch on the parent node a .
2. A searcher reaches node a . At this time, the global NSN increases by 7 the split of node c .
3. The searcher goes down with the increased global NSN(8).
4. The inserter acquires an x-latch on the parent node a and releases latches on nodes c and d .
5. The searcher acquires an s-latch on node c and compares the NSN(8) to the NSN of node c .

Figure 1. Unrecognized split detection



6. The NSN(8) is equal to the NSN of node c' ; the searcher cannot detect the split of node c .

To prevent this situation, the inserter must acquire an x-latch on node a before splitting node c and increasing global NSN.

The concurrency control algorithms briefly explained above get multiple locks or latches exclusively on index nodes from multiple levels participating in node splits or MBR updates. The exclusive locks or latches block concurrent searchers. As the result, the overall search performance is extremely degenerated. Kanth, Serena, and Ambuj (1998) try to solve the problems mentioned in the previous paragraph. They introduce a top-down index region modification (TDIM) technique; that is, when an insert operation traverses an index tree to find the most suitable node for a new entry, MBR updates are performed. In addition, the locks that are placed on nodes during MBR updates are compatible with search operations. It is achieved by the modification of MBR in a piecemeal fashion. In addition to the TDIM technique, Kanth et al. (1998) propose optimized split algorithms such as copy-based concurrent update (CCU) and copy-based concurrent update with non-blocking queries (CCUNQ).

The TDIM technique has some problems. It eliminates the necessity of lock-coupling during insert operations (Kornacker & Banks, 1995; Kornacker, Mohan, & Hellerstein, 1997). However, to our knowledge it never considers deletes. Deleters need to perform an exact match like a tree traversal to find a target entry. Since multi-

dimensional index structures may have multiple paths from the root node to a target node that contains the target entry to be deleted, deleters can not assure that the node they are currently visiting is the correct ancestor of the target node. Consequently, to modify MBR in top-down fashion, we must modify MBR once completing the location of the target entry.

When deleters and inserters that use TDIM are performed concurrently, index trees may reach inconsistent states since TDIM does not perform lock-coupling. We can image the following situation easily. An inserter starts to insert a new entry (NE) to an index tree, visits an internal node (N), and chooses an entry (E) that is a pair of a pointer for a child node (CN) and the CN's MBR. The inserter concludes that the MBR of E does not need to be modified and proceeds with its tree traversal. Subsequently, a deleter visits N and modifies the MBR of CN of E. This MBR shrinking may exclude NE from the MBR of CN, and the index tree reaches an inconsistent state. Therefore, the TDIM can not be applied in real-life applications without modifying the MBR updates algorithm in some or most part because it can not handle delete operations that are necessary in real-life applications.

Also, the CCU and CCUNQ greatly reduce the delay of queries but they are not efficient. They need extra space to perform split operations, and the CCUNQ must perform garbage collection works periodically. These features make the implementation of the algorithm very difficult. The simplicity of an algorithm reduces the development costs.

To our knowledge, Song, Kim, and Yoo (2004) presents the most recent link-based concurrency control algorithm for multi-dimensional index structures. This work addresses some problems in achieving high performance in multi-dimensional index structures, as follows.

First, the entries of internal nodes are not usually ordered so calculating split dimensions and positions is expensive. Therefore, split operations of multi-dimensional index structures take a longer time than in uni-dimensional index structures such as B-tree and B+-tree. Most concurrency control algorithms for multi-dimensional index structures hold x-locks or x-latches on the nodes where split operations are being performed. These x-locks and x-latches block search operations during the whole split time. A split operation ascends an index tree to propagate split to ancestor nodes, and may cause another split on ancestor nodes. A split operation is one of the primary factors that deteriorate the concurrency of multi-dimensional index structures.

Second, minimum bounding region (MBR) update operations block search operations. The MBR update of a node is less expensive than a split operation. However, MBR updates are much more frequent than split operations, so they significantly deteriorate the concurrency of index structures. Even though several concurrency control algorithms have been proposed for multi-dimensional index structures, none of them can completely prevent the delay of search operations. Actually, it is impossible to eliminate the above search delay completely, but we can minimize it. Song, Kim, and Yoo (2004) introduce a partial lock-coupling (PLC) technique to decrease the search delay by MBR updates. To reduce blocking time by split operations, these authors propose a split method that optimizes x-latch time during node splits. Also, they address how to support the phantom protection in our algorithm.

All of the existing concurrency control algorithms briefly described above get multiple locks or latches exclusively on nodes from multiple levels participating in node splits and MBR updates. The exclusive locks block concurrent search operations. As a

result, overall search performance is degenerated. Also, since they do not consider reinsert operations at all, they cannot be applied to the multi-dimensional index structures using reinsert operations. In contrast, we propose a new concurrency control technique that focuses on reducing the blocking time of search operations with little sacrifice of insert performance and supports concurrency control methods for reinsert operations.

Phantom Protection Methods

Several matured phantom protection methods for B+-tree exist, for example, key-range locking (Mohan, 1990) and next-key locking (Mohan & Levine, 1992). They rely on the presence of a total order over the underlying data based on their key values. However, in multi-dimensional index structures, no such ordering between keys exists, so the existing phantom protection methods for B+-tree are not applicable. Therefore, the first developed phantom protection method for multi-dimensional index structures uses the modified predicate locking mechanism of Kornacker, Mohan, and Hellerstein (1997) instead of the techniques of B+-tree of Mohan, (1990) and Mohan and Levine (1992).

To our knowledge, the phantom protection method proposed by Kornacker, Mohan, and Hellerstein (1997) is the first such method. It addressed the above problems of a predicate-locking mechanism and proposed hybrid approaches that synthesize the two-phase locking of data record with predicate locking. In the hybrid mechanism, data records that are scanned, inserted, or deleted are protected by the two-phase locking protocol. In addition, searchers set predicate locks to prevent phantoms. Furthermore, the predicate locks are not registered in a tree-global list before the searcher starts traversing the tree. Instead, it is directly attached to nodes.

Predicate attachments are performed so that the following invariant is true at all times. If a searcher's predicate overlaps a node's MBR, the predicate must be attached to the node. An inserter checks only the predicates attached to its target leaf. A deleter performs a logical delete, that is, a leaf entry is not physically deleted but is only marked as deleted. Searchers attach their predicates to the nodes that they visit. The predicates of the nodes are only removed when the the owner transactions commit. Since the tree structure changes dynamically as nodes split and MBRs are expanded during key insertions, the attached predicates have to adapt to the structural changes. In order to handle this problem, Kornacker, Mohan, and Hellerstein (1997) replicate existing predicates to newly overlapped nodes by the structural changes. Possible structural changes are node splits and MBR updates.

The first case is a node split, which creates a new node whose MBR might be consistent with some of the predicates attached to the original node. The invariant is maintained by attaching those predicates to the new node. The second case involves the expansion of a node's MBR, causing it to become consistent with additional search predicates. The additional search predicates at other nodes must be attached to the node. The updater that expanded the MBR must traverse tree to find predicates.

The hybrid mechanism of Kornacker, Mohan, and Hellerstein (1997) has some drawbacks. First, each node of the index trees has an additional space for a predicate table consisting of predicates of searchers, inserters, and deleters. The size of the table is variable, and the contents of the table must be changed whenever the MBR updates or node splits are performed. These properties make the maintenance of predicate tables expensive. Second, the lock range is not expanded gradually because predicates have

to be attached to the visited nodes top-down, starting with the root. This can block an insertion into the search range, even if the leaf where the insertion takes place has not been visited by the search operation.

To overcome the shortcomings of the hybrid mechanism of Kornacker, Mohan, and Hellerstein (1997), Chakrabarti and Mehrotra (1998, 1999b) have proposed a granular-locking approach. The predicate locking offers potentially higher concurrency; typically, the granular locking is preferred since the lock overhead of a predicate-locking approach is much higher than that of a granular-locking approach. Chakrabarti and Mehrotra (1998, 1999b) define the lowest level MBRs as the lockable granules. Each lowest level MBR corresponds to a leaf node of the R-tree. The granules dynamically grow and shrink with insertions and deletions of entries to adapt data space to the distribution of the objects. The lowest level MBRs alone may not fully cover the embedded space, that is, the set of granules may not be able to properly protect search predicates resulting in phantoms. Accordingly, they define additional granules called external granules for each non-leaf node in the tree, such that the lowest level MBRs together with the external granules fully cover the embedded space.

Updaters (inserters and deleters) acquire ix-locks on a minimal set of granules sufficient to fully cover the object followed by an x-lock on the object itself. Searchers acquire s-locks on all granules that overlap with the predicate being scanned. In this strategy, the insertion of an object that overlaps with the search region of a query is not permitted to execute concurrently, thereby preventing phantoms from arising. This strategy is referred to as the cover-for-insert and overlap-for-search policy. The reverse policy could also be followed; namely, overlap-for-insert and cover-for-search, in which ix-locks are acquired on all overlapping granules for inserters and deleters and s-locks are acquired on the minimal set of granules that cover the scan predicate for search.

However, the above two locking policies are not sufficient to prevent phantoms from arising when the granules are dynamically changing due to insertions and deletions. Therefore, some additional locking strategies are proposed. The ultimate lock protocols are summarized as follows. First, inserters acquire ix-locks on all granules that contain the newly inserted object. If the MBR of a node is changed by a new entry, they obtain short duration ix-locks on all overlapping nodes. If overflow occurs, they acquire six-lock on the overflowed node before split, and acquire ix-locks on the original node and the newly created node after split and s-lock on its parent node's external granule. Second, searchers obtain s-locks on all overlapping granules with the search predicate. Finally, deleters acquire ix-locks on all granules that contain the object to be deleted when logically deleting it and physically obtain short duration ix-locks on the granule that contains the object when deleting the entry.

The granular-locking mechanism is much more efficient than the predicate-locking mechanism. The lockable granules are nodes of the index trees so it uses the existing object locking mechanism of database systems. Also, unlike predicate locking mechanism, it does not need to maintain additional information at each node for storing predicates. However, when the granules are changed or overflow occurs, it must acquire ix-locks on all nodes overlapped with the object. This requires inserters to traverse the index tree from its root to find overlapping nodes. Since it acquires locks on index nodes, it is difficult to integrate with existing concurrency control algorithms because of the locks conflicting purposes.

THE PROPOSED CONCURRENCY CONTROL ALGORITHM

Detection of an Unrecognized Split

CGiST uses global NSN to detect unrecognized node split and to get rid of LSNs assigned at each internal node of R^{Link} -tree (Kornacker & Banks, 1995). However, global NSN is accompanied by side effects. As described in the second chapter, it must keep an x-latch on current node's parent node during split the current node. In this chapter, we introduce *max_child_nsn*. It is assigned to each internal node. Then, when a node is split, the *max_child_nsn* of the parent node of the split node is replaced with the *nsn* of the split node. Since a split operation must add an internal entry for a newly created node to its parent node, we can ensure that *max_child_nsn* always is the maximum *nsn* of child nodes.

When a transaction traverses an index tree to find a leaf node for a new entry to be inserted or to process a query, it compares the parent node's *max_child_nsn* to the current visiting node's *nsn*. If the *max_child_nsn* of the parent node is smaller than the *nsn* of the current node, it traverses the right link. Otherwise, it goes down to the next child node. Figure 2 shows the pseudo code of the above process. With this algorithm, we do not need to keep an x-latch on the parent node during node split because the *max_child_nsn* always is increased after the node split is completed and *max_child_nsn* is local to an internal node.

Properties of Our Proposed Algorithm

The properties of the proposed algorithm can be summarized as follows. First, the proposed algorithm is based on the link technique. The link technique used in this chapter is from Kornacker, Mohan, and Hellerstein (1997), which introduces the global counter as the method for reducing the extra information of internal node entries. Second, the proposed algorithm supports concurrency control methods for reinsert operations by using reinsert nodes. The reinsert operation was proposed originally in R^* -tree by Beckmann et al. (1990). To achieve dynamic reorganizations, the R^* -tree forces entries to be reinserted during the insertion routine. Consequently, the result is a performance improvement of 20% to 50%. Several index structures such as TV-tree, SS-tree, SR-tree,

Figure 2. Pseudo code of detecting an unrecognized split

```

parent_max_child_nsn = current_node.max_child_node;
decide the child node to traverse;
if (parent_max_child_nsn < current_node.nsn )
    traverse the right link;
else
    decide the child node to traverse;
  
```


and CIR-tree employ the forced reinsert to improve search performance. Especially, CIR-tree proposed an improved reinsert algorithm that uses weighted center to select entries to be reinserted (Yoo et al., 1998). Existing concurrency control algorithms do not consider the reinsert operation seriously. To perform reinsert operations, the entries to be reinserted should first be removed from index trees. After that, other search operations cannot recognize the entries until the entries are inserted again. This may cause search operations to fail.

In our proposed algorithm, the removed reinsert entries are stored in a reinsert node that can be shared by other transactions. The reinsert node is allocated outside the tree structure. Figure 3 shows the structure of a reinsert node. The first entry of the reinsert node consists of a node identifier, the MBR that covers the reinsert entries, and the level where the reinsert operation is being performed. When a search operation traverses the tree, it visits the reinsert node and compares the MBR of the reinsert node with search predicates. If the MBR satisfies the search predicates, the search operation accesses the reinsert entries.

Third, we use latches and locks on the index nodes. The latches on index nodes synchronize transactions accessing an index node concurrently and guarantee the physical consistency of the index node. The locks on index nodes solve the path-loss problem caused by reinsert operations. To perform reinsert operations, the entries to be reinserted first should be removed from index trees. When MBR updates or node splits are performed in the sub-tree of the internal node on which the reinsert operation is performed, the path-loss problem occurs. The situation is depicted in Figure 4.

As we can see from Figure 4, the transaction T_1 may be lost the path to ascend by the reinsert operation of transaction T_2 . To solve the path-loss problem, a transaction performing insert operations gets s-locks besides latches on index nodes that the transaction visits, and releases the obtained s-locks when finishing the insert operation. On the other hand, before a transaction performs the reinsert operation on a node, the transaction must get x-lock on the node. This scheme solves the path-loss problem because the transaction trying to perform the reinsert operation on the node cannot get x-locks if other transactions are performing insert operations in the sub-tree rooted at the node. The lock on the root node, called a tree lock, has special meaning. It is used as a tree lock that serializes structure modification operations such as node splits and MBR updates.

Figure 3. Structure of a reinsert node

Node ID (where reinsert being performed)	MBR (covers reinsert entries)	Level (the level of Node ID)
Reinsert Entries		

Figure 4. Path-loss problem

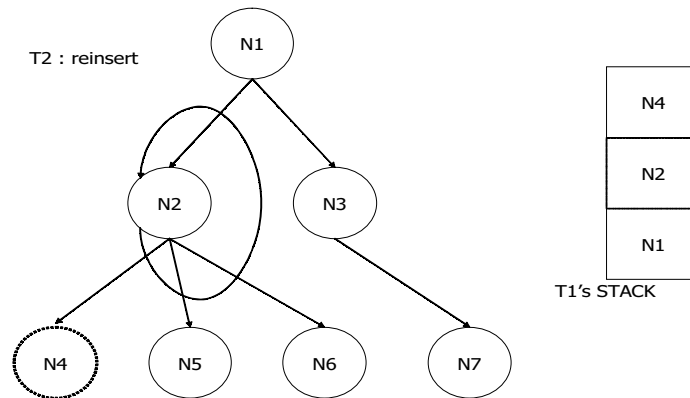


Figure 5. Pseudo code for the insert operation - InsertEntry

```

Function InsertEntry( Entry leafentry, Node rootnode )

Start Function
  leafnode = FindNode(leafentry, root, path, level);
  If ( overflow occurred in leafnode due to leafentry )
    Obtain tree lock conditionally (if failed, release x-latch on leafnode and request
    unconditionally);
    TreatOverflow(leafentry, leafnode, path);
    Release tree lock;
    Release all locks;
    End Function;
  End If
  Add leafentry to leafnode;
  If ( the MBR of leafnode is changed )
    Obtain tree lock;
    Release x-latch on leafnode;
    FixMBR( leafnode, path );
    Release tree lock;
    Release all locks;
    End Function;
  End If
  Release all locks;
End Function

```

Figure 6. Pseudo code for the insert operation - FindNode

```

Function FindNode( Entry entry, Node node, PathStack path, Level level )

Start Function
Obtain s-latch on node;
currentlevel = node.level;
Push [node, node.nsn] into path;
Start Loop
  Select the child entry childentry[Node node, MBR mbr] from node;
  node = childentry.node;
  Subtract 1 from currentlevel;
  Release s-latch on node;
  If ( currentlevel == level )
    Obtain x-lock and x-latch on node;
  Otherwise
    Obtain s-lock and s-latch on node;
  End If
  node = use the check module of Figure 2 on page 7;
  If ( currentlevel == level )
    Exit Loop;
  End If
  Push [node, node.nsn] into path;
End Loop;
End Function

```

Finally, the proposed algorithm always guarantees that insert operations keep x-latches simultaneously on nodes from only one level in all cases. Even though we employ the link technique of Kornacker, Mohan, and Hellerstein (1997), we do not need to obtain a latch on the parent node before splitting the current node while processing node split, and lock coupling is no longer necessary while processing MBR updates since node splits and MBR updates are serialized through tree locks. In our algorithm, search operations are blocked by only x-latches so the delay time of search operations is reduced.

Insert Operation

Figures 5, 6, 7, and 8 show the pseudo code of the insert operation of concurrency control algorithm proposed in this chapter. Our insert algorithm (*InsertEntry*) consists of *FindNode*, *TreatOverflow* and *FixMBR*. The individual procedures are described in the following. The function *FindNode* in Figure 6 descends to the leaf node to insert a new entry, obtaining s-latch on internal nodes and recording the path along the way, and finally gets exclusive latches on the leaf. The function *TreatOverflow* in Figure 7 describes a situation where the leaf node does not have enough room to accommodate the new entry. It performs reinsert or split to cope with the overflow as described above

Figure 7. Pseudo code for the insert operation - *TreatOverflow*

```

Function TreatOverflow( Node node, PathStack path )

Start Function
Obtain x-lock on node conditionally;
If ( success to obtain x-lock on node )
    Obtain s-latch on node, select entries to be reinserted, and release the s-latch;
    Obtain x-latch on ReinsertNode, and copy the entries to it, release x-latch;
    latch;
    Obtain x-latch on node, delete entries, and release x-latch;
    Perform reinsert operations (insert all of entries into index);
    If ( the result of reinsert operation does not cause overflow )
        End Function;
    End If
End If
Obtain x-latch on node, split node to node and newnode;
Assign sibling pointer value of node to newnode. Set sibling pointer to newnode;
Assign node.nsn of node to newnode;
Increase global_nsn and install its values as the node.nsn.;
Create an internal entry internalentry[newnode, mbr];
parentnode = POP( path );
Obtain x-latch, and modify the mbr for node in parent parentnode;
If ( overflow occurred in parentnode )
    TreatOverflow(internalentry, parentnode, path );
End If
Add internalentry to parentnode;
If ( the MBR of parentnode is changed )
    Release x-latch on parentnode;
    FixMBR( parentnode, path );
End If
End Function

```

Figure 8. Pseudo code for the insert operation - *FixMBR*

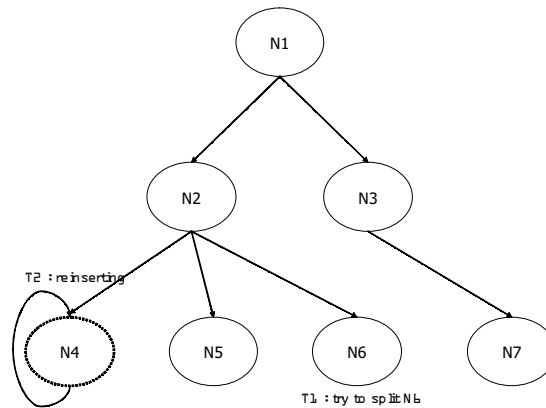
```

Function FixMBR ( Node node, PathStack path )

Start Function
parentnode = POP( path );
Obtain x-latch, and modify the mbr for node in parent parentnode;
Release x-latch;
If ( the MBR of parentnode is changed )
    FixMBR ( parentnode, path );
End If
End Function

```

Figure 9. Example of obtaining a tree lock



recursively. The function *FixMBR* in Figure 8 is used to propagate the changed MBR when the leaf's MBR is changed, and after a leaf is split into an old leaf node and a new leaf node to propagate the changed MBR of the old leaf node.

An insert operation is carried out in two stages. In the first stage, we traverse the tree from the root node to find the leaf node for the new entry to be inserted. In this stage, we store the path we take while descending the tree to a stack, called the path stack. In the second stage, the new entry is inserted to the leaf node. If the leaf's MBR has been changed, after adding the new entry to the node, we propagate the changed MBR to its ancestor nodes until we reach a node that no longer needs to be changed. On the other hand, a reinsert operation proceeds in the leaf node if it does not have enough room to accommodate the new entry. After performing the reinsert operation, if the leaf node is still full, we split the node. If the leaf node is split, we must insert a new internal entry in the parent node and modify the MBR for the split node. If overflow occurs in the parent node recursively, we determine if the reinsert operation is able to be performed in the node, that is, we request a conditional x-lock on the node and if it is accepted, we perform the reinsert operation. In contrast, if the reinsert operation is not possible, we split the parent node. The above steps are repeated until we reach a node with enough room to accommodate the new entry or split the root.

As previously described, MBR updates, splits, and reinserts are serialized by a tree lock. Before performing splits, reinserts or MBR updates, we must obtain an exclusive tree lock first. Obtaining a tree lock is not a trivial matter. If we request an exclusive tree lock unconditionally, deadlock may occur. For example, in Figure 9, the transaction T_1 that keeps x-latch on node N_6 requests an exclusive tree lock unconditionally to split the node N_6 . The transaction T_2 is concurrently performing a reinsert operation in node N_4 . The T_2 requests x-latch on node N_6 to insert one of the reinsert entries into it. However, the T_1 that is keeping x-latch on N_6 is waiting for an exclusive tree lock. T_2 never obtains x-latch on N_6 . This situation is a deadlock. Therefore, we always request an exclusive tree lock conditionally. If a transaction fails to get tree lock, it releases x-latch on leaf node

and requests exclusive tree lock unconditionally. Since MBR updates, splits and reinserts are serialized by tree lock, inserters do not need to employ latch-coupling as described by Kornacker et al. (1997) and Kornacker and Banks (1995) when ascending trees. Therefore, the blocking overhead of queries by insert operations is reduced.

Search Operation

We acquire latches on index nodes instead of locks to guarantee the physical consistency of data. Processing search operations in the proposed algorithm is the same as those of Kornacker et al. (1997) and Kornacker and Banks (1995) in a normal case. When an index tree employs reinsert operations, the search operation should be modified. Because the reinsert entries are stored on the reinsert node, the algorithm of search operation must be modified properly to be able to access the reinsert node. When the search operations reference the reinsert node, they first have to get latches on the node.

Figure 10. Pseudo code of search operation

Function RangeSearch (QueryFeatureVector *qfv*, Range *r*)

Start Function

Queue *queue*;

PriorityQueue *resultset*;

Push *root* to *queue*;

currentlevel = *currentnode.level* ;

Obtain s-latch on *reinsertnode*;

If (reinsert operation is busy)

reinsertlevel = *reinsertnode.level* ;

End If

Loop

 If (*queue* is empty)

 Exit Loop;

 End If

currentnode = POP(*queue*);

 Obtain s-latch on *currentnode*;

 If (*currentlevel* > *currentnode.level*)

currentlevel = *currentnode.level*;

 If (*currentlevel* == *reinsertlevel*)

 Push *entries* of *reinsertnode* within *r* to *queue*;

 End If

 End If

 If (*currentnode* != *leaf*)

 Push *entries* of *currentnode* within *r* to *queue*;

 Else If

 Push *entries* of *currentnode* within *r* to *resultset*;

 End If

 Release s-latch on *currentnode*;

Exit Loop;

Release s-latch on *reinsertnode*;

End Function

To access the reinsert node more efficiently, our algorithm does not use a depth-first search scheme but rather a breadth-first search scheme. Figure 10 shows the pseudo code of range search operations. Like the algorithm of insert operations, when a search operation visits a node, it uses the module in Figure 2 and pushes entries in the *r* to *queue* or *resultset*. Also, we can use the K-NN query of Seidl and Kriegel (1998) with some modifications for reinsert operations as illustrated in Figure 10. We omit the pseudo code of K-NN query for brevity.

PERFORMANCE EVALUATION

We implemented concurrency control algorithms such as RPLC and CGiST and phantom protection methods such as our method and a granular-locking method based on MIDAS (a multi process storage system for BADA DBMS). To evaluate the RPLC, the phantom protection part of CGIST was omitted from implementations for fairness. CGIST employs a hybrid mechanism, which synthesizes two-phase locking of data records with predicate locking. It maintains predicates at nodes of index trees instead of global area. Searchers attach their predicates to the nodes that are overlapped with their predicates and set predicate locks. The attached predicates at each node must be maintained during node splits and MBR updates. Inserters are blocked by the attached predicates. We eliminated all of those phantom protection actions from CGIST when implementing it on MIDAS. Also, we did not implement the signal-lock method of CGIST that prevents invalid pointer by node deletions. The two concurrency control algorithms are implemented with locks, latches, and the logging application program interfaces (APIs) of MIDAS.

Our experiments were performed for various sized data sets and various performance parameters such as node size, number of MiDAS page buffers, number of data items, and so on. Table 3 shows the notations, the descriptions, and the values of the performance parameters. To save space, we discuss the performance comparison only when 100,000 real data with 9-dimensional feature vectors are used, the node size is 16 Kbytes, and the number of page buffers is 120, because the experimental results of most of cases are very similar. The platform used in our experiments was dual Ultra Sparc. processor, Solaris 2.5 with 128 Mbytes of main memory. The maximum number of

Table 3. Performance parameters

Parameters	Descriptions	Values
DS	Database Size	50000 ~ 300000
NS	Node Size	4K ~ 16K
NP	Number of Page Buffers	40 ~ 120
ND	Number of Dimension	8 ~ 12,
K	Number of the K of K-NN Queries	5 ~ 20
DST	Distribution of data set	Real, normal, uniform

Figure 11. Response time of search operations (K of $KNN = 10$, $MPL = 40$, data size = 50K)

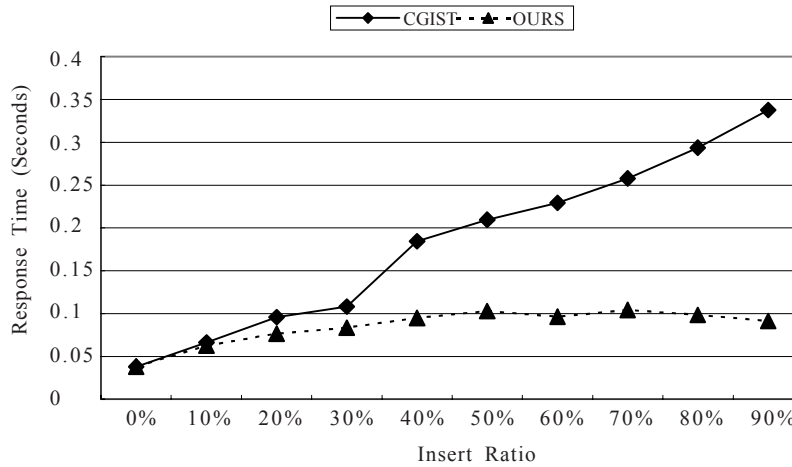
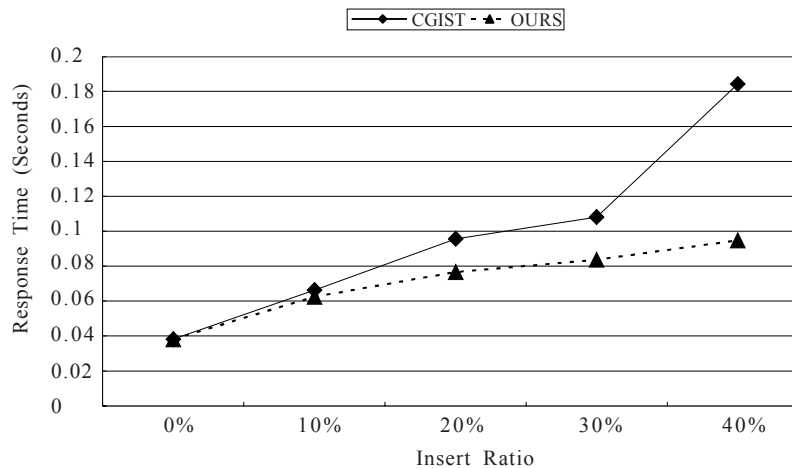
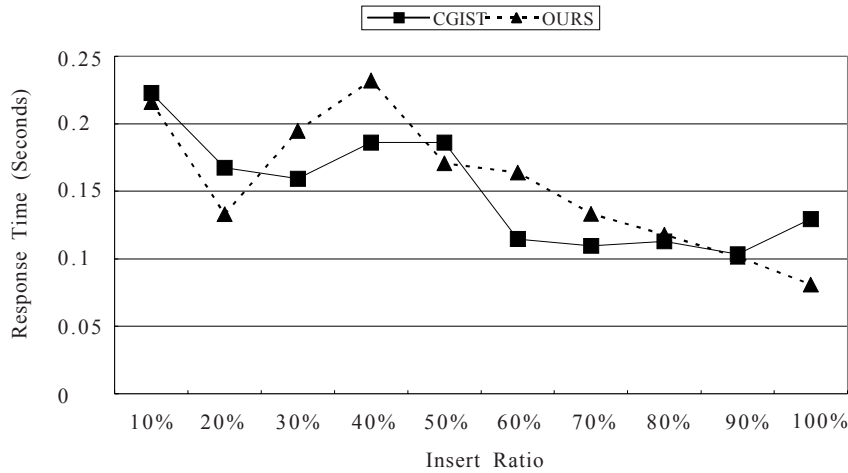


Figure 12. Response time of search operations through 10% ~ 40% insert ratio (K of $KNN = 10$, $MPL = 40$, data size = 50K)



concurrent processes is 80. We experimented with different workloads of insert and search operations. We fixed the number of concurrent processes at 80 and varied the ratios of the insert processes to the search processes from 0% to 100%. Also, we performed experiments with varying multi programming levels (MPL) ranging from 20 to 80. We did not perform comparisons of reinsert operations because the CGiST does not support a concurrency control mechanism for reinsert operations.

Figure 13. Response time of insert operations (K of KNN = 10, MPL = 40, data size = 50K)



Initially, CIR-trees were constructed by bulk-loading techniques. Subsequently, feature vectors were inserted concurrently by multiple processes under certain workload. According to the input parameters, the workload generators decide the number of search and insert processes, the number of concurrent processes, the initial number of feature vectors to construct index trees, and the number of K of K -NN queries or the range of range queries. Subsequently, the workload generators pass the decided values to a driver program that is written with C and MIDAS APIs. The driver executes search and insert processes. It randomly selects feature vectors from an already inserted data set for queries and from a data set to be inserted for insertions. Each process executes multiple transactions. We measured the total execution time of each process and took the average time of a transaction. We fixed the number of buffer pools at 100 when initiating MIDAS.

Figure 11 shows the response time of search operations of both algorithms. The graph of our algorithm stayed almost constant, whereas that of CGiST deteriorated considerably. Our algorithm achieved about 45% performance improvement over CGiST through 10 ~ 100 % insert ratios. As we described in the introduction, generally in the application of multi-dimensional index structures, the ratio of insert operations to search operations is small. Therefore, we need to concentrate on small insert ratios. Figure 12 shows the response times of both methods when the ratio of insert operations is from 10% to 40%. In this case, our algorithm achieved about a 24% improvement over CGiST. Figure 12 shows the performance results when the K was 10. Also, we experimented with varying K . As K grew, the performance improvement of our algorithm also grew.

Figure 13 shows the response time of insert operations. As described earlier, we sacrificed the performance of insert operations a little for much more efficient search operations. The overall insert performance of our algorithm was slightly lower than that of CGiST. As shown in Figure 13, however, as a whole, the performance result of the proposed algorithm was similar to that of CGiST.

Table 4. Proportions of *FindNode*, *Split*, and *FixMBR*

	<i>FindNode</i>	<i>Split</i>	<i>FixMBR</i>
Proportions	87.8 %	11%	0.07%
Numbers	30,000	145	2518

Our algorithm does not need to obtain a latch on the parent node before splitting the current node while processing node split. Latch coupling is not necessary any more while processing MBR updates since node splits and MBR updates are serialized through tree locks. The first stage of an insert operation, that is, the *FindNode* function, needs a large amount of computation time and disk I/O time, and it occurs in every insert operation. On the other hand, MBR updates are caused less frequently and take less computation time than *FindNode*. Also, since the ancestor nodes to be modified usually are on the buffer pool, less disk I/O time is required. Node splits are more expensive operations than MBR updates, but they occur less frequently than MBR updates.

Table 4 shows the proportion of execution time of *FixMBR*, *TreatOverflow*, and *FindNode* to the overall execution time of an insert operation. We performed 30,000 insert operations and calculated average execution time of each operation. Therefore, increasing throughput of *FindNode* increased the overall concurrency performance. We serialized *TreatOverflow* and *FixMBR* by using exclusive tree locks. However, since *TreatOverflows* rarely occurred and *FixMBRs* took little time, as shown in Table 4, the degradation of overall concurrency was small. This reduced the number of simultaneous x-latches in index trees so more *FindNodes* and *Searchs* could be performed. For that reason, the overall concurrency is increased. Clearly, search performance of our scheme was superior to that of CGiST. On the other hand, the insert performance was almost the same as that of CGiST. However, as we mentioned earlier in this chapter, we mainly focused on increasing search performance while maintaining reasonable insert performance.

Figures 14 and 15 show the response time of insert and search operations when varying MPL from 20 to 80. As the MPL was increased, the performance gap of search operations of both scheme became larger. That means that our scheme is scalable for increasing MPL. However, in case of insert operations, as the MPL was increased, CGiST outperformed ours slightly.

CONCLUSION

In this chapter, we have proposed an efficient concurrency control algorithm for high-dimensional index structures. Even though the proposed algorithm is based on the link technique of CGiST, it does not employ lock coupling while ascending the index tree to process node splits and MBR updates by serializing structure modifications (*TreatOverflows* in our algorithm). It also provides the concurrency control mechanisms

Figure 14. Response time of search operations (Selectivity = 0.05%, Insert Ratio = 40%, data size = 50K)

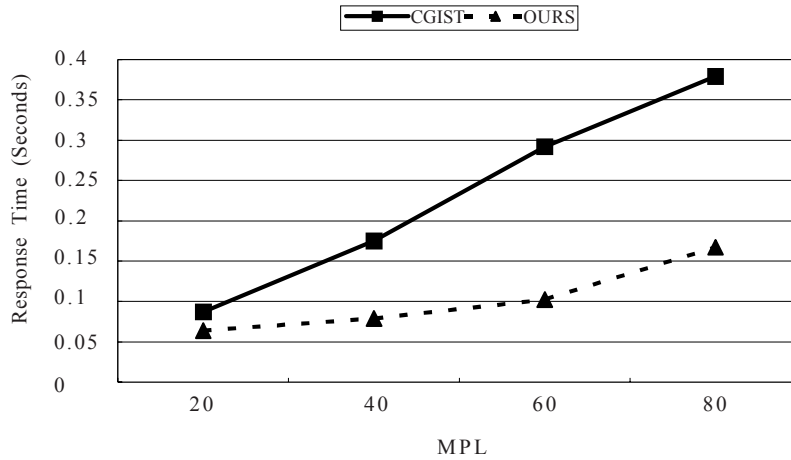
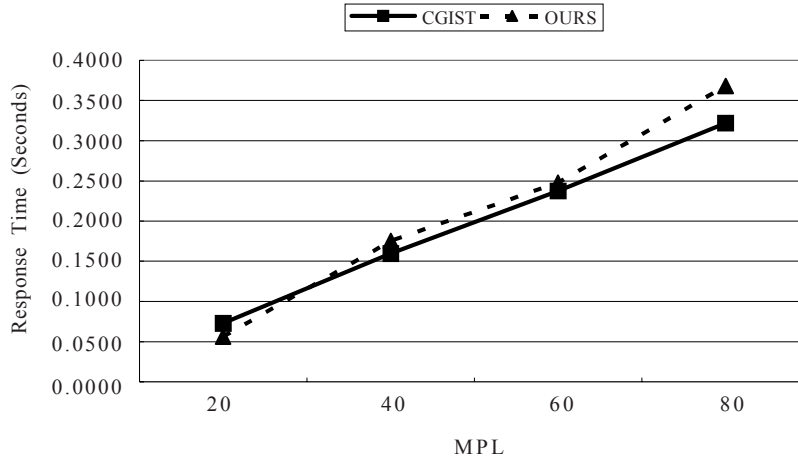


Figure 15. Response time of insert operations (Selectivity = 0.05%, Insert Ratio = 40%, data size = 50K)



for forced reinsert operations that are used to improve search performance in multi-dimensional index trees. In experimental comparisons with CGiST, we have shown that our proposed algorithm outperforms CGiST in terms of response time of search operations, with about 45 % performance improvement. Currently, our proposed algorithm supports repeatable read isolation. In the further research, we will consider no phantom read consistency. Also, we will design the proper recovery scheme for the proposed concurrency control algorithm.

REFERENCES

- Beckmann, N., Kornacker, H. P., Schneider, R., & Seeger, B. (1990). The R*-Tree: An efficient and robust access method for points and rectangles. In *Proceedings of ACM Special Interest Group on Management of Data (SIGMOD)* (pp. 322-331).
- Berchtold, S., Keim, D. A., & Kriegel, H. P. (1996). The X-Tree: An index structure for high-dimensional data. In *Proceedings of Very Large Data Bases (VLDB)* (pp. 28-39).
- Chae, M., Hong, K., Lee, M., Kim, J., Joe, O., Jeon, S., & Kim, Y. (1995). Design of the object kernel of BADA-III: An object-oriented database management system for multimedia data service. *Workshop on Network and System Management*.
- Chakrabarti, K., & Mehrotra, S. (1998). Dynamic granular locking approach to phantom protection in R-Trees. In *Proceedings of International Conference on Data Engineering (ICDE)* (pp. 446-454).
- Chakrabarti, K., & Mehrotra, S. (1999a). The Hybrid Tree: An index structure for high-dimensional feature spaces. In *Proceedings of International Conference on Data Engineering (ICDE)* (pp. 440-447).
- Chakrabarti, K., & Mehrotra, S. (1999b). Efficient concurrency control in multi-dimensional access methods. In *Proceedings of ACM Special Interest Group on Management of Data (SIGMOD)* (pp. 25-36).
- Chen, J. K., & Huang, Y. F. (1997). A study of concurrent operations on R-Trees. In *Proceedings of Information Sciences* (pp. 263-300).
- Ciaccia, P., Patella, M., & Zezula, P. (1997). M-tree: An efficient access method for similarity search in metric spaces. *Proceedings of Very Large Data Bases (VLDB)* (pp. 426-435).
- Finkel, R. A., & Bentley, J. L. (1974). Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4, 1-9.
- Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *Proceedings of ACM Special Interest Group on Management of Data (SIGMOD)* (pp. 47-57).
- Indyk, P., & Motwani, R. (1998). Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of ACM Symposium on Theory of Computing (STOC)* (pp. 604-613).
- Kanth, K. V., Serena, D., & Ambuj, K. (1998). Improved concurrency control techniques for multi-dimensional index structures. In *Proceedings of the Symposium on Parallel and Distributed Processing* (pp. 580-586).
- Katayama, N., & Satoh, S. (1997). The SR-tree: An index structure for high-dimensional nearest neighbor queries. In *Proceedings of ACM Special Interest Group on Management of Data (SIGMOD)* (pp. 369-380).
- Kornacker, M., & Banks, D. (1995). High-concurrency locking in R-trees. In *Proceedings of Very Large Data Bases (VLDB)* (pp. 134-145).
- Kornacker, M., Mohan, C., & Hellerstein, J. M. (1997). Concurrency and recovery in generalized search trees. In *Proceedings of ACM Special Interest Group on Management of Data (SIGMOD)* (pp. 62-72).
- Lehmann, P. L., & Yao, S. B. (1981). Efficient locking for concurrent operations on B-Trees. *Journal of ACM Transaction on Database Systems (TODS)*, 6(4), 650-670.
- Lin, K. I., Jagadish, H., & Faloutsos, C. (1994). The TV-tree: An index structure for high dimensional data. *Journal of Very Large Data Bases (VLDB)*, 3, 517-542.

- Mohan, C. (1990). ARIES/KVL: A key value locking method for concurrency control of multi-action transactions operating on b-tree indexes. In *Proceedings of Very Large Data Bases (VLDB)* (pp. 392-405).
- Mohan, C., Harderle, D., Lindsay, B., Pirahesh, H., & Schwarz, P. (1992). ARIES: A transaction recovery method supporting fine-granularity locking and partial roll-backs using write ahead logging. *Journal of ACM Transaction on Database Systems (TODS)*, 17(1), 94-162.
- Mohan, C., & Levine, F. (1992). ARIES/IM: An efficient and high concurrency index management method using write-ahead logging. In *Proceedings of ACM Special Interest Group on Management of Data (SIGMOD)* (pp. 371-380).
- Ng, V., & Kamada, T. (1993). Concurrent accesses to R-Trees. In *Proceedings of Symposium on Large Spatial Databases* (pp. 142-161).
- Nievergelt, J., Hinterberger, H., & Sevcik, K. C. (1984). The grid file: An adaptable, symmetric multikey structure. *Journal of ACM Transaction on Database Systems (TODS)*, 9(1), 38-71.
- Ravi, K. V., Kanth, Serena, D., & Singh, A. K. (1998). Improved concurrency control techniques for multi-dimensional index structures. In *Proceedings of Symposium on Parallel and Distributed Processing* (pp. 580-586).
- Robinson, J. T. (1981). The K-D-B-tree: A search structure for large multi-dimensional dynamic indexes. In *Proceedings of ACM Special Interest Group on Management of Data (SIGMOD)* (pp. 10-18).
- Seidl, T., & Kriegel, H. P. (1998). Optimal multi-step k-nearest neighbor search. In *Proceedings of ACM Special Interest Group on Management of Data (SIGMOD)*.
- Sellis, T., Roussopoulos, N., & Faloutsos, C. (1987). The R+-tree: A dynamic index for multi-dimensional objects. In *Proceedings of Very Large Data Bases (VLDB)* (pp. 507-519).
- Song, S. I., Kim, Y. H., & Yoo, J. S. (2004). An enhanced concurrency control algorithm for multi-dimensional index structures. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16(1), 97-111.
- Weber, R., Schek, H., & Blott, S. (1998). A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of Very Large Data Bases (VLDB)* (pp. 194-205).
- White, D. A., & Jain, R. (1996). Similarity indexing with the SS-tree. In *Proceedings of International Conference on Data Engineering (ICDE)* (pp. 516-523).
- Yoo, J., Shin, M., Lee, S., Choi, K., Cho, K., & Hur, D. (1998). *An efficient index structure for high dimensional image* (pp. 134-147).

Section III: Database Design Issues and Solutions

Chapter XV

Modeling Fuzzy Information in the IF_2O and Relational Data Models

Z. M. Ma, Northeastern University, China

ABSTRACT

Computer applications in non-traditional areas have put requirements on conceptual data modeling. Some conceptual data models, being the tool of design databases, have been proposed. However, information in real-world applications is often vague or ambiguous. Currently, less research has been done in modeling imprecision and uncertainty in conceptual data models and the design of databases with imprecision and uncertainty. In this chapter, a different level of fuzziness based on fuzzy set and possibility distribution theory will be introduced into the IFO data model and the corresponding graphical representations will be given. The IFO data model is then extended to a fuzzy IFO data model, denoted IF_2O . In particular, we provide the approach to mapping an IF_2O model to a fuzzy relational database schema.

INTRODUCTION

A major goal for database research has been the incorporation of additional semantics into data models. Databases have gone through the development from hierarchical and network databases to relational databases. As computer technologies

move into non-transaction processing such as CAD/CAM, knowledge-based systems, multimedia and Internet systems, many feel the limitation of a relational database in these data-intensive applications. So some non-traditional data models for databases—such as conceptual data models (e.g., entity relationships/enhanced entity relationships [ER/EER]) (Chen, 1976), Unified Modeling Language (UML) (Siau & Cao, 2001), and IFO (Abiteboul & Hull, 1987)), object-oriented data models, and logic data models—have been proposed. Conceptual data models can capture and represent rich and complex semantics at a high abstract level (Fong, Karlapalem, Li, & Kwan, 1999; Halpin, 2002; Shoval & Frummermann, 1994); therefore, various conceptual data models have been used for conceptual design of databases. For example, the relational databases were designed by first developing a high-level conceptual data model, the ER model, and then the developed conceptual model was mapped to an actual implementation (Teorey, Yang, & Fry, 1986). As to the IFO model, it was extended into a formal object model IFO₂, and then the IFO₂ model was mapped into object-oriented databases by Poncelet, Teisseire, Cicchetti, and Lakhali (1993).

However, information is often imperfect in real-world applications. Therefore, different kinds of imperfect information have been extensively introduced into databases (Yazici & George, 1998). There have been some attempts to classify various possible kinds of imperfect information, although there are no unified points of view and definitions. But inconsistency, imprecision, vagueness, uncertainty, and ambiguity are viewed as the basic kinds of imperfect information in database systems (Bosc & Prade, 1993). Rather than giving the definitions of this imperfect information, we explain its meanings in the following:

- Inconsistency is a kind of semantic conflict, meaning the same aspect of the real world is irreconcilably represented more than once in a database or in several different databases. For example, the age of George is stored as 34 and 37 simultaneously. Information inconsistency usually comes from information integration.
- Intuitively, the imprecision and vagueness are relevant to the content of an attribute value, and it means that a choice must be made from a given range (interval or set) of values, but we do not know exactly which one to choose at present. In general, vague information is represented by linguistic values. For example, the age of Michael is a set {18, 19, 20, 21}, a piece of imprecise information, and the age of John is a linguistic “old,” a piece of vague information.
- The uncertainty is related to the degree of truth of its attribute value, and it means that we can apportion some but not all of our belief to a given value or a group of values. For example, the possibility that the age of Chris is 35 right now should be 98%. The random uncertainty, described using probability theory, is not considered in this chapter.
- The ambiguity means that some elements of the model lack complete semantics leading to several possible interpretations.

Generally, several different kinds of imperfection can co-exist with respect to the same piece of information. For example, the age of Michael is a set {18, 19, 20, 21} and their possibilities are 70%, 95%, 98%, and 85%, respectively. Imprecision, uncertainty, and vagueness are three major types of imperfect information and can be modeled with

possibility theory (Zadeh, 1978). Many of the existing approaches dealing with imprecision and uncertainty are based on the theory of fuzzy sets.

Fuzzy information has been extensively investigated in the context of the relational model (Buckles & Petry, 1982; Ma & Mili, 2002; Ma, Zhang, & Ma, 1999; Prade & Testemale, 1984). Current efforts have been concentrated on fuzzy object-oriented databases and some related notions such as class, superclass/subclass, inheritance, and the like are extended (Bordogna, Pasi, & Lucarella, 1999; Cross, Caluwe, & Vangyseghe, 1997; Dubois, Prade, & Rossazza, 1991; George, Srikanth, Petry, & Buckles, 1996; Gyseghem & Caluwe, 1998; Ma, Zhang, & Ma, 2004; Ma, 2004). The fuzzy object-relational databases can also be found in Cubero, Marin, Medina, Pons, and Vila (2004).

However, less research has been done in modeling fuzzy information in the conceptual data model. It is particularly true in developing design methodologies for implementing fuzzy databases (Ma, Zhang, Ma, & Chen, 2001). Zvieli and Chen (1986) allowed fuzzy attributes in entities and relationships, and they introduced three levels of fuzziness in the ER model. At the first level, entity sets, relationships, and attribute sets may be fuzzy; that is, they have membership degree to the model. The second level is related to the fuzzy occurrences of entities and relationships. The third level concerns the fuzzy values of attributes of special entities and relationships. In Chaudhry, Moyne, and Rundensteiner (1999), the fuzzy relational databases were designed by using the fuzzy ER model proposed in Zvieli and Chen (1986). In Chen and Kerre (1998), the fuzzy extension of several major EER concepts (superclass, subclass, generalization, specialization, category, and shared subclass) was introduced without including graphical representations. Ma et al. (2001) worked with the three levels of Zvieli and Chen (1986) and introduced a fuzzy extended entity-relationship (FEER) model to cope with imperfect as well as complex objects in the real world at a conceptual level. They also provided an approach to mapping a FEER model to a fuzzy object-oriented database schema. Galindo, Urrutia, Carrasco, and Piattini (2004) relaxed constraints in enhanced entity-relationship (EER) models using fuzzy quantifiers. In addition, they studied new constraints that are not considered in classic EER models and examined the representation of these constraints in an EER model and their practical representations. More recently, the fuzzy UML data model and the fuzzy Extensible Mark-Up Language (XML) data model have also been introduced by Ma (2005), based on fuzzy sets and possibility distributions.

In this chapter, fuzzy information is represented via the relational databases and the IFO model. Here the IFO model was proposed by Abiteboul and Hull (1987) as a formally defined conceptual database model that comprises a rich set of high-level primitives for database design. The reason why the IFO model is employed instead of the ER model for the conceptual modeling of fuzzy information might be because the IFO model subsumes the ER model and other semantic and functional data models as claimed by Abiteboul and Hull (1987). In addition, the IFO model provides a formal representation of the main data structuring features found in previous semantics data models (Abiteboul & Hull, 1995; Hanna, 1995).

Therefore, in this chapter, we extend the IFO model to handle fuzzy information. The fuzzy IFO model is called the IF₂O model. A mapping process from the IF₂O model to the fuzzy relational model is developed in this chapter. It should be noticed that the IFO model has been extended for the conceptual modeling of fuzzy information in Vila, Cubero, Medina, and Pons (1996) and Yazici, Buckles, and Petry (1999). This chapter differs from the research effort in Vila et al. (1996) in that the conceptual design of fuzzy databases

was not provided there. Based on similarity relations (Buckles & Petry, 1982), Yazici, Buckles, and Petry (1999) extended the IFO model into the ExIFO (Extended IFO) model to represent uncertainties at the levels of the attribute, the object, and class. They also used a fuzzy extended NF² relation (non-first normal form) to transform a conceptual design — ExIFO model — into a logical design. Consequently, the strategy is to analyze the attributes that compose the conceptual model in order to establish an NF² model. Our study uses the possibility distribution theory to extend the IFO model into the IF₂O model to represent different levels of fuzziness. Based on the corresponding graphical representations, the IF₂O model is mapped into the fuzzy relational databases.

The remainder of this chapter is organized as follows. The next section presents fuzzy sets and fuzzy relational databases. Then, we introduce the fuzzy extension of the IFO model to the IF₂O model. Next, the approaches to mapping the IF₂O model to a fuzzy relational database schema are provided. The chapter ends with our conclusions.

FUZZY SETS AND FUZZY RELATIONAL DATABASES

In this section, we discuss the basic definitions and characteristics of the models and concepts used. Topics include a brief background on the IFO model, the possibility distribution, and the fuzzy relational model.

Fuzzy Sets and Possibility Distributions

Fuzzy data is originally described as fuzzy set by Zadeh (1965). Let U be a universe of discourse, then a fuzzy value on U is characterized by a fuzzy set F in U . A membership function $m_F: U \rightarrow [0, 1]$ is defined for the fuzzy set F , where $\mu_F(u)$, for each $u \in U$, denotes the degree of membership of u in the fuzzy set F . Thus the fuzzy set F is described as follows:

$$F = \{\mu(u_1)/u_1, \mu(u_2)/u_2, \dots, \mu(u_n)/u_n\}$$

When the $\mu_F(u)$ above is explained to be a measure of the possibility that a variable X has the value u in this approach, where X takes values in U , a fuzzy value is described by a possibility distribution π_X (Zadeh, 1978).

$$\pi_X = \{\pi_X(u_1)/u_1, \pi_X(u_2)/u_2, \dots, \pi_X(u_n)/u_n\}$$

Here, $\pi_X(u_i)$, $u_i \in U$, denotes the possibility that u_i is the actual value of X . Let π_X and F be the possibility distribution representation and the fuzzy set representation for a fuzzy value, respectively. It is clear that $\pi_X = F$ is true (Raju & Majumdar, 1988).

In addition, fuzzy data is represented by similarity relations in domain elements (Buckles & Petry, 1982), in which the fuzziness comes from the similarity relations between two values in a universe of discourse, not from the status of an object itself. Similarity relations are thus used to describe the degree of similarity of two values from the same universe of discourse. A similarity relation *Sim* on the universe of discourse

U is a mapping: $U \times U \rightarrow [0, 1]$ such that:

- a. for $\forall x \in U, \text{Sim}(x, x) = 1$, (reflexivity)
- b. for $\forall x, y \in U, \text{Sim}(x, y) = \text{Sim}_i(y, x)$, and (symmetry)
- c. for $\forall x, y, z \in U, \text{Sim}(x, z) \geq \max_y (\min(\text{Sim}(x, y), \text{Sim}(y, z)))$. (transitivity)

Fuzzy Relational Database Modeling

In connection with the three types of fuzzy data representations, there exist two basic extended data models for fuzzy relational databases. One of the data models is based on similarity relations (Buckles & Petry, 1982), proximity relation (Shenoi & Melton, 1989), or resemblance (Rundensteiner, Hawkes, & Bandler, 1989). The other one is based on possibility distribution (Prade & Testemale, 1984; Raju & Majumdar, 1988). The latter can be further classified into two categories: tuples associated with possibilities and attribute values represented possibility distributions. In Raju and Majumdar (1988), these two categories were called type-1 and type-2 fuzzy relational models, respectively. The form of an n -tuple in each of the above-mentioned basic fuzzy relational models can be expressed, respectively, as:

$$t = \langle p_1, p_2, \dots, p_i, \dots, p_n \rangle, t = \langle a_1, a_2, \dots, a_i, \dots, a_n, d \rangle \text{ and } t = \langle \pi_{A_1}, \pi_{A_2}, \dots, \pi_{A_i}, \dots, \pi_{A_n} \rangle,$$

where $p_i \subseteq D_i$ with D_i being the domain of attribute A_i , $a_i \in D_i$, $d \in [0, 1]$, π_{A_i} is the possibility distribution of attribute A_i on its domain D_i , and $\pi_{A_i}(x)$, $x \in D_i$, denotes the possibility that x is true.

The fuzzy relational instances in Figure 1 clearly show these three kinds of basic fuzzy relational models. The similarity relations for attributes popularity and category are shown in Figure 2.

Based on the above-mentioned basic fuzzy relational models, there should be two kinds of extended fuzzy relational models. One is the extended fuzzy relational model that is formed through combining the type-1 and type-2 fuzzy relational models. Another is the extended fuzzy relational model where possibility distribution and similarity (prox-

Figure 1. Three kinds of basic fuzzy relational models

Similarity-based Fuzzy Relation			
ID	Name	Category	Popularity
J001	CACM	[CS, CE, ME]	[very-popular]
J002	AI	[CS, CE]	[popular, mod-popular]
J003	ME	[IE, ME]	[not-popular]

Type-1 Fuzzy Relation			
ID	Name	Status	μ
S001	CACM	Falcity	0.7
S002	AI	Staff	0.9
S003	ME	Student	0.3

Type-2 Fuzzy Relation			
ID	Name	Age	Position
F001	Chris	young	Assis. Prof.
F002	John	more or less old	Assoc. Prof.
F003	Tom	old	Prof.

Figure 2. The similarity relations for attributes Popularity and Category

	<i>very-popular</i>	<i>popular</i>	<i>mod-popular</i>	<i>not-popular</i>		<i>CS</i>	<i>CE</i>	<i>IE</i>	<i>ME</i>
<i>very-popular</i>	1.0	0.8	0.6	0.0	<i>CS</i>	1.0	0.9	0.4	0.1
<i>popular</i>	0.8	1.0	0.8	0.1	<i>CE</i>	0.9	1.0	0.8	0.3
<i>mod-popular</i>	0.6	0.8	1.0	0.4	<i>IE</i>	0.4	0.8	1.0	0.7
<i>not-popular</i>	0.0	0.1	0.4	1.0	<i>ME</i>	0.1	0.3	0.7	1.0

$\text{Dom}(\text{Popularity}) = \{\text{very-popular}, \text{popular}, \text{mod-popular}, \text{not-popular}\},$

$\text{Dom}(\text{Category}) = \{\text{CS}, \text{CE}, \text{IE}, \text{ME}\}$

imity or resemblance) relation arises in a relation simultaneously (Ma, Zhang, & Ma, 1999). Of course, these two kinds of the extended fuzzy relational models can be combined further to form a more complex fuzzy relational model. In this chapter, we focus on the first kind of the extended fuzzy relational models, where the form of an n -tuple is $t = \langle \pi_{A_1}, \pi_{A_2}, \dots, \pi_{A_i}, \dots, \pi_{A_n}, d \rangle$.

Fuzzy relational model: A fuzzy relation r on a relational schema $R(A_1, A_2, \dots, A_n, A_{n+1})$ is a subset of the Cartesian product of $\text{Dom}(A_1) \times \text{Dom}(A_2) \times \dots \times \text{Dom}(A_n)$, where $\text{Dom}(A_i)$ ($1 \leq i \leq n$) may be a fuzzy subset or even a set of fuzzy subset in which each fuzzy set is represented by a possibility distribution, and $\text{Dom}(A_{n+1})$ is $[0, 1]$. Attribute A_{n+1} , called the attribute of membership degree, is used to indicate the possibility that the tuples belong to the corresponding relation.

Based on various fuzzy relational database models, many studies have also been done for data integrity constraints (Bosc & Pivert, 2003; Bosc, Dubois, & Prade, 1998; Liu, 1997; Raju & Majumdar, 1988; Sözat & Yazici, 2001). There have also been research studies on fuzzy query languages (Bosc & Pivert, 1995; Takahashi, 1993) and fuzzy relational algebra (Ma & Mili, 2002; Umamo & Fukami, 1994). In Bosc and Pivert (1995), an existing query language — SQL — for fuzzy queries was extended and some fuzzy aggregation operators were developed. In Zemankova and Kandel (1985), the fuzzy relational database (FRDB) model architecture and query language were presented, and the possible applications of the FRDB in imprecise information processing were discussed. For a comprehensive review of what has been done in the development of the fuzzy relational databases, please refer to Chen (1999), Petry (1996), Yazici, Buckles, and Petry (1992), Yazici and George (1999), and Yazici, Buckles and Petry (1992).

FUZZY IFO DATA MODEL: IF_2O

In this section, we extend the IFO model to represent fuzzy information. The fuzzy extended IFO model is denoted IF_2O . Since the constructs of the IFO model contain printable type, abstract type, free type, grouping, aggregation, fragment, and ISA relationships, the extension to these constructs must be conducted based on fuzzy set and possibility distribution theory. Before we introduce the IF_2O , we present the IFO.

IFO Data Model

The IFO model (Abiteboul & Hull, 1987) is a formally defined conceptual data model that incorporates the fundamental principles of semantic database modeling within a graph-based representational framework (Vila et al., 1996; Yazici, Buckles, & Petry, 1999). More formally, an IFO schema is a directed graph with various types of vertices and edges, representing atomic objects, constructed objects, fragments, and ISA relationships. A basic IFO schema is a combination of these pieces. The formal definitions of the IFO model were given in Abiteboul and Hull (1987), and readers may consult this for further information. The comparisons between the IFO data model and the semantic data model as well as *ER* data model can be found in Abiteboul and Hull (1995) and Hanna (1995). This chapter is concerned with an intuitive description of the model that will be used in the fuzziness representation. In this context, we consider the following relevant features of the IFO model.

Objects

The representation of the different object structures is called types, which constitute the basis of any IFO schema and correspond to the nodes in the schema graph representation. There exist three kinds of atomic types in the IFO model. The complex types can be built by utilizing two constructs applied to these three atomic types. Atomic types are those that have not been built from other ones, and are distinguished as follows:

- a. Printable types, which correspond to predefined data types of object such as string, numbers, and so forth.
- b. Abstract types, which correspond to real-world objects that have no underlying structure. Roughly speaking, an abstract type should be equivalent to an entity type in the ER model context.
- c. Free types, which correspond to entities obtained via ISA relationships.

Non-atomic objects are built from underlying types by utilizing two constructs as follows.

- a. Grouping, which is used to describe a finite set of objects of a given structure.
- b. Aggregation, which forms ordered n-tuples of instances, those are associated with a type.

Note that these two constructs could be applied recursively in any order to form more complex types.

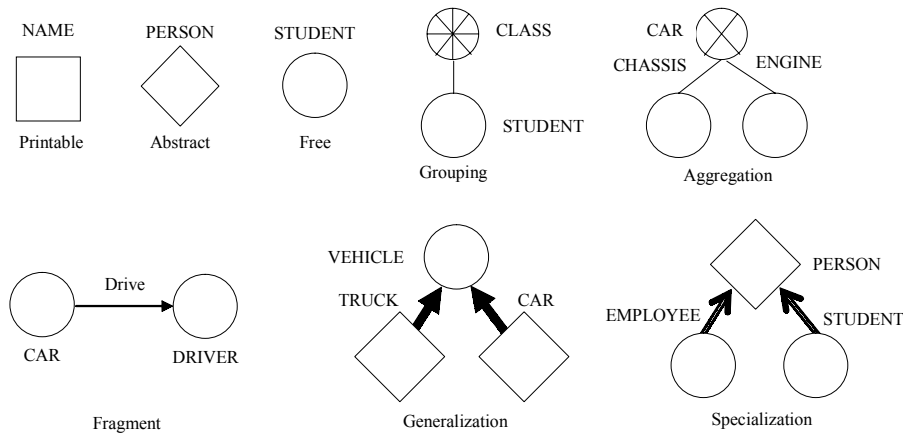
Fragments

Another main structural component of the IFO model is fragments for the representation of functional relationships. Fragments provide naturally clustered representations of types and their associated functions.

ISA Relationships

The final structural component of the IFO model is the representation of ISA relationships denoted by the arcs of the graphic schema. Two kinds of ISA relationships are distinguished:

Figure 3. The building blocks of the IFO data model



- Specialization, denoted by a double arrow, can be used to define possible roles for members of a given type. The attribute inheritance is verified in this case.
- Generalization, denoted by a broad arrow, represents situations where distinct, preexisting types are combined to form a virtual type.

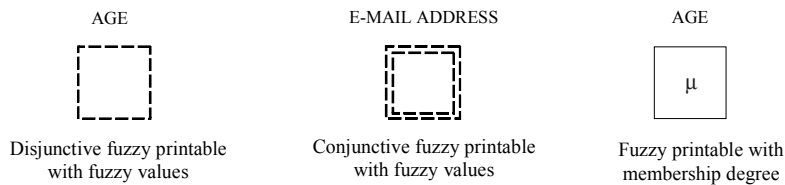
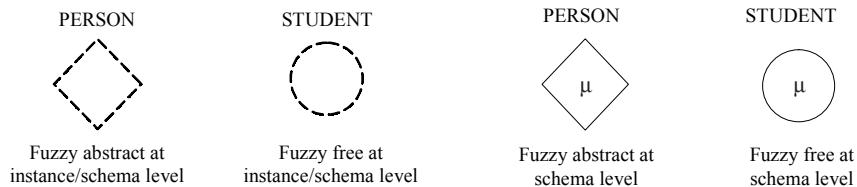
Combining the basic building blocks described above, the IFO schemas can be formed. Note that traditional IFO model cannot model the imprecision and uncertainty that extensively exist in the real world. Figure 3 shows the building blocks of the IFO model.

IF₂O Data Model

The IF₂O model contains the constructs of fuzzy printable type, fuzzy abstract type, fuzzy free type, fuzzy grouping, fuzzy aggregation, fuzzy fragment, and fuzzy ISA relationship.

Fuzzy Printable Types

In the IF₂O model, the fuzziness at the attribute level can be represented with fuzzy printable types. The fuzzy printable types can be distinguished into two levels. For fuzzy printable type with fuzzy values, its instance may have a fuzzy value on the corresponding attribute. Note that a fuzzy printable type with fuzzy values may have two kinds of interpretation: disjunctive fuzzy printable type with fuzzy values, and conjunctive fuzzy printable type with fuzzy values. The former means that only one choice must be made from among several alternatives, whereas the latter means that more than one choice may be made from among several alternatives. For a fuzzy printable type AGE, for example, it is unknown how old one guy is, but it is certain that this guy only has one number for the age. For a fuzzy printable type e-mail address, however, it is possible that one guy has several e-mail addresses, although we do not know what they are. In addition, a

Figure 4. Three fuzzy printable IF₂O typesFigure 5. Four fuzzy abstract and free IF₂O types

printable type of the object may be fuzzy corresponding to the data model, and it has its membership degree. Such fuzzy printable can be represented through placing memberships inside the diagrams of printable in the IF₂O model. Consider a fuzzy printable type AGE with 0.9 membership degree in an abstract type, say PERSON. That means the possibility printable type AGE is connected with abstract type PERSON is 0.9.

Figure 4 shows the graphical representation of disjunctive fuzzy printable type with fuzzy values, conjunctive fuzzy printable type with fuzzy values, and fuzzy printable type with membership degree.

Fuzzy Abstract and Free Types

The fuzziness in the abstract and the free types can be distinguished into two levels of fuzziness: instance/schema level and schema level. The fuzziness at the instance/schema level is related with the instances of special objects, which means that an object instance belongs to the corresponding object fuzzily. For example, it is uncertain if one person John is a Ph.D. student. Fuzziness at the schema level means that objects may be fuzzy corresponding to the data model, that is, they have their degree of membership. Consider a fuzzy free type STUDENT with 0.8 membership degree. We can place memberships inside the diagrams of abstract and free in the IF₂O model. For example, let A be an abstract type and m be its degree of membership in the model, then “ m ” is enclosed in the rhombus ($0 < m \leq 1$). If $m = 1.0$, “1.0” is usually omitted.

The graphical representations of fuzzy abstract and fuzzy free at instance/schema and schema levels are shown in Figure 5.

Fuzzy Constructs

First of all, let us look at the aggregation constructs. This constructor connects a subtype representing a part of object to the type representing the entire object. A high-

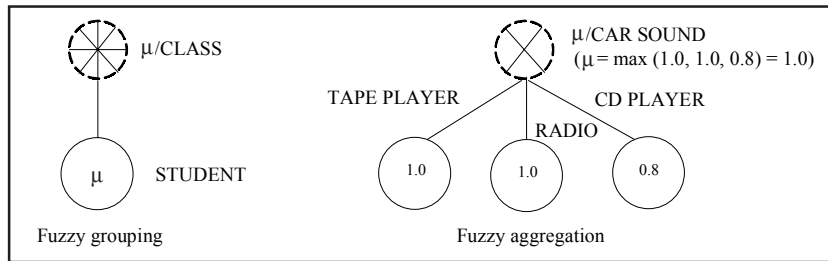
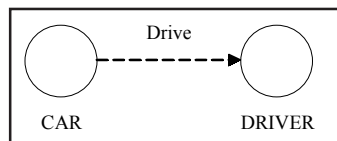
Figure 6. Two fuzzy constructed IF_2O objects

Figure 7. Fuzzy fragment



level object is thus formed. The subtypes may be the atomic types, perfect and fuzzy ones mentioned above, or the constructed types applying aggregation and grouping constructs. When any subtype that participates in aggregation constructor is a fuzzy type with degree of membership, the corresponding aggregation is a fuzzy aggregation with degree of membership, which is the maximum of membership degrees in all subtypes participated in the aggregation. For a fuzzy aggregation CAR SOUND, for example, it is aggregated by three free types: RADIO, TAPE PLAYER, and CD PLAYER. But free type CD PLAYER is fuzzy one with 0.8 membership degree.

Being similar to the fuzzy aggregation constructor, the grouping is a fuzzy grouping with degree of membership when the subtype that participates in grouping constructor is a fuzzy type with degree of membership. The membership degree of fuzzy grouping is the membership degree of the subtype participated in the grouping.

Figure 6 shows the graphical representation of fuzzy grouping and aggregation with membership degree.

Fuzzy Fragments

Under a fuzzy information environment, the fragments that are used for connections between abstract and abstract, abstract and free, and free and free may have fuzziness. There are two kinds of interpretation for such fuzziness. One is that the functional relationships between objects are certain, but instances of the functional relationships are fuzzy. For a fuzzy fragment Drive, for example, it is uncertain if driver John drives car Ford Focus, although a driver can drives a car. Figure 7 gives the graphic representation of such fuzzy fragments. Another interruption is that the functional relationships

Figure 8. Fuzzy fragment with membership degree

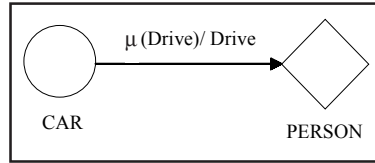
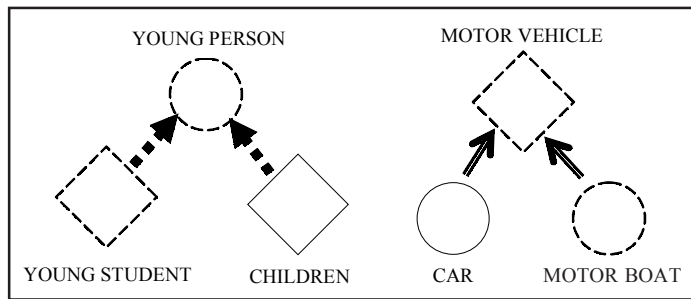


Figure 9. Fuzzy generalization and specialization



between objects are uncertain. The degree of membership is necessary for such fuzzy fragment. For a fuzzy fragment Drive with 0.6 membership degree, for example, it is uncertain if there is a relationship “drive” between person and car. The possibility is 0.6. Figure 8 gives the graphic representation.

Fuzzy ISA relationships

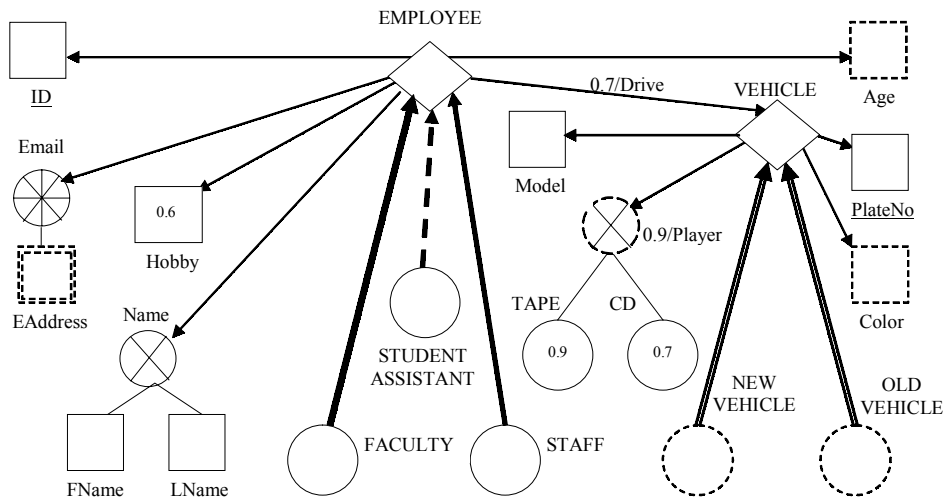
ISA relationships are related to the notion of subclass/superclass. Let $E, S_1, S_2, \dots$, and S_n be non-printable object types in the IF₂O model. We say $S_1, S_2, \dots$, and S_n are fuzzy subclass of E and E is a fuzzy superclass of $S_1, S_2, \dots$, and S_n if and only if there exist the fuzziness at instance/schema level in $E, S_1, S_2, \dots$, and S_n and the following is true, where e is object instance in universe.

$$(\forall e)(\forall S)(S \in \{S_1, S_2, \dots, S_n\} \wedge \mu_S(e) \leq \mu_E(e))$$

Figure 9 shows the graphical representations of fuzzy generalization and fuzzy specialization.

An Example Illustration

Let us see the EMPLOYEE-VEHICLE example represented with the IF₂O data model in Figure 10. Abstract type EMPLOYEE is connected with printable types ID, Hobby, and

Figure 10. A fuzzy IF_2O data model

Age, grouping E-mail, and aggregation Name. Here, Age can take fuzzy values; Hobby is related to EMPLOYEE with membership degree 0.6; E-mail may have no value, one, or more (fuzzy) values. Also, there are generalization relationships between EMPLOYEE and FACULTY, STAFF, and STUDENT ASSISTANT, where the generalization relationship between EMPLOYEE and *STUDENT ASSISTANT* is fuzzy one. In addition, a relationship Drive with membership degree 0.7 exists between EMPLOYEE and VEHICLE. Abstract type VEHICLE is connected with printable types PlateNo, Color, and Model, and aggregation Player. Here, Color can take fuzzy values; Player aggregates two free types TAPE and CD that have membership degrees 0.9 and 0.7, respectively. There are also specialization relationships between VEHICLE and OLD VEHICLE, as well as NEW VEHICLE, where OLD VEHICLE and NEW VEHICLE are fuzzy ones. The IF_2O model for the above state descriptions is defined in Figure 10 by utilizing the notations introduced in this section.

MAPPING AN IF_2O SCHEMA TO A FUZZY RELATIONAL DATABASE SCHEMA

The abstract types and the free types in the IFO model, in general, correspond to the tables (relations) in relational databases. The printable types in an abstract type and a free type correspond to the attributes in the relational table. In the IF_2O model, printable types, abstract types, free types, and ISA relationships may be fuzzy. In the following, we give a formal approach to transform an IF_2O schema to a fuzzy relational schema. First, let us consider the printable types.

Printable Type Transformation

Printable types used in abstract types or free types are mapped into the attributes of the relational tables that are created through mapping the corresponding abstract or free types just like shown in Section 4.2. In the IF₂O model, we can distinguish three kinds of fragments:

- a. printable types without any fuzziness,
- b. printable types taking fuzzy values, and
- c. printable types with membership degrees.

The first kind of printable types can be directly mapped into attributes in the relation transformed from the corresponding abstract or free type. The second kind of printable types are also mapped into attributes in the relation transformed from the corresponding abstract or free type. The difference between these two kinds of attributes is that the domains of the latter attributes are fuzzy ones. That means the values of tuples on such attributes may take fuzzy values. It should be noticed, however, that the relational model and the fuzzy relational model only focus on instance modeling (attribute values and tuples) and their meta-structures are implicitly represented in the schemas. So the fuzzy printable types with membership degrees cannot be mapped into the created fuzzy relational databases. Similarly, the fuzzy printable types and the fuzzy fragments with membership degrees cannot be mapped into the created fuzzy relational databases.

Abstract Type and Free Type Transformations

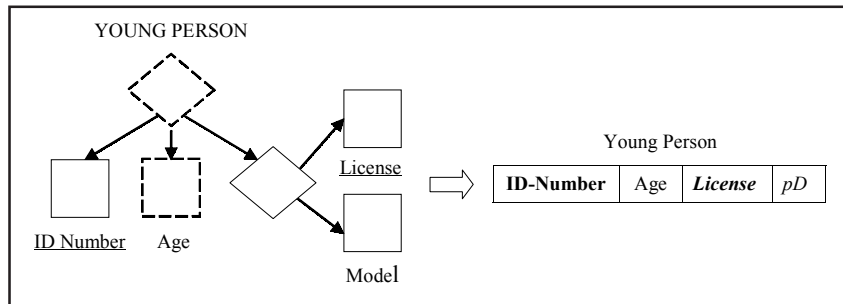
Each abstract type is mapped to a relational table, and all printable types connected with this abstract type — crisp or fuzzy — become the attributes in the table. Here, we assume that the abstract type has no ISA relationship and ignore the fragments connected with the abstract type, whose mapping will be discussed below. We can distinguish three kinds of abstract types in the IF₂O model:

- a. abstract types without any fuzziness,
- b. abstract types with the fuzziness at instance/schema level, and
- c. abstract types with the fuzziness at schema level.

The first kind of abstract types can be mapped into relations directly. For the second kind of abstract types, an additional attribute, denoted by *pD*, must be added to each relation transformed from the corresponding abstract type, which is used to denote the membership degree of an object instance to the type. As to printable types whose value may be fuzzy, the created attributes should have a fuzzy attribute domain. The third kind of abstract types and printable types with membership degrees cannot be mapped into the created relations.

Figure 11 shows the transformations of printable type and abstract type. Here, abstract type YOUNG PERSON is an abstract type with the fuzziness at instance/schema level. That means that an instance may belong to abstract type YOUNG PERSON fuzzily, that is, with a membership degree. Then, abstract type YOUNG PERSON is mapped into relation Young Person with the membership degree attribute *pD*. Then, each tuple in the relation can be associated with a membership degree (greater than 0.0 and less than or equal to 1.0). Also printable types *ID* Number and Age connected with abstract type YOUNG PERSON are directly mapped into attributes *ID* Number and Age of relation

Figure 11. Transformation of abstract type



Young Person. Since printable type Age is a fuzzy one taking fuzzy values, attribute Age has a fuzzy attribute domain. For the tuples in relation Young Person, their values on attribute Age may be fuzzy ones. However, it should be noticed that abstract type YOUNG PERSON is connected with another abstract type with a fragment. Therefore, the printable type License in that abstract type must be mapped to attribute License of relation Young Person also as a foreign key. The transformation process for the fragment is given in Section 4.3. The transformation process for the free type is the same as for the abstract type.

Fragment Transformation

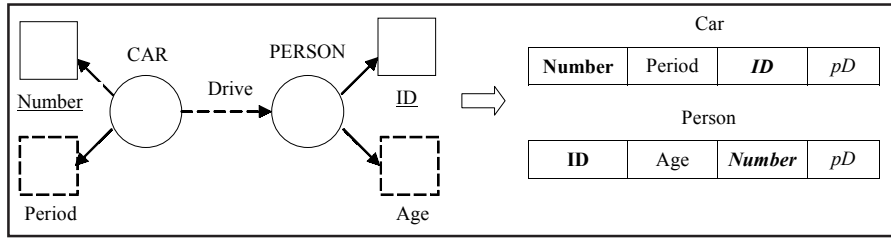
Fragments are used to connect abstract and abstract types, abstract and free types, or free and free types. In the IF₂O model, we can distinguish three kinds of fragments:

- fragments without any fuzziness,
- fragments with the fuzziness at instance/schema level, and
- fragments with the fuzziness at schema level (i.e., with membership degrees).

For the first kind of fragments, two additional attribute sets can directly be appended into the relations transformed from the corresponding two free types (or two abstract types, or one free type and another abstract type), respectively, which are used to indicate the association of tuples in the relations. The additional attribute set that is added into the created relation must be one of another created relation, which should correspond to the printable types and serve as primary keys in the free type or abstract type creating the later relation. For the second kind of fragments, in addition to the transformations given above, two additional attributes denoting membership degree of tuples to the relation should be added to the created relations, respectively. For the fragments with membership degree (the third kind of fragments), relational databases do not support their transformations.

Figure 12 shows the transformation of fragment. According to the transformation processes for the free types and printable types, free type CAR is mapped into relation Car with attributes Number and Period first. Then, printable type ID in free type PERSON

Figure 12. Transformation of fragment



is mapped into attribute ID of relation Car as a foreign key because there is fragment Drive between free types CAR and PERSON. Since this fragment is fuzzy one with the fuzziness at instance/schema level, membership degree attribute pD is added to relation Car to capture the uncertain degree of functional relationship Drive between the instances of free type CAR and the instances of free type PERSON. Similarly, free type PERSON is mapped into relation Person with attributes ID, Age, Number, and pD.

ISA Relationship Transformation

Now we focus on the transformation of abstract types and free types in ISA relationships. In general, the above-mentioned basic transformation rules for abstract types and free types can be used, that is, they are mapped into relations, and the printable types in them are mapped into the attributes in the corresponding relations. In addition, if these abstract types and free types are fuzzy ones with the fuzziness at instance/schema level, an additional attribute (membership degree attribute) *pD* should be added. However, the process of the primary keys in the ISA relationship transformation is different.

Let *S* be an abstract type with printable types named *K*, *A*₁, *A*₂, ..., and *A*_{*n*}, where *K* is its key. Let a free type *S*₁ with printable types named *A*₁₁, *A*₁₂, ..., and *A*_{1*k*} and a free type *S*₂ with printable types named *A*₂₁, *A*₂₂, ..., and *A*_{2*m*} be subclasses of *S*. Since *S*₁ and *S*₂ are subclasses of *S*, there are no keys in *S*₁ and *S*₂. At this point, *S* is mapped into the relational schema {*K*, *A*₁, *A*₂, ..., *A*_{*n*}}, and *S*₁ and *S*₂ are mapped into schemas {*K*, *A*₁₁, *A*₁₂, ..., *A*_{1*k*}} and {*K*, *A*₂₁, *A*₂₂, ..., *A*_{2*m*}}, respectively.

Figure 13 shows the transformation of specialization. Using the transformation rules for abstract types given in Section 4.2, abstract type *ENGINE* is mapped into relation Engine with attributes ID-Number and Model. As to two free types PLANE ENGINE and CAR ENGINE, they are mapped into relations Plane Engine with attributes Name and Usage and relations Car Engine with attributes Designer and Rate, respectively, using the transformation rules for free types given in Section 4.2. But free types PLANE ENGINE and CAR ENGINE are subclasses of abstract type ENGINE and they do not have key. Therefore, the key ID-Number in abstract type ENGINE is added to relations Plane Engine and Car Engine, respectively.

Figure 13 Transformation of specialization

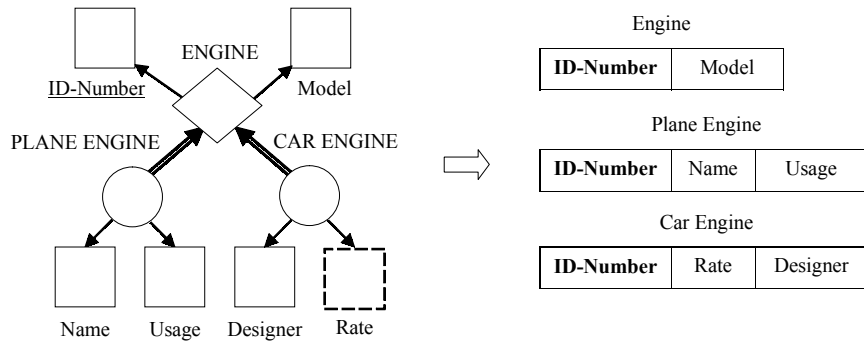
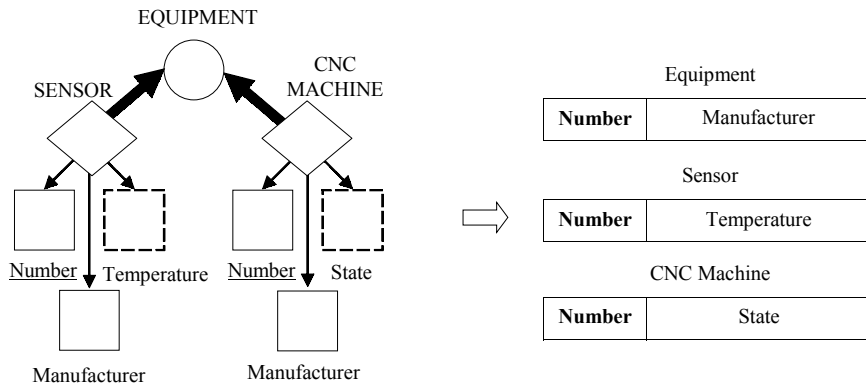
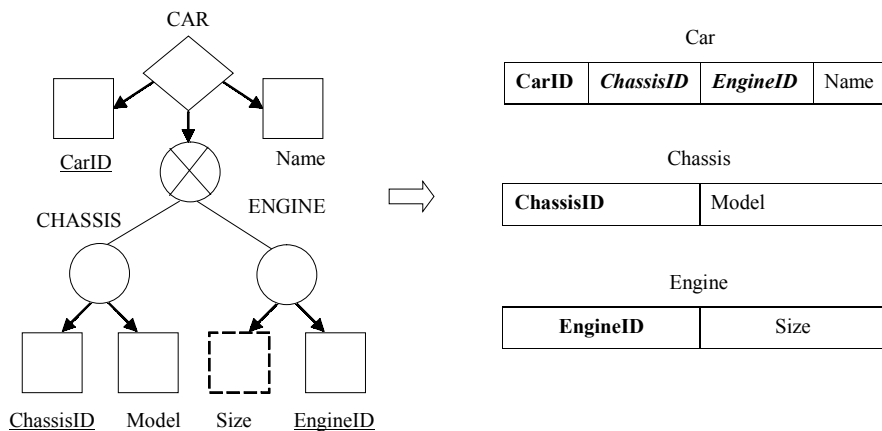


Figure 14. Transformation of generalization

Figure 15. Transformation of aggregation in IF_2O 

The transformation for generalization is more complex than for specialization. Let E_1 with attribute types named $K_1, A_1, A_2, \dots$, and A_k and E_2 with attribute types named $K_2, B_1, B_2, \dots$, and B_m be generated to supertype S . Assume $\{A_1, A_2, \dots, A_k\} \subset \{B_1, B_2, \dots, B_m\} = \{C_1, C_2, \dots, C_n\}$. Generally speaking, E_1 and E_2 are mapped into schemas $\{K_1, A_1, A_2, \dots, A_k\} - \{C_1, C_2, \dots, C_n\}$ and $\{K_2, B_1, B_2, \dots, B_m\} - \{C_1, C_2, \dots, C_n\}$, respectively. As to the transformation of S , depending on K_1 and K_2 , we distinguish the following two cases.

- a. K_1 and K_2 are identical. Then S is mapped into the relational schema $\{K_1\} \cup \{C_1, C_2, \dots, C_n\}$,
- b. K_1 and K_2 are different. Then S is mapped into the relational schema $\{K'\} \cup \{C_1, C_2, \dots, C_n\}$, where K' denotes the surrogate key created by K_1 and K_2 (Yazici, Buckles, & Petry, 1999).

Considering the fuzziness in entities, the following cases for the transformation of generalization are distinguished.

- a. E_1 and E_2 are crisp. Then E_1 and E_2 are transformed to relations r_1 and r_2 with attributes $\{K_1, A_1, A_2, \dots, A_k\} - \{C_1, C_2, \dots, C_n\}$ and $\{K_2, B_1, B_2, \dots, B_m\} - \{C_1, C_2, \dots, C_n\}$, respectively. S is transformed to a relation r with attributes $\{K, C_1, C_2, \dots, C_n\}$ just like the discussion above.
- b. When there is fuzziness of instance/schema level in E_1 and/or E_2 , being similar to case (a) also, relation r , as well as relations r_1 and r_2 , are formed. Not that r , r_1 and (or r_2) created by E_1 and (or) E_2 with the instance/schema level of fuzziness should include the attribute pD .
- c. When there is fuzziness of schema level in E_1 and (or) E_2 , relation r , as well as relations r_1 and r_2 , is formed. But the fuzziness at this level cannot be modeled in the created relations.

Figure 14 shows the transformations of generalization. Two free types SENSOR and CNC MACHINE are generalized into an abstract type EQUIPMENT. In spite of key Number, free types SENSOR and CNC MACHINE have a common Manufacturer. According to the transformation rules above, free types SENSOR and CNC MACHINE are mapped into relations Sensor with attributes Number and Temperature and CNC Machine with attributes Number and State, respectively. Abstract type EQUIPMENT is mapped into relation Equipment with attributes Number and Manufacturer.

Note that the transformation for abstract types and free types is suitable for aggregation. Let us consider the following example shown in Figure 15.

It can be seen that the aggregation and ISA relationships in the IF₂O model are not directly supported in the created relations. The relationships among abstract types as well as free types are modeled by the relationships between the same attributes in different relations. It is clear that such relationships are implicit and inefficient in information retrieval.

CONCLUSION

Conceptual data model, being the tool of modeling databases and the potential post-relational database model, has been proposed for non-traditional application. In addition, incorporation of imprecise and uncertain information in database model has

been an important topic of database research because such information extensively exists in real-world applications. Classical conceptual data models and logical database models do not satisfy the need of handling information imprecision and uncertainty. Therefore, current efforts have been concentrated on extending conceptual data model, relational databases and object-oriented databases.

In this chapter, we focus on conceptual data modeling and logical database modeling of fuzzy information. The IFO model is extended with fuzzy set and possibility theory to cope with fuzzy information at a conceptual level and the corresponding graphical representations are given. The approach to mapping the IF₂O model to the fuzzy relational database schema is developed.

One can conduct the conceptual design of a database model with fuzzy information and then transform it into the fuzzy database. This approach has value to some data/knowledge-intensive application domains, where complex objects are involved and the data/information is generally imperfect. Then the conceptual model can be utilized to represent the complex objects with uncertainty and the database model can be used to effectively handle data manipulations and information queries. For example, Yazici, Buckles, and Petry (1999) have pointed out that two of the important applications of databases that assimilate both complex objects and uncertainty are expert system interfaces and data warehouses.

ACKNOWLEDGMENTS

Work is supported by the Program for New Century Excellent Talents in University and in part by the MOE Funds for Doctoral Programs (20050145024).

REFERENCES

- Abiteboul, S., & Hull, R. (1987). IFO: A formal semantic database model. *ACM Transactions on Database Systems*, 12(4), 525-565.
- Abiteboul, S., & Hull, R. (1995). Response to "A close look at the IFO data model." *SIGMOD Record*, 24(3), 4-4.
- Bordogna, G., Pasi, G., & Lucarella, D. (1999). A fuzzy object-oriented data model for managing vague and uncertain information. *International Journal of Intelligent Systems*, 14, 623-651.
- Bosc, P., Dubois, D., & Prade, H. (1998). Fuzzy functional dependencies and redundancy elimination. *Journal of the American Society for Information Science*, 49(3), 217-235.
- Bosc, P., & Pivert, O. (1995). SQLf: A relational database language for fuzzy querying. *IEEE Transactions on Fuzzy Systems*, 3(1), 1-17.
- Bosc, P., & Pivert, O. (2003). On the impact of regular functional dependencies when moving to a possibilistic database framework. *Fuzzy Sets and Systems*, 140(1), 207-227.
- Bosc, P., & Prade, H. (1993). An introduction to fuzzy set and possibility theory based approaches to the treatment of uncertainty and imprecision in database management systems. In *Proceedings of the Second Workshop on Uncertainty Management in Information Systems: From Needs to Solutions*.

- Buckles, B. P., & Petry, F. E. (1982). A fuzzy representation of data for relational database. *Fuzzy Sets and Systems*, 7(3), 213-226.
- Chaudhry, N. A., Moyne, J. R., & Rundensteiner, E. A. (1999). An extended database design methodology for uncertain data management. *Information Sciences*, 121(1-2), 83-112.
- Chen, G. Q. (1999). *Fuzzy logic in data modeling: Semantics, constraints, and database design*. Boston: Kluwer Academic Publisher.
- Chen, G. Q., & Kerre, E. E. (1998). Extending ER/EER concepts towards fuzzy conceptual data modeling. In *Proceedings of the 1998 IEEE International Conference on Fuzzy Systems* (Vol. 2, pp. 1320-1325).
- Chen, P. P. (1976). The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, 1(1), 9-36.
- Cross, V., Caluwe, R., & Vangysegheem, N. (1997). A perspective from the fuzzy object data management group (FODMG). In *Proceedings of the 1997 IEEE International Conference on Fuzzy Systems* (Vol. 2, pp. 721-728).
- Cubero, J. C., Marin, N., Medina, J. M., Pons, O., & Vila, M. A. (2004). Fuzzy object management in an object-relational framework. In *Proceedings of the 2004 International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems* (pp. 1767-1774).
- Dubois, D., Prade, H., & Rossazza, J. P. (1991). Vagueness, typicality, and uncertainty in class hierarchies. *International Journal of Intelligent Systems*, 6, 167-183.
- Fong, J., Karlapalem, K., Li, Q., & Kwan, I. S. Y. (1999). Methodology of schema integration for new database applications: A practitioner's approach. *Journal of Database Management*, 10(1), 3-18.
- Galindo, J., Urrutia, A., Carrasco, R. A., & Piattini, M. (2004). Relaxing constraints in enhanced entity-relationship models using fuzzy Quantifiers. *IEEE Transactions on Fuzzy Systems*, 12(6), 780-796.
- George, R., Srikanth, R., Petry, F. E., & Buckles, B. P. (1996). Uncertainty management issues in the object-oriented data model. *IEEE Transactions on Fuzzy Systems*, 4(2), 179-192.
- Gysegheem, N. V., & Caluwe, R. D. (1998). Imprecision and uncertainty in UFO database model. *Journal of the American Society for Information Science*, 49(3), 236-252.
- Halpin, T. A. (2002). Metaschemas for ER, ORM and UML data models: A comparison. *Journal of Database Management*, 13(2), 20-30.
- Hanna, M. S. (1995). A close look at the IFO data model. *SIGMOD Record*, 24(1), 21-26.
- Liu, W. Y. (1997). Fuzzy data dependencies and implication of fuzzy data dependencies. *Fuzzy Sets and Systems*, 92(3), 341-348.
- Ma, Z. M. (2004). *Advances in fuzzy object-oriented databases: Modeling and applications*. Hershey, PA: Idea Group Publishing.
- Ma, Z. M. (2005). Fuzzy database modeling with XML. In *The Kluwer international series on advances in database systems*. New York: Springer.
- Ma, Z. M., & Mili, F. (2002). Handling fuzzy information in extended possibility-based fuzzy relational databases. *International Journal of Intelligent Systems*, 17(10), 925-942.
- Ma, Z. M., Zhang, W. J., & Ma, W. Y. (1999). Assessment of data redundancy in fuzzy relational databases based on semantic inclusion degree. *Information Processing Letters*, 72(1-2), 25-29.

- Ma, Z. M., Zhang, W. J., & Ma, W. Y. (2004). Extending object-oriented databases for fuzzy information modeling. *Information Systems*, 29(5), 421-435.
- Ma, Z. M., Zhang, W. J., Ma, W. Y., & Chen, G. Q. (2001). Conceptual design of fuzzy object-oriented databases utilizing extended entity-relationship model. *International Journal of Intelligent Systems*, 16(6), 697-711.
- Petry, F. E. (1996). *Fuzzy databases: Principles and applications*. Boston: Kluwer Academic Publisher.
- Poncelet, P., Teisseire, M., Cicchetti, R., & Lakhal, L. (1993). Towards a formal approach for object database design. In *Proceedings of the 19th International Conference on Very Large Data Bases* (pp. 278-289).
- Prade, H., & Testemale, C. (1984). Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries. *Information Sciences*, 34, 115-143.
- Raju, K. V. S. V. N., & Majumdar, A. K. (1988). Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems. *ACM Transactions on Database Systems*, 13(2), 129-166.
- Rundensteiner, E. A., Hawkes, L. W., & Bandler, W. (1989). On nearness measures in fuzzy relational data models. *International Journal of Approximate Reasoning*, 3, 267-298.
- Shenoi, S., & Melton, A. (1989). Proximity relations in the fuzzy relational databases. *Fuzzy Sets and Systems*, 31(3), 285-296.
- Shoval, P., & Frummermann, I. (1994). OO and EER conceptual schemas: A comparison of use comprehension. *Journal of Database Management*, 5(4), 28-38.
- Siau, K., & Cao, Q. (2001). Unified modeling language: A complexity analysis. *Journal of Database Management*, 12(1), 26-34.
- Sözat, M. I., & Yazici, A. (2001). A complete axiomatization for fuzzy functional and multivalued dependencies in fuzzy database relations. *Fuzzy Sets and Systems*, 117(2), 161-181.
- Takahashi, Y. (1993). Fuzzy database query languages and their relational completeness theorem. *IEEE Transactions on Knowledge and Data Engineering*, 5(1), 122-125.
- Teorey, T. J., Yang, D. Q., & Fry, J. P. (1986). A logical design methodology for relational databases using the extended entity-relationship model. *ACM Computing Surveys*, 18(2), 197-222.
- Umano, M., & Fukami, S. (1994). Fuzzy relational algebra for possibility-distribution-fuzzy-relational model of fuzzy data. *Journal of Intelligent Information Systems*, 3, 7-27.
- Vila, M. A., Cubero, J. C., Medina, J. M., & Pons, O. (1996). A conceptual approach for dealing with imprecision and uncertainty in object-based data models. *International Journal of Intelligent Systems*, 11, 791-806.
- Yazici, A., Buckles, B. P., & Petry, F. E. (1992). A survey of conceptual and logical data models for uncertainty management. In L. Zadeh & J. Kacprzyk (Eds.), *Fuzzy logic for management of uncertainty* (pp. 607-644). New York: John Wiley & Sons.
- Yazici, A., Buckles, B. P., & Petry, F. E. (1999). Handling complex and uncertain information in the ExIFO and NF2 Data Models. *IEEE Transactions on Fuzzy Systems*, 7(6), 659-676.

- Yazici, A., & George, R. (1998). Fuzzy database modeling. *Journal of Database Management*, 9(4), 36-36.
- Yazici, A., & George, R. (1999). *Fuzzy database modeling*. Heidelberg, Germany: Physica-Verlag.
- Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8(3), 338-353.
- Zadeh, L. A. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1(1), 3-28.
- Zemankova, M., & Kandel, A. (1985). Implementing imprecision in information systems. *Information Sciences*, 37(1-3), 107-141.
- Zicari, R., & Milano, P. (1990). Incomplete information in object-oriented databases. *ACM SIGMOD Record*, 19(3), 5-16.
- Zvieli, A., & Chen, P. P. (1986). Entity-relationship modeling and fuzzy databases. In *Proceedings of the 1986 IEEE International Conference on Data Engineering* (pp. 320-327).

Chapter XVI

Evaluating the Performance of Dynamic Database Applications

Zhen He, La Trobe University, Australia

Jérôme Darmont, Université Lumière Lyon 2, France

ABSTRACT

This chapter explores the effect that changing access patterns has on the performance of database management systems. Changes in access patterns play an important role in determining the efficiency of key performance optimization techniques, such as dynamic clustering, prefetching, and buffer replacement. However, all existing benchmarks or evaluation frameworks produce static access patterns in which objects are always accessed in the same order repeatedly. Hence, we have proposed the dynamic evaluation framework (DEF) that simulates access pattern changes using configurable styles of change. DEF has been designed to be open and fully extensible (e.g., new access pattern change models can also be added easily). In this chapter, we instantiate DEF into the dynamic object evaluation framework (DoEF), which is designed for object databases, that is, object-oriented or object-relational databases such as multimedia databases or most eXtensible Mark-up Language (XML) databases.

INTRODUCTION

In database management systems (DBMSs), architectural or optimisation choices, efficiency comparison, or tuning all require the assessment of system performance. Traditionally, this is achieved with the use of benchmarks, namely, synthetic workload models (databases and operations) and sets of performance metrics. To the best of our knowledge, none of the existing database benchmarks incorporate the possibility of change in the access patterns, whereas in real life, almost no application always accesses the same data in the same order repeatedly. Furthermore, the ability to adapt to changes in access patterns is critical to database performance. Highly tuning a database to perform well for only one particular access pattern can lead to poor performance when different access patterns are used. In addition, the performance of a database on a particular trace provides little insight into the reasons behind its performance, and thus is of limited use to database researchers or engineers, who are interested in the identification and improvement in the performance of particular components of the system. Hence, this chapter aims to present a new perspective on DBMS performance evaluation by exploring how to assess the dynamic behaviour of DBMSs.

More precisely, this chapter presents a benchmarking framework that allows users to explore the performance of databases under different styles of access pattern change. In contrast, benchmarks of the Transaction Processing Performance Council (TPC) family aim to provide standardised means of comparing systems for vendors and customers. In this chapter, we take a look at how dynamic application behaviour can be modelled and propose the dynamic evaluation framework (DEF). DEF contains a set of protocols that define a set of styles of access pattern change. DEF by no means has exhausted all possible styles of access pattern change. However, it is designed to be fully extensible and its design allows new styles of change to be easily incorporated. Finally, DEF is a generic platform that can be specialized to suit the particular needs of a given family of DBMS (e.g., relational, object, object-relational, or XML). In particular, it is designed to be implemented on top of an existing benchmark so that previous benchmarking research and standards can be reused.

In this chapter, we show the utility of DEF by creating an instance of DEF called the dynamic object evaluation framework (DoEF) (He & Darmont, 2003). DoEF is designed for object databases. Note that in the remainder of this chapter, we term object database management systems (ODBMSs) both object-oriented and object-relational systems, indifferently. ODBMSs include most multimedia and XML DBMSs, for example.

To illustrate the effectiveness of DoEF, this chapter presents the results of two sets of experiments. First, it presents benchmark results of four state-of-the-art dynamic clustering algorithms (Bullat & Schneider, 1996; Darmont, Fromantin, Regnier, Gruenwald, & Schneider, 2000; He, Marquez, & Blackburn, 2000). There are three reasons for choosing to test the effectiveness of DoEF using dynamic clustering algorithms:

1. ever since the “early days” of object database management systems, clustering has been proven to be one of the most effective performance enhancement techniques (Gerlhof, Kemper, & Moerkotte, 1996);
2. the performance of dynamic clustering algorithms is very sensitive to changing access patterns; and
3. despite this sensitivity, no previous attempt has been made to benchmark these algorithms in this way.

Next, the utility of DoEF is further demonstrated by benchmarking two transactional object stores: Platypus (He, Blackburn, Kirby, & Zigman, 2000) and SHORE (Carey et al., 1994).

The remainder of this chapter is organised as follows. The next section provides an overview of existing DBMS benchmarks. Then, in the next two sections, we describe in detail the DEF framework and its object-oriented instance DoEF. Next, we present and discuss the experimental results we achieved with DoEF. We finally conclude this chapter and provide future research directions in the last section.

STATE-OF-THE-ART: EXISTING DATABASE BENCHMARKS

We provide in this section an overview of the prevalent benchmarks that have been proposed in the literature for evaluating the performances of DBMSs. Note that, to the best of our knowledge, none of these benchmarks incorporate any dynamic application behaviour.

In the world of relational databases, the Transaction Processing Performance Council (TPC), a non-profit institute founded in 1988, defines standard benchmarks, verifies their correct application, and publishes the results. The TPC benchmarks include TPC-C (TPC, 2005) for OLTP; and TPC-H (TPC, 2003a) and TPC-R (TPC, 2003b) for decision support. These last benchmarks were to be replaced by the TPC-DS data warehouse benchmark (Poess, Smith, Kollar, & Larson, 2002), but it is not completed yet and alternatives have appeared, such as the data warehouse engineering benchmark (Darmont, Bentayeb, & Boussaid, 2005). Finally, the TPC has also specified benchmarks for Web commerce: TPC-W (TPC, 2002) and Web services: TPC-App (TPC, 2004). All these benchmarks feature an elaborate database and set of operations, and, except for DWEB, both are fixed. In the TPC benchmarks, the only parameter is indeed the database size (scale factor).

In contrast, there is no standard object-oriented database benchmark. However, the OO1 benchmark (Cattell, 1991), the HyperModel benchmark (Anderson, Berre, Mallison, Porter, & Schneider, 1990), and the OO7 benchmark (Carey, DeWitt, & Naughton, 1993) may be considered as *de facto* standards. They are all designed to mimic engineering applications such as CAD, CAM, or CASE applications. They range from OO1, which has a very simple schema (two classes) and only three simple operations, to OO7, which is more generic and provides both a much richer and more customisable schema (ten classes), and a wider range of operations (15 complex operations). However, even OO7's schema is static and still not generic enough to model other types of applications like financial, telecommunications, and multimedia applications (Tiwary, Narasayya, & Levy, 1995). Furthermore, each step in adding complexity makes these benchmarks harder to implement. Finally, the object clustering benchmark (OCB) has been proposed as a generic benchmark that is able to simulate the behaviour of other main object-oriented benchmarks (Darmont, Petit, & Schneider, 1998; Darmont & Schneider, 2000). OCB is further detailed in the object clustering benchmark section of this chapter.

Object-relational benchmarks, such as the BUCKY benchmark (Carey et al., 1997) and the benchmark for object-relational databases (BORD) (Lee, Kim, & Kim, 2000), are

query-oriented benchmarks that are specifically aimed at evaluating the performances of object-relational database systems. For instance, BUCKY only features operations that are specific to object-relational systems, since typical object navigation has already been tested by other benchmarks (see above). Hence, these benchmarks focus on queries involving object identifiers, inheritance, joins, class references, inter-object references, set-valued attributes, flattening queries, object methods, and various abstract data types. The database schema is also static in these benchmarks.

Carey and Franklin have also designed a set of workloads for measuring the performance of their client-server object-oriented database management systems (OODBMSs) (Carey, Franklin, Livny, & Shekita, 1991; Franklin, Carey, & Livny, 1993). These workloads operate at the page grain instead of the object grain, that is, synthetic transactions read or write pages instead of objects. The workloads contain the notion of hot and cold regions (some areas of database are more frequently accessed compared to others), attempting to approximate real application behaviour. However, the hot region never moves, meaning no attempt is made to model dynamic application behaviour.

Finally, a new family of benchmarks has recently appeared to specifically evaluate the performances of XML databases in various contexts: data-centric or document-centric XML databases, single or multi-document XML databases, global or micro benchmark, and so on (Lu et al., 2005). These so-called XML benchmarks include XMach-1 (Böhme & Rahm, 2001), XOO7, an XML extension of OO7 (Bressan, Lee, Li, Lacroix, & Nambiar, 2002), the Michigan benchmark (Runapongsa, Patel, Jagadish, & Al-Khalifa, 2002), XMark (Schmidt et al., 2002), and XBench (Yao, Ozsu, & Khandelwal, 2004). However, none of them evaluate the dynamic behaviour of XML database applications.

THE DYNAMIC EVALUATION FRAMEWORK (DEF)

The primary goal of DEF is to evaluate the dynamic performance of DBMSs. To make the work of DEF more general, we have made two key decisions: define DEF as an extensible framework; and reuse existing and standard benchmarks when available.

Dynamic Framework

We start by giving an example scenario that the framework can mimic. Suppose we are modelling an online book store in which certain groups of books are popular at certain times. For example, travel guides to Australia may have been very popular during the 2000 Olympics. However, once the Olympics were over, these books suddenly or gradually became less popular.

Once the desired book has been selected, information relating to the book may be required. Examples of required information include customer reviews of the book, excerpts from the book, picture of the cover, and the like. If the data are stored in an ODBMS, retrieving the related information is translated into an object graph navigation with the traversal root being the selected book. After looking at the related information for the selected book, the user may choose to look at another book by the same author. When information relating to the newly selected book is requested, the newly selected book becomes the root of a new object graph traversal.

Next, we give an overview of the five main steps of the dynamic framework and in the process show how the above example scenario fits in.

1. **H-region parameters specification:** The dynamic framework divides the database into regions of homogeneous access probability (H-regions). In our example, each H-region represents a different group of books, each group having its own probability of access. In this step, we specify the characteristics of each H-region, for example, its size, initial access probability, and so on.
2. **Workload specification:** H-regions are responsible for assigning access probability to pieces of data (tuples or objects). However, H-regions do not dictate what to do then. We term the selected tuple or object workload root. In the remainder of this chapter, we use the term “root” to mean workload root. In this step, we select the type of workload to execute after selecting the root.
3. **Regional protocol specification:** Regional protocols use H-regions to accomplish access pattern change. Different styles of access pattern change can be accomplished by changing the H-region parameter values with time. For example, a regional protocol may initially define one H-region with a high-access probability, while the remaining H-regions are assigned low-access probabilities. After a certain time interval, a different H-region may become the high-access probability region. This, when translated to the book store example, is similar to Australian travel books becoming less popular after the 2000 Olympics ended.
4. **Dependency protocol specification:** Dependency protocols allow us to specify a relationship between the currently selected root and the next root. In our example, this is reflected in a customer deciding to select a book that is by the same author as the previously selected book.
5. **Regional and dependency protocol integration specification:** In this step, regional and dependency protocols are integrated to model changes in dependency between successive roots. An example is a customer using our online book store, who selects a book of interest, and then is confronted with a list of currently popular books by the same author. The customer then selects one of the listed books (modelled by dependency protocol). The set of currently popular books by the same author may change with time (modelled by regional protocol).

The first three steps we have described are generic, that is, they can be applied on any selected benchmark and system type (relational, object-oriented, or object-relational). The two last steps are similar when varying the system type, but are nonetheless different because access paths and methods are substantially different in a relational system (with tables, tuples, and joins) and an object-oriented system (with objects and references), for instance.

Next, we further detail the concept of H-region and the generic regional protocol specification.

H-Regions

H-regions are created by partitioning the objects of the database into non-overlapping sets. All objects in the same H-region have the same access probability. Here we use the term access probability to mean the likelihood that an individual object of the H-region will be accessed at a given moment in time. The parameters that define an H-region are listed as follows:

- **HR_SIZE:** The size of the H-region is specified as a fraction of the database size. *Constraint:* The sum size of all regions must equal 1.
- **INIT_PROB_W:** The initial probability weight that is assigned to the region. The actual probability is derived from the probability weight, by dividing the probability weight of the region by the sum probability weight of all regions.
- **LOWEST_PROB_W:** The lowest probability weight this region can go have.
- **HIGHEST_PROB_W:** The highest probability weight this region can have.
- **PROB_W_INCR_SIZE:** The amount by which the probability weight of this region increases or decreases when change is requested.
- **OBJECT_ASSIGN_METHOD:** This determines the way objects are assigned into this region. The options are random selection and by class selection. Random selection picks objects randomly from anywhere in the database. By class selection places attempts to assign objects of the same class into the same H-region, as much as possible. It first sorts objects by class ID and then picks the first N objects (in sorted order), where N is the number of objects allocated to the H-region.
- **INIT_DIR:** The initial direction in which the probability weight increment moves.
- The access probability of an H-region can never be below *LOWEST_PROB_W* or above *HIGHEST_PROB_W*.

Regional Protocols

Regional protocols simulate access pattern change by first initializing the parameters of every H-region, and then periodically changing the parameter values in certain predefined ways. This chapter documents three styles of regional change: moving window of change, gradual moving window of change, and cycles of change. Although these three styles of change together provide a good spectrum of ways in which access pattern can change, they are by no means exhaustive. Other researchers or framework users are encouraged to create new regional protocols of their own.

Moving Window of Change Protocol

This regional protocol simulates sudden changes in access pattern. In our online book store, this is translated to books suddenly becoming popular due to some event, and once the event passes, the books become unpopular very fast. For instance, books that are recommended in a TV show may become very popular in the few days after the show, but may quickly become unpopular when the next set of books are introduced. This style of change is accomplished by moving a window through the database. The objects in the window have a much higher probability of being chosen as root when compared to the remainder of the database. This is done by breaking up the database into N H-regions of equal size. One H-region is first initialised to be the hot region (where heat is used to denote probability of reference), and then after H root selections, a different H-region becomes the hot region. H is a user-defined parameter that reflects the rate of access pattern change.

- The database is broken up into N regions of equal size.
- All H-regions have the same value for *HIGHEST_PROB_W*, *LOWEST_PROB_W* and *PROB_W_INCR_SIZE*.

- Set the *INIT_PROB_W* of one of the H-regions to equal *HIGHEST_PROB_W* (the hot region) and the rest of the H-regions get their *INIT_PROB_W* assigned to *LOWEST_PROB_W*.
- Set *PROB_W_INCR_SIZE* of every region to equal *HIGHEST_PROB_W - LOWEST_PROB_W*.
- The *INIT_DIR* parameter of all the H-regions is set to move downwards. Initially, the window is placed at the hot region. After every *H* root selections, the window moves from one H-region to another. The H-region that the window is moving *from* has its direction set to *down*. The H-region that the window is moving *into* has its direction set to *up*. Then, probability weights of the H-regions are incremented or decremented depending on the current direction of movement.

Gradual Moving Window of Change Protocol

The way this protocol differs from the previous one is that the hot region cools down gradually instead of suddenly. The cold regions also heat up gradually as the window is moved onto them. In our book store example, this style of change may depict travel guides to Australia gradually becoming less popular after the 2000 Sydney Olympics. As a consequence, travel guides to other countries may gradually become more popular. Gradual changes of heat may be more common in the real world.

This protocol is specified in the same way as the previous protocol with two exceptions. First, *PROB_W_INCR_SIZE* is now user-specified instead of being the difference between *HIGHEST_PROB_W* and *LOWEST_PROB_W*. The value of *PROB_W_INCR_SIZE* determines how vigorously access pattern changes at every change iteration. We use the term change iteration to mean the changing of access probabilities of the H-regions after every *H* (defined in the previous section) root selections. The second exception is in the way the H-regions change direction. The H-region that the window moves into has its direction toggled. The direction of the H-region that the window is moving from is unchanged. This way, the previous

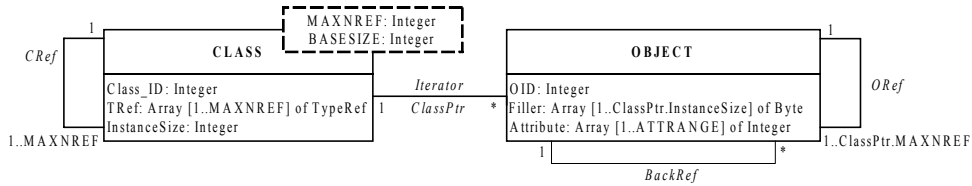
H-region is able to continue cooling down gradually or heating up gradually. When the access probability of a cooling H-region reaches its *LOWEST_PROB_W*, it stops cooling and similarly a heating up H-region stops heating up when it reaches its *HIGHEST_PROB_W*.

Cycles of Change Protocol

This style of change mimics something like a bank where customers in the morning tend to be of one type (e.g., social category), and in the afternoon of another type. This, when repeated, creates a cycle of change. Cycles of change can be simulated using the following steps. Members of a set are not ordered.

- Break up the database into three H-regions. The first two H-regions represent objects going through the cycle of change. The third H-region represents the remaining unchanged part of the database. The *HR_SIZE* of the first two H-regions are equal to each other and user-specified. The *HR_SIZE* of the third H-region is equal to the remaining fraction of the database.
- Set the *LOWEST_PROB_W* and *HIGHEST_PROB_W* parameters of the first two H-regions to values that reflect the two extremes of the cycle.

Figure 1. OCB database schema



- Set the *PROB_W_INCR_SIZE* of the first two H-regions to both equal *HIGHEST_PROB_W-LOWEST_PROB_W*. Set the *PROB_W_INCR_SIZE* of the third H-region to equal zero.
- The *INIT_PROB_W* of the first H-region is set to *HIGHEST_PROB_W* and the second to *LOWEST_PROB_W*.
- Set the *INIT_DIR* of the hot H-region to down and the *INIT_DIR* of the cold H-region to up.
- Again, the *H* parameter is used to vary the rate of access pattern change.

THE DYNAMIC OBJECT EVALUATION FRAMEWORK (DOEF)

In this section, we describe DoEF, which is an instance of DEF. DoEF is built on top of the object clustering benchmark (OCB) and uses both the database built from the rich schema of OCB and the operations offered by OCB. Since OCB's generic model can be implemented within an object-relational system and most of its operations are relevant for such a system, DoEF can also be used in the object-relational context.

Next, we present the OCB benchmark and then detail the steps in DEF that are specific to the object-oriented context, namely, the specification of the dependency protocols and their integration with the regional protocols.

The Object Clustering Benchmark (OCB)

OCB is a generic, tunable benchmark aimed at evaluating the performances of OODBMSs. It was first oriented toward testing clustering strategies (Darmont et al., 1998) and was later extended to become fully generic (Darmont & Schneider, 2000). The flexibility and scalability of OCB is achieved through an extensive set of parameters. OCB is able to simulate the behaviour of the de facto standards in object-oriented benchmarking, namely OO1 (Cattell, 1991), HyperModel (Anderson et al., 1990), and OO7 (Carey et al., 1993). Furthermore, OCB's generic model can be easily implemented within an object-relational system, and most of its operations are relevant for such a system. We only provide here an overview of OCB. Its complete specification is available in Darmont and Schneider (2000). The two main components of OCB are its database and workload.

Table 1. OCB database main parameters

Parameter Name	Parameter	Default Value
NC	Number of classes in the database	50
MAXNREF(<i>i</i>)	Maximum number of references, per class	10
BASESIZE(<i>i</i>)	Instances base size, per class	50 bytes
NO	Total number of objects	20,000
NREFT	Number of reference types	4
ATTRANGE	Number of integer attributes in an object	1
CLOCREF	Class locality of reference	NC
OLOCREF	Object locality of reference	NO

Database

The OCB database is made up of *NC* classes derived from the same metaclass (Figure 1). Classes are defined by two parameters: *MAXNREF*, the maximum number of references in the instances, and *BASESIZE*, an increment size used to compute the *InstanceSize*. Each *CRef* (class reference) has a type: *TRef*. There are *NTREF* different types of references (e.g., inheritance, aggregation...). Finally, an *Iterator* is maintained within each class to save references toward all its instances.

Each object possesses *ATTRANGE* integer attributes that may be read and updated by transactions. A *Filler* string of size *InstanceSize* is used to simulate the actual size of the object. After instantiating the schema, an object *O* of class *C* points through the *ORef* references to at most *C.MAXNREF* objects. There is also a backward reference (*BackRef*) from each referenced object toward the referring object *O*.

The database generation proceeds through three steps:

1. **Instantiation of the *CLASS* metaclass into *NC* classes and selection of class level references:** Class references are selected to belong to the [*Class_ID* - *CLOCREF*, *Class_ID* + *CLOCREF*] interval. This models locality of reference at the class level.
2. **Database consistency check-up:** Suppression of all cycles and discrepancies within the graphs that do not allow them, for example, inheritance graphs or composition hierarchies.
3. **Instantiation of the *NC* classes into *NO* objects and random selection of the object references:** Object references are selected to belong to the [*OID* - *OLOCREF*, *OID* + *OLOCREF*] interval. This models locality of reference at the instance level.

The main database parameters are summarized in Table 1.

Workload

The operations of OCB are broken up into four categories:

1. **Random Access:** Access to *NRND* randomly selected objects.
2. **Sequential Scan:** Randomly select a class and then access all its instances. A *Range Lookup* additionally performs a test on the value of *NTEST* attributes, for each accessed instance.

3. **Traversal:** There are two types of traversals in OCB. *Set-oriented accesses* (or associative accesses) perform a breadth-first search. *Navigational Accesses* are further divided into *Simple Traversals* (depth-first searches), *Hierarchy Traversals* that always follow the same reference type, and *Stochastic Traversals* that select the next link to cross at random. Each traversal proceeds from a randomly chosen root object, and up to a predefined depth. All the traversals can be reversed by following the backward links.
4. **Update:** Update operations are also subdivided into different types. *Schema Evolutions* are random insertions and deletions of *Class* objects (one at a time). *Database Evolutions* are random insertions and deletions of objects. *Attribute Updates* randomly select *NUPDT* objects to update, or randomly select a class and update all of its objects (*Sequential Update*).

In DoEF, the workload type is selected from these types. For *sequential scans*, the class of the root object is used to decide which objects are scanned; for *traversals*, the root object becomes the root of the traversal; and for *updates*, either the class of the root object or just the root object is used to decide which objects are updated (depending on the particular update workload selected).

Dependency Protocols

There are many scenarios in which a person executes a query and then decides to execute another query based on the results of the first query, thus establishing a dependency between the two queries. In this chapter, we have specified four dependency protocols: random selection protocol, by reference selection protocol, traversed objects selection protocol, and same class selection protocol. Again, these protocols are not meant to be exhaustive, and other researchers or benchmark users are encouraged to extend DoEF beyond these dependency protocols.

Random Selection Protocol

This method simply uses some random function to select the current root. This protocol mimics a person starting a completely new query after finishing the previous one.

$$r_i = \text{RANDI}()$$

r_i is the ID of the i^{th} root object. The function $\text{RANDI}()$ can be any random function. An example of $\text{RANDI}()$ is a skewed random function that selects a certain group of root objects with a higher probability than others.

By Reference Selection Protocol

The current root is chosen to be an object referenced by the previous root. An example of this protocol in our online book store scenario is a person who, having finished with a selected book, then decides to look at the next book in the series (assuming the books of the same series are linked together by structural references).

$$r_{i+1} = \text{RAND2}(\text{RefSet}(r_i, D))$$

$\text{RefSet}(r_i, D)$ is a function that returns the set of objects that the i^{th} root references. $\text{RAND2}()$, like $\text{RAND1}()$ can be any random function. Two types of references can be used: structure references (S-references) and D-references. Structure references are simply the references obtained from the object graph. D-references are a new type of reference used for the sole purpose of establishing dependencies between roots of traversals. The parameter D is used to specify the number of D-references per object. Note if structure references are specified, then parameter D is not used.

Traversed Objects Selection Protocol

The current root is selected from the set of objects that are referenced in the previous traversal. An example is a customer in the first query requesting a list of books along with their authors and publishers (thus requiring the book objects themselves to be retrieved), who then decides to read an excerpt from one of the books listed.

$$r_{i+1} = \text{RAND3}(\text{TraversedSet}(r_i, C))$$

$\text{TraversedSet}(r_i, C)$ returns the set of objects referenced during the traversal that began with the i^{th} root. $\text{RAND3}()$, like $\text{RAND1}()$ can be any random function. The parameter C is used to restrict the number of objects returned by $\text{TraversedSet}(r_i, C)$. C is specified as a fraction of the objects traversed. This way, the degree of locality of objects returned by $\text{TraversedSet}(r_i, C)$ can be controlled (smaller C means higher degree of locality).

Same Class Selection Protocol

In same class selection, the currently selected root must belong to the same class as the previous root. Root selection is further restricted to a subset of objects of the class. The subset is chosen by a function that takes the previous root as a parameter. That is, the subset chosen is dependent on the previous root object. An example of this protocol is a customer deciding to select a book from our online book store that is by the same author as the previous selected book. In this case, the same class selection function returns books by the same author as the selected book.

$$r_{i+1} = \text{RAND4}(f(r_i, \text{Class}(r_i), U))$$

$\text{Class}(r_i)$ returns the class of the i^{th} root. $\text{RAND4}()$, like $\text{RAND1}()$ can be any random function. The parameter U is user-defined and specifies the size of the set returned by function $f()$. U is specified as a fraction of the total class size. U can be used to increase or decrease the degree of locality between the objects returned by $f()$. $f()$ always returns the same set of objects given the same set of parameters.

Hybrid Setting

The hybrid setting allows an experiment to use a mixture of the dependency protocols outlined above. Its use is important since it simulates a user starting a fresh random query after having followed a few dependencies. Thus, the hybrid setting is

implemented in two phases. The first *randomisation phase* uses the random selection protocol to randomly select a root. In the second *dependency phase*, one of the dependency protocols outlined in the previous section is used to select the next root. R iterations of the second phase are repeated before going back to the first phase. The two phases are repeated continuously.

The probability of selecting a particular dependency protocol during the *dependency phase* is specified via the following settings: *RANDOM_DEP_PROB* (random selection), *SREF_DEP_PROB* (by reference selection using structure references), *DREF_DEP_PROB* (by reference selection using D-references), *TRAVERSED_DEP_PROB* (traversed objects selection), and *CLASS_DEP_PROB* (same class selection).

Integration of Regional and Dependency Protocols

Dependency protocols model user behaviour. Since user behaviour can change with time, dependency protocols should also be able to change with time. The integration of regional and dependency protocols allows us to simulate changes in the dependency between successive root selections. This is easily accomplished by exploiting the dependency protocols' property of returning a candidate set of objects when given a particular previous root. Up to now, the next root is selected from the candidate set by the use of a random function. Instead of using the random function, we partition the candidate set using H-regions and then apply regional protocols on these H-regions. When integrating with the traversed objects dependency protocol, the following property must hold: whenever given the same root object, the same set of objects is always traversed. This way, the same previous root will return the same candidate set.

EXPERIMENTS AND RESULTS

This section details two sets of experiments we have conducted to evaluate the effectiveness of DoEF. In the first set of experiments, four state-of-the-art dynamic clustering algorithms are benchmarked. In the second set, two real object stores are benchmarked.

For dynamic clustering algorithms, we have conducted two sets of experiments: moving and gradual moving window of change regional protocol experiments; and moving and gradual moving S-reference protocol experiments. For the real object stores, we also conducted two sets of experiments: moving window of change protocol experiments, and moving window of change traversed objects experiment. There are two reasons for choosing these set of protocols to test: the space constraints prohibit us from showing results obtained using all combinations of protocols; and after testing many of the possible combinations we found for the particular clustering algorithms and real OODBs we have tested, the experiments presented gives the greatest insight into the effectiveness of DoEF.

Tested Systems and Algorithms

In this section, we briefly describe the dynamic clustering algorithms and object stores we have used in our experiments.

Dynamic Clustering Algorithms

Dynamic clustering is the periodic online reorganisation of objects in an ODBMS. The aim is to allow the physical placement of objects on disk to more closely reflect the pervading pattern of database access. Objects that are likely to be accessed together in the near future are placed in the same page, thereby reducing the number of disk I/Os.

Dynamic, statistical, and tuneable clustering (DSTC) (Bullat & Schneider, 1996) is a dynamic clustering algorithm that has the feature of achieving dynamicity without adding high statistics collection overhead or excessive volumes of statistics. However, it does not take care to reduce the I/O generated by the clustering process itself. The clustering algorithm is not very selective when deciding which pages to re-cluster. The effect is a page is re-clustered even if there is a slight benefit in re-clustering it. However, the slight benefit gained from re-clustering is often outweighed by the cost of loading the page into memory for re-clustering. This situation (re-clustering of slightly badly clustered pages) will become more frequent as access pattern changes more rapidly. For this reason, we expect that DSTC will perform poorly when access pattern changes rapidly.

Detection and recluster of objects (DRO) (Darmont et al., 2000) capitalizes on the experiences of DSTC and StatClust (Gay & Gruenwald, 1997) to produce less clustering I/O overhead and use less statistics. DRO uses various thresholds to limit the pages involved in re-clustering to only the pages that are most in need of re-clustering. We term this flexible conservative re-clustering. Experiments conducted using OCB show that DRO outperforms DSTC (Darmont et al., 2000). The improvement in performance is mainly attributed to the low clustering I/O overhead of DRO. In order to limit statistics collection overhead, DRO only uses object frequency and page usage rate information. In contrast, DSTC stores object transition information, which is much more costly. Since DRO chooses only a limited number of the worst clustered pages to re-cluster (flexible conservative re-clustering), it should perform better than DSTC when access pattern changes rapidly. This is because when access pattern changes rapidly, the benefits in re-clustering pages become lower and thus there will be more pages that only benefit slightly from re-clustering. DRO does not re-cluster these pages, whereas DSTC does. This leads DSTC to generate larger clustering overhead for very slight improvements in clustering quality.

Opportunistic prioritised clustering framework (OPCF) (He, Marquez, & Blackburn, 2000) is a framework for translating any static clustering algorithm (where re-clustering occurs off-line) into a dynamic clustering algorithm. OPCF creates algorithms that have the following key properties: read and write I/O opportunism and prioritisation of re-clustering. Read and write I/O opportunism refers to limiting re-clustering to pages that are currently in memory (in the case of read opportunism) and dirty (in the case of write opportunism). This approach reduces the I/O overhead associated with re-clustering. Prioritisation of re-clustering refers to choosing a limited number of the worst clustered pages to be re-clustered first. This also reduces clustering overhead by reducing the number of pages re-clustered. Therefore, OPCF clustering algorithms also perform flexible conservative re-clustering. Two dynamic clustering algorithms produced from the OPCF framework are presented in He et al. (2000): dynamic graph partitioning algorithm (GP) and dynamic probability ranking principle algorithm (PRP). Since OPCF, like DRO, performs flexible conservative re-clustering, it should also perform well when access pattern changes very rapidly. We use the term flexible clustering algorithms to refer to DRO and the OPCF dynamic clustering algorithms.

Object Stores

Platypus (He, Blackburn, Kirby, & Zigman, 2000) is a flexible high-performance transactional object store, designed to be used as the storage manager for persistent programming languages. The design includes support for symmetric multiprocessing (SMP) concurrency: stand-alone, client-server, and client-peer distribution configurations; configurable logging and recovery; and object management that can accommodate garbage collection and clustering mechanisms. In addition to these features, Platypus is built for speed. It features a new recovery algorithm derived from the popular ARIES (Mohan, Haderle, Lindsay, Pirahesh, & Schwarz, 1992) recovery algorithm, which removes the need for log sequence numbers to be present in store pages; a zero-copy memory-mapped buffer manager with controlled write-back behaviour; and a novel fast and scalable data structure (splay trees) used extensively for accessing metadata.

SHORE (Carey et al., 1994) is a transactional persistent object system that is designed to serve the needs of a wide variety of target applications, including persistent programming languages. It has a peer-to-peer distribution configuration. Like Platypus, it also has a focus on performance.

Dynamic Clustering Experiments

These experiments use DoEF to compare the performance of four state-of-the-art dynamic clustering algorithms: DSTC, DRO, OPCF-PRP, and OPCF-GP (see test systems and algorithms section for more details). The parameters we have used for the dynamic clustering algorithms are shown in Table 2. In the interest of space, we do not include their description in this chapter; however, they are wholly described in their respective papers. The clustering techniques have been parameterized for the same behaviour and best performance.

The experiments are conducted on the Virtual Object-Oriented Database simulator (VOODB) (Darmont & Schneider, 1999). VOOB is based on a generic discrete-event simulation framework. Its purpose is to allow performance evaluations of OODBMSs in general, and optimisation methods like clustering in particular. VOOB has been validated for two real-world OODBMSs, O2 (Deux, 1991) and Texas (Singhal, Kakkad, & Wilson, 1992). The VOOB parameter values we have used are depicted in Table 3 (a). Simulation is chosen for this experiment for two reasons. First, it allows rapid development and

Table 2. DSTC, DRO, OPCF-PRP, and OPCF-GP parameters

Parameter	Value
n	200
n _p	1
p	1000
T _{fa}	1.0
T _{fè}	1.0
T _{fç}	1.0
w	0.3

(a) DSTC

Parameter	Value
M _{inUR}	0.001
M _{inLT}	2
P _{CRate}	0.002
M _{axD}	1
M _{axDR}	0.2
M _{axRR}	0.95
SU _{Ind}	true

(b) DRO

Parameter	PRP Value	GP Value
N	200	200
CBT	0.1	0.1
NPA	50	50
NRI	25	25

(c) OPCF

Table 3. VOODB (a) and OCB parameters (b)

Parameter Description	Value
System class	Centralized
Disk page size	4096 bytes
Buffer size	4 MB
Buffer replacement policy	LRU-1
Pre-fetching policy	None
Multiprogramming level	1
Number of users	1
Object	Sequential

(a) VOODB parameters

Parameter Description	Value
Number of classes in the database	50
Maximum number of references, per class	10
Instances base size, per class	50
Total number of objects	100,000
Number of reference types	4
Reference types random distribution	Uniform
Class reference random distribution	Uniform
Objects in classes random distribution	Uniform
Objects references random distribution	Uniform

(b) OCB parameters

testing of a large number of dynamic clustering algorithms (all previous dynamic clustering papers compared at most two algorithms). Second, it is relatively easy to simulate accurately, read, write, and clustering I/O (the dominating metrics that determine the performance of dynamic clustering algorithms).

Since DoEF uses the OCB database and operations, it is important for us to document the OCB settings we have used for these experiments. The values of the database parameters we have used are shown in Table 3 (b). The size of the objects we have used varies from 50 to 1600 bytes, with the average size being 233 bytes. A total of 100,000 objects are generated for a total database size of 23.3 MB. Although this is a small database size, we have also used a small buffer size (4 MB) to keep the database to buffer size ratio large. Clustering algorithm performance is indeed more sensitive to database to buffer size ratio than database size alone.

The operation we have used for all the experiments is the simple, depth-first traversal with traversal depth 2. The simple traversal is chosen since it is the only traversal that always accesses the same set of objects given a particular root. This establishes a direct relationship between varying root selection and changes in access pattern. Each experiment involved executing 10,000 transactions.

Table 4. DoEF parameters

Parameter Name	Value
<i>HR_SIZE</i>	0.003
<i>HIGHEST_PROB_W</i>	0.80
<i>LOWEST_PROB_W</i>	0.0006
<i>PROB_W_INCR_SIZE</i>	0.02
<i>OBJECT_ASSIGN_METHOD</i>	Random object assignment

The main DoEF parameter settings we have used in this study are shown in Table 4. These DoEF settings are common to all experiments in this chapter. The *HR_SIZE* setting of 0.003 (remember this is the database population from which the traversal *root* is selected) creates a hot region about 3% the size of the database (each traversal touches approximately 10 objects). This fact is verified from statistical analysis of the trace generated. The *HIGHEST_PROB_W* setting of 0.8 and *LOWEST_PROB_W* setting of 0.0006, produces a hot region with 80% probability of reference and the remaining cold regions with a combined reference probability of 20%. These settings are chosen to represent typical database application behaviour. Gray and Putzolu (1987) cite statistics from a real videotext application in which 3% of the records got 80% of the references. Carey et al. (1991) use a hot region size of 4% with a 80% probability of being referenced in the HOTCOLD workload we have used to measure data caching tradeoffs in client-server OODBMSs. Franklin, Carey, and Livny (1993) use a hot region size of 2% with a 80% probability of being referenced in the HOTCOLD workload we have used to measure the effects of local disk caching for client server OODBMSs. In addition to the results reported in this chapter, we also tested the sensitivity of the results to variations in hot region size and probability of reference. We found the algorithms show similar general tendencies at different hot region sizes and probability of reference. It is for this reason and in the interest of space that we omit these results.

The dynamic clustering algorithms shown on the graphs in this section are labelled as follows:

- **NC:** No Clustering;
- **DSTC:** Dynamic Statistical Tunable Clustering;
- **GP:** OPCF (greedy graph partitioning);
- **PRP:** OPCF (probability ranking principle);
- **DRO:** Detection & Reclustering of Objects.

As we discuss the results of these experiments, we focus our discussion on the relative ability of each algorithm to adapt to changes in access pattern; that is, as rate of access pattern change increases, we seek to know which algorithm exhibits more rapid performance deterioration. This contrasts from discussing which algorithm gives the best absolute performance. All the results presented here are in terms of total I/O. Total I/O is the sum of transaction read I/O, clustering read, and clustering write I/O. Thus, the results give an overall indication of the performance of each clustering algorithm, including each algorithm's clustering I/O overhead.

A Priori Analysis

In this section we analyse the relative performances of the dynamic clustering algorithms based on the characteristics of DSTC, DRO, OPCF-PRP, and OPCF-GP. For the moving window of change protocol experiments, we expect the relative difference in performance between DSTC and the flexible clustering algorithms to increase with increasing rate of change. This is because DSTC does not do flexible conservative re-clustering and thus incurs high re-clustering overheads. The relative difference between the different flexible clustering algorithms should not change by much with increasing rate of change, since they all limit the clustering overheads to a bounded amount. In terms of the shapes of the curves, we expect DSTC to perform linearly worse with increasing rate of change. This is because it does not bind the clustering overhead. In contrast, the flexible dynamic clustering algorithms' performance will increase with increasing rate of change but will be flat after a certain point (we call this the saturation point). This is because these algorithms bound the clustering overhead and thus there is a bound on its performance.

In terms of the gradual moving window of change experiments, we expect the relative differences between the algorithms to stay similar as the rate of change increases. The reason is this change protocol is very mild and therefore does not cause the flexible clustering algorithms to reach their saturation point. In terms of the shapes of the curves, we expect the performance of all the algorithms to perform close to linear with increasing rate of change of access pattern. This is because increases in the rate of change of access pattern cause the benefit of re-clustering to diminish; this increase is constant and does not reach a saturation point due to the mild style of change.

Moving and Gradual Moving Regional Experiments

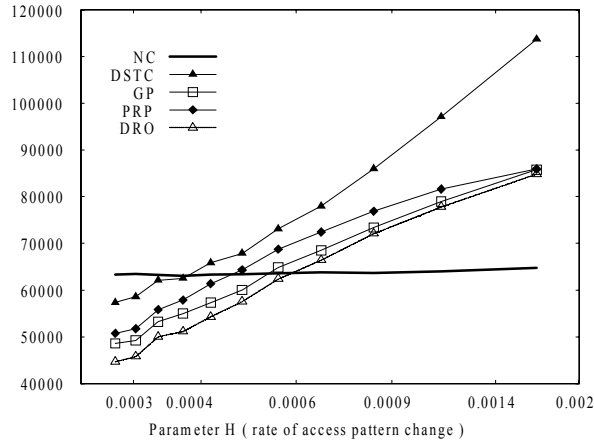
In these experiments, we have used the regional protocols moving window of change and gradual moving window of change to test each of the dynamic clustering algorithms' ability to adapt to changes in access pattern. The regional protocol settings we have used are shown in Table 4. We vary the parameter H , rate of access pattern change.

The results for these experiments are shown in Figure 2. There are three main results from this experiment. Firstly, when rate of access pattern change is small [when parameter H is less than 0.0006 in Figure 2 (a) and all of Figure 2 (b)], all algorithms show similar performance trends (rate of performance degradation). This implies that at moderate levels of access pattern change all algorithms are approximately equal in their ability to adapt to the change.

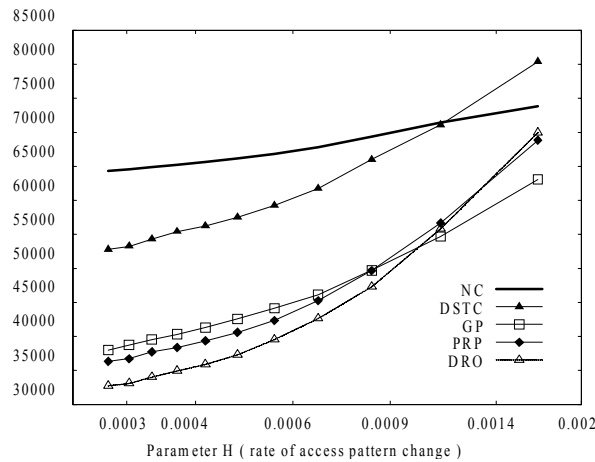
Secondly, when the more vigorous style of change is applied [Figure 2 (a)], all dynamic clustering algorithms' performance quickly degrades to worse than no clustering. Thirdly, when access pattern change is very vigorous [when parameter H is greater than 0.0006 in Figure 2 (a)], DRO and OPCF algorithms GP and PRP show a better trend performance (rate of performance degradation), implying DRO and OPCF are more robust to access pattern change. This supports our analysis in the a priori analysis section.

Moving and Gradual Moving S-Reference Experiments

In these experiments, we explore the effect that changing pattern of access has on the S-reference dependency protocol. This is accomplished by using the integrated regional dependency protocol method outlined in the integration of regional and depen-

Figure 2. Regional dependency results (x-axis is in $\log_2$ scale)

(a) Moving window of change

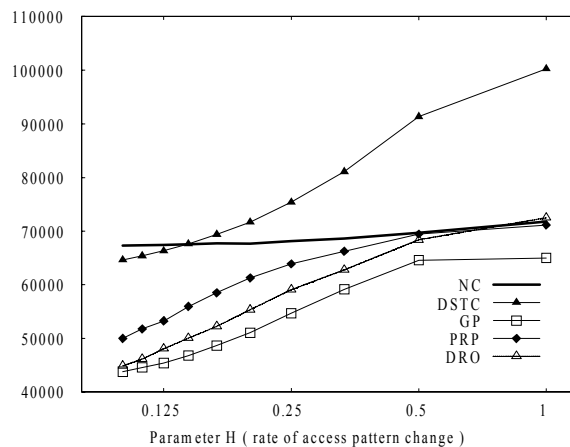


(b) Gradual moving window of change

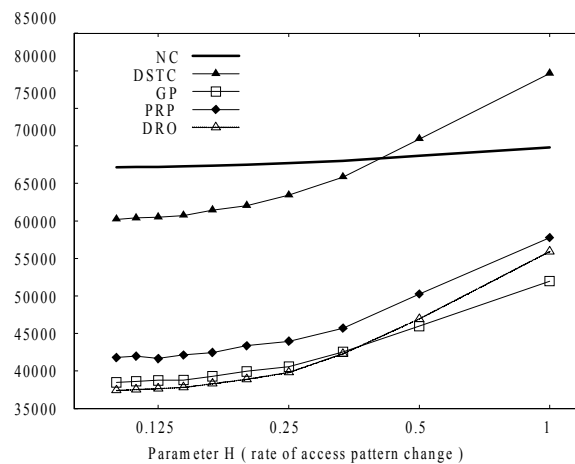
dependency protocols section. We integrated S-reference dependency with the moving and gradual moving window of change regional protocols. For this experiment, we use the hybrid dependency setting detailed in the hybrid setting section. R is set to 1. The first phase (random phase) of the hybrid setting requires a random dependency function. The random function we use partitions the database into one hot and one cold region. The hot region is set to be 3% of the database size and has an 80% probability of reference (typical database application behaviour (Carey et al., 1991; Franklin et al., 1993; Gray & Putzolu, 1987)). S-reference dependency is the only dependency protocol used. The regional protocol settings are as described in Table 4.

The results for these experiments are shown in Figure 3. In the moving window of change results [Figure 3 (a)], DRO and the OPCF algorithms (GP and PRP) are again more robust to changes in access pattern than DSTC for moving window of change. However, in contrast to the previous experiment, DRO and OPCF algorithms never perform worse than NC by much, even when parameter H is 1 (access pattern changes after every transaction). The reason is that the cooling and heating of S-references is a milder form of access pattern change than the pure moving window of change regional protocol of the previous experiment. In the gradual moving window of change results shown in Figure 3 (b), all dynamic clustering algorithms show approximately the same trend performance. This is similar to the observation made in the previous experiment. The analysis in the a priori analysis section supports these observations.

Figure 3. S-reference dependency results (x-axis is in $\log_2$ scale)



(a) Moving window of change



(b) Gradual moving window of change

Object Store Experiments

In this section, we report the results of using DoEF to compare the performance of two real object stores: SHORE and Platypus.

SHORE has a layered architecture that allows users to choose the level of support appropriate for a particular application. The lowest layer of SHORE is the SHORE Storage Manager (SSM), which provides basic object reading and writing primitives. Using the SHORE SSM, we have constructed PSI-SSM, a thin layer providing PSI (Blackburn, 1998) compliance for SSM. By using the PSI interface, the same DoEF code could be used for both Platypus and SHORE. The buffer replacement policy that SHORE uses is CLOCK. We use SHORE version 2.0 for all of the experiments reported in this chapter.

The Platypus implementation we have used for this set of experiments has the following features: physical object IDs; the NOFORCE/STEAL recovery policy (Franklin, 1997); zero-copy memory mapped buffer cache; the use of hash-splay trees to manage metadata; PSI compliance; the system is SMP re-entrant and supports multiple concurrent client threads per process as well as multiple concurrent client processes. Limitations of the Platypus implementation at the time of this writing include: the failure-recovery process is untested (although logging *is* fully implemented); virtual address space reserved for metadata is not recycled; the store lacks a sophisticated object manager with support for garbage collection and dynamic clustering, and lacks support for distributed store configurations. Platypus uses the LRU replacement policy.

In this set of experiments, the SHORE and Platypus implementations do not include dynamic clustering algorithms. In contrast to the previous experiment, we are interested here in comparing the other factors (besides clustering) that affect system performance. The experiments in this section are conducted using Solaris 7 on an Intel machine with dual Celeron 433Mhz processors, 512 MB of memory, and a 4 GB hard disk. The OCB database and workload settings we have used for this experiment are the same as for the previous set of experiments (see the dynamic clustering experiments section), except that a total of 400,000 objects are generated instead of 100,000. The reason for using a larger database size is that the real object stores are configured with a larger buffer cache, therefore, we need to increase the database size in order to test the swapping. The sizes of the objects we have used vary from 50 to 1200 bytes, with the average size being 269 bytes. Therefore, the total database size is 108 MB.

A Priori Analysis

For the moving window of change protocol experiments, we expect the performance of Platypus to start well in front of SHORE, but its lead should rapidly diminish as the rate of access pattern change increases. The reason lies in the change in access locality when the rate of access pattern changes. When the rate of access pattern change is low, access locality is high (due to a small and slow-moving hot region), and thus most object requests can be satisfied from the buffer cache. However, as the rate of access pattern change increases, access locality diminishes, which results in more buffer cache misses. Thus, the reason behind Platypus' poor performance lies in its poor swapping performance. Platypus' poor swapping performance is due to the low degree of concurrency (coarse-grained locking) between the page server and the client process when swapping is in progress (a deficiency in the implementation).

In the traversed objects protocol experiments, we expect the results to again show that the performance of Platypus diminishes at a faster rate than SHORE. The reason for this behaviour can again be explained by Platypus' poor swapping performance. However, the saturation is expected to occur at a later point than for the moving window of change protocol since the degree of locality in this protocol is higher.

Figure 4. Moving window of change regional protocol results (x-axis is in log2 scale)

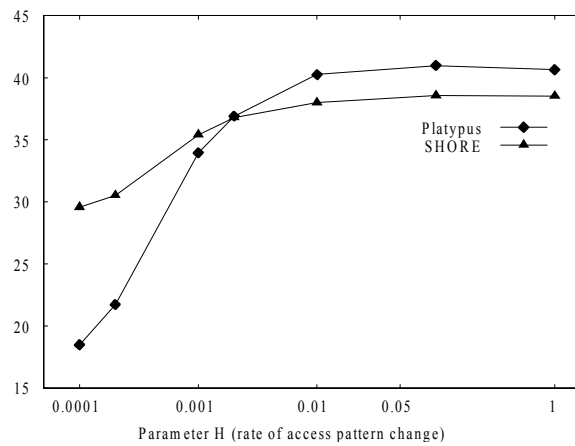
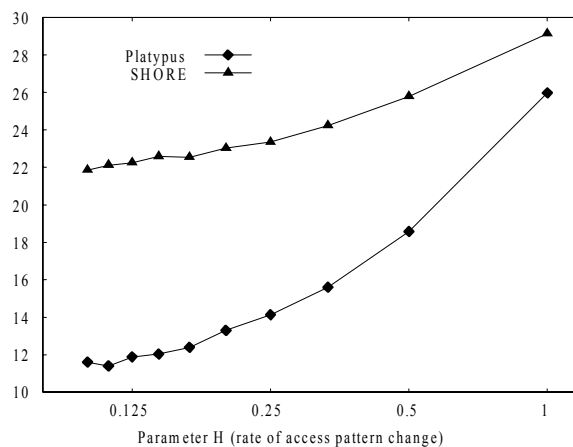


Figure 5. Traversed objects results. The x-axis is in log2 scale. The minimum and maximum coefficients of variation are 0.005 and 0.037, respectively.



Moving Window of Change Regional Experiment

In this experiment, we use the moving window of change protocol to compare the effects of changing pattern of access on Platypus and SHORE. The regional protocol settings we have used are the same as shown in Table 4. The buffer size is set to 61 MB. Note that both Platypus and SHORE have their own buffer managers with user-defined buffer sizes.

The results for this experiment are shown in Figure 4. The results show the trend predicted in the a priori analysis section, namely, the performance of Platypus starts well in front of SHORE, but its lead rapidly diminishes as the rate of access pattern change increases.

Moving Window of Change Traversed Objects Experiment

In this experiment, we compare the performance of Platypus and SHORE in the context of moving traversed objects dependency protocol. This is accomplished by using the integrated regional dependency protocol method outlined in the integration of regional and dependency protocols section. We have integrated traversed objects dependency protocol with the moving window regional protocol. For this experiment, we use the hybrid dependency setting detailed in the hybrid setting section. R is set to 1. The random function we use partitions the database into one hot and one cold region. The hot region is set to be 0.01 fraction of the database size, and the cold region is assigned the remaining portion of the database. 99% of the roots are selected from the hot region. The C parameter is set to 1.0. Traversed objects dependency is the only dependency protocol we have used. The regional protocol parameters we have used are identical to those used in the previous experiment, except that HR_SIZE is set to 0.05. In this experiment, the buffer size we have used is only 20 MB as opposed to 61 MB in the previous experiment, because this experiment has a smaller working set size; thus, at 61 MB, swapping would not occur (even when H is one). The reason behind the small working set size lies in the fact that the random function we have used does not move its hot region.

The results for this experiment are shown in Figure 5. Its behaviour is consistent with that described in the a priori analysis section.

CONCLUSION

In this chapter, we have conducted a short survey of existing benchmarking techniques. We have identified that no existing benchmark evaluates the dynamic performance of database applications. We then presented in detail the specification of a generic framework for database benchmarking, DEF, which allows DBMSs designers and users to test the performances of a given system in a dynamic setting. We have also instantiated DEF in an object-oriented context under the name of DoEF to illustrate how such specialization can be performed.

DEF is designed to be readily extensible along two axes. First, since, to the best of our knowledge, this is the first attempt at studying the dynamic behaviour of DBMSs, we

have taken great care to make the incorporation of new styles of access pattern change as painless as possible, mainly through the definition of H-regions. We actually view the DEF software as an open platform that is available to other researchers for testing their own ideas. The DoEF code we have used in both our object-clustering simulation experiments and our implementation for Platypus and SHORE is freely available for download.

Second, although we have considered an object-oriented environment in this study with DoEF, we can also apply the concepts developed in this chapter to other types of databases. Instantiating DEF for object-relational databases, for instance, should be relatively easy. Since OCB can be quite easily adapted to the object-relational context (even if extensions would be required, such as abstract data types or nested tables, for instance), DEF can be used in the object-relational context too.

The main objective of DEF is to allow researchers and engineers to explore the performance of databases (identify components that are causing poor performance) within the context of changing patterns of data access. Our experimental results involving dynamic clustering algorithms and real object stores have indeed demonstrated DoEF's ability to meet this objective. Within the dynamic clustering context, two new insights are gained: (1) dynamic clustering algorithms can deal with moderate levels of access pattern change, but performance rapidly degrades to be worse than no clustering when vigorous styles of access pattern change are applied; and (2) flexible conservative re-clustering is the key in determining a clustering algorithm's ability to adapt to changes in access pattern. In the performance comparison between the real object stores Platypus and SHORE, the use of DoEF allowed us to identify Platypus' poor swapping performance.

FUTURE TRENDS

In the past, most research has focused on the static optimization of database systems. As a result, this is now a very mature area. The next frontier in database optimization is to optimize queries while taking query patterns into consideration. This leads to the need to evaluate such systems in a quantitative manner. This study takes the first step in developing a benchmark for this purpose.

An interesting direction for future work is to use DEF to keep on acquiring knowledge about the dynamic behaviour of various commercial and research DBMSs. This knowledge could of course be used to improve the performance of these systems. Furthermore, comparing the dynamic behaviour of different systems, though an interesting task in itself, may inspire us to develop new styles of access pattern change. New styles of access pattern change identified in this and other ways may be incorporated into DEF.

Finally, the effectiveness of DEF at evaluating other aspects of database performance could also be explored. Data clustering is indeed an important performance optimisation technique, but other strategies such as buffer replacement and prefetching should also be evaluated.

REFERENCES

- Anderson, T. L., Berre, A.-J., Mallison, M., Porter, H. H., & Schneider, B. (1990). The HyperModel benchmark. In *International Conference on Extending Database Technology (EDBT 90)*, Venice, Italy (LNCS 416, pp. 317-331). Berlin: Springer-Verlag.
- Blackburn, S. M. (1998). *Persistent store interface: A foundation for scalable persistent system design*. PhD thesis, Australian National University, Canberra, Australia.
- Böhme, T., & Rahm, E. (2001). XMach-1: A benchmark for XML data management. In *Datenbanksysteme in Büro, Technik und Wissenschaft (BTW 01)*. Oldenburg, Germany (pp. 264-273). Berlin: Springer.
- Bressan, S., Lee, M.-L., Li, Y. G., Lacroix, Z., & Nambiar, U. (2002). The XOO7 benchmark. In *Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web, VLDB 2002 Workshop EEXTT, Hong Kong, China* (LNCS 2590, pp. 146-147). Berlin: Springer-Verlag.
- Bullat, F., & Schneider, M. (1996). Dynamic clustering in object databases exploiting effective use of relationships between objects. In *10th European Conference on Object-Oriented Programming (ECOOP 96)*, Linz, Austria (LNCS 1098, pp. 344-365). Berlin: Springer-Verlag.
- Carey, M. J., DeWitt, D. J., Franklin, M. J., Hall, N. E., McAuliffe, M., Naughton, J. F., et al. (1994, May 24-27). Shoring up persistent applications. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, Minneapolis, MN (pp. 383-394). New York: ACM Press.
- Carey, M., DeWitt, D., & Naughton, J. (1993, May 26-28). The OO7 benchmark. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, DC (pp. 12-21). New York: ACM Press.
- Carey, M., DeWitt, D., Naughton, J., Asgarian, M., Brown, P., Gehrke, J., & Shah, D. (1997, May 13-15). The BUCKY object-relational benchmark. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, Tucson, AZ (pp. 135-146). New York: ACM Press.
- Carey, M. J., Franklin, M. J., Livny, M., & Shekita, E. J. (1991, May 29-31). Data caching tradeoffs in client-server DBMS Architectures. In *Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data*, Denver, CO (pp. 357-366). New York: ACM Press.
- Cattell, R. (1991). An engineering database benchmark. In J. Gray (Ed.), *The benchmark handbook for database transaction processing systems* (pp. 247-281). San Francisco: Morgan Kaufmann.
- Darmont, J., Bentayeb, F., & Boussaid, O. (2005). DWEB: A data warehouse engineering benchmark. In *Proceedings of the 7th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 05)*, Copenhagen, Denmark (LNCS 3589, pp. 85-94). Berlin: Springer-Verlag.
- Darmont, J., Fromantin, C., Regnier, S., Gruenwald, L., & Schneider, M. (2000). Dynamic clustering in object-oriented databases: An advocacy for simplicity. In *ECOOP 2000 Symposium on Objects and Databases*, Sophia Antipolis, France (LNCS 1944, pp. 71-85). Berlin: Springer-Verlag.

- Darmont, J., Petit, B., & Schneider, M. (1998). OCB: A generic benchmark to evaluate the performances of object-oriented database systems. In *6th International Conference on Extending Database Technology (EDBT 98)*, Valencia, Spain (LNCS 1377, pp. 326-340). Berlin: Springer-Verlag.
- Darmont, J., & Schneider, M. (1999, September 7-10). VOODB: A generic discrete-event random simulation model to evaluate the performances of OODBs. In *Proceedings of the 25th International Conference on Very Large Databases (VLDB 99)*, Edinburgh, Scotland (pp. 254-265). San Francisco: Morgan Kaufmann.
- Darmont, J., & Schneider, M. (2000). Benchmarking OODBs with a generic tool. *Journal of Database Management*, 11(3), 16-27.
- Deux, O. (1991). The O2 system. *Communications of the ACM*, 34(10), 34-48.
- Franklin, M. J. (1997). Concurrency control and recovery. In A. B. Tucker (Ed.), *The computer science and engineering handbook* (pp. 1058-1077). Boca Raton, FL: CRC Press.
- Franklin, M. J., Carey, M. J., & Livny, M. (1993, August 24-27). Local disk caching for client-server database systems. In *Proceedings of the 19th International Conference on Very Large Data Bases (VLDB 93)*, Dublin, Ireland (pp. 641-655). San Francisco: Morgan Kaufmann.
- Gay, J., & Gruenwald, L. (1997). A clustering technique for object-oriented databases. In *Proceedings of the 8th International Conference on Database and Expert Systems Application (DEXA 97)*, Toulouse, France (LNCS 1308, pp. 81-90). Berlin: Springer-Verlag.
- Gerlhof, C., Kemper, A., & Moerkotte, G. (1996). On the cost of monitoring and reorganization of object bases for clustering. *SIGMOD Record*, 25(3), 22-27.
- Gray, J., & Putzolu, G. R. (1987). The 5-minute rule for trading memory for disk accesses and the 10-byte rule for trading memory for CPU time. In *Proceedings of the ACM SIGMOD 1987 Annual Conference*, San Francisco (pp. 395-398).
- He, Z., Blackburn, S. M., Kirby, L., & Zigman, J. (2000, September 6-8). Platypus: The design and implementation of a flexible high performance object store. In *Proceedings of the 9th International Workshop on Persistent Object Systems (POS-9)*, Lillehammer, Norway (pp. 100-124). Berlin: Springer.
- He, Z., & Darmont, J. (2003). DOEF: A dynamic object evaluation framework. In *Proceedings of the 14th International Conference on Database and Expert Systems Applications (DEXA 03)*, Prague, Czech Republic (LNCS 2736, pp. 662-671). Berlin: Springer-Verlag.
- He, Z., Marquez, A., & Blackburn, S. (2000). Opportunistic prioritised clustering framework (OPCF). In *Proceedings of the ECOOP 2000 Symposium on Objects and Databases*, Sophia Antipolis, France (LNCS 1944, pp. 86-100). Berlin: Springer-Verlag.
- Lee, S., Kim, S., & Kim, W. (2000). The BORD benchmark for object-relational databases. In *Proceedings of the 11th International Conference on Database and Expert Systems Applications (DEXA 00)*, London (LNCS 1873, pp. 6-20). Berlin: Springer-Verlag.
- Lu, H., Yu, J. X., Wang, G., Zheng, S., Jiang, H., Yu, G., & Zhou, A. (2005). What makes the differences: Benchmarking XML database implementations. *ACM Transactions on Internet Technology*, 5(1), 154-194.

- Mohan, C., Haderle, D., Lindsay, B., Pirahesh, H., & Schwarz, P. (1992). ARIES: A transaction recovery method supporting fine-granularity locking and partial roll-backs using write-ahead logging. *TODS*, 17(1), 94-162.
- Poess, M., Smith, B., Kollar, L., & Larson, P.-A. (2002, June 3-6). TPC-DS: Taking decision support benchmarking to the next level. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, Madison, WI (pp. 582-587). New York: ACM Press.
- Runapongsa, K., Patel, J. M., Jagadish, H. V., & Al-Khalifa, S. The Michigan benchmark: A microbenchmark for XML query processing systems. In *Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web, VLDB 2002 Workshop EEXTT*, Hong Kong, China (LNCS 2590).
- Schmidt, A., Waas, F., Kersten, M., Carey, M. J., Manolescu, I., & Busse, R. (2002, August 20-23). XMark: A benchmark for XML data management. In *Proceedings of the 28th International Conference on Very Large Databases (VLDB 02)*, Hong Kong, China (pp. 974-985). San Francisco: Morgan Kaufmann.
- Singhal, V., Kakkad, S. V., & Wilson, P. R. (1992, September 1-4). Texas: An efficient, portable persistent store. In *Proceedings of the 5th International Workshop on Persistent Object Systems (POS 92)*, San Miniato, Italy (pp. 11-33). Berlin: Springer.
- Tiway, A., Narasayya, V., & Levy, H. (1995, October 5-19). Evaluation of OO7 as a system and an application benchmark. In *Proceedings of the OOPSLA '95 Workshop on Object Database Behavior, Benchmarks and Performance*, Austin, TX. *SIGPLAN Notices* 30(10). New York: ACM Press.
- TPC. (2002). *TPC Benchmark W (Web Commerce), Specification version 1.8*. Transaction Processing Performance Council. San Francisco.
- TPC. (2003a). *TPC Benchmark H Standard, Specification revision 2.2.0*. Transaction Processing Council. San Francisco.
- TPC. (2003b). *TPC Benchmark R Standard, Specification revision 2.2.0*. Transaction Processing Performance Council. San Francisco.
- TPC. (2004). *TPC Benchmark App (Application Server), Specification version 1.0*. Transaction Processing Performance Council. San Francisco.
- TPC. (2005). *TPC Benchmark C Standard, Specification revision 5.4*. Transaction Processing Performance Council. San Francisco.
- Yao, B. B., Ozsu, M. T., & Khandelwal, N. (2004). XBenck benchmark and performance testing of XML DBMSs. In *Proceedings of the 20th International Conference on Data Engineering (ICDE 04)*, Boston (pp. 621-633).

Chapter XVII

MAMDAS: A Mobile Agent-Based Secure Mobile Data Access System Framework

Yu Jiao, Pennsylvania State University, USA

Ali R. Hurson, Pennsylvania State University, USA

ABSTRACT

Creating a global information-sharing environment in the presence of autonomy and heterogeneity of data sources is a difficult task. When adding mobility and wireless media to this mix, the constraints on bandwidth, connectivity, and resources worsen the problem. Our past research in global information-sharing systems resulted in the design, implementation, and prototype of a search engine, the summary-schemas model, which supports imprecise global accesses to the data sources while preserving local autonomy. We extended the scope of our search engine by incorporating mobile agent technology to alleviate many problems associated with wireless communication. We designed and prototyped a mobile agent-based secure mobile data access system (MAMDAS) framework for information retrieval in large, distributed, and heterogeneous databases. In order to address the mounting concerns for information security, we also proposed a security architecture for MAMDAS. As shown by our experimental study, MAMDAS demonstrates good performance, scalability, portability, and robustness.

INTRODUCTION

Database systems play important roles in information storing and sharing. They are widely used in business, military, and research fields. However, since they have been developed, evolved, and applied in isolation over a relatively long period of time, the inevitable heterogeneity and autonomy become unavoidable characteristics of any information-sharing environment. Moreover, for many practical and performance purposes, the creation of databases is usually close to the application domains. Consequently, information resources are distributed in nature. This distribution of information worsens the problem of global information sharing.

To overcome the obstacles brought by the local database heterogeneity, two possible solutions have been studied in the literature:

- Redesign the existing databases to form a homogeneous information-sharing system, or
- Develop a global system on top of the heterogeneous local databases to provide a uniform information access method (a multidatabase system).

The first solution is not economically feasible due to its high cost; hence, the second approach (multidatabases) is recognized as a more practical solution (Sheth & Larson, 1990). Within the scope of multidatabases, the summary-schemas model (SSM) proposed by Bright, Hurson, and Pakzad (1994) is a solution that utilizes a hierarchical meta-data in which a parent node maintains an abstract form of its children's data semantics, namely, a summary schema. The hierarchical structure and the automated schema abstraction significantly improve the robustness and provide dynamic expansion capability to the system. By using an online thesaurus, the SSM also supports imprecise queries.

As mobile communication technology advances and the cost and functionality of mobile devices improves, more and more users desire and sometimes demand anytime, anywhere access to information sources. The flexibility of such mobile data access systems (MDASs) comes at the expense of system complexity caused by technological limitations (i.e., low network bandwidth, unreliable connectivity, and limited resources).

The mobile agent-based distributed system design paradigm can alleviate some of these limitations. When mobile agents are introduced into the system, mobile users only need to maintain the network connectivity during the agent submission and retraction. Therefore, the use of mobile agents alleviates constraints such as connectivity, bandwidth, energy, and so forth.

We have designed and prototyped a novel MDAS framework, called MAMDAS — a mobile agent-based secure mobile data access system framework. This framework adopts SSM as its underlying multidatabase organization model. The design of MAMDAS intends to address two major issues: achieving high performance and supporting mobility. In this chapter, we focus on the performance issues. Studies addressing the second issue can be found in Jiao and Hurson (2004b).

The use of mobile agents alleviates many problems associated with mobile computing. However, it also has brought upon new challenges in ensuring information security. In order to address this problem, we propose a security architecture for MAMDAS that can protect the hosts, agents, and communication channels.

The rest of this chapter is organized as follows: the next section introduces the background, followed by a discussion of related work. The MAMDAS architecture and its implementation is then described in detail. The following section proposes a security architecture for MAMDAS. Then, we present the performance evaluation of MAMDAS. Last, we discuss future research trends and conclusions.

BACKGROUND

Multidatabase Systems

Database systems serve critical functions in government projects, business applications, and academic research. In many cases, existing geographically distributed, autonomous, and heterogeneous data sources must share information and perform conjoint functions. Since designing and building a database requires a large capital and time investment, it is not practical to redesign and rebuild a homogenous system out of a collection of heterogeneous databases. As an alternative, it would be of interest to design a global system on top of the existing heterogeneous local databases and generate an impression of uniform access with reasonable cost. Such systems are often referred to as multidatabase systems (MDBSs) in the literature.

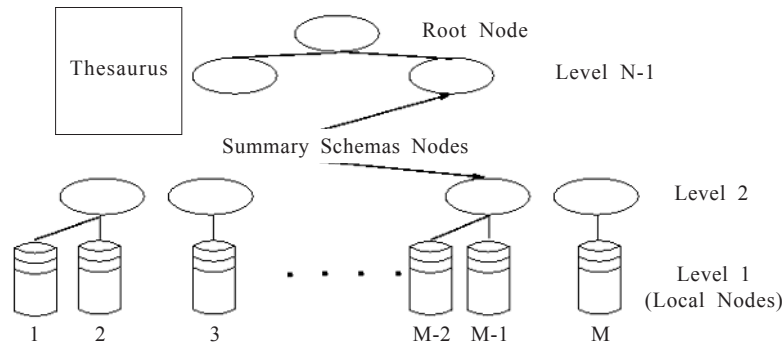
According to the taxonomy introduced by Sheth and Larson (1990), multidatabase systems can be further divided into federated and non-federated database systems (FDBSs). Due to the fact that non-federated database systems do not support local autonomy but federated database systems do, the latter is more favorable in practice. An FDBS consists of component databases that are autonomous and yet share information within the federation (Sheth & Larson, 1990). To overcome the local schema heterogeneity problem and support global transactions, FDBSs normally adopt the layered schema architecture — evolving from heterogeneous local-level data models to a uniform global-level data model. This global-level data model is called canonical or common data model (CDM). Two problems are often associated with the layered schema architecture: (1) schema redundancy exists between different layers, and (2) as the size of the FDBS grows, the size of global-level schema also increases; therefore, it becomes more difficult to automatically maintain and manipulate the global-level schema.

Based on who creates, maintains, and controls the federation, federated database solutions can be loosely or tightly coupled. It is the user's responsibility in the former, while it is the FDBS administrator's task in the latter. The summary-schemas model (SSM) (Bright et al., 1994) is, as reported in the literature, a tightly coupled FDBS that can address the two problems associated with the layered schema architecture.

The Summary-Schemas Model (SSM)

The SSM consists of three major components: a thesaurus, local nodes, and summary-schemas nodes. Figure 1 depicts the structure of the SSM. The thesaurus defines a set of standard terms that can be recognized by the system, namely, global terms, and the categories to which they belong. Each physical database (local nodes) may have its own dialect of those terms, called local terms. In order to share information among

Figure 1. A summary schemas model with M local nodes and N levels



databases that speak in different dialects, each physical database maintains a local-global schema meta-data that maps each local term into a global term in the format of “local term: global term.”. Global terms are related through synonym, hypernym, and hyponym links. The thesaurus uses a semantic-distance metric (SDM) to provide a quantitative measurement of “semantic similarity” between terms. The implementation detail of the thesaurus can be found in the work of Byrne and McCracken (1999).

The cylinders and the ovals in Figure 1 represent local nodes and summary-schemas nodes, respectively. A local node is a physical database containing real data. A summary-schemas node is a logical database that contains a meta-data called summary schema, which stores global terms and lists of locations where each global term can be found. The summary schema represents the schemas of the summary-schema node’s children in a more abstract manner — it contains the hypernyms of the input data. As a result, fewer terms are used to describe the information than the union of the terms in the input schemas.

The SSM is a tightly coupled FDBS solution and, therefore, the administrator is responsible for determining the logical structure of the SSM. In other words, when a node joins or leaves the system, the administrator is notified and changes to the SSM are made accordingly. Note that once the logical structure is determined, the schema population process is automated and does not require the administrator’s attention.

The SSM was simulated and its prototype was developed. The performance of the model was evaluated under various schema distributions, query complexity, and network topology (Bright et al., 1994). The major contributions of the SSM include preservation of the local autonomy, high expandability and scalability, short response time, and resolution of imprecise queries. Because of the unique advantages of the SSM, we chose it as our underlying multidatabase organization model.

The Mobile Agent Technology

An agent is a computer program that acts autonomously on behalf of a person or organization (Lange & Oshima, 1998). A *mobile agent* is an agent that can move through the heterogeneous network autonomously, migrate from host to host, and interact with

other agents (Gray, Kotz, Cybenko, & Rus, 2002). Agent-based distributed application design is gaining prevalence, not because it is an application-specific solution — any application can be realized as efficiently using a combination of traditional techniques. , but rather because it provides a single framework that allows a wide range of distributed applications to be implemented easily, efficiently, and robustly.

Mobile agents have many advantages. We only highlight some of them that motivated our choice. A quantitative simulation study of mobile agents' effect on reducing communication cost, improving query response time, and conserving energy can be found in Jiao and Hurson (2004a).

- **Support disconnected operation:** Mobile agents can roam the network and fulfill their tasks without the owner's intervention. Thus, the owner only needs to maintain the physical connection during submission and retraction of the agent. This asset makes mobile agents desirable in the mobile computing environment where intermittent network connection is often inevitable.
- **Balance workload:** By migrating from the mobile device to the core network, the agents can take full advantage of the high bandwidth of the wired portion of the network and the high computation capability of servers/workstations. This feature enables mobile devices that have limited resources to provide functions beyond their original capability.
- **Reduce network traffic:** Mobile agents' migration capability allows them to handle tasks locally instead of passing messages between the involved databases. Therefore, fewer messages are needed in accomplishing a task. Consequently, this reduces the chance of message losses and the overhead of retransmission.

One should note that the agent-based computation model also has some limitations. For instance, the overhead of mobile agent execution and migration can sometimes overshadow the performance gain obtained by reduced communication costs. In addition, the ability to move and execute code fragments at remote sites introduces serious security implications that cannot be addressed by existing technology.

Contemporary mobile agent system implementations fall into two main groups: Java-based and non-Java-based. We argue that Java-based agent systems are better in that the Java language's platform independent features make it ideal for distributed application design. We chose the IBM Aglet Workbench SDK 2.0 (<http://www.trl.ibm.co.jp/aglets/index.html>) as the MAMDAS' implementation tool. The IBM Aglet API provides high-level methods for sending and receiving messages. Low-level communication details are addressed by the mobile agent platform. The major advantage of the agent-based model, compared to the client/server-based model, is not the superiority of its communication mechanism. Rather, it is beneficial because the agent mobility allows tasks to be accomplished by using a fewer number of messages, which in turn improves the system performance in a congested network environment.

RELATED WORK

Developing agent-based MDAS involves research in many different directions: multidatabase architecture design (Sheth & Larson, 1990), ontology definition, which

enables semantic interoperability among distributed data sources (Maedche, Motik, Stojanovic, Studer, & Volz, 2003; Ouksel & Sheth, 1999; Sheth & Meersman, 2002), mobile agent platform development (Gray et al., 2002), and global transaction management (Dunham, Helal, & Balakrishnan, 1997), and so on. In this chapter, we focus on exploring and evaluating the practicality and performance of mobile agents in global information retrieval applications.

Agent-Based Distributed Data Access Systems

A work closely related to our research is the InfoSleuth project (Bayardo et al., 1997). The InfoSleuth addresses dynamic information integration and sharing problems in an open environment — the Internet — by using the agent-oriented design and ontology technology. In this system, entities and functions are represented by agents, and the agents communicate using the Knowledge Query and Manipulation Language (KQML) standard. Specialized broker agents semantically match information needs with available resources by consulting the ontology definition. The user accesses the InfoSleuth via a Java Applet-enabled Web browser.

A technical report-searcher system was implemented by Brewington et al. (1999) using the D'Agents mobile agent system. The SMART information-retrieval engine (Buckley, 1985) was used to measure the textual similarity between documents and a Yellow Page directory in order to determine the location of the documents. Mobile agents are exploited to retrieve reports across multiple machines. In this system, a mobile agent stays on the user's mobile device and performs the task by making remote procedure call (RPC)-like calls if the connection between the user and the network is reliable. Otherwise, the mobile agent will migrate to the closest proxy and start searching from there. When a task involves a large amount of intermediate data, the agent sends out child agents to the source of the documents. In the converse situation, where the query requires only a few operations, the agent simply makes RPC-like calls. Brewington et al. concluded that mobile agent technology has the potential to be a single, general framework in distributed information-retrieval applications. They also pointed out that the significant overhead of inter-agent communication and migration cannot be ignored.

Papastavrou, Samaras, and Pitoura (2000) proposed the DBMA-aglet framework for World Wide Web distributed database access. The system uses mobile agents, between the client and the server machine, as a means of providing database connectivity, processing, and communication. The authors also proposed a DBMS-aglet multidatabase framework, which is an extension of the original DBMS-aglet framework. In this extended framework, a coordinator DBMS-aglet is responsible for creating and dispatching multiple DBMS-aglets to different data sources in parallel. Finally, the coordinator DBMS-aglet compiles the result and returns it to the client. The authors claimed that the DBMS-aglet framework and its extension allow the aglet to be portable, light, independent, autonomous, flexible, and robust.

Vlach, Lana, Marek, and Navara (2000) described a system called mobile fatabase agent system (MDBAS). The system intends to integrate heterogeneous databases under one virtual global database schema to transparently manage distributed execution. The MDBAS aims to preserve local autonomy and execute distributed transactions using the two-phase commit protocol. Based on the experiences gained in the development of

MDBAS, the authors claimed that mobile agent technology will play an important role in the software industry in a short time.

Agent-based information retrieval engine design has also received attention from the business sector. Das, Shuster, and Wu (2002) of the Charles River Analytics Inc. prototyped such an information retrieval engine, called ACQUIRE, using a Java-based mobile agent platform. It aims to handle complex queries for large, heterogeneous, and distributed data sources. ACQUIRE translates each user query into a set of sub-queries by employing traditional query planning and optimization techniques. A mobile agent is spawned for each of these sub-queries. Mobile agents carry data-processing code with them to the remote site and perform local execution. Finally, when all mobile agents have returned, ACQUIRE filters and merges the retrieved data and presents the result to the user. Experiments have shown that ACQUIRE can effectively decompose and retrieve data from distributed databases. The system is easy to use and fast in query retrieval times.

Zhang, Croft, Levine, and Lesser (2004) proposed a multi-agent approach for purely decentralized P2P information retrieval. In this system, each database is associated with an intelligent agent that is cooperating with other agents in the distributed search process. The agent society is connected through an agent-view structure maintained by each agent. The agent-view structure contains the content-location information and is analogous to the routing table of a network router. The agent-view structures are initially formed randomly, and they dynamically evolve using the agent-view reorganization algorithm (AVRA) in order to improve search efficiency. The author observed that the system can achieve good performance when appropriate organizational structures and context-sensitive distributed search algorithms are deployed.

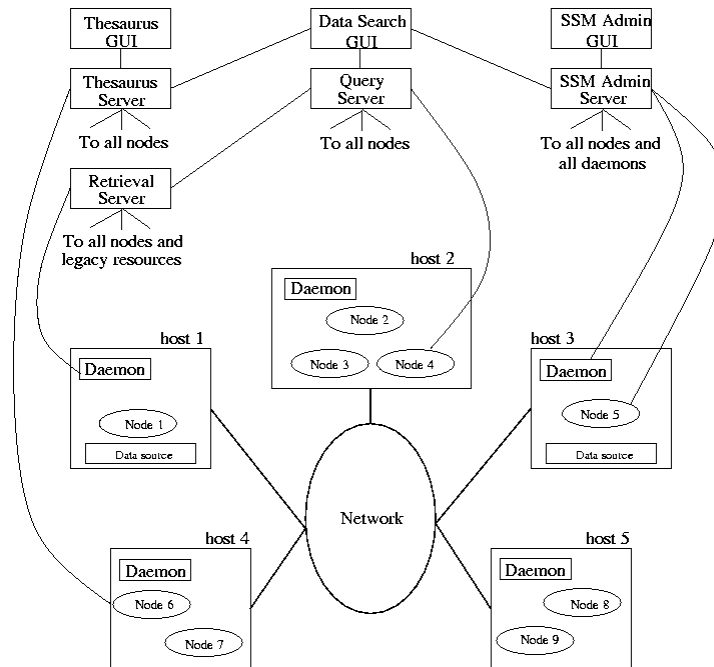
The above research projects have proven the practicality of agent-based distributed database access system design. However, these projects either did not investigate the multidatabase architecture or adopted the global schema approach and, consequently, will suffer from the two problems associated with it.

We proposed and prototyped a secure mobile multidatabase access system called MAMDAS, which stands for mobile agent-based secure mobile data access system framework. It takes full advantage of the SSM and the mobile agent-based computation paradigm. We expect that by adopting the SSM as the underlying multidatabase platform, the MAMDAS framework will satisfy the requirements of large-scale multidatabase systems, such as preserving local autonomy, achieving high performance, and providing good scalability and expandability. We also anticipate that the mobile agent technology will allow MAMDAS to provide better support for mobile users when dealing with intermittent network connectivity and congested networks.

The Information Broker System: A Client/Server-Based SSM Prototype

The information broker (IB) is an SSM prototype (Byrne & McCracken., 1999) based on the conventional client/server computation model. The system consists of four servers: a Thesaurus server, a SSM Administration server, a Retrieval server, and a Query

Figure 2. An overview of the Information Broker system architecture



server. Each server has a graphical user interface (GUI) that eases the user's interaction with the server. Local nodes and summary-schemas nodes run on a set of hosts connected through a network. The administrator can start and stop a node by sending commands to the Daemon program residing on each host and construct the summary-schemas hierarchy through the SSM Admin GUI. Figure 2 illustrates the architecture of the IB system.

Users submit queries through the data search GUI. To form a query, the user needs to supply the following information: the category preference (categorizing terms into different fields helps to narrow down the search scope), the node to start the search, the keyword, and a preferred semantic distance (loose match or close match). After accepting a query, the Query server starts searching the SSM hierarchy from the user-chosen node and performs the search over the multidatabase. Figure 3 presents the search algorithm performed at each node.

When presenting the results, the Query server displays all the terms that satisfy the user's preferred semantic distance. The IB system has proven that the SSM is a practical multidatabase solution. However, its operation relies on network connectivity, and its performance degrades significantly when the network is congested.

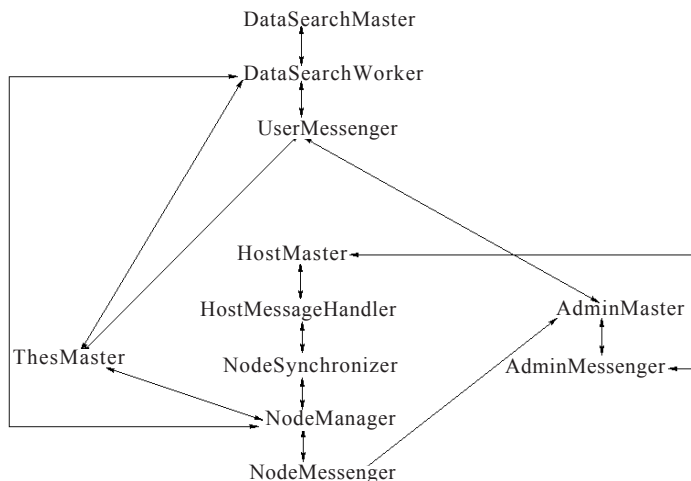
Figure 3. The search algorithm of the enhanced Information Broker system

```

1  Set all child-nodes to be unmarked;
2  WHILE (there exist unexamined terms)
3    IF (term is of interest)
      Mark all the child nodes that contain this term;
    ELSE
      CONTINUE;
5  END IF
6  END WHILE
7  IF (no marked child-node)
8    Go to the parent node of the current node and repeat the search algorithm;
9  ELSE
10   Notify each marked child node to execute the search algorithm.
11  END IF

```

Figure 4. The acquaintance model of MAMDAS



MAMDAS

MAMDAS Design

We chose Gaia, a general agent-oriented analysis and design methodology as the MAMDAS design methodology (Wooldridge, Jennings, & Kinny, 2000). Tables 1 and 2 show the agent model and the service model of MAMDAS (Jiao & Hurson, 2002), respectively. Figure 4 captures the acquaintance relation among agents in MAMDAS, where the arrows represent the communication direction.

Table 1. The agent model of MAMDAS

Role Name (= Agent Name)	Agent Mobility	Agent Instance
HostMaster	Stationary	Occurs n times
NodeManager	Stationary	Occurs m times
NodeSynchronizer	Stationary	Occurs n times
HostMessageHandler	Stationary	Occurs one or more times
NodeMessenger	Mobile	Occurs one or more times
AdminMaster	Stationary	Occurs once
AdminMessenger	Mobile	Occurs n times
ThesMaster	Stationary	Occurs once
DataSearchMaster	Stationary	Occurs zero or more times
DataSearchWorker	Mobile	Occurs zero or more times
UserMessenger	Mobile	Occurs zero or more times

Table 2. The service model of MAMDAS

Service	Accept users queries
Inputs	Keyword, preferred semantic distance, category, starting node
Outputs	Query result
Pre-condition	The AdminMaster is ready, the ThesMaster is ready, and the summary-schemas hierarchy is ready.
Post-condition	True

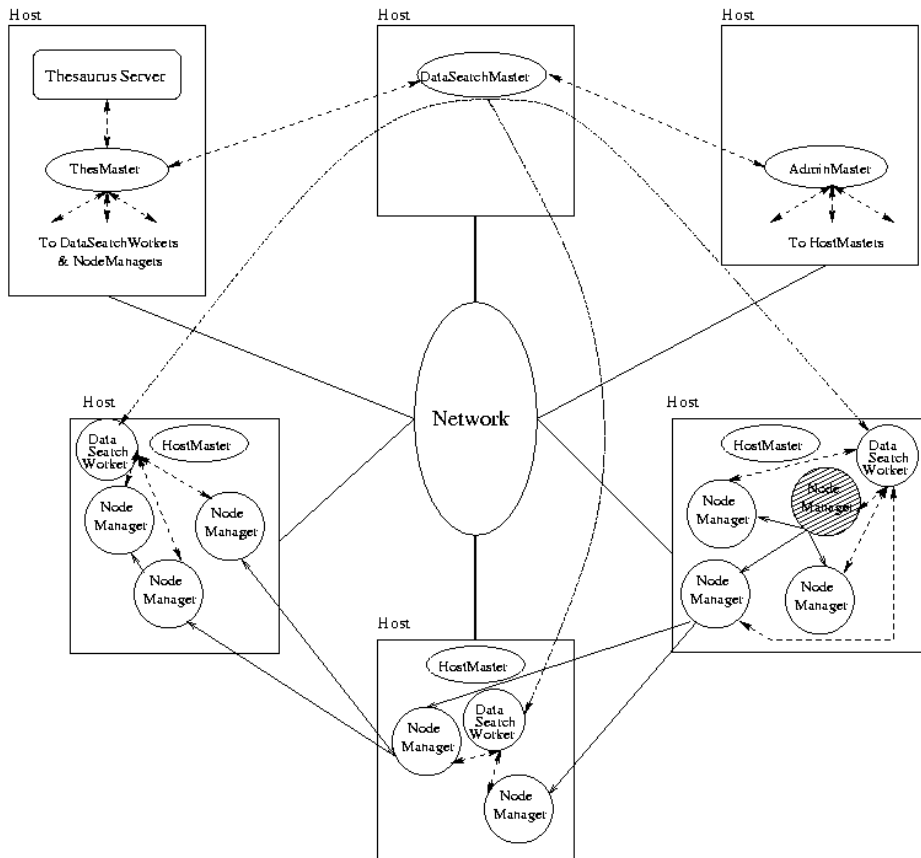
System Overview

The MAMDAS consists of four major logical components: the host, the administrator, the thesaurus, and the user. Figure 5 illustrates the overall architecture of the MAMDAS. To avoid complications, the figure shows only the most important agent types. Some secondary agents have been omitted.

The MAMDAS can accommodate an arbitrary number of hosts. A HostMaster agent resides on each host. A host can maintain any number and any type of nodes (local nodes or summary-schemas nodes) based on its resources. Each NodeManager agent monitors and manipulates a node. The HostMaster agent is in charge of all the NodeManager agents on that host. Nodes are logically organized into a summary-schemas hierarchy. The system administrators have full control over the structure of the hierarchy. They can construct the structure by using the graphical tools provided by the AdminMaster agent.

In Figure 5, the solid lines depict a possible summary-schemas hierarchy with the arrows indicating the hierarchical relation. The ThesMaster agent acts as a mediator between the thesaurus server and other agents. The dashed lines with arrows indicate the communication between the agents. The DataSearchMaster agent provides a query interface, the data search window, to the user. It generates a DataSearchWorker agent for each query. The three dashed-dot-dot lines depict the scenario in which three DataSearchWorker agents are dispatched to different hosts and work concurrently.

Figure 5. An overview of the MAMDAS system architecture



Once the administrator decides the summary-schemas hierarchy, commands will be sent out to each involved NodeManager agent to build the structure. NodeManagers at the lower levels export their schemas to their parents. Parent nodes contact the thesaurus and generate an abstract version of their children's schemas. When this process reaches the root, the MAMDAS is ready to accept queries.

The user can start querying by launching the DataSearchMaster on his/her own device, which can be a computer attached to the network or a mobile device. The DataSearchMaster sends out two UserMessengers (not shown in the figure) to the AdminMaster and the ThesMaster, respectively. The UserMessengers will return to the DataSearchMaster with the summary-schemas hierarchy and the category information. The DataSearchMaster then creates a data search window that shows the user the summary-schemas hierarchy and the tree structure of the category. The user can enter the keyword, specify the preferred semantic distance, choose a category, and select a node to start the search. After the user clicks on the "Submit" button, the DataSearchMaster packs the inputs, creates a DataSearchWorker, and passes the inputs to it as parameters.

Since the DataSearchMaster creates a DataSearchWorker to handle each query, the user can submit multiple queries concurrently.

Once dispatched, the DataSearchWorker can intelligently and independently accomplish the search task by making local decisions without the owner's interference. The search process can be described as follows:

- First, the DataSearchWorker contacts the NodeManager to obtain its schema and children and parent information.
- Second, the DataSearchWorker performs the search algorithm with the help of the ThesMaster. Note that this is the step that involves the most communication between agents.
- Third, if there is no resolution on the current node, based on the principle of the SSM, if the current node is the root, the DataSearchWorker will return to its home (where it is created) and display "no result." If the current node is not the root, the worker agent will recursively migrate to the current node's parent and conduct the same search algorithm until it reaches the root or finds a result. Another possibility is that the current node does indicate potential results. If the current node is a leaf-node, the DataSearchWorker will collect all the local terms that satisfy the semantic distance and go home to display the results. If the current node is a non-leaf-node, the DataSearchWorker will generate a clone for each node that may have results. To clarify the difference between the DataSearchWorker and its clones, we name the clone DataSearchSlave, even though they perform essentially the same functions. The cloning process will be invoked recursively till the slaves finally reach the leaf nodes. Slaves perform the search algorithm on their destinations in parallel. To reduce unnecessary network traffic, the slaves only report the results to its originator and then die on the local host.
- Fourth, when the final report reaches the DataSearchWorker, it knows that the task is done. It then returns home and displays the results. After the user clicks on the "ok" button or closes the result display window, the DataSearchWorker will dispose itself and release all the resources it occupies.

Two implementation choices need to be noted. First, we decided to program the data search algorithm into mobile agents instead of letting the nodes provide the search function for two reasons: to give users the freedom to tailor the search algorithm to fit specific needs and to reduce the maintenance work of the MAMDAS participants. Second, when multiple query resolutions are found, the mobile agent simply returns all the results. Functions such as data filtering and fusion, summary and statistics generation, and so on, can be easily added to mobile agents according to specific application requirements.

Optimizing the SSM Search Algorithm

According to the SSM search algorithm implemented in the IB system, when a DataSearchWorker searches a node, it must compare each global term in the node's schema with the keyword. If the node is a local node, the user-specified semantic distance is used as the criterion to determine whether the term is of interest. If the node is a summary-schemas node, other criteria depending on the implementation can be applied to determine whether a global term indicates potential resolution or not.

Several characteristics of the SSM have drawn our attention. Observe the following facts:

- The centralized thesaurus can only compare one pair of terms at a time.
- As we will see from the experimental results, the centralized thesaurus is actually the bottleneck of the whole system. Therefore, minimizing the number of term comparisons is the key to improving the system performance.
- When searching a local node, the DataSearchWorker must compare each global term in the node's local-global schema in order to obtain all local terms that satisfy the user-specified semantic distance.
- When searching a summary-schemas node, the DataSearchWorker can stop as soon as it finds that all the children of the current node contain potential resolution, because all the children must be searched regardless of the results of the remaining comparison.
- If the search on summary-schemas node *A* indicates that there is no resolution in this subtree, then the DataSearchWorker moves to *A*'s parent node, if a global term only exists on *A* (there is an entry which looks like "global term: <summary-schemas node *A*>"), this global term does not need to be checked. The reason is that we already know that there is no resolution on *A*.
- When the administrator organizes the summary-schemas hierarchy, naturally, he/she would prefer to connect nodes that contain similar contents to the same parent. Consequently, as we search down the tree, it is likely that all the children of a node have terms that are of interest.

Based on these observations, we claim that when searching a summary-schema node, there is an opportunity to reduce the number of comparisons of the SSM search algorithm.

- We represent the node's summary schema as a two-dimensional array with node names as row indices and global terms as column indices. If a global term's hyponym exists on a child node, the corresponding array element is set to 1; otherwise, it is set to 0. Table 3 shows an example of such an array.
- By re-organizing the terms, we move the columns that have more 1's to the left. In other words, we should examine the terms that exist on more child nodes first. Table 4 shows the re-organization of Table 3.

Table 3. The array representation of a summary schema

	Term ₁	Term ₂	Term ₃	Term ₄	Term ₅	Term ₆	Term ₇	Term ₈
Child ₁	1	0	1	0	1	1	0	1
Child ₂	0	1	0	1	0	0	0	1
Child ₃	1	0	0	1	1	0	1	0

Table 4. The array in Table 3 after re-organization

	Term ₁	Term ₄	Term ₅	Term ₈	Term ₂	Term ₃	Term ₆	Term ₇
Child ₁	1	0	1	1	0	1	1	0
Child ₂	0	1	0	1	1	0	0	0
Child ₃	1	1	1	0	0	0	0	1

Figure 6. Optimized search algorithm

```

1   Set all child-nodes to be unmarked;
2   WHILE (NOT (all term(s) are examined OR all child node(s) are marked))
    IF (term is of interest)
3       Mark all the child nodes that contain this term;
4       ELSE
5           CONTINUE;
6       END IF
7   END WHILE
8   IF (no marked child node)
    Go to the parent node of the current node and repeat the search
    algorithm (if a summary schema term of the parent node only exists on the
    current node, we can skip this term);
9   ELSE
10      Create a DataSearch Slave for each marked child node;
11      Dispatch the slaves to the destinations and repeat the search algorithm;
12  END IF

```

As a result, the search algorithm was modified as depicted in Figure 6. Assume that Term₁ and Term₄ in Table 4 indicate potential results in the subtree rooted at the current node. The DataSearchWorker only needs to make two comparisons before it proceeds to other nodes: “Term₁, keyword” and “Term₄, keyword.” In contrast, the search algorithm used in the IB and the enhanced IB systems will incur eight comparisons.

The network traffic reduction of the algorithm depends on many factors: the summary-schemas hierarchy, the thesaurus implementation, the query distribution, and so on; thus, a quantitative measurement of the reduction is difficult. However, one thing that is clear is that the worst-case performance of the optimized algorithm is the same as the original search algorithm used in the other two SSM prototypes: compare every summary schema’s global term with the keyword. As we will see later, the thesaurus contributes to as high as 80% of the total query-response time. Therefore, any reduction of communication involving the thesaurus will significantly improve the overall response time.

An Example of the Query Process

Initialization Phase

The ThesMaster and HostMaster agents are launched from the aglet server — Tahiti. The HostMaster will start all NodeManagers that are in charge of the nodes on this host and the AdminMaster agent on a separate host. The AdminMaster reads the summary-schemas tree configuration file and sends out commands containing the structural information to corresponding NodeManagers. The NodeManagers then start exporting schemas to their parents. The summary-schemas nodes will summarize the schemas exported by their children. When the schema population process reaches the root of the summary-schemas hierarchy, the MAMDAS is ready to accept queries. Figure 7 depicts the result of this summary-schemas hierarchy building process.

Search Phrase

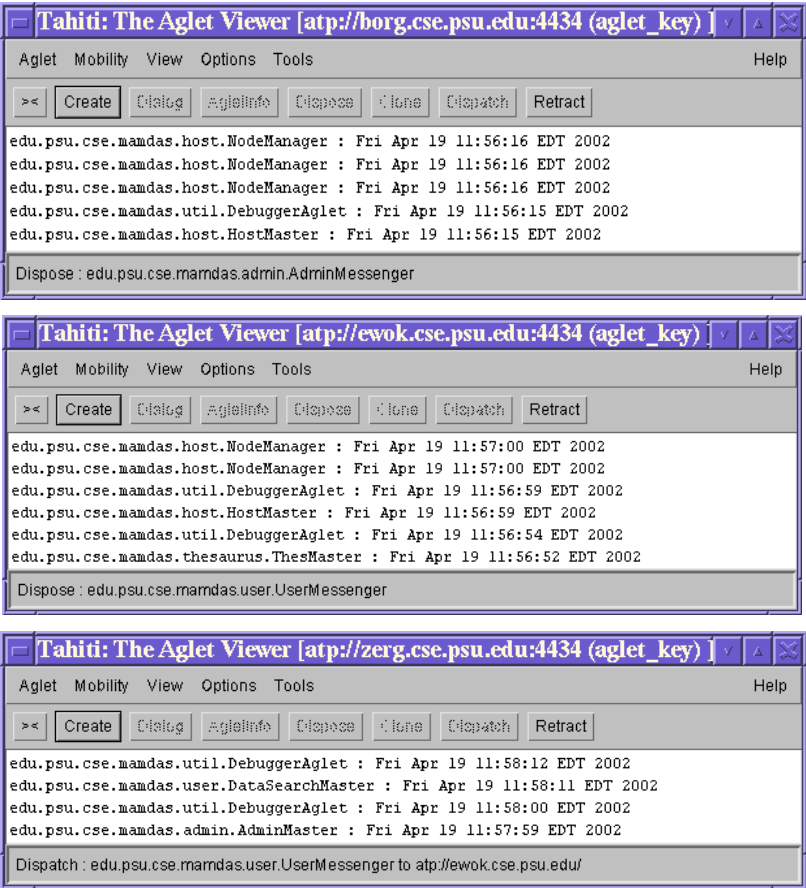
Users can start the DataSearchMaster on any computer (“zerg.cse.psu.edu” in this example). The DataSearchMaster will create the data search GUI on which the user can enter the keyword, choose the semantic distance, category, and a node to start the search. Figure 8 is a snapshot of the GUI. In this example, the summary-schemas hierarchy consists of five nodes: “borg bssn1,” “borg bssn2,” “borg bln2,” “ewok ewok1,” and “ewok ewok3” (we use “machine name + node name” to identify a node). These nodes form a summary-schemas hierarchy with the root “borg bssn1.” The user specifies the following information:

- The keyword is “damage” (actually, the term “damage” exists on both the node “ewok ewok1” and “ewok ewok3”, but it does not exist on “borg bln2”).
- The category is “heavy_industry.”
- Start search at node “borg bln2.”
- The preferred semantic distance is “0” (search for exact match).

Once the user clicks the “Submit” button, a DataSearchWorker mobile agent will be sent to the host “borg.cse.psu.edu.” The worker contacts the NodeManager of “bln2” and performs the search algorithm against the local-global schema of “bln2.” Since the term “damage” does not exist on the “bln2,” the DataSearchWorker tries to search the summary schema of its parent node — “bssn1,” which runs on the same host. The DataSearchWorker will find out that on one of the children of “borg bssn1,” “borg bssn2,” has potential result.

The DataSearchWorker then contacts node “borg bssn2” running on the same host “borg.cse.psu.edu.” The search result shows that both node “ewok ewok1” and node “ewok ewok3” may have terms that exactly match the keyword. The DataSearchWorker then clones two slaves (let us call them slave₁ and slave₂, respectively) and dispatches them to the host “ewok.cse.psu.edu” to search node “ewok1” and node “ewok3” in parallel. Slave₁ and slave₂ will find the local terms that have semantic distance 0 with the keyword “damage” on the two nodes they are searching. Slave₁ and slave₂ report to their originator DataSearchWorker about the resolution they found and dispose themselves locally. When the DataSearchWorker has obtained reports from all its slaves, it returns to the host where it was created — in this case the host is “zerg.cse.psu.edu” — and displays the result. The result display window is shown in Figure 9.

Figure 7. Building the summary-schemas hierarchy



THE SECURITY ARCHITECTURE OF MAMDAS

Security is one of the most difficult to achieve and most essential objectives in building global information sharing systems. A small flaw may render the entire system useless. In this section, we propose a security architecture for MAMDAS that can protect the hosts, the agents, and the communication channels. It constitutes two major components: IBM Aglets Workbench security extension and MAMDAS security polices. The IBM Aglets Workbench security extension includes security primitives and mechanisms that are essential for host, agent, and communication protection in any application built on this platform. MAMDAS security policies consist of certificate management and policies for host and agent protection.

Figure 8. The data search GUI

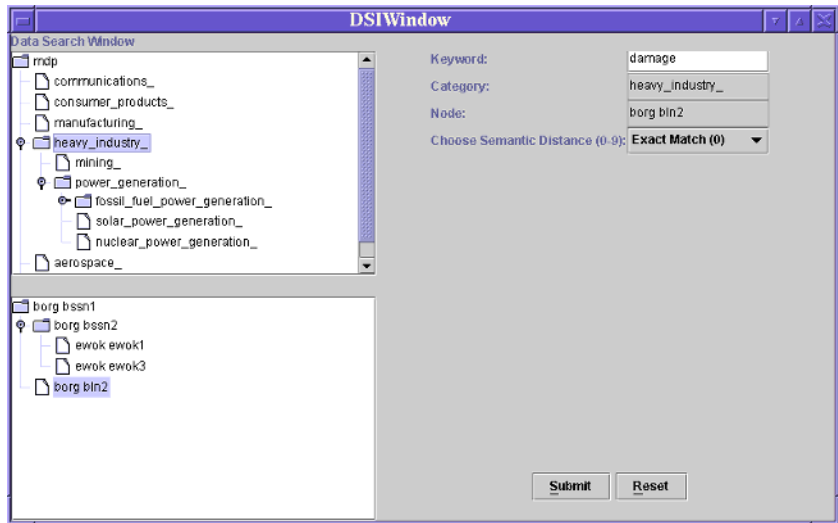
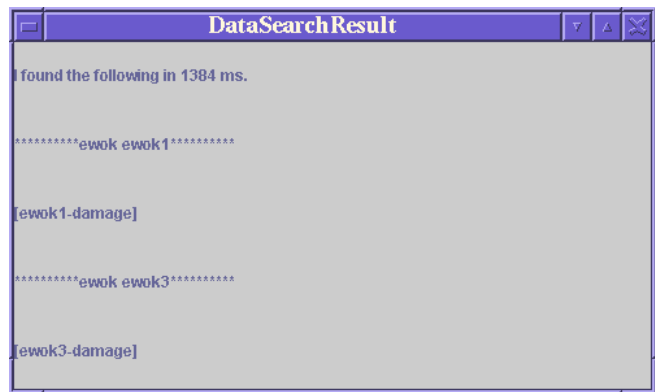


Figure 9. The result display window



IBM Aglets Workbench Security Extension

Key Management Mechanisms

The X.509 public key infrastructure (PKI) is a widely accepted industrial standard (RFC, 2002). Its well-defined structure and rigorous rules of certificate verification make it ideal for key management in corporate systems. However, it relies on a centralized certificate authority that everyone in the system trusts. In a global information system, which may have a scale as large as the Internet, it is often impossible to identify such a

central authority. The pretty good privacy (PGP) protocol (2004) addresses this problem by using a different approach for trust management: Trust is distributed and based on reputation. PGP is widely used over the Internet. Thus, the IBM Aglets Workbench security extension should provide both X.509 PKI and PGP services.

Secure Communication Channels

Following the suggestion made by many other agent security studies, we designate the task of securing communication channels to the secure socket layer (SSL) protocol (<http://wp.netscape.com/eng/ssl3>). One thing we would like to point out is that, originally, SSL was designed to ensure secure communication over the Internet. Typically, the server is authenticated by the client during the authentication phase, but not vice versa. For use in securing communication of mobile agent-based applications, the default mode of SSL should be set to mutual authentication: the client and server must authenticate each other before communication.

Authentication Mechanisms

As a preceding procedure to access control, symmetric and asymmetric-key cipher algorithms can be used to compose mutual authentication protocols for both the agent and host. Note that there is a difference between the mutual authentication in SSL and the one mentioned here: parties involved in a communication are typically agent platforms (sometimes referred to as agent servers or agent contexts). However, for access control purposes, the host must authenticate the agent in order to determine its access privileges, and the agent must authenticate the host in order to be sure that the host is a legitimate service provider.

Private Information-Retrieval Mechanisms

The Aglets Workbench security extension should include the private information retrieval (PIR) protocols (Chor, Goldreich, Kushilevitz, & Sudan, 1998). When multiple copies of the same data are available, the user can invoke these protocols to ensure the privacy of information retrieval. Note that existing PIR protocols are expensive: they typically involve high-communication complexity. An agent host may choose to conditionally provide this service, for example, according to a quality of service (QoS) agreement.

Auditing and Intrusion-Detection Mechanisms

Currently, the IBM Aglets Workbench does not provide any auditing functions. We propose to implement active online intrusion-detection mechanisms in its security extension. Each agent should submit a resource requirement to the host upon arrival. Once the request is granted, an intrusion detector (a stationary agent) will monitor the resource consumption of the requestor thereafter. The intrusion detector has the right to immediately terminate any misbehaving agents.

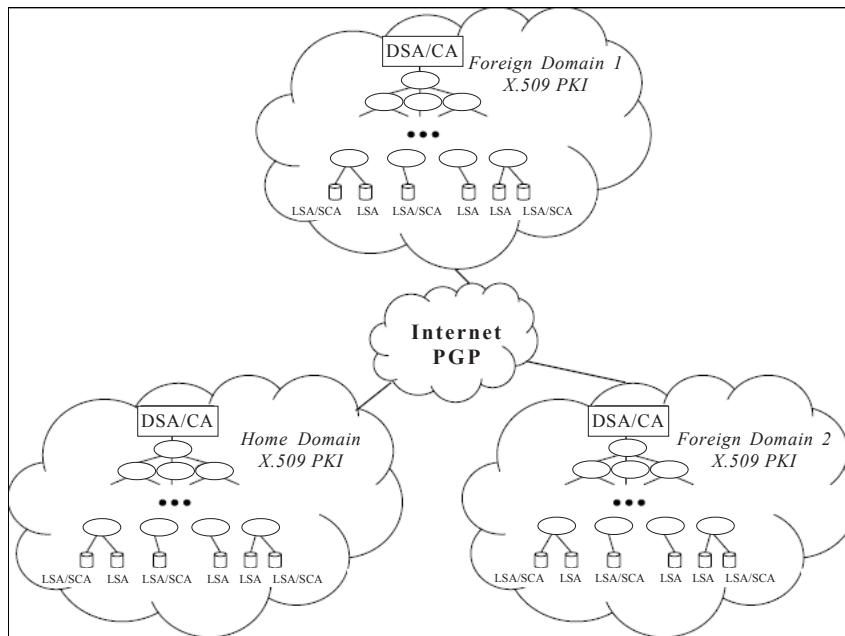
MAMDAS Security Policies

We define five principals in MAMDAS (Table 5). Here, we assume that when the aglet owner differs from the aglet manufacturer, it is the owner's responsibility to ensure the proper implementation of the aglet program. The same is also true for the aglet-context

Table 5. MAMDAS principals

Principal	Description
Aglet	Instantiation of an aglet program. Each aglet has a globally unique identifier.
Aglet Owner (AO)	Individual that launched the aglet. This is the principal that has legal responsibility for the aglet's behavior.
Context Owner (CO)	Individual or organization that maintains the agent context. This is the principal responsible for the context's behavior.
Local Security Authority (LSA)	Owner/administrator of the local host who establishes and enforces the local security policies.
Domain Security Authority (DSA)	Owner/administrator of the domain. It also acts as the certificate authority for the whole domain.

Figure 10. MAMDAS certificate management



owner and the aglet-context manufacturer. Thus, in our environment, we do not consider aglet and context manufacturers principals. A digital certificate is associated with and used to uniquely identify each principal. The next section elaborates the creation and management of digital certificates in MAMDAS.

Certificate Management

We view a MAMDAS shared by a particular user group (e.g., an organization) as the home domain of those users. MAMDAS built for different organizations can

collaborate to further improve information sharing. We refer to a MAMDAS that belongs to a different organization as the foreign domain, as opposed to the home domain. As a solution for global information sharing, MAMDAS must be able to handle security within its home domain as well as inter-domain security. We propose to use the X.509 standard for intra-domain key management and PGP for inter-domain key management.

Within the home domain, the domain security authority (DSA) acts also as the root certificate authority (CA) in X.509. The DSA/CA can appoint, some or all local security authorities (LSAs) to be the subordinate certificate authorities (SCAs). LSAs that are also SCAs can register and sign digital certificates on the root CA's behalf. As depicted in Figure 10, in MAMDAS, the DSA/CA is an independent host that is not associated with any particular agent context. Some or all of the local nodes can be designated as SCA. Users of the domain may register with either the DSA or an LSA in order to obtain a certificate.

When establishing trust between domains, however, it is difficult, if not impossible, to find a central authority on which both sides can rely. Therefore, we use PGP for inter-domain key/trust management. The purely distributed nature of PGP makes it ideal for large-scale information sharing where a central authority does not exist. If an agent wishes to contact a host in a foreign domain, it has two choices: (1) it can gain access by authenticating itself to the host using the PGP protocol in a peer-to-peer fashion, or (2) it can contact the DSA/LSA to obtain a temporary certificate and join the foreign X.509 infrastructure. With the first option, the agent needs to perform authentication on each host it visits, while, with the second option, it will be authenticated once when applying for the temporary certificate. This temporary certificate has very short lifetime and therefore, no revocation is needed.

Security Policies

As any other agent-based application, MAMDAS must protect hosts, agents, and communication channels. Since communication channels can be effectively protected using SSL with the mutual authentication setting, we will mainly focus on host and agent protection.

- **Host Protection:** We assume that when traveling within its home domain, a mobile aglet bares the same identity (certificate) and access rights as its owner. The mutual authentication between agent and host is done using the digital certificate. Within a domain, MAMDAS uses a hierarchical role-based access control method proposed by Ngamsuriyaroj, Hurson, and Keefe (2002) for host protection. This method maps local subjects to common roles defined at the global system level and tags access terms in the SSM hierarchy with a set of roles that are allowed to access those objects. The generalization of roles at the summary-schemas nodes leads to more relaxed access control. Any aglets that attempts to access objects beyond its privilege will be terminated immediately, because as the aglet moves down the SSM hierarchy, the access control rules are becoming more and more strict. If an access right cannot be granted at the high level, neither can it be granted at a lower level. Thus, by terminating the aglet as soon as possible, we reduce unnecessary resource consumption.

When an aglet travels to a foreign domain using its PGP certificate, the access control decision is made by the LSA of the host being visited. In order to achieve maximum protection of hosts against malicious foreign aglets, closed discretionary

access control policy is deployed by the LSA — an access is denied unless specified otherwise by the authorization rules.

When an aglet travels to a foreign domain using a temporary X.509 certificate issued by the DSA of that domain, LSAs will assume minimum access rights for this aglet. This means that only data/resources that are available to everyone can be accessed by this aglet.

To further protect the hosts, MAMDAS also applies limitation-based techniques and online intrusion detection. Before dispatching an aglet to MAMDAS, the aglet owner should compose a resource requirement (e.g., memory, CPU time, communication bandwidth, number of clones allowed, etc.) and send it along with the aglet. When an aglet arrives at a host, it must submit this request for resources to the LSA and get approval. The LSA monitors aglets running in its context and immediately terminates those that have exceeded their maximum allowable resource quota.

- **Agent Protection:** The first layer of protection for aglets is to ensure that only honest hosts are permitted to participate in MAMDAS. In MAMDAS, all contexts (participants) belong to the same domain, and they are registered with and authenticated by a trusted third party (DSA). As a complementary agent-protection scheme, the append-only data log can be used to detect tampering with data collected by the aglet. In addition, if an aglet owner requests private information retrieval, the PIR protocol will be invoked whenever multiple copies of required data are available.

EXPERIMENTS AND RESULT ANALYSIS

Experimental Environment

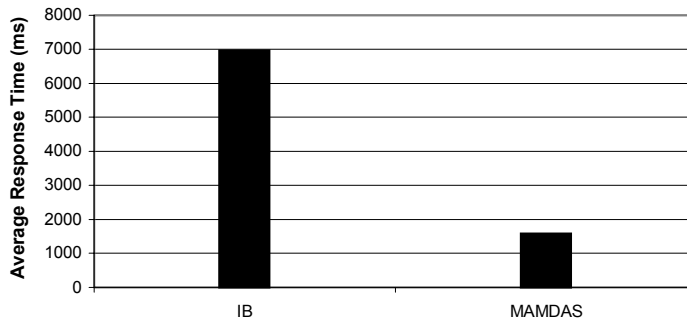
We performed most of our experiments on Sun Ultra 5 workstations running Solaris 8. The machines are connected through a fast Ethernet network that supports up to 100Mbps. Some of our experiments were carried out on PCs with various processors running different versions of the Windows operating system. We chose to conduct our experiments in a public computer lab when the machines were lightly loaded. We believe that this choice makes the experimental results more representative of typical system behavior in a realistic environment, where the machines are not dedicated to the database application and users' behavior is random. In general, the MAMDAS can be set up on any collection of machines that satisfy the following requirements:

- Each machine has a fixed IP address.
- All machines have J2SE installed (free software).
- All machines have IBM aglet Workbench SDK 2.0 installed (free software).

Average Response Time

We anticipate that the MAMDAS can improve the average response time because of the reduced communication cost, optimized search algorithm, and full exploitation of parallelism. Results (Figure 11) show that with the same SSM configuration and the same set of query, on average, MAMDAS is six times faster than IB.

Figure 11. Performance comparison of three SSM prototypes



Impact of the SSM Configuration

The query-response time depends strongly on the SSM configuration. Therefore, the organization of the summary-schemas hierarchy should be of interest to the global DBA. Intuitively, the global DBA may apply the following configuration strategies:

- **The semantic-aware configuration:** Clusters the local databases based on their semantic contents and assign semantically similar data sources to the same entry-level summary-schemas node.
- **Non-semantic-aware configuration:** Based on the physical connectivity of the network, assigns local data sources to the nearest entry-level summary-schemas nodes. The semantically similar data sources are distributed across the summary-schemas hierarchy.

The first strategy reduces contention at higher level summary-schemas nodes at the expense of creating a bottleneck at certain hot nodes in the network. The second approach distributes the workload among nodes and minimizes the communication distance between nodes on adjacent levels at the cost of longer search times at higher level nodes and possible longer search paths. It is a difficult task to form a well-balanced summary-schemas hierarchy and optimize the performance. The purpose of this experiment was to compare the effects of the two configurations and identify critical factors that affect the overall performance. The result can serve as a hint to help global DBA to make configuration decisions.

Semantic-Aware Configuration vs. Non-Semantic-Aware Configuration

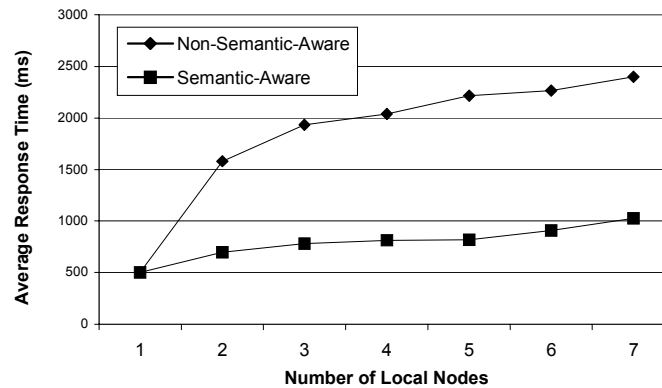
To clearly demonstrate the impact of the aforementioned strategies, we designed two extreme cases of the two configurations. The experiment was setup as follows:

- The total number of local nodes varied from 1 to 7. All local nodes have similar semantic content.
- By manipulating the local-global schemas, we ensured that the search result exists in all local nodes but one for each simulation run. Queries are always initiated at

Figure 12. The semantic-aware configuration and the non-semantic-aware configuration with 3 local nodes



Figure 13. Impact of SSM configurations



the node that does not contain the resolution. The purpose is to force the agent to travel in order to find resolutions. Different SSM configurations will result in different agent travel paths. Consequently, the average response time will be different.

- The semantic-aware configuration assigns all nodes to the same entry-level summary-schemas node because they all have similar semantic content.
- The non-semantic-aware configuration creates a new path starting at the root for each newly added local node.

Figure 12 illustrates structures of both configurations when the number of local nodes is 3. Note that when no resolution is found at the first node (we forced a search miss), in the semantic-aware configuration, the agent only needs to go up one level in order to find other possible solutions. In contrast, when the non-semantic-aware configuration is applied, the agent has to go all the way up to the root before it can find any other potential resolutions. After potential resolutions are identified, both configurations conduct searches in parallel. Intuitively, we anticipated that a shorter search path will demonstrate better performance. Figure 13 shows the experimental results. As

expected, the semantic-aware configuration outperforms the non-semantic-aware configuration. However, after a closer examination of this experimental result, we noticed performance degradation when the number of local nodes searched in parallel reaches 5 (the total number of local nodes is 6). This phenomenon raises a question: from the performance point of view, is it a good idea to build a wide summary-schemas hierarchy? In order to answer this question, we conducted the following experiment.

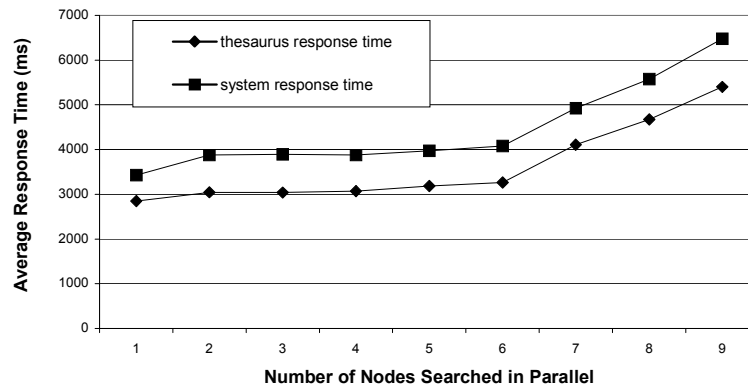
Scalability of Parallel Searches

From the optimized search algorithm, we can see that the system response time mainly consists of the thesaurus response time and agent creation and migration overhead. In order to identify the contribution of each of them, we designed an experiment to separate the thesaurus response time from the system response time. In this experiment, the semantic-aware configuration was applied and the number of nodes searched in parallel ranged from 1 to 9. We also set the result to be found on every local node. All queries in this experiment are submitted to the root.

Figure 14 shows the scalability of parallel searches. For configurations with local nodes less than 7, the average response time is almost the same, regardless of the increase in the number of local nodes. A sudden increase in the response time occurs when the number of local databases grows greater than 7. The thesaurus server makes the major contribution to this performance degradation. Although the thesaurus server supports multithreading, the number of concurrent clients it can support without performance degradation is still limited. When the number reaches a certain threshold (7 in this case), the server's performance degrades dramatically. Further analysis indicates that agent cloning introduces nearly a fixed amount of overhead when agent instances increase from 1 to 10. The reason is that most parts of the agent migration and execution time overlap. These results suggest that a fan out in the range of 3 to 5 of the summary-schemas hierarchy is suitable based on the present MAMDAS implementation. However, the choice of fan out is not universal. It must be calibrated for multidatabases of different sizes and local database characteristics. We recommend using a simulator to find out the most suitable fan out range. Figure 14 also implies that the optimization of the thesaurus server's performance is very important, since it contributes to almost 80% of the execution time.

Robustness and Portability Evaluation

As noted before, the IB system is vulnerable to message losses and exceptions. Thus, the system is not stable and it is difficult to debug. The MAMDAS is much more stable than the IB system for several reasons: the robustness of agents, the reduced communication, and good exception handling mechanism. During the course of our evaluation, we did not experience any crashes or stalls. Moreover, the MAMDAS demonstrates its robustness by handling intermittent network connections gracefully. When a DataSearchWorker fails to contact the owner upon finishing the task, it assumes that the owner is disconnected from the network. The mobile agent will then return to the node to which it is first submitted and wait. Because each agent has a universally unique identification number (ID), when the owner reconnects to the network, he/she can retract the agent from the node by using its ID. This supports our expectation that the agent-based computation model is superior to the client-server computation model when

Figure 14. The scalability of parallel searches

frequent disconnections exist. We intended to apply MAMDAS in a distributed environment and provide special services to mobile users. One challenge we must face is the heterogeneity of the machines. Thanks to the Java language's platform independent features, our system can be easily ported to any machine that supports the JVM version 1.3. We have successfully tested the system on PCs that run different versions of the Windows operating system without any modification.

FUTURE TRENDS

As the ubiquitous computing and wireless network research advances, computation and data sources are gradually moved to the background in the computing environment. In the foreground, more and more services are provided without having to be specifically asked. Mobile agent-based information retrieval systems naturally fit into this big picture: large, distributed, and heterogeneous multidatabase systems serve as the background knowledge base; users are represented by mobile intelligent agents that are capable of caring out various tasks; according to the user's profile and preset itinerary, many user needs can be anticipated and mobile agents are created automatically and launched to perform those tasks. Often, agents will take advantage of the databases for learning, information exchange, and collaboration purposes. For example, Chen, Perich, Chakraborty, Finin, and Joshi (2004) described an example of pervasive computing — a smart meeting room system that explores the use of agent technology, ontology, and logic reasoning. In this system, relevant service and information are provided to meeting participants based on their situational needs, i.e. all necessary equipments needed for the meeting can be reserved automatically and even the presentation slides can be preloaded right before the meeting. We believe that ubiquitous computing will prevail in the near future, and it is envisioned that the mobile agent-based information retrieval framework will become a dominant infrastructure choice for such applications.

CONCLUSION

Our study addressed the performance and security issues in multidatabase information retrieval while focusing on providing special support for mobile users. Employing the Gaia agent-based application design methodology (Wooldridge et al., 2000), we have successfully devised and implemented a new mobile data access system MAMDAS — a mobile agent-based secure mobile data access system framework. MAMDAS uses the SSM as its multidatabase organization model and the Java-based IBM aglet Workbench SDK 2.0 as its implementation tool.

The MAMDAS framework benefits from the assets of the SSM. It can effectively organize large-scale multidatabase systems and support imprecise queries. In addition, MAMDAS inherits the advantages of mobile agents, and thus can alleviate the limitations posed by wireless communication technology. In order to address the security concerns, we established a security architecture for MAMDAS that can protect the hosts, agents, and communication channels. One major advantage of this security architecture is that it separates the security mechanisms from the security policies, which allows the mechanisms provided by the agent platform to be shared by all applications built on top of it. Moreover, applications have the flexibility to decide security policies that suit their needs.

Our experimental results showed that MAMDAS significantly improved the average response time compared to the earlier SSM prototype. It is six times faster than the original prototype. The MAMDAS demonstrated great system scalability because of the employment of parallelism. Moreover, MAMDAS' platform-independent and security ensuring nature makes it an ideal choice for distributed information retrieval system design.

ACKNOWLEDGMENT

The Office of Naval Research under contract N00014-02-1-0282 and the National Science Foundation under contract IIS-0324835 in part have supported this work.

REFERENCES

- Bayardo, R.J., Bohrer, W., Brice, R., Cichocki, A., Fowler, J., Helal, A., et al. (1997). InfoSleuth: Agent-based semantic integration of information in open and dynamic environments. *ACM SIGMOD Record*, 26(2), 195-206.
- Brewington, B., Gray, R., Moizumi, K., Kotz, D., Cybenko, G., & Rus, D. (1999). Mobile agents in distributed information retrieval. In M. Klusch (Ed.), *Intelligent information agents* (pp. 355-395). Berlin: Springer-Verlag.
- Bright, M. W., Hurson, A. R., & Pakzad, S. H. (1994). Automated resolution of semantic heterogeneity in multidatabases. *ACM Transactions on Databases Systems*, 19(2), 212-253.
- Buckley, C. (1985). *Implementation of the SMART information retrieval system* (Tech. Rep. No. TR85-686). Ithaca, NY: Cornell University.

- Byrne, C., & McCracken S.A. (1999). An adaptive thesaurus employing semantic distance, relational inheritance and post-coordination for linguistic support of information search and retrieval. *Journal of Information Science*, 25(2), 113-131.
- Chen, H., Perich, F., Chakraborty, D., Finin, T., & Joshi, A. (2004, July 19-23). Intelligent agents meet semantic Web in a smart meeting room. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, New York (pp. 854-861). Washington, DC: IEEE Computer Society.
- Chor, B., Goldreich, O., Kushilevitz, E., & Sudan, M. (1998). Private information retrieval. *Journal of the ACM*, 45(6), 965-982.
- Das S., Shuster, K., & Wu, C. (2002, July 15-19). ACQUIRE: Agent-based complex query and information retrieval engine. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, Bologna, Italy (pp. 631-638). New York: ACM Press.
- Dunham, M. H., Helal, A., & Balakrishnan, S. (1997). A mobile transaction model that captures both the data and movement behavior. *Mobile Network Applications*, 2(2), 149-162.
- Gray, R. S., Kotz, D., Cybenko, G., & Rus, D. (2002). *Mobile agents: Motivations and state-of-the-art systems* (Tech. Rep. No. TR2000-365). Hanover, NH: Dartmouth College, Department of Computer Science.
- Jiao, Y., & Hurson, A. R. (2002, October 30-November 1). Mobile agents in mobile data access systems. In *Proceedings of the 10th International Conference on Cooperative Information Systems (COOPIS)*, Irvine, CA (pp. 144-162). Berlin: Springer-Verlag.
- Jiao, Y., & Hurson, A. R. (2004a, April 18-22). Mobile agents in mobile heterogeneous database environment — Performance and power consumption analysis. In *Proceedings of the Advanced Simulation Technologies Conference 2004 (ASTC'04)*, Arlington, VA (pp. 185-190). San Diego, CA: Society for Modeling and Simulation International.
- Jiao, Y., & Hurson, A. R. (2004b, March 29-31). Mobile agents and energy-efficient multidatabase design. In *Proceedings of the 18th International Conference on Advanced Information Networking and Applications (AINA'04)*, Fukuoka, Japan (pp. 255-260). Washington, DC: IEEE Computer Society.
- Lange, D., & Oshima, M. (1998). *Programming and developing Java mobile agents with aglets*. Reading, MA: Addison Wesley Longman.
- Maedche, A., Motik, B., Stojanovic, L., Studer, R., & Volz, R. (2003, May 20-24). An infrastructure for searching, reusing and evolving distributed ontologies. In *Proceedings of the ACM WWW2003*, Budapest, Hungary (pp. 439-448). New York: ACM Press.
- Ngamsuriyaroj, S., Hurson, A. R., & Keefe, T. F. (2002, July 17-19). Authorization model for summary schemas model. In *Proceedings of the International Database Engineering and Applications Symposium (IDEAS'02)*, Edmonton, Canada (pp. 182-191). Washington, DC: IEEE Computer Society.
- Ouksel, A. M., & Sheth, A. (1999). Semantic interoperability in global information systems: A brief introduction to the research area and the special section. *ACM SIGMOD Record*, 28(1), 5-12.

- Papastavrou, S., Samaras, G., & Pitoura, E. (2000). Mobile agents for World Wide Web distributed database access. *Transaction on Knowledge and Data Engineering*, 12(5), 802-820.
- PGP Corporation. (2004). *An introduction to cryptography*. Retrieved December 10, 2004, from <http://www.pgp.com>
- RFC-3280. (2002). *Internet X.509 public key infrastructure certificate and certificate revocation list profile*. Retrieved December 10, 2004, from <http://www.ietf.org/rfc/rfc3280.txt>
- Sheth, A., & Larson, J (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3), 183-236.
- Sheth, A., & Meersman, R. (2002). Amicalola report: Database and information systems research challenges and opportunities in semantic web and enterprises. *ACM SIGMOD Record*, 31(4), 98-106.
- Vlach, R., Lana, J., Marek, J., & Navara, D. (2000, December 2). MDBAS — A prototype of a multidatabase management system based on mobile agents. In *Proceedings of the 27th Annual Conference on Current Trends in Theory and Practice of Informatics (SOFEM'00)*, Milovy, Czech Republic (pp. 440-449). Berlin: Springer-Verlag.
- Wooldridge W., Jennings, N. R., & Kinny D. (2000). The Gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3), 285-312.
- Zhang, H., Croft, W.B., Levine, B., & Lesser, V. (2004, July 19-23). A multi-agent approach for peer-to-peer-based information retrieval systems. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, New York (pp. 456-463). Washington, DC: IEEE Computer Society.

Chapter XVIII

Indexing Regional Objects in High-Dimensional Spaces

Byunggu Yu, University of Wyoming, USA

Ratko Orlandic, University of Illinois at Springfield, USA

ABSTRACT

Many spatial access methods, such as the R-tree, have been designed to support spatial search operators (e.g., overlap, containment, and enclosure) over both points and regional objects in multi-dimensional spaces. Unfortunately, contemporary spatial access methods are limited by many problems that significantly degrade the query performance in high-dimensional spaces. This chapter reviews the problems of contemporary spatial access methods in spaces with many dimensions and presents an efficient approach to building advanced spatial access methods that effectively attack these problems. It also discusses the importance of high-dimensional spatial access methods for the emerging database applications, such as location-based services.

INTRODUCTION

There is a large body of literature on accessing data in high-dimensional spaces: Berchtold, Bohm, and Kriegel (1998); Berchtold, Keim, and Kriegel (1996), Lin, Jagadish, and Faloutsos (1995), Orlandic and Yu (2002), Sakurai, Yoshikawa, Uemura, and Kojima (2000), Weber, Schek, and Blott (1998), and White and Jain (1996). However, the proposed techniques almost always assume data sets representing points in the space. In many applications, effective representation of extended (regional) data is also important.

Regional data are usually associated with low-dimensional spaces of geographic applications. However, due to approximation, aggregation or clustering, such data may naturally appear in high-dimensional spaces as well.

For example, when the massive high-dimensional data of advanced scientific applications are clustered in files on tertiary storage, storage considerations often prevent the corresponding access structure from keeping the descriptors of all items in the repository. Instead, the content of each file can be approximated in the access structure by the minimal bounding rectangle (MBR) enclosing all data points in the given file (Orlandic, 2003).

Similarly, in order to reduce the cost of dynamic updates, multi-dimensional databases of location-based services frequently approximate the position of a moving object by the bounded rectangle of a larger area in which the object currently resides. Since the position is usually only one of many relevant parameters describing a moving object, the index (access) structure appropriate for these environments must deal with regional data in spaces with possibly many dimensions. The storage and retrieval of regional data representing moving objects are discussed later in this chapter.

Other applications in which regional objects naturally appear in high-dimensional spaces include multimedia and image-recognition systems. In these applications, objects are usually mapped onto long d -dimensional feature vectors. For the purposes of recognition, the feature vectors are projected onto a “reduced space” defined by $c \leq d$ principal components of the data (Swets & Weng, 1996). After populating the reduced space, images are grouped into classes, each of which can be represented by its approximate region and stored in a spatial access method. In order to identify the most likely class for the given object, the image recognition system must employ a form of spatial retrieval with a probabilistic ranking of the retrieved objects.

Unlike point access methods (PAMs), spatial access methods (SAMs) are designed to support different search operators (e.g., overlap, containment, and enclosure) over both points and regional objects in multi-dimensional spaces (Gaede & Gunther, 1998). Unfortunately, contemporary SAMs are limited by many problems, including some conceptual flaws that have a tendency to accelerate as dimensionality increases. The problems significantly degrade query performance in high-dimensional spaces.

This chapter reviews the problems of contemporary SAMs and presents an efficient approach to building advanced SAM techniques that effectively attack the limitations of traditional spatial access methods in spaces with many dimensions. The approach is based on three complementary measures. Through a special kind of object transformation, the first measure addresses the conceptual flaws of previous SAMs. The second measure reduces the number of false drops into index pages that contain no object satisfying the query. The third measure addresses a structural degradation of the underlying index.

The resulting technique, called the *cQSF-tree*, is not the ultimate achievement in the area of indexing regional data in high-dimensional spaces. However, it effectively attacks the limitations of traditional SAMs in spaces with many dimensions. The results of an extensive experimental study (presented later in this chapter) show that the performance improvements also increase with more skewed data distributions. In the experiments, the sQSF-tree (Yu, Orlandic, & Evens, 1999) and an optimized version of the R*-tree (Beckmann, Kriegel, Schneider, & Seeger, 1990; Papadias, Theodoridis, Sellis, & Egenhofer, 1995) are used as benchmarks for comparison.

The chapter concludes by emphasizing the importance of high-dimensional SAMs on the emerging spatiotemporal database applications with continuously changing multi-dimensional objects and by summarizing the results of this ongoing research.

BACKGROUND

To reduce the storage overhead of the index structure, extended regional objects are typically approximated by their MBRs, which tend to provide a good balance between accuracy and storage efficiency. There are many MBR-based SAMs, usually classified into region-overlapping (Beckmann et al., 1990; Guttman, 1984), object-clipping (Sellis, Roussopoulos, & Faloutsos, 1987) and object-transformation (Pagel, Six, & Toben, 1993) schemes.

Unfortunately, each group of traditional SAMs suffers from major conceptual problems that have a tendency to grow with data dimensionality (Orlandic & Yu, 2000). We call these problems conceptual because they tend to be associated with the very idea underlying a group of SAMs. For example, region overlap in R-trees (Guttman, 1984) and R*-trees (Beckmann et al., 1990) requires the traversal of many index paths, which increases the number of accessed nodes (index pages). The amount of overlap in these structures grows rapidly with data dimensionality (Berchtold et al., 1996). Object clipping (Sellis et al., 1987) creates multiple clips of a single regional object, which increases the size of the structure and degrades retrieval performance. Because the probability of clipping an object grows with dimensionality, these negative effects of clipping are more pronounced in higher dimensional spaces. A major drawback of object-transformation schemes (Pagel et al., 1993) is that a relatively small query window in the original space may map into a relatively large search region in the transformed space. The magnitude of this problem increases rapidly as the number of dimensions grows (Orlandic & Yu, 2000).

Few access methods for high-dimensional data can accommodate extended regional objects. X-trees (Berchtold et al., 1996) and simple QSF-trees, or just sQSF-trees (Orlandic & Yu, 2000; Yu et al., 1999), are exceptions. X-trees are designed to address the problem of region overlap in R*-trees. Instead of allowing splits that introduce high overlap, they extend index pages over the usual size. These clusters of pages, called super-nodes, are searched sequentially. Therefore, the advantages of the reduced overlap come at the expense of scanning the super-nodes and more complex dynamic updates.

By attacking the conceptual problems of traditional SAMs, sQSF-trees improve the performance of multi-dimensional queries in high-dimensional spaces. Unlike R-trees and R*-trees, which maintain hierarchies of possibly overlapping MBRs, they employ a simple modification of a PAM to avoid any region overlap. In contrast to traditional object transformations (Pagel et al., 1993), MBRs are not mapped to points in higher dimensional space. Instead, sQSF-trees apply an original query transformation that calculates search-and-filtering regions from the given query and uses these regions to search the index tree and to filter the result set. This is the origin of the name "Query-to-Search-and-Filter"-trees (QSF-trees). Prior experiments (Orlandic & Yu, 2000) have shown that sQSF-trees outperform an improved variant of R*-trees (Papadias et al., 1995) and an object-transformation scheme (Seeger & Kriegel, 1988).

While sQSF-trees eliminate certain conceptual problems of contemporary spatial access methods, they are not immune to the problems of high data dimensionality. As noted before, sQSF-trees index only the low endpoints of object MBRs. As a result, they may incur many false drops, especially in high-dimensional situations where the regions enclosing the high endpoints of object MBRs in leaf pages tend to be relatively small.

Moreover, since the structure of sQSF-trees is a simple modification of a point access method, it inherits all problems of the underlying PAM (Point Access Method). For example, when KDB-trees are used as the underlying PAM structure, the size of index entries increases proportionally to the dimensionality d . The growing storage overhead decreases retrieval performance (Orlandic and Yu, 2002). This is only one facet of a structural degradation of the underlying index as dimensionality grows.

CONCEPTUAL FLAWS

The research presented in this chapter evolved from three major observations:

1. By adopting suitable transformations, one can effectively attack the conceptual limitations of traditional SAMs in high-dimensional spaces.
2. By maintaining additional information in the interior levels of the index tree, many false drops can be eliminated.
3. The retrieval performance can be further improved by adopting a PAM structure that addresses the degradation of the index structure as dimensionality grows.

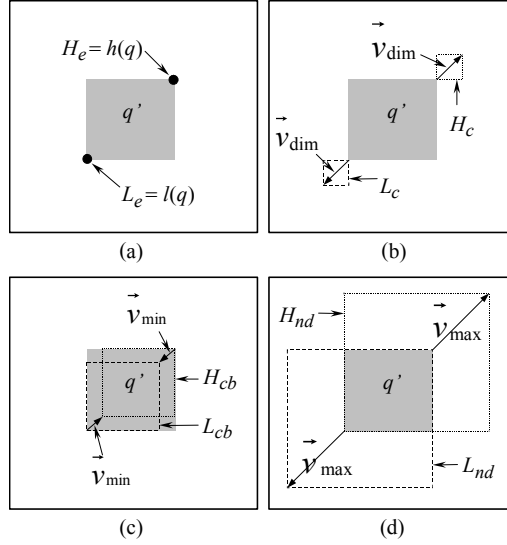
The design goals stem from the needs of advanced applications discussed in the introduction. In addition to improved retrieval performance, the goals include simplicity, portability, and faster updates.

Beginning with this section, we describe the incremental approach that leads to the modular design of cQSF-trees. The first milestone was the object transformation of the sQSF-tree (Yu et al., 1999). Like any object-transformation scheme, the sQSF-tree performs an explicit query transformation. However, while traditional object-transformation schemes (Seeger & Kriegel, 1988) map d -dimensional objects and queries onto their equivalents in a $2d$ -dimensional space, this query transformation takes place in the original d -dimensional space.

To describe the query transformation of sQSF-trees, we consider four *topological relations* (*query predicates*) between two MBRs r' and q' . The symbols $\subseteq$ and $\supseteq$ represent the *subset* and *superset* relations, respectively (e.g., $r' \supseteq q'$ means that every point of q' is also in the interior or on the boundary of r'). The relations are: *equal* ($r', q' \rightarrow r' = q'$); *covers* ($r', q' \rightarrow r' \supseteq q'$); *covered_by* ($r', q' \rightarrow r' \subseteq q'$); and *not_disjoint* ($r', q' \rightarrow (r' \cap q' \neq \emptyset)$).

We assume a square d -dimensional universe U and universal sets R and P of all rectangles and points in U , respectively. Next, we define two functions $l, h: R \rightarrow P$. For each d -dimensional rectangle $r' \in R$, these functions give its low endpoint $l(r')$ and high endpoint $h(r')$, respectively. Due to the geometry of rectangles, the low and high endpoints are the vertices of the given rectangle (the d -dimensional vectors) with the lowest (highest) coordinates along each dimension $i = 1, \dots, d$. The coordinates of the low and the high endpoint of r' along each axis i are denoted by $l_i(r')$ and $h_i(r')$, respectively.

Figure 1. Query transformation of sQSF-trees



The sQSF-tree represents each object MBR r' by a pair $\langle l(r'), h(r') \rangle$ of its endpoints in the original d -dimensional space. For each dimension i , the implementation dynamically keeps track of two values, m_i and M_i . Given an MBR r' , let $r'_i = h_i(r') - l_i(r')$ be the *length* of its side along the axis i . Then, m_i and M_i are respectively the minimum and maximum r'_i among all object MBRs r' in the data set.

The basic question behind the query transformation of sQSF-trees can be formulated as follows: where could the low and high endpoints of the object MBRs that satisfy the query predicate possibly lie in the space? To answer this, the transformation uses the notions of the *L-region* and *H-region*, which are defined as the portions of space containing the low (high) endpoints of all possible object MBRs that could satisfy the given query predicate (*equal*, *covers*, *covered_by*, or *not_disjoint*). The precise coordinates of the corresponding L- and H-regions for each type of query are defined in Yu et al. (1999).

Figure 1 illustrates the L- and H-regions generated for different topological relations with a query window q' . As in the rest of the chapter, the origin of the universe is assumed to be in the lower left corner of each figure. In the figure, v_{dim} is a d -dimensional vector whose component along each dimension i has magnitude $M_i - q'_i$, where q'_i is the length of the query window along this dimension. Similarly, v_{min} and v_{max} are d -dimensional vectors whose lengths along each dimension i are m_i and M_i , respectively.

For the relation *equal*, the regions L_e and H_e are just the low and high endpoints of the query window. For queries with the relation *covers*, the length of the regions L_c and H_c along the axis i is $M_i - q'_i$, unless it is truncated. For queries with the relation *covered_by*, the extents of the regions L_{cb} and H_{cb} along the same axis are $q'_i - m_i$. Finally, for the relation *not_disjoint*, the lengths of the i -th sides of the regions L_{nd} and H_{nd} , if they are not truncated, are $q'_i + M_i$.

Table 1. Search predicates of sQSF-trees

Query Predicate	Search Predicate of sQSF-trees
<i>equal</i> (r', q')	<i>covers</i> (R', L_e)
<i>covers</i> (r', q')	<i>not disjoint</i> (R', L_e)
<i>covered by</i> (r', q')	<i>not disjoint</i> (R', L_{cb})
<i>not disjoint</i> (r', q')	<i>not disjoint</i> (R', L_{nd})

The structure of sQSF-trees is a slightly modified PAM (Yu et al., 1999). When KDB-trees (Robinson, 1981) underlie the implementation of sQSF-trees, the insertion algorithm requires a simple modification to accommodate both low and high endpoints (l and h , respectively) of object MBRs at the leaf level of the tree structure. However, the leaf-level entries are indexed solely by the low endpoints. Since the space-partitioning strategy is that of KDB-trees when taking into account only the low endpoints of object MBRs, the format of interior entries is the same as in KDB-trees.

As in Orlandic and Yu (2000) and Yu et al. (1999), we assume here the splitting policy for interior pages based on the first-division plane (Freeston, 1995; Orlandic & Yu, 2001), which avoids downward propagation of splits associated with forced splitting of the original KDB-trees (Robinson, 1981). This enables greater storage utilization than in the original KDB-trees and an improved retrieval performance (Orlandic & Yu, 2001). As a forward reference, the space partition and the structure of sQSF-trees are illustrated in Figure 2a.

With the L- and H-regions, the original query is translated into the problem of finding MBRs whose low endpoints lie in the L-region and whose high endpoints lie in the H-region. While traversing the interior nodes (pages) of the index tree, the search operations rely solely on the L-region. Table 1 shows the search predicates applied to the interior entries. In the table, R' denotes the rectangular region representing an index page of the sQSF-tree at one level below. When searching a leaf page, the algorithm checks each object MBR whose low endpoint lies in the L-region to see whether its high endpoint falls in the H-region.

How does the transformation of sQSF-trees address the conceptual flaws of traditional SAMs? In its essence, this transformation translates the original problem of retrieving extended objects into a problem of finding relevant points in the space. Since the latter problem is resolved by employing a KDB-tree index structure that partitions the space into non-overlapping regions, the sQSF-tree automatically eliminates the possibility of region overlap. Since the structure indexes only the low endpoints of object MBRs, it avoids the need for object clipping. Moreover, since the query transformation takes place in the original space, the need to transform object MBRs into the points of a dual ($2d$ -dimensional) space is eliminated too. Finally, because the L- and H-regions are tuned to the semantics of individual queries, the sQSF-tree achieves a differentiation of search operations with different query predicates that can further reduce the number of accessed pages per average query.

FALSE DROPS

While sQSF-trees eliminate certain conceptual problems of contemporary spatial access methods, they are not immune to the problems associated with high data dimensionality. The expected number of page accesses per query, which is generally used as a measure of retrieval performance, is determined by the probability that an interior region and the given L-region overlap (are not-disjoint). Thus, even though sQSF-trees calculate both L- and H-region, only the L-region figures into the expected retrieval performance. This is because sQSF-trees index only the low endpoints of object MBRs. As a result, they can incur many false drops, especially in high-dimensional situations where the regions enclosing the high endpoints of object MBRs in leaf pages tend to be relatively small.

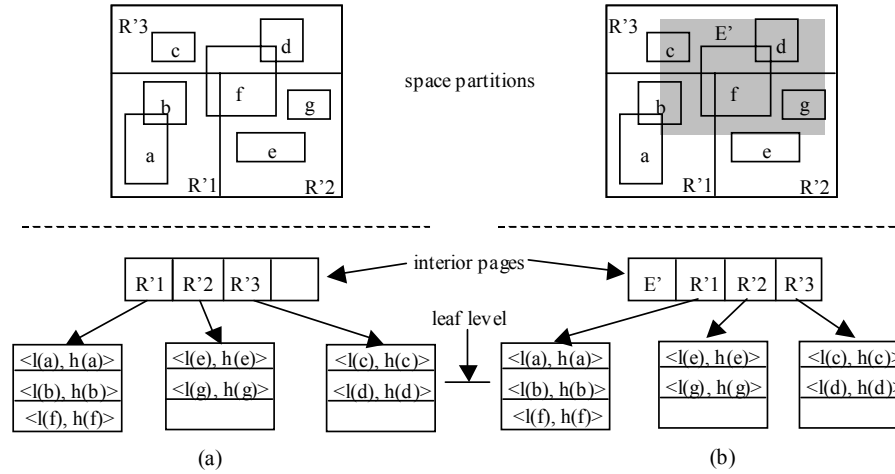
One way to propagate information about the high endpoints of object MBRs to the interior levels of the index tree is to extend each interior entry e_i in the index tree to include the MBR enclosing the high endpoints of all subordinate object MBRs in the tree (object MBRs appearing in the subtree rooted at e_i). Unfortunately, this would almost double the size of interior entries, reducing the capacity of interior pages by almost half. Several pilot experiments, conducted to observe the effects of this optimization on the retrieval performance, revealed that the improved selectivity of the search predicates does not compensate for the reduced capacity of interior pages.

However, a different heuristic optimization can lead to significant performance improvements. Instead of keeping the information about high endpoints of MBRs in every interior entry, one can assign to each interior page only one entry that would keep the information about the high endpoints of MBRs located in all sub-trees of the given page. In other words, for each interior page, the additional entry of the form $\langle l(E'), h(E') \rangle$, called the *H-entry*, would represent the minimum bounding hyper-rectangle E' enclosing the high endpoints of object MBRs that appear in any branch spawning from the given interior page.

The resulting structure is called the *scalable QSF-tree*, or just *cQSF-tree*. The addition of the H-entry can decrease the capacity of interior pages by at most one entry. (If the unused space that typically appears in index pages is sufficiently large, the page capacity need not be reduced.) On the other hand, since the leaf page structure is unchanged, its capacity remains the same as in the equivalent sQSF-tree.

Assuming that the QSF-tree variants are implemented using KDB-trees, Figure 2 illustrates the structures of the sQSF-tree and cQSF-tree built on the same set of object MBRs. The figure also shows the corresponding space partitions. In the figures, R'_1 , R'_2 , and R'_3 represent the index regions. Contrasting Figure 2a with 2b, one can see that the only structural difference is the appearance of the H-entries in the interior pages of cQSF-trees (in Figure 2, each structure has only one interior page). The H-entry in an interior page maintains the MBR E' enclosing the high endpoints of all object MBRs stored in the leaf pages within the sub-tree rooted at the given interior page. The shaded window in Figure 2b represents the region E' associated with the only H-entry in the given cQSF-tree structure.

To develop a cQSF-tree, the maintenance algorithms of sQSF-trees must be modified so that each H-entry is up-to-date. In particular, the *insertion* of an object MBR $r' = \langle l(r'), h(r') \rangle$ proceeds as follows:

Figure 2. The space partition and structure of (a) *sQSF*-trees and (b) *cQSF*-trees

1. **Search:** Starting from the root page and using only $l(r')$, search the underlying KDB-tree with the point-search algorithm of Robinson (1981) in order to locate the leaf page where r' belongs.
2. **Insertion:** Insert the object MBR r' into the leaf page. For each dimension i , if necessary, update the minimum m_i and/or maximum M_i extension along the axis i among all object MBRs in the data set.
3. **Updating H-entries:** If the new entry enlarges the MBR E' corresponding to the H-entry of the parent page, the H-entry needs to be updated. These updates may propagate upwards, up to the root of the *cQSF*-tree. (Note that, if node splitting occurs, the updates of H-entries can be propagated upwards along with the splitting of index regions.)
4. **Splitting leaf pages:** If the leaf page overfills, perform the split operation according to the splitting algorithm for leaf pages given in Robinson (1981), taking into account only the low endpoints of object MBRs. Split the index region of the old leaf in its parent page.
5. **Splitting interior pages:** If an interior page overfills, perform the split operation according to the rules of first-division splitting (Orlandic & Yu, 2001). Split the index region of the old interior page in its parent page, if any. The splitting may propagate up to the root, in which case a new root page is created and the number of levels in the index tree is incremented by one.

The approximate information about the high endpoints of object MBRs maintained in the H-entries can be used effectively to improve the performance of search operations. Table 2 shows the search predicates applied to the interior entries of the *cQSF*-tree. As before, R' denotes an index region stored in the given interior entry and E' represents the MBR corresponding to the H-entry of the given interior page. In contrast to the search predicates of *sQSF*-trees (Table 1), the predicates of Table 2 test whether the MBR E' overlaps the given H-region. The test can be performed once for each interior page.

Table 2. Search predicates of cQSF-trees

Query Predicate	Search Predicate
$equal(r', q')$	$covers(R', L_e) \wedge covers(E', H_e)$
$covers(r', q')$	$not\_disjoint(R', L_c) \wedge not\_disjoint(E', H_c)$
$covered\_by(r', q')$	$not\_disjoint(R', L_{cb}) \wedge not\_disjoint(E', H_{cb})$
$not\_disjoint(r', q')$	$not\_disjoint(R', L_{nd}) \wedge not\_disjoint(E', H_{nd})$

Given a query of the form “find all object MBRs r' that satisfy topological relation t with respect to the given query window q' ,” cQSF-trees perform the *search* operation as follows:

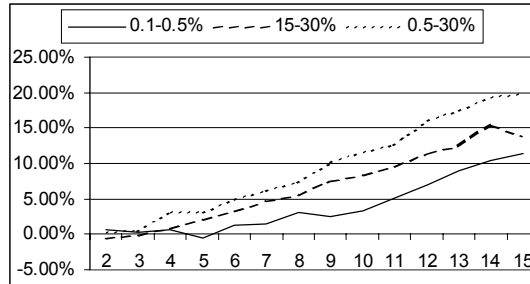
1. **Initialization:** For the given query window q' and the relation t of the query predicate, calculate the corresponding L-region and H-region (Yu et al., 1999), as illustrated in Figure 1.
2. **Search:** Starting from the root page, perform the breadth-first search of the underlying tree structure by applying the corresponding search predicate of the second column of Table 2 to the index entries and the H-entry of every accessed interior page in the cQSF-tree.
3. **Selection:** When a leaf page is accessed, include in the result set each object MBR r' that satisfies the topological relation t with respect to q' .

The main trust of cQSF-trees is that they reduce the false drops in sQSF-trees. Since the number of false drops tends to grow with dimensionality, the effects of this measure are likely to be more pronounced in high-dimensional spaces. More precisely, as the number of dimensions grows, the size of index entries increases, thus decreasing the effective capacity of the index pages. When the number of objects in the data set is constant and page capacity reduces, the MBRs E' of the H-entries enclose fewer points and they become progressively smaller than the universe. Due to smaller regions E' , fewer pages must be visited while traversing the tree. With more skewed data distributions, the MBRs E' are likely to be even smaller. As a result, the probability of accessing an index page is likely to be reduced, which enables greater performance improvements over the equivalent sQSF-trees.

However, the expected improvements of cQSF-trees over sQSF-trees are by no means guaranteed. They may depend on a number of different factors, including the dimensionality of data as well as their volume and distribution in space. To investigate the performance of the two variants of QSF-trees in various scenarios, we conducted several experiments with different test cases. Both versions of QSF-trees were implemented by modifying KDB-trees to accommodate extended entries at the leaf level of the tree and using the first-division splitting of interior pages (Orlandic & Yu, 2001).

In each experiment, the number of dimensions was varied between 2 and 15. The page size of every structure was 2K bytes. The retrieval performance was measured in terms of the average number of page accesses over 2,000 randomly generated queries (500 for each type of query). Every side of a query window was obtained as a pair of random

Figure 3. Percentage improvements for uniform data distribution as dimensionality grows



numbers between 0 and 1. The experiments were performed for both uniform and highly skewed data distributions.

The first set of experiments involved uniformly distributed data. For each d -dimensional space, we constructed three data files with small, large, or widely varying objects. Each file contained 65,536 (2^{16}) random d -dimensional rectangles whose lengths along each axis, relative to the side of the universe, were between: 0.1% and 0.5% (small objects), 15% and 30% (large objects), or 0.5% and 30% (widely varying objects). Objects of each file were inserted into an sQSF-tree and the equivalent cQSF-tree.

Figure 3 shows the percentage improvements of cQSF-trees over sQSF-trees for an average query with varying data dimensionality. For each input file in every d -dimensional space, the improvement was measured using the following formula: $100 \cdot (T_s(d) - T_c(d)) / T_s(d)$, where $T_s(d)$ and $T_c(d)$ are the total page accesses generated by all queries performed on sQSF-trees and the corresponding cQSF-trees, respectively. As expected, the improvements had a tendency to grow with data dimensionality.

As Figure 3 shows, the highest improvements were obtained for objects of widely varying size. To see the reason for this, observe first that the volumes of the L-regions L_{nd} , L_c , and L_e are the same for both widely varying and large objects (recall Figure 1). However, since the size of the L-regions L_{cb} for *covered_by* queries is greater for widely varying than for large objects, sQSF-trees generate more false drops for widely varying objects. As a result, the more restrictive search predicates of cQSF-trees have greater impact for objects with widely varying sizes than for large objects. The lowest percentage improvements were obtained for small objects. This can be explained by considering the *not_disjoint* queries, which generally dominate the average performance. For small objects, the enlargement of the search regions (the L-regions L_{nd}) is relatively small, and so few false drops are generated by the sQSF-tree. When this is the case, the more restrictive search predicates of cQSF-trees have relatively smaller impact on the retrieval performance.

To observe the performance of cQSF-trees and sQSF-trees for non-uniform data distributions, for each d -dimensional space, we constructed an input file with exactly 32,768 (2^{15}) synthesized objects of varying size, which were concentrated in two different clusters. While one of the clusters appeared close to the origin of the space, the other

Figure 4. A skewed distribution of data in a 2-dimensional space

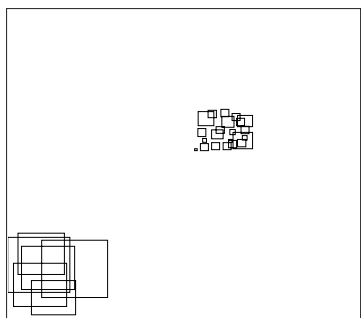
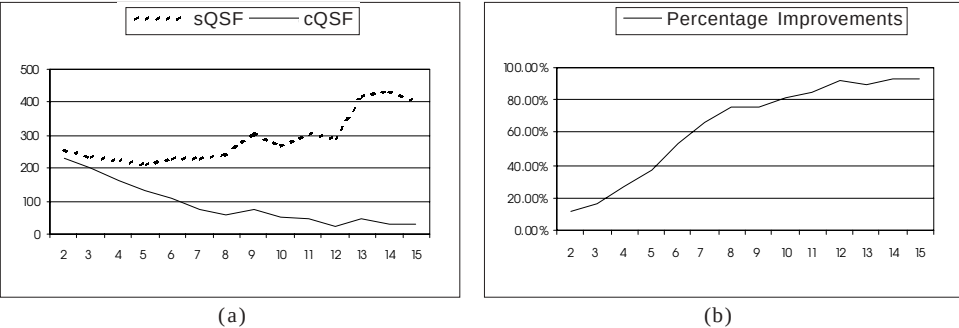


Figure 5. Relative performance for the skewed distribution of Figure 4 as data dimensionality grows



was placed near the center of the universe. Figure 4 illustrates the distribution in the 2-dimensional space. For each d -dimensional space, the objects of the corresponding data file were inserted into both sQSF-trees and cQSF-trees.

Figure 5 shows the difference in the retrieval performance as data dimensionality grows, which is expressed in terms of the average page accesses per query (Figure 5a) and the percentage improvement (Figure 5b). For skewed data distribution, the performance improvements of cQSF-trees over the sQSF-trees can be dramatic. In the 15-dimensional space, cQSF-trees generated about 13.5 times fewer page accesses than the corresponding sQSF-trees. This also confirms the anticipated impact, stated earlier in this section, of the optimization applied by cQSF-trees.

STRUCTURAL DEGRADATION

Since the structures of sQSF-trees and cQSF-trees are simple modifications of a point access method, they inherit all problems of the underlying PAM. Since we have

chosen KDB-trees as the underlying PAM structure, the size of index entries increases proportionally to the dimensionality d . This increases the number of pages that need to be accessed, which decreases retrieval performance.

Moreover, since the index regions at any given level of the KDB-tree completely cover the space, they can be much larger than the areas occupied by their enclosed points. As a result, they may incur a significant amount of dead (empty) space, which tends to increase as dimensionality grows. As a somewhat contrived example, consider an index region whose every side is twice as long as the corresponding side of the MBR enclosing all points in the index region. In a d -dimensional space, the index region is 2^d times larger than it needs to be. With the enlargement of the index region, the probability that the corresponding index page is accessed increases.

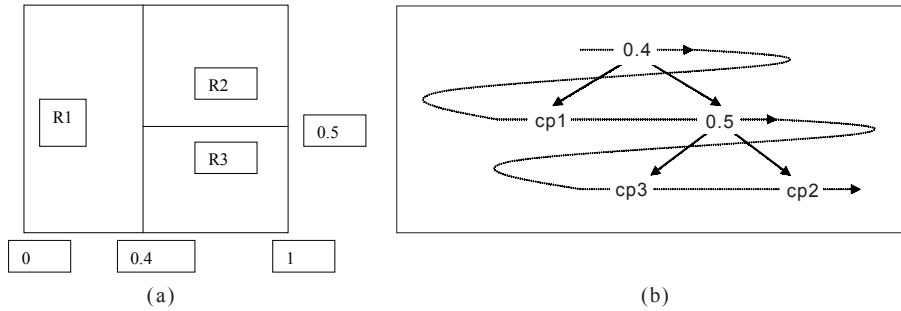
We use the term structural degradation of the index to refer to the enlargement of both index entries and regions as dimensionality grows. This section introduces a new PAM called the RM-tree (Reduced-Margin-tree), which effectively attacks this degradation without creating overlap between the index regions. Note that the margin of an index page that is associated with a d -dimensional hyper-rectangle R in the data space is defined as $2 \cdot \sum_{i=1}^d (h_i - l_i)$ where, for all $i=1, \dots, d$, h_i and l_i are the high- and low-endpoints of R along the dimension i .

Several point access methods (PAMs), such as the TV-tree (Lin et al., 1995), the Pyramid Technique (Berchtold et al., 1998), and the KDB_{HD}-tree (Orlandic & Yu, 2002), have been proposed for high-dimensional point data. For example, the approach of the TV-tree is based on the observation that in typical high-dimensional data sets, only a small number of dimensions carry most of the relevant information. The idea is to store in the interior pages only a small number of features that discriminate well between the point objects and ignore the rest of the dimensions. Since fewer features are stored in the interior pages, the interior levels of the index structure are more compact and the spatial searches are more efficient. However, prior to inserting any object into the structure, one must decide which dimensions are important and how many of these dimensions should be used. These factors have significant bearing on the performance of the TV-tree.

The Pyramid Technique partitions a d -dimensional universe into $2d$ pyramids that meet at the center of the universe. The d -dimensional vectors, each of which represents a multi-dimensional data point, are approximated by one-dimensional quantities, called pyramid values, which are indexed by a regular B⁺-tree. Due to the one-dimensional transformation, every pyramid is implicitly “sliced” into variable-size index regions parallel to the basis of the pyramid. However, the transformation results in a loss of spatial proximity as well as the enlargement of queries falling near the boundaries of the space.

In order to improve the retrieval performance in high-dimensional spaces, the KDB_{HD}-trees use two heuristic measures. One measure relates to the policy of node splitting; the other reduces the size of index entries. In a high-dimensional space, every index region in the KDB-tree is split along a small subset of dimensions. Since each remaining dimension of the region extends over the entire side of the universe, it contributes nothing to the selectivity of the structure (Orlandic & Yu, 2002). In the KDB_{HD}-tree, these remaining dimensions are eliminated from index-region descriptors. This enables a greater index compression.

Figure 6. (a) Three index regions and (b) binary-tree-like page structure



However, the child-region descriptors of the interior pages still contain redundant information. For example, Figure 6 shows three index regions. In the KDB_{HD} -tree, the index entry for the region R1 is $\langle 1, 0, 0.4, cp1 \rangle$; for the region R2, $\langle 2, 0.4, 0.5, 1, 1, cp2 \rangle$; and for the region R3, $\langle 2, 0.4, 0, 1, 0.5, cp3 \rangle$, where cp1, cp2, and cp3 are child-page pointers. In this example, the value of 0.4 is shared by all three index entries, and the value 0.5 is shared by two index entries. While the redundancies tend to be negligible in low-dimensional spaces, they can become significant in high-dimensional spaces.

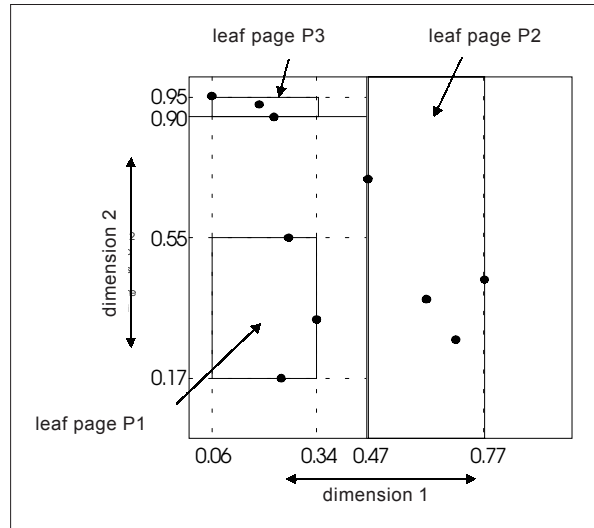
The RM-tree structure further reduces redundant information and produces tighter bounding index regions using an improved page-splitting policy. The removal of redundant information increases the capacity of the index pages, and tighter bounding index regions decrease the amount of dead space covered by the index regions.

The elimination of redundant information can be achieved by changing the structure of the interior page from a list of entries to a multi-dimensional binary search tree (or KD-tree) (Bentley, 1975). All entries in an interior page are represented by a single binary search tree. For example, Figure 6a can be represented by the binary tree shown in Figure 6b. To store this tree in a page, we do a recursive, pre-order traversal of the tree. For example, the tree in Figure 6b is stored as $[0.4|cp1|0.5|cp3|cp2]$. We modify the original KD-tree so that the leaf nodes (pointer nodes) point to child pages of the given index page and the interior nodes (division nodes) represent the split values that divide the index region of the given page into those of the child pages. In this example, the division-node values are 0.4 and 0.5, and the pointer-node values are cp1, cp2, and cp3.

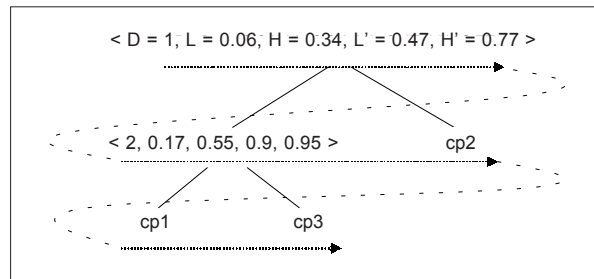
This page structure significantly reduces the size of the interior levels of the index, thus providing room for additional information in the interior levels. In the RM-tree, each division node (stored in an interior page) consists of five values: $\langle D, L, H, L', H' \rangle$, where D is the split dimension, L (respectively, H) is the smallest (respectively, largest) data coordinate in the left child region along the dimension D ; L' (respectively, H') is the smallest (respectively, largest) data coordinate in the right child region along the dimension D . The additional value D in the division node enables the RM-tree to split an overfilled page along the best possible dimension, depending on the distribution of data points. Figure 7 gives an example of an RM-tree structure. In the Figures 7b and 7c, cp1, cp2, and cp3 represent pointers to leaf pages P1, P2, and P3, respectively.

To insert a new object, the tree is traversed from the root to a leaf page. At each interior page, the binary-tree-like page structure (e.g., Figure 7b) is traversed to locate

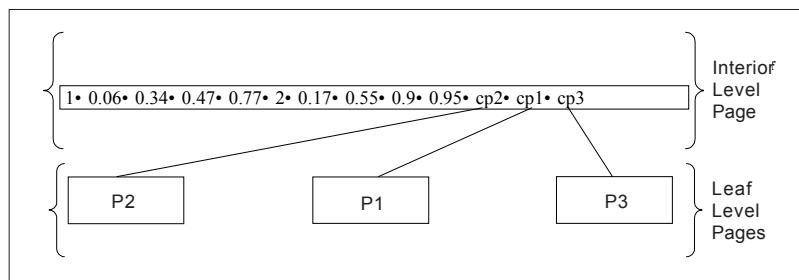
Figure 7. A simple two-level example of an RM-tree structure



(a) Data Space

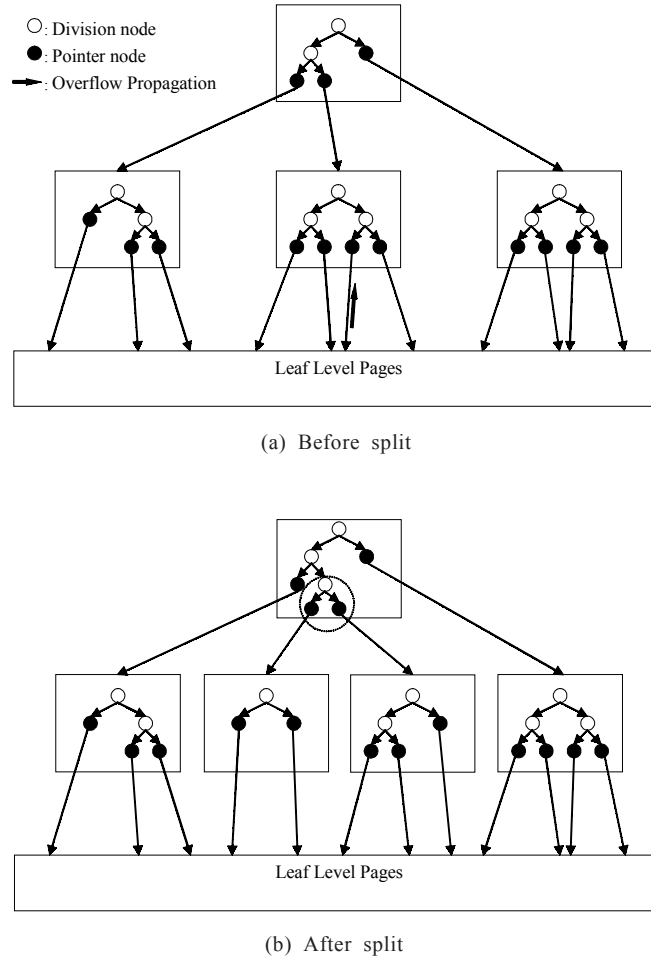


(b) Page Structure



(c) Index Structure

Figure 8. Splitting an interior page (Assumption: the capacity of the interior pages is 7 nodes)



the child-page pointer (pointer node). The page structure is traversed from the root node to a pointer node. At each division node, the left (respectively, right) branch is chosen if the new data point's coordinate P_D in the dimension D is within $[L, H]$ (respectively, $[L', H']$). If the coordinate is between H and L' , but closer to H (respectively, L'), P_D becomes the value of H (respectively, L'). In that case, follow the left (respectively, right) branch.

Each overfilled interior page is split as follows: Let O be an interior page that needs to be split. Then, O is split into the left page O and the right page O' . The left page O retains the left sub-tree, whereas the right sub-tree of O is moved to O' . Finally, in the parent page of O , the pointer to O is replaced by a small two-level binary tree. The root, the left leaf node, and the right leaf node of this two-level binary-tree represent the before-split root

node of the O 's tree, the pointer to O , and the pointer to O' , respectively. Figure 8 gives a simple example of this interior split procedure.

To insert a data point into the index structure, the insertion algorithm finds the leaf page whose region encloses the given point. The same procedure is used to process point queries. When the search reaches the leaf page, all data entries in the leaf page are tested, and the entries whose point coordinates are the same as the coordinates of the given reference point constitute the result set.

For window (range) queries, the search starts at the root page and propagates downward. At each interior page, the procedure selects all child pages whose regions intersect the given query window. When the search reaches the leaf level, the data entries of the selected leaf pages are tested. Those that are enclosed by the given query window are selected.

EXPERIMENTAL VALIDATION

We performed a comprehensive set of experiments to observe the performance of:

- the cQSF-tree with both the RM-tree and KDB-tree;
- an optimized version of the R*-tree [the topologically improved (Papadias et al., 1995) quadratic version with 30% forced-reinsertion at the leaf level (Beckmann et al., 1990)]; and
- the sQSF-tree based on the KDB-tree.

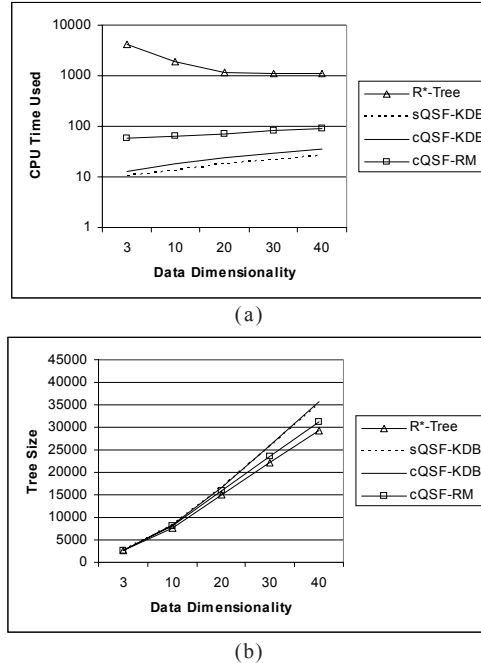
The experiments were performed on a PC with a 1.7 GHz CPU, 384MB memory, and 512KB CPU cache.

In the first experiment with skewed synthetic data, the number of dimensions was varied between 3 and 40. The page size was fixed at 4K (4,096) bytes, and all values stored in the index structures were 4 bytes long. For each d -dimensional space $[0, 1]^d$, we created a data file with 262,144 (2^{18}) randomly generated hyper-rectangular regions mainly focused in ten clusters randomly located in the universe (data space). The ten clusters, each of which had a linear extension along each dimension between 0.05 and 0.1, were populated with 209,715 random center points (about 80% of all data objects). 52,429 additional center points were randomly scattered throughout the universe. Around each center point, a hyper-rectangle was drawn with a random side length within $[0, 0.05]$ along every dimension. The data rectangles that intersected the boundary of the universe were clipped (truncated). Objects of each file were inserted into all of the tested SAMs.

Figure 9 shows the index tree construction times and the tree sizes of the four SAMs. Although the index trees of the QSF-tree variants were somewhat larger, their construction times (see also Figure 12a) were significantly lower. Observe, however, that the performance of the R*-tree's insertions (including the page splitting) is determined primarily by the logical page capacity, that is, the maximum number of entries that can be stored in a single page.

The retrieval performance of the access methods was measured using four sets of queries. Each set contained 1,000 *equal*, 1,000 *covers*, 1,000 *covered_by*, and 1,000 *not_disjoint* queries (see Tables 1 and 2). In the first three query sets, each side of a random query rectangle was obtained by generating a random center and the axis-parallel linear extension of 0.02, 0.1, or 0.25. In the last query set, each query with a randomly-

Figure 9. Synthetic skewed distribution: CPU time (sec.) used for (a) constructing the index tree and (b) the index tree size



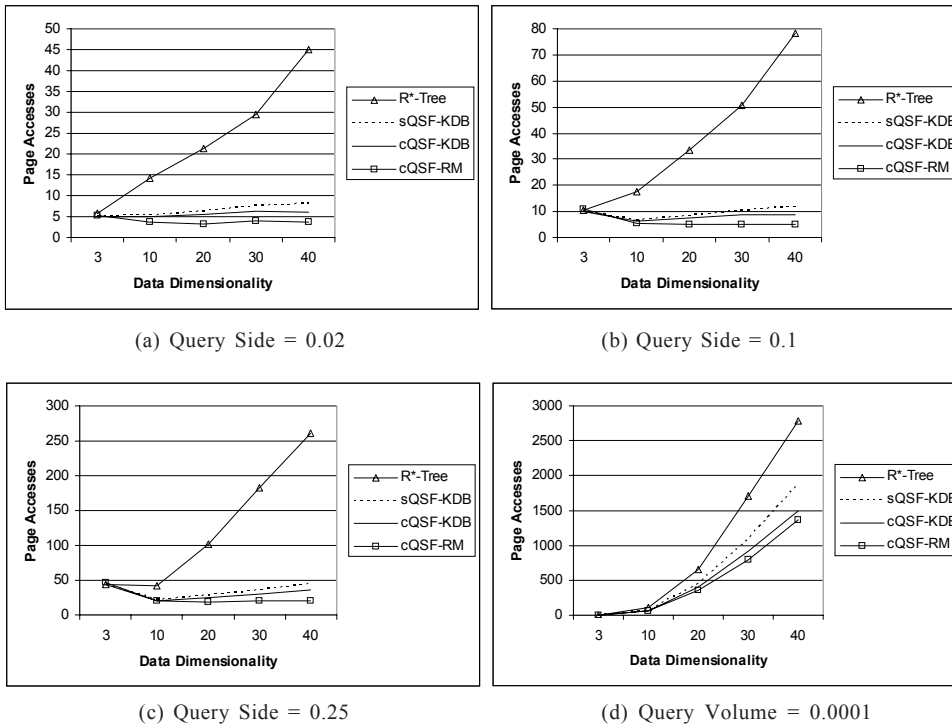
generated center had a fixed volume of 0.0001, unless it was clipped against the boundaries of the space. Thus, each unclipped query covered 0.01% of the space.

The average page accesses per query are given in Figure 10. In the figure, the notations sQSF-KDB, cQSF-KDB, and cQSF-RM include the name of the QSF-tree variant and the underlying PAM. The results confirm the efficiency of the simple QSF-tree and show that the cQSF-tree based on the RM-tree significantly improves the average retrieval performance. Observe also that the improvements increase as dimensionality grows.

We also recorded the average CPU time of query processing. Figure 11 shows that the cQSF-tree needs more CPU time due to the addition of the search predicate for the H-entries. Moreover, the RM-tree slightly increases the CPU overhead because of the additional index-region boundary values used during the search.

The last experiment involved a real data set of moderate size, called *covtype*, obtained from the UCI Machine Learning Repository (www.ics.uci.edu/~mllearn/MLRepository.html). This data set has 581,012 data points with 10 real-value attributes (dimensions), 44 binary attributes, and one category (class) attribute. We used only the first 10 attributes to create a normalized set of 10-dimensional data. We assumed that each data value is within an error of $[\pm 0.0001, \pm 0.001]$, which could be the result of inaccurate readings or normalization, limited sensor resolution, communication noise, or inaccurate timer devices (measurement and instrument errors). Thus, each data point in the file was used as a center point, around which we drew a hyper-rectangle with a random side length

Figure 10. Synthetic skewed distribution: average performance of the tested spatial access methods (the average of equal, covers, covered_by, and not_disjoint queries)



between 0.0002 and 0.002. (Small-size rectangles are appropriate here because, typically, errors are small, and with larger rectangles, the original data distribution is not preserved well.) As in the earlier experiment, all data rectangles intersecting the boundary of the universe were clipped. Figure 12 gives the index tree construction times and the tree sizes.

To measure the query (retrieval) performance of the SAMs, three sets of queries were generated. As before, each set contained 1,000 *equal*, 1,000 *covers*, 1,000 *covered_by*, and 1,000 *not_disjoint* queries. In order to produce non-empty results on this very skewed data set, each query rectangle was generated around a randomly selected data point as the center of the query with the linear extension along every dimension of 0.01 (for the first query file), 0.05 (for the second query file), and 0.1 (for the third query file). All query rectangles intersecting the boundary of the universe were clipped. The results, which are given in Figure 13, are in line with those obtained on synthetic data.

The experiments on the real data also revealed that the R*-tree had better performance, sometimes even better than the tested QSF-tree variants, for very large random queries. On this specific real data set, such queries cover significant amounts of dead space, which the R*-tree structure can eliminate effectively. However, in practical situations, the query distribution generally follows the data distribution and the average queries are not so large. For the situations when this is not the case, the QSF-trees can be built using the R*-tree as the underlying PAM.

Figure 11. Synthetic skewed distributions: Average CPU time (sec.) for query processing

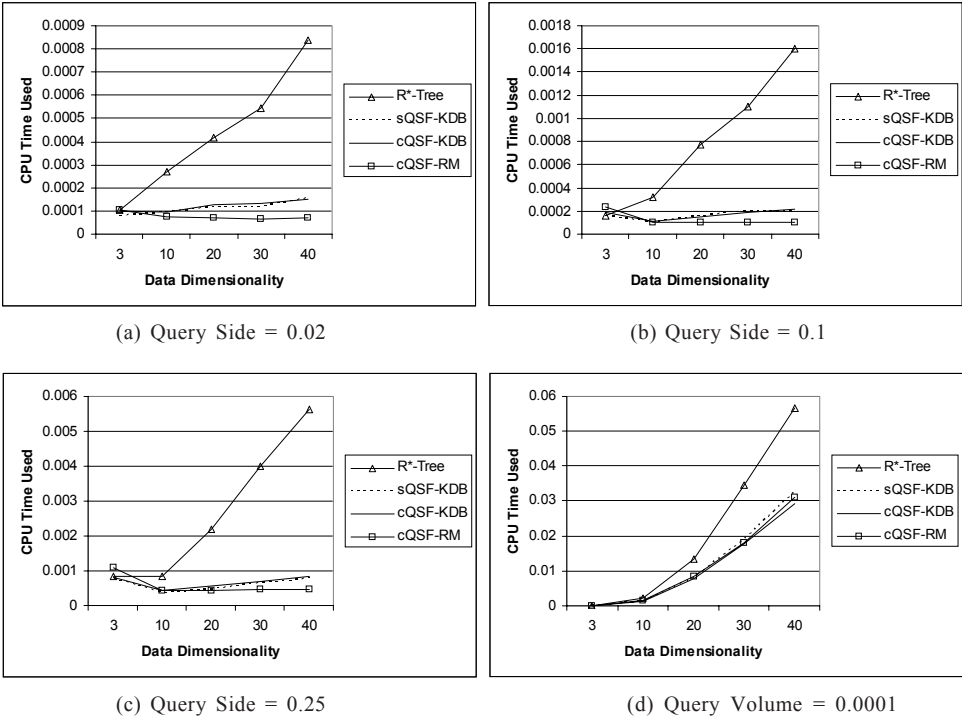


Figure 12. Real data distribution: (a) CPU time (sec.) used for constructing the index tree and (b) the index tree size

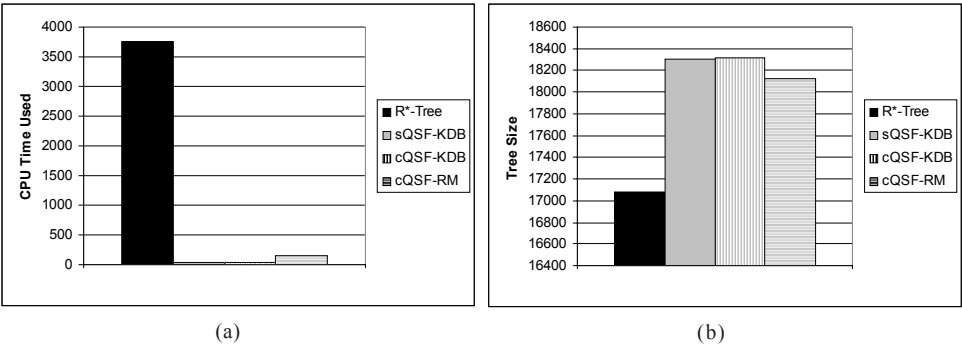
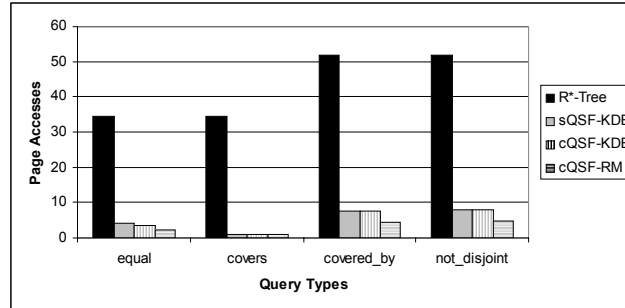
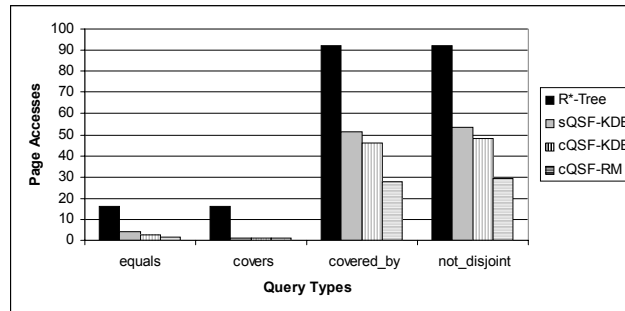


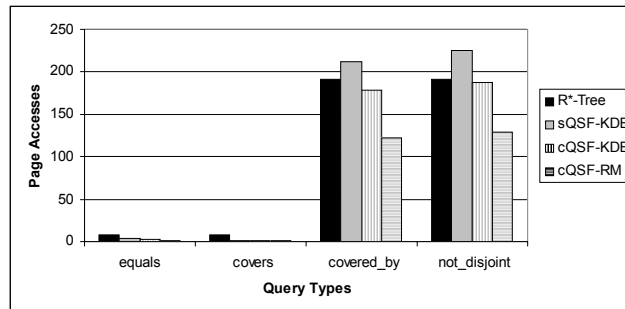
Figure 13. Real data distribution: performance of the tested access methods for each type of the query predicates in Table 1



(a) Query Side = 0.01



(b) Query Side = 0.05



(c) Query Side = 0.1

TRENDS AND ISSUES

An increasing number of emerging applications deal with a large number of continuously changing (or moving) data objects (CCDOs). CCDOs, such as vehicles, humans, animals, sensors, nano-robots, orbital objects, economic indicators, temporal geographic objects, sensor data streams, and bank portfolios (or assets), range from continuously moving objects in a 2-, 3-, or 4-dimensional space-time to conceptual entities that can continuously change in a high-dimensional space-time.

For example, several models of watches and handheld devices equipped with a GPS are already available to consumers. Accordingly, new services and applications dealing with large sets of objects that can continuously move in a geographic space are appearing. A sensor that can detect and report $n \geq 1$ distinct stimuli draws a trajectory in the $(n+1)$ -dimensional data space-time. In earth-science applications, temperature, wind speed and direction, radio or microwave image, and various other measures (e.g., the level of CO₂) associated with a certain geographic region can change continuously. There is much common ground among these different CCDOs; each CCDO can continuously change over time.

Although actual CCDOs can continuously move or change, computer systems cannot deal with continuously occurring infinitesimal changes — this would require infinite computational speed and sensor resolution. Thus, each object's spatiotemporal attribute values can only be discretely updated. Hence, the location of an object in the data space-time is always associated with a certain degree of uncertainty at every point in time.

The current and future locations of each object are estimated (via extrapolation), and the past locations of an object are represented by a sequence of connected segments, each of which joins two consecutive reported locations in the space-time (Yu, Kim, Bailey, & Gamboa, 2004). Each segment is associated with a certain degree of uncertainty (i.e., a spatiotemporal region) that encloses all possible in-between location-times of the object (Yu, Prager, & Bailey, 2005). Spatiotemporal queries are generally processed over the estimates characterizing the uncertainty of the trajectory. Therefore, the importance of access methods that can efficiently index the (low to high) multi-dimensional uncertainty regions of CCDO trajectories cannot be understated.

A number of trajectory access methods have been proposed in recent years, which can be classified into:

1. **Past trajectory access methods (PTAM):** These spatiotemporal access methods support spatiotemporal queries referring to past trajectories (Jun, Hong, & Yu, 2003; Pfoser & Jensen, 2001; Pfoser, Jensen, & Theodoridis, 2000). These spatiotemporal access methods index the minimum bounding rectangles of the trajectory segments.
2. **Future trajectory access methods (FTAM):** For the spatiotemporal queries that refer only to the current (or future) locations of CCDOs, some spatiotemporal access methods such as Saltenis, Jensen, Leutenegger, and Lopez (2000) and Papadias, Tao, and Sun (2003) have been proposed. In these access methods, each trajectory is represented by the straight line passing through the last reported location with the last reported direction and speed.

Both PTAM and FTAM trajectory access methods are based on traditional SAM structures (typically, the R-tree variants) and are designed mainly for low-dimensional CCDOs that can continuously move in a 2- or 3-dimensional geographic space. While the QSF-tree family can provide a better basis for developing efficient trajectory access methods for the emerging database applications (e.g., sensor network databases) that deal with higher dimensional CCDOs, further research is necessary to satisfy all the requirements posed by these real-time applications, which include probabilistic query processing and real-time updates. Equally important issues include page caching, concurrent reads and updates, and recoverability concerns.

CONCLUSION

Numerous database applications must deal with regional data in high-dimensional spaces. Unfortunately, traditional spatial access methods for regional objects do not scale well to higher dimensionalities. Simple QSF-trees (sQSF-trees) were designed to attack the conceptual problems that traditional spatial access methods experience in spaces with many dimensions. sQSF-trees eliminate certain conceptual problems of region-overlapping schemes, while avoiding the conceptual problems of both object clipping and object transformation. Using an original query transformation that results in two regions in the original space as well as a space-partitioning strategy of a point access method that incurs no region overlap, sQSF-trees adapt more gracefully to the growing dimensionality of data.

An improved variant of QSF-trees, called the cQSF-tree, reduces the number of false drops into index pages containing no objects that can satisfy the query. These false drops are due to the fact that sQSF-trees index only the low endpoints of object MBRs. By efficiently indexing not only the low endpoints of the object MBRs but also some approximate information about their high endpoints, and by using an efficient PAM structure called the RM-tree, cQSF-trees can increase the selectivity of search predicates and improve the performance of multi-dimensional selections. The experimental evidence shows that cQSF-trees are more scalable than sQSF-trees and R-trees with respect to increasing data dimensionality.

The proposed organization is an attractive alternative to the existing spatial access methods for low-dimensional data of geographic applications. More importantly, its ability to scale well to spaces with many dimensions makes it highly appropriate for situations when the aggregation or clustering of high-dimensional data requires efficient handling of not only points but also regional objects. As noted in this chapter, these situations regularly arise in advanced scientific applications, location-based services with moving objects, and multimedia systems.

Other than the higher performance and scalability of multi-dimensional selections, a common advantage of the QSF-tree family over traditional spatial access methods is the lower cost of dynamic updates. For example, since page splitting in R^* -trees employs several complex optimizations (Beckmann et al., 1990), the dynamic construction of R^* -trees tends to be slow (Gaede & Gunther, 1998). Even with the upward propagation of H-entry updates and the binary-tree page structure of the underlying index, the dynamic updates in cQSF-trees are much faster. As a result, cQSF-trees are more appropriate for environments where the cost of updates is an important factor.

With a few simple modifications, any PAM structure can be used to implement cQSF-trees. The required modifications of the given PAM are a simple change in the structure of the leaf entries to differentiate between the low and high endpoints of object MBRs, the maintenance of H-entries in the interior nodes, and the application of new search predicates in the selection process. Other than that, the cQSF-tree is just a simple layer of software on top of any existing PAM structure that supports a diverse set of search operations over both points and regional objects. It is possible to use a PAM based on a variant of R-trees or an access method that can index points using B⁺-trees (Yu, 2005).

The flexibility and simplicity of a cQSF-tree implementation is desirable in many practical environments. In particular, the provision for reuse of indexing techniques already deployed in many database management systems can enable rapid integration of advanced multi-dimensional capabilities into these systems. With the query transformation of sQSF- and cQSF-trees, these systems could support highly dimensional regional data they have not been able to support previously.

Further research on high-dimensional access methods is necessary to satisfy all the requirements posed by the emerging spatiotemporal database applications. These requirements include not only high access and update performance, but also effective page caching, robust concurrency, and recovery mechanisms.

ACKNOWLEDGMENTS

This research was supported in part by the National Science Foundation (NSF), Grant IIS-0312266; and the NSF Wyoming EPSCoR, Grant NSFLOC4304. We would like to thank Dr. Martha Evens and Dr. Soochan Hwang for useful discussions about a preliminary version of the QSF-tree. We would also like to thank Dr. Mario Lopez, Dr. Scott Leutenegger, Dr. Seon Ho Kim, and Dr. Thomas Bailey for useful discussions about a preliminary version of the RM-tree and spatiotemporal database applications.

REFERENCES

- Beckmann, N., Kriegel, H., Schneider, R., & Seeger, B. (1990, May 23-25). The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Atlantic City, NJ (pp. 322-331). New York: ACM Press.
- Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of ACM*, 18(9), 509-517.
- Berchtold, S., Bohm, C., & Kriegel, H. P. (1998, June 2-4). The pyramid-technique: Towards breaking the curse of dimensionality. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Seattle, WA (pp. 142-153).
- Berchtold, S., Keim, D. A., & Kriegel, H. (1996, September 3-6). The X-tree: An index structure for high-dimensional data. In *Proceedings of the 22nd International Conference on Very Large Data Bases*, Bombay, India (pp. 28-39). San Francisco: Morgan Kaufmann.

- Freeston, M. (1995, May 22-25). A general solution of the N-dimensional B-tree problem. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, San Jose, CA (pp. 80-91). New York: ACM Press.
- Gaede, V., & Gunther, O. (1998). Multidimensional access methods. *ACM Computing Surveys*, 30(2), 170-23.
- Guttman, A. (1984, June 18-21). R-trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Boston (pp. 47-54). New York: ACM Press.
- Jun, B., Hong, B. H., & Yu, B. (2003). Dynamic splitting policies of the adaptive 3DR-tree for indexing continuously moving objects. In G. Goos, J. Hartmanis, & J. van Leeuwen (Eds.), *Database and expert systems applications* (LNCS 2736, pp. 308-317). Berlin; Heidelberg: Springer-Verlag.
- Lin, K., Jagadish, H., & Faloutsos, C. (1995). The TV-tree: An index structure for high-dimensional data. *VLDB Journal*, 3, 517-542.
- Orlandic, R. (2003, April 7-10). Effective management of hierarchical storage using two levels of data clustering. In *Proceedings of the 20th IEEE / 11th NASA Goddard Conference on Mass Storage Systems and Technologies*, San Diego, CA (pp. 270-279). Los Alamitos, CA: IEEE Computer Society.
- Orlandic, R., & Yu, B. (2000, September 18-20). A study of MBR-based spatial access methods: How well they perform in high-dimensional spaces. In *Proceedings of the International Database Engineering and Applications Symposium*, Yokohama, Japan (pp. 306-315). Los Alamitos, CA: IEEE Computer Society.
- Orlandic, R., & Yu, B. (2001, July 16-18). Implementing KDB-trees to support high-dimensional data. In *Proceedings of the International Database Engineering and Applications Symposium*, Grenoble, France (pp. 58-67). Los Alamitos, CA: IEEE Computer Society.
- Orlandic, R., & Yu, B. (2002). A retrieval technique for high-dimensional data and partially specified queries. *Data and Knowledge Engineering*, 42(1), 1-21.
- Pagel, B.-U., Six, H.-W., & Toben, H. (1993). The transformation technique for spatial objects revisited. In D. Abel & B. C. Ooi (Eds.), *Advances in spatial databases* (LNCS 692, pp. 73-88). Berlin: Springer-Verlag.
- Papadias, D., Tao, Y., & Sun, J. (2003, September 9-12). The TPR*-tree: An optimized spatio-temporal access method for predictive queries. In *Proceedings of the International Conference on Very Large Databases*, Berlin, Germany (pp. 790-801). San Francisco: Morgan Kaufmann.
- Papadias, D., Theodoridis, Y., Sellis, T., & Egenhofer, M. J. (1995, May 22-25). Topological relations in the world of minimum bounding rectangles: A study with R-trees. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, San Jose, CA (pp. 92-103). New York: ACM Press.
- Pfoser, D., & Jensen, C. S. (2001, May 20). Querying the trajectories of on-line mobile objects. In *Proceedings of the International Workshop on Data Engineering for Wireless and Mobile Access*, Santa Barbara, CA (pp. 66-73). New York: ACM Press.
- Pfoser, D., Jensen, C. S., & Theodoridis, Y. (2000, September 10-14). Novel approaches to the indexing of moving object trajectories. In *Proceedings of the Very Large Data Base Conference*, Cairo, Egypt (pp. 395-406). San Francisco: Morgan Kaufmann.

- Robinson, J. T. (1981, April 29-May 1). The K-D-B tree: A search structure for large multidimensional dynamic indexes. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Ann Arbor, MI (pp. 10-18). New York: ACM Press.
- Sakurai, Y., Yoshikawa, M., Uemura, S., & Kojima, H. (2000, September 10-14). The A-tree: An index structure for high-dimensional spaces using relative approximation. In *Proceedings of the 26th International Conference on Very Large Data Bases*, Cairo, Egypt (pp. 516-526). San Francisco: Morgan Kaufmann.
- Saltenis, S., Jensen, C. S., Leutenegger, S. T., & Lopez, M. A. (2000, May 16-18). Indexing the positions of continuously moving objects. In *Proceedings of the International Conference on Management of Data*, Dallas, TX (pp. 331-342). New York: ACM Press.
- Seeger, B. & Kriegel, H. P. (1988, August 19-September 1). Techniques for design and implementation of efficient spatial access methods. In *Proceedings of the 14th International Conference on Very Large Data Bases*, Los Angeles, CA (pp. 360-371). San Francisco: Morgan Kaufmann.
- Sellis, T., Roussopoulos, N., & Faloutsos, C. (1987, September 1-4). The R+-tree: A dynamic index for multi-dimensional objects. In *Proceedings of the 13th International Conference on Very Large Data Bases*, Brighton, UK (pp. 507-518). San Francisco: Morgan Kaufmann.
- Swets, D. L., & Weng, J. (1996). Using discriminant eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8), 831-836.
- Weber, R., Schek, H.-J., & Blott, S. A. (1998, August 24-27). A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24th International Conference on Very Large Data Bases*, New York (pp. 194-205). San Francisco: Morgan Kaufmann.
- White, D. A., & Jain, R. (1996, February 26-March 1). Similarity indexing with the SS-tree. In *Proceedings of the International Conference on Data Engineering*, New Orleans, LA (pp. 516-523). Los Alamitos, CA: IEEE Computer Society.
- Yu, B. (2005). Adaptive query processing in point-transformation schemes. In K. V. Andersen, J. Debenham, & R. Wagner (Eds.), *Database and expert systems* (LNCS 3588, pp. 197-206), Berlin Heidelberg: Springer-Verlag.
- Yu, B., Kim, S., Bailey, T., & Gamboa, R. (2004, July 7-9). Curve-based representation of moving object trajectories. In *Proceedings of the International Database Engineering and Applications*, Coimbra, Portugal (pp. 419-425). Los Alamitos, CA: IEEE Computer Society.
- Yu, B., Orlandic, R., & Evens, M. (1999, November 2-6). Simple QSF-trees: An efficient and scalable spatial access method. In *Proceedings of the 8th International Conference on Information and Knowledge Management*, Kansas City, MO (pp. 5-14). New York: ACM Press.
- Yu, B., Prager, S. D., & Bailey, T. (2005). The isosceles-triangle uncertainty model: A spatiotemporal uncertainty model for continuously changing data. In C. Gold (Ed.), *Workshop on Dynamic & Multi-Dimensional GIS, International Society for Photogrammetry and Remote Sensing* (Vol. XXXVI [2/W29], pp. 179-183). The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences.

Section IV: Semantic Database Analysis

Chapter XIX

A Concept-Based Query Language Not Using Proper Association Names

Vladimir Ovchinnikov, Lipetsk State Technical University, Russia

ABSTRACT

This chapter is focused on a concept-based query language that permits querying by means of application domain concepts only. The query language has features making it simple and transparent for end-users: a query signature represents an unordered set of application domain concepts; each query operation is completely defined by its result signature and nested operation's signatures; join predicates are not to be specified in an explicit form, and the like. In addition, the chapter introduces constructions of closures and contexts as applied to the language which permits querying some indirectly associated concepts as if they are associated directly and adopting queries to users' needs without rewriting. All the properties make query creation and reading simpler in comparison with other known query languages. The author believes that the proposed language opens new ways of solving tasks of semantic human-computer interaction and semantic data integration.

INTRODUCTION

Conceptual models serve for application domain modeling as opposed to means of system implementation modeling. A conceptual model does not concern implementation details and describes an application domain's essence. Conceptual models underlie conceptual query languages that are meant for querying schemas of the models (here and throughout the chapter, a model is considered to be a mean of modeling, and a schema

is considered to be a result of modeling). The languages have dual use. On the one hand, conceptual queries play the key role in constraint formalization—any constraint can be formulated as a query and an assertion upon it. On the other hand, the queries can be used for requesting data from an information system wrapped by a conceptual schema. In both cases, conceptual query transparency and simplicity are very important.

Aiming at more transparency and simplicity of conceptual queries, the author proposes Semantically Complete Query Language (SCQL) (Ovchinnikov, 2004b, 2005b; Ovchinnikov & Vahromeev, 2005). The language is founded on the semantically complete model (SCM) (Ovchinnikov, 2004a, 2005b, 2004c), the main property of which is semantic completeness that endows the model and query language with their names. A schema of the model is a set of application domain concepts, concept associations, and constraints defined over. The semantic completeness property implies a SCM schema does not include associations describing interrelation of application domain concepts differently; in other words, each association describes semantics of concept interrelation completely (more precise definition will be given in the section, *Restrictions Imposed on Underlying Model*). The main consequence of the property is that associations are based on unique (within a schema) concept sets; an association is identified with a set of underlying concepts, and not a proper name. As a result, SCQL is created that uses concept sets for referring to associations. The language permits querying by means of application domain concepts completely; proper names of associations are not used within it.

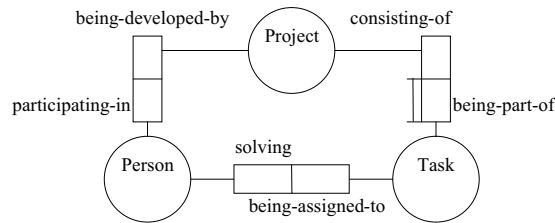
There are several other properties of SCQL that resulted in more simplicity and transparency of its queries: each query operation is completely defined by its signature¹ and nested operations' signatures; a signature of any query is an unordered set of application domain concepts; and join predicates have not to be specified in an explicit form. In addition, this chapter introduces conceptions of closures and contexts as applied to the language. The conceptions permit querying some indirectly associated concepts as if they are associated directly and adopting queries to users' needs without rewriting. All the properties make query creation and reading simpler in comparison with other known query languages, which will be proved in the subsequent sections. The author believes that all these properties and others discussed permit usage of the language by end-users who are not specialists in information technologies (IT).

The chapter considers restrictions imposed on an underlying model by SCQL, the way of referring to associations within it, the structure of SCQL expressions, and the context mechanism. All ideas are illustrated using the running example introduced in the next section. Finally, the chapter shows the ways of application and development of the query language.

QUERY SIMPLIFICATION METHODS

Now there exist many conceptual and data models and modeling approaches: entity-relationship (ER) (Chen, 1976; Chen, 1981), object-role modeling (ORM) (Bronts, Brouwer, Martens, & Proper, 1995; Halpin, 1995, 2001) and its particular cases (Brouwer, Martens, Bronts, & Proper, 1994; Bommel, Hofstede, & Weide, 1991; Halpin & Orłowska, 1992; Hofstede & Weide, 1993; Nijssen & Halpin, 1989; Troyer, 1991), fully communication oriented information modeling (FCO-IM) (Bakema, Zwart, & Lek, 1994), conceptual

Figure 1. ORM model of project management domain



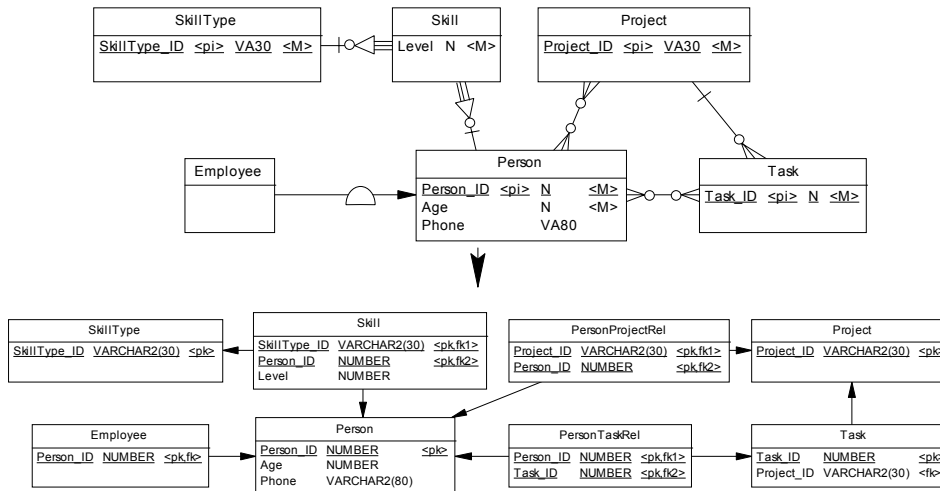
graphs (CG) (Dibie-Barthelemy, Haemmerle, & Loiseau, 2001), Web Ontology Language OWL (W3C, 2004b), resource description framework (RDF) (W3C, 2004a), relational model (RM) (Codd, 1979), and others. Almost every existing model underlies one or several query languages used for accessing information through the models, for instance, ORM underlies LISA-D (Hofstede, Proper, & Weide, 1993, 1996) and Conquer-II (Bloesch & Halpin, 1997), RDF underlies RDQL (Seaborne, 2004), RDL and other query languages, RM underlies relational algebra (Codd, 1972) and partially SQL, and so forth. Unfortunately, all existing query languages are not sufficiently transparent for end-users not being specialists in IT, though several simplification methods were applied to some of them.

Query simplification can be achieved by using natural names for entities and relations when modeling and querying (Halpin, 2004; Hofstede, Proper, & Weide, 1997; Owei, 2000; Owei & Navathe, 2001b). The method significantly simplifies end-user work as interaction with a system takes place directly in application domain terms. Examples of such query languages are LISA-D (Hofstede, Proper, & Weide, 1993, 1996) and CQL (Owei & Navathe, 2001b). But this simplification is not structural: the query structure remains complex. Let us illustrate the fact with the example of project management domain. Persons, tasks, and projects are the main entities of the domain: projects consist of tasks that are assigned to persons; persons can participate in project teams directly. The application domain has the formalization in ORM as seen in Figure 1.

The query “select all tasks assigned to persons participating in the project MES’s team” is formulated in LISA-D as “Task being-assigned-to Person participating-in Project MES.” The path expression has the following complexity factors: (a) the order of entities and roles is important and should be kept correct; (b) the appropriate role names should be remembered precisely (for instance, “being-assigned-to” or “participating-in”). A user is not insured against creation of senseless queries like “Task solving Person,” “Person consisting-of Project,” or mistakes like “Person solved Task,” as a result of incorrect usage or remembering of the role names. Using the SCQL language discussed further, the query is formulated as (Task–Person–Project=“MES”). Here proper relation or role precise names are not used, and one does not have to remember them.

Unfortunately, LISA-D did not become an industrial standard for information system development and user interaction, and it is slightly supported by tools. Now the standard is SQL. Therefore, let us use SQL for the comparison purpose below. It is allowable because SCQL and SQL applications have one common field—both languages

Figure 2. ER and relational schemas of project management domain



can be used as a mean of end-user interaction with an information system. In the case of SCQL, such information systems are to be wrapped by SCM and may be backed by relational or other DBMS [prototype system implementation can be found in Ovchinnikov (2005a)].

The project management domain can be formalized, using ER (Chen, 1976, 1981) and relational model, as shown in Figure 2. To use the example as a running one, we have introduced new entities: person's phone, age, skills, and an employee as a particular case of a person.

The query “select all tasks assigned to persons participating in the project MES's team” considered above is formulated in SQL as follows:

```
SELECT Task_ID FROM PersonTaskRel ptr, PersonProjectRel ppr
WHERE ptr.Person_ID = ppr.Person_ID AND ppr.Project_ID = 'MES'.
```

The given SQL query has the following complexity factors in comparison with the SCQL query (Task–Person–Project=“MES”):

- the join predicate “ptr.Person_ID = ppr.Person_ID” is defined explicitly;
- the appropriate precise table names should be remembered;
- the query's signature is lacking in semantics since it consists of abstract columns not associated with the application domain's concepts;
- names of fields and tables are noticeably far from the natural language.

Finally, the SCQL query is shorter and easier to understand.

Known query languages use proper names for referring to associations (relations, fact types); one has to remember many precise names to formulate queries using the languages. The reason lies in models underlying the languages: ORM, ER, RM, and

others. The models require identification of relations by their proper names. Any two entities of a schema of the models can be associated in many ways, and each of the ways takes its own unique name. As a result, one cannot think about entities as if they are merely associated. At the same time, it is not necessary to remember a precise name of the association of “Person” and “Task” if one refers to it by (Person, Task) as the proposed language implies. Such language behavior impacts properties of the underlying model, which will be considered in the next section.

Not all associations can be named clearly and shortly. Sometimes full names of associations are whole sentences actually enumerating participated concepts and only. For instance, the association of “Person” and “Task” can be named “Persons solving tasks,” “Tasks being solved by persons,” or “Assignments of tasks to persons.” Formulating a query within any known query language, one should remember the way of association naming. This is not necessary when a concept enumeration is used for referring to an association, for instance, (Person, Task). Detailed discussion of referring to associations within SCQL will be given in the section, “Association Referring And Context Mechanism Within SCQL Expressions.”

Many query languages have another complexity factor — query signatures are not based on application domain concepts; in such languages, interpretation of a query result is completely determined by a structure of the query. For instance, the column “Task_ID” in the previous SQL query can mean anything, even phone number. One should analyze the query’s structure to understand the real meaning of the column. Moreover, one is not insured against formulation of senseless queries, for instance, joining the tables “Skill” and “Person” with “Age = Level.” As a result, the languages are too complicated for end-users. The offered solutions for the problems will be discussed in the following sections.

Another way of query simplification is usage of a GUI application concealing query complexity, as, for instance, Conquer-II (Bloesch & Halpin, 1997) and OSM-QL (Embley, Wu, Pinkston, & Czejdo, 1996) offer. Using intuitively clear interface elements like trees, one can easily construct conceptual queries. Nevertheless, the approach’s extent of simplification has the limit imposed by strong impact of a query language structure to GUI: tree node types, node connectivity, and node attributes are dictated by the structure. Since each operation of the language proposed is completely defined by its resulting signature and nested operations’ signatures, the author believes the language has simpler structure than existing query languages, well suits the purpose of GUI-based query languages, and should be developed in that direction in the future.

Existing query languages still remain complex for end-users as they have the following main complexity factors: (a) queries are formulated using association proper names, and not application domain concepts; (b) queries have structure including many complicated elements; (c) there is no context mechanism that would permit using some indirectly associated concepts as if they are associated directly according to pre-adjusted context. The chapter introduces Semantically Complete Query Language (SCQL) which attempts to solve the complexity factors.

Let us summarize characteristics of SCQL and the well-known query languages LISA-D, Conquer, and SQL (see Table 1). The languages were selected as they are representative specimens of the very different query language categories. Analyzing the table, one could conclude that the most important distinction of SCQL is pure concept-

Table 1. Summary of characteristics of the languages LISA-D, Conquer, SQL, and SCQL

Characteristic	LISA-D	Conquer	SQL	SCQL
Declarative queries	+	+	+	+
Natural names for entities and associations (relations)	+	+	-	+
GUI-based query formulation	-	+	-/+	-
Semantic result signatures (referring to domain concepts)	+	+	-	+
Purely concept-based query formulation (uselessness of proper association names)	-	-	-	+
Capability of implicit join predicates	-/+	-/+	-	+
Prohibition of senseless queries	-/+	-/+	-	+
Formulation of queries as concept chains	-/+	-/+	-	+
Formulation of join-like queries as a resulting signature merely	-	-	-	+
Query adaptation without rewriting	-	-	-	+

based query formulation without resorting to association proper names. In the next sections, all the listed characteristics will be considered in detail, in addition to GUI-based query formulation, which is perspective of SCQL development.

RESTRICTIONS IMPOSED ON UNDERLYING MODEL

Disusing of proper association (relation, fact type) names promises the most noticeable increase of query language simplicity. The only way of referring to associations without the use of explicit names is to use concept (entity, object type) combinations as references to associations so that each concept combination gets identification of an appropriate association. The identification would imply a sequence of concepts, but this method is not transparent. Therefore, the language being proposed uses the method of relation identification by means of sets of application domain concepts and does not use proper association names.

As a result, not any model can be used as basis for the query language; such model has to permit identification of relations by domain concept sets. A model having the identification property was proposed by Ovchinnikov (2004a, 2005a) and was named the semantically complete model (SCM). Moreover, SCM is more restricted than the identification property imposes — it is semantically complete. The semantic completeness property means that (a) within a schema, each association is uniquely identified with a set of concepts underlying it, (b) an association can not be based on a concept set being a proper subset of another concept set underlying another association of the same schema; in other words, each association describes semantics of concept interrelation completely. Any schema that satisfies the semantic completeness property (SCM schema) also satisfies the identification property since each association covers a unique set of concepts.

SCM is a full-scale modeling technique having textual notation that is near to natural languages [see Ovchinnikov (2004a, 2004b, 2005a) for details]. Continuing the above-running example, let us present a SCM schema of the example domain in the textual notation:

Person solves Tasks [Task]

Person has a Phone →

Person has a Skill Level for a Skill Type

Employee is a Person =

[(Person, Skill Type) → Skill Level]

Project consists of Tasks ← [Task]

Person is of Age →

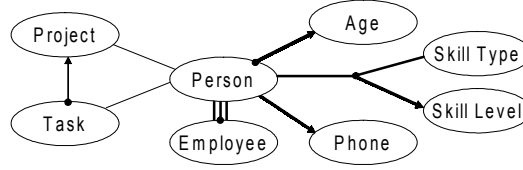
Project has a team of Persons [Team]

Here the associations and concepts are self-described as each association is represented by a sentence where application domain concepts are marked with capital first letters. The most general constraints are given within the sentences: functional constraints of binary associations (→), equivalent constraints of binary associations (=), mandatory constraints (_). For instance, the association “Person is of Age →” is constrained as “each person must correspond to only one age,” the association “Employee is a Person =” is constrained as “each employee must correspond to only one person and a person can correspond to only one employee,” and the association “Person solves Tasks” is not constrained at all.

More complex constraints are placed in square brackets after sentences with indent, for instance, “each combination of person and skill type can determine only one skill level” is formulated as “[(Person, Skill Type) → Skill Level].” Such constraints can be a lot more complex when being based on SCQL queries or statements formulated using SCQL-extended predicate calculus. The running example does not include all existing types of SCM constraints. Detailed definition of textual and graphical SCM notations, including the constraint language, is out of the scope of the chapter. The same SCM schema in graphical notation is presented in Figure 3.

One can see from Figure 3 that SCM graphical notation is an extension of a type of hypergraph notation: concepts are nodes and associations are edges. Concepts are

Figure 3. Graphical notation of SCM schema of project management application domain



designated with ellipses and associations – with lines or star-lines connecting appropriate concepts. General constraints are placed upon concepts and associations: functional constraints as arrows, equivalent constraints as triple lines, mandatory constraints as dots. For instance, the association “Person has a Skill Level for a Skill Type” is designated with a star-line pointed at “Skill Level” as it is constrained with “[(Person, Skill Type) → Skill Level].”

Any SCM schema is a set of associations based on sets of concepts; also a SCM schema includes a set of constraints, but this question is out of the scope of the chapter. Let m be a set of SCM schemas, a be a set of associations, c be a set of concepts. Then $ma \subseteq m \times a$ determines correspondence of associations and schemas, and $ac \subseteq a \times c$ determines correspondence of concepts and associations. The association identification constraint “a model cannot have two associations based on the same set of concepts” can be formulated as follows:

$$[C1] \quad \forall m' \in m \forall a' \in a \forall a'' \in a \left\{ \begin{array}{l} (m', a') \in ma \wedge (m', a'') \in ma \rightarrow \\ \{c' \mid (a', c') \in ac\} \neq \{c' \mid (a'', c') \in ac\} \end{array} \right\}$$

The main property of SCM is semantic completeness, which means that within a schema there is no an association based on a concept set being a proper subset of a concept set of another association. This restriction guarantees that each association defines semantics of interrelation of underlying concepts completely.

$$[C2] \quad \forall m' \in m \forall a' \in a \forall a'' \in a \left\{ \begin{array}{l} (m', a') \in ma \wedge (m', a'') \in ma \rightarrow \\ \{c' \mid (a', c') \in ac\} \not\subseteq \{c' \mid (a'', c') \in ac\} \end{array} \right\}$$

The identification constraint C1 is sufficient for referring to associations without using proper names and, therefore, it is sufficient for creation of a query language not using proper association names. The semantic completeness constraint C2 is introduced since it increases schema and query simplicity and transparency; one can think about interrelation of a set of concepts as complete phenomenon, knowing that there are no alternatives for this interrelation (Ovchinnikov, 2004b). This constraint impacts the context mechanism, which will be discussed in the next section.

The conceptual query language based on SCM and not using proper association names for query formulation was named the Semantically Complete Query Language (SCQL) (Ovchinnikov, 2004b, 2005a) and will be discussed in the following sections.

ASSOCIATION REFERRING AND CONTEXT MECHANISM WITHIN SCQL EXPRESSIONS

As a result of the identification constraint C1, SCQL uses concept sets for referring to associations and not proper names. SCQL provides for simple notation of such references, namely, enumeration of concepts by comma in round brackets; concept order in the enumerations is not important. For instance, both of the references (Person, Skill Level, Skill Type) and (Person, Skill Type, Skill Level) are correct and point to the same association. One can see that one does not have to remember proper association names to refer to the association and must only know the fact of interrelation of the concepts.

A reference to an association is considered as a selection of all its instances. For example, the expression (Person, Skill Level, Skill Type) is a selection of all instances of the appropriate association. The analogous SQL query is the following: “SELECT SkillType_ID, Person_ID, Level FROM Skill.” One can see from the example that the SQL expression includes the proper name of the table “Skill,” while the appropriate SCQL expression has no such element.

Here, a composition operation of SCQL can be considered as a mathematical composition (a natural join) of subqueries (see the next section for details). When a composition operation is built over selections of associations, it uses direct references to associations. For this case, there are two special notations that make an expression more simple and transparent for end-users: path and star notations. Each notation has its own scope of application where it is the most usable.

The path notation is used when several binary associations forming a connected chain are composed. A path expression is a chain of concepts separated by dashes. Each adjacent concept pair is considered as a concept set referring to an appropriate association. Therefore, a chain as a whole is a composition of all associations referred by adjacent pairs. For instance, the expression (Person–Task–Project) selects project and persons for each task by composing the associations (Person, Task) and (Task, Project). One can see that SCQL chains have no any attributes besides concepts themselves as opposed to, for example, LISA-D where names of used relations (predicators, saying more precisely) are to be indicated explicitly: “Person solving Task being-part-of Project.” SCQL path expressions can be written down starting from any edge concept, for example, (Project–Task–Person) is equivalent to (Person–Task–Project). The analogous SQL query is as follows:

```
SELECT t.Project_ID, t.Task_ID, ptr.Person_ID
FROM Task t, PersonTaskRel ptr WHERE t.Task_ID = ptr.Task_ID.
```

This SQL expression has the following complication factors that the above SCQL expressions do not have: (a) the resulting signature is not semantic since a result column could mean anything (for example, Person_ID could mean even “phone number”); one must analyze the SQL expression structure to understand the real semantics of each

column; (b) the explicit join predicate “ $t.Task\_ID = ptr.Task\_ID$ ” has been defined; and (c) the proper table names “Task” and “PersonTaskRel” have been used.

The star notation is used when several binary associations forming a star with one central concept are composed. A star expression is a comma-separated list of non-central concepts in square brackets chained with a central concept (by means of dash). For instance, one can use the star expression (Person–[Project, Phone]) instead of the path expression (Phone–Person–Project). The star notation is the most convenient when there are more than two non-central concepts in a star. The star notation has the same advantages relative to analogous SQL and LISA-D queries as the path notation.

One concept can play several roles within an expression, for example, when using one association several times. For this purpose, SCQL introduces the concept of a “role concept.” A role concept is a concept extended with a role name that indicates the concept’s role in a given expression. Role names are placed in round brackets after concepts if different roles are necessary. For instance, consider the expression (Project(Task’s)–Task–Person–Project(Person’s)). Here both projects are semantically diverse columns of the expression and have the role names “Task’s” and “Person’s.” The analogous SQL query is as follows:

```
SELECT t.Project_ID, t.Task_ID, ppr.Person_ID, ppr.Project_ID
FROM Task t, PersonTaskRel ptr, PersonProjectRel ppr
WHERE t.Task_ID = ptr.Task_ID AND ptr.Person_ID = ppr.Person_ID.
```

If one does not use the roles in the expression, it becomes cyclic with one project column: (Project–Task–Person–Project), which reads as “select persons with their tasks being part of projects of which the persons are members.” Since the expression is cyclic, it can be equivalently reformulated starting from any concept, for instance, as (Person–Task–Project–Person). In both cases of cyclic expressions, their resulting signatures contain only these three elements: “Person,” “Task,” and “Project.” The analogous SQL query is the following:

```
SELECT t.Project_ID, t.Task_ID, ppr.Person_ID
FROM Task t, PersonTaskRel ptr, PersonProjectRel ppr
WHERE t.Task_ID = ptr.Task_ID AND ptr.Person_ID = ppr.Person_ID
AND t.Project_ID = ppr.Project_ID
```

As the SQL query is cyclic, it contains the additional condition “ $t.Project\_ID = ppr.Project\_ID$ ” that completes the cycle, and it has the only resulting “Project_ID” column.

Saying this formally, let rc be a set of role concepts, rn be a set of role names. Then the maps $rcc : rc \rightarrow c$ and $rcrn : rc \rightarrow rn$ reflect the facts that a role concept pertains to a concept and can have a role name; at that, each role concept is to pertain to a concept:

$$[C3] \quad \forall rc' \in rc \exists c' \in c \{ (rc', c') \in rcc \}$$

Concept enumerations are used within SCQL not only for referring to associations, but also for requesting interrelation of indirectly associated concepts by using the context mechanism of SCQL. The mechanism increases simplicity and transparency of

queries to a greater extent since it permits omitting “trivial” inter-concept transition details. For example, if (Employee, Person) and (Person, Phone) are included to the current context, one can execute the query “select phones of employees” using (Employee, Phone) or (Employee–Phone) instead of (Employee–Person–Phone). Here the transition “Employee–Person–Phone” is considered as “trivial” and therefore can be shortened to “Employee–Phone.” Comparing the query (Employee–Phone) and the analogous SQL query: `SELECT e.Person_ID, p.Phone FROM Employee e, Person p WHERE e.Person_ID = p.Person_ID`, one can conclude that the SCQL query is a lot more simple and transparent than the SQL query. The analogous LISA-D query is also more complicated than the SCQL query: “Employee being Person having Phone.”

The context mechanism permits for some composition-projection queries to be shortened to simple enumeration of required concepts. The core concepts of the context mechanism are “association closure” and “execution context.” An association closure serves as an agreement on query shortenings and is characterized by unity of effect, that is, either all or none of shortenings implied by the agreement take effect. An association closure is defined over a SCM schema and is a set of associations of the schema. An execution context is a set of association closures or associations directly. An SCQL query-execution system has a single execution context at a time named as current one. The current context is used for executing any shortened SCQL query.

The context mechanism increases query transparency and simplicity; a composition-projection query can be shortened to simple concept enumeration. Therefore, a concept enumeration can mean a selection of an association as well as a shortened query. If the queried schema has an association based on the specified concept set, then the enumeration is considered as an association selection; otherwise, the enumeration is considered as a shortened query. For example, the query (Employee, Phone) is a shortening of the composition (Employee–Person–Phone), while (Person, Phone) is the reference to the appropriate association.

Any shortened query is subject to execution in the following way. Consider as a hypergraph all associations included to a context directly or indirectly by means of closures. Pick out all connected sub-hypergraphs existing in the hypergraph. Each of the connected hypergraphs has its own set of concepts underlying its associations. If the desired shortened query enumerates concepts pertaining to different connected hypergraphs, it is concluded that the query is mistaken and cannot be executed. Otherwise, it is taken as a minimum set of associations that connect the required concepts, including all alternative connecting paths. The taken associations are composed and then projected by the required concepts. The result of the projection is the result of the shortened query.

Using the mechanism, all non-cycle path and star expressions can be written as simple enumerations of required concepts. For instance, the shortened query (Employee, Phone) is executed as a composition of the associations (Employee, Person) and (Person, Phone), and then the result is projected on “Employee” and “Phone.” The context mechanism serves for similar purpose as the abbreviated concept-based query language presented in Owei and Navathe (2001a) and Owei, Navathe, and Rhee (2002) that does not require entire query paths to be specified but only their terminal points. Formally, let ac be a set of closures and cx be a set of contexts. Then $aca \subseteq ac \times a$ defines associations included to each closure and $cxac \subseteq cx \times ac$ defines closures constituting each context.

Association closures can be of different types. There can be some default closures for each SCM schema; the default closures are always included to the current context. Other closures are to be uniquely named since they are to be included to or excluded from a context explicitly. Both the default closure set *dac* and the named closure set *ncc* are subsets of the general closure set: $dac \subseteq ac$, $nac \subseteq ac$. A set of associations included with a closure can be specified explicitly or can be calculated from a schema according to an algorithm. For instance, a closure can be calculated as all associations not part of cycles on a schema. The calculated closure set *cac* is also subset of the general closure set: $cac \subseteq ac$.

Closures are designated on a SCM schema as follows. Associations of the main default closure are designated with bold as it is done in the running example above. Associations of named closures are followed by closure names in square brackets, as, for instance, the association “Project has a team of Persons [Team].” If an association is included with more than one named closure, they are enumerated with comma within the brackets.

The simplest context is an empty one when closures are not used and all concept enumerations refer to associations directly, as, for instance, (Person, Phone). Closures are added to and removed from the current context explicitly. Even if the context is not empty, one may decide not to use the context mechanism by writing full queries without shortenings. This is recommended if a query should not change semantics when changing the current context; otherwise, if a query must be context-sensitive, it should be written down using context-sensitive shortenings.

The context mechanism makes SCQL more flexible, but if one uses the context mechanism heedlessly, query semantics can change unpredictably. Therefore, context change should be closely controlled. For instance, if the current context contains the closure “Team” (in addition to the default closure of course) of the running example, the query (Project, Phone) or (Project–Phone) will select all phones of persons being members of project teams. If the current context contains the closure “Task,” then the same query (Project–Phone) will select all phones of persons solving tasks of the projects. So the query has different semantics in different contexts. One could make semantics stable if he/she would write it fully as the query (Project–Person–Phone) for the first semantics or as the query (Project–Task–Person–Phone) for the second semantics. Translating the shortened query (Project–Phone) to SQL, one gets two different SQL queries depending on the current context. The first SQL query is:

```
SELECT e.Person_ID, ppr.Project_ID FROM Employee e, PersonProjectRel ppr
WHERE e.Person_ID = ppr.Person_ID,
```

and the second one is:

```
SELECT e.Person_ID, t.Project_ID FROM Employee e, Task t, PersonTaskRel ptr
WHERE e.Person_ID = ptr.Person_ID AND ptr.Task_ID = t.Task_ID.
```

Both queries are a lot more complicated than the SCQL query (Person–Phone).

Context mechanism is very useful when a schema is evolving. If one modifies a schema not removing concepts and not changing their semantics, the modification can be done absolutely transparently by means of default closure configuring. For instance,

introduce a new concept to the running example: “Communication Address,” and replace the association “Person has a Phone $\rightarrow$ ” of the schema with the following associations:

Person has a Communication Addresses –

Phone is a Communication Address $\equiv$

$[(\text{Person-Phone}): \text{Person} \rightarrow \text{Phone}]$

Since the new associations are in the default closure (they are in boldface)—and so always in the current context—all queries that used the association (Person, Phone) do not change their semantics. The concept enumeration (Person, Phone), which was the association selection, becomes a shortening for the composition (Person–Communication Address–Phone) and subsequent projection on the concepts “Person” and “Phone.” Therefore, the modification has passed unnoticed by schema users.

SCQL context can be created according to different strategies. An obvious strategy is to reflect users’ preferences for data browsing. This approach is suitable for simple queries when users go from one concept to another without writing complex expressions. In this case, a context can be changed explicitly or automatically by using browsing statistics, for instance, an association usage frequency.

Another strategy of context creation aims to reflect shortenings generally accepted by a community or an application domain. The generally accepted shortenings underlie default closures; other shortenings underlie several named closures and are optional for some part of a community or an application domain. The options are activated when necessary by user or automatically.

And the last strategy can be used in natural language recognition systems. The strategy implies a context changes dynamically for each new text part. According to this strategy, a context of a previous text part is used as basis for a context of a next text part and the last context is modified by using some statistics of both text parts. Context mechanism of SCQL is unique; other known query languages have no such mechanism at so deep an architectural level; context changes do not require query rewriting.

SCQL EXPRESSION STRUCTURE AND PROPERTIES

This chapter is focused on the following main SCQL property: queries are formulated by using application domain concepts completely; the property is guaranteed by the fact that associations are identified by concept sets; and it increases transparency and simplicity of query expressions, especially when using contexts, path and star expressions. In addition, SCQL has other interesting properties based on characteristics of its operations that are considered below.

An expression of any query language represents a tree of operations. A set of possible operation types varies from one language to another, but leaf operations always are selections from relations of an underlying schema. Let e be a set of expressions, o

be a set of operations, and ot be a set of operation types. Then $oe : o \rightarrow e$ determine operations of each expression, and $oot : o \rightarrow ot$ determine an operation type for each operation. Leaf operation types are subset of all operation types ($lot \subset ot$) and leaf operations are subset of all operations ($lo \subset o$). It is true that all and only leaf operations are to be of leaf operation types:

$$[C4] \forall o' \in o \forall ot' \in ot \left\{ \begin{array}{l} (o', ot') \in oot \rightarrow \\ \{ (o' \in lo \wedge ot' \in lot) \vee (o' \notin lo \wedge ot' \notin lot) \} \end{array} \right\}$$

Operations can be nested to other operations: $oo : o \rightarrow o$. All and only leaf operations have no nested operations:

$$[C5] \forall o' \in o \left\{ \begin{array}{l} (o' \in lo \rightarrow \{o'' \mid (o'', o') \in oo\} = \emptyset) \wedge \\ (o' \notin lo \rightarrow \{o'' \mid (o'', o') \in oo\} \neq \emptyset) \end{array} \right\}$$

A signature $sign$ of any SCQL operation is a set of role concepts: $osign : o \rightarrow sign$, $sign \subseteq rc$. Each SCQL operation is to have a signature:

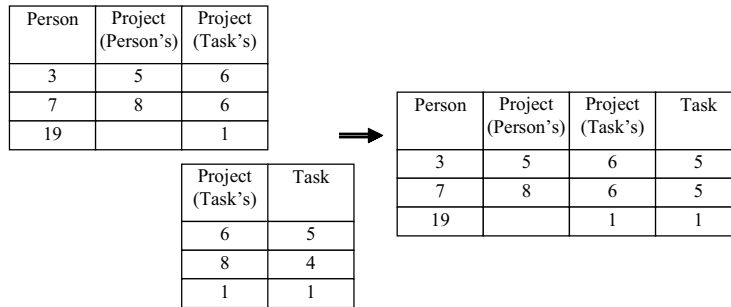
$$[C6] \forall o' \in o \exists sign' \in sign \{ (o', sign') \in osign \}$$

SCQL provides for the following operation types serving as non-leaf ones: composition, transformation, union, and minus. Operations of the types will further be named as composition operation, transformation operation, and so on. Let $comp$ be a set of composition operations, $trans$ be a transformation operation set, $union$ be a union operation set, and $minus$ be a minus operation set. All they are subsets of the general operation set: $trans \subset o$, $union \subset o$, $minus \subset o$, $comp \subset o$; and they are non-leaf operations:

$$[C7] \forall o' \in (comp \cup trans \cup minus \cup union) \{ o' \notin lo \}$$

A composition operation is a mathematical superposition defined over role concepts as sets and SCQL subqueries as relations. A composition operation fulfills join-like transformation of nested operations: a) selects all instances, having the same values of identical role concepts, from Cartesian product of nested operations; and b) projects the result to avoid duplication of role concepts. The composition is analogous to the natural join of the relational algebra (Codd, 1972), but there is the following important distinction: composition fulfillment considers coincidence of application domain concept identities, while natural join fulfillment considers coincidence of attribute names. The natural join is not semantic as attribute names within Relational Model are not associated with application domain concepts directly. Two attributes representing one concept can have different names, and two attributes having the same name can represent different application domain concepts. At the same time, composition operations are semantic as they are based on application domain concepts directly.

Figure 4. SCQL composition fulfillment example



For example, if given two nested operations with signatures (Person, Project(Person's), Project(Task's)) and (Project(Task's), Task), then their composition has the resulting signature (Person, Project(Person's), Project(Task's), Task), as shown at the Figure 4.

The composition has three different notations, two of which, path and star notations, were discussed above; those two notations are applicable to binary associations only. There is another notation that is applicable to any subqueries and, therefore, is the most general one. Using the general notation, one composes several subqueries by enumerating them in round brackets with comma. For instance, the query “((Employee, Person), (Person, Skill Level, Skill Type), (Person, Phone))” is composition of three association selections: “(Employee, Person),” “(Person, Skill Level, Skill Type),” and “(Person, Phone).” A composition operation does not have any parameters besides a set of nested operations. Just this fact enables all the notations: general, path, and star ones.

Composition signatures contain a union of role concepts of nested operation signatures and do not include one role concept several times:

$$\begin{aligned}
 & \forall comp' \in comp \forall sign' \in sign \\
 [C8] & \left\{ \begin{array}{l} (comp', sign') \in osign \rightarrow \\ sign' = \bigcup \{ sign'' \mid \exists o' \in o \{ (o', comp') \in oo \wedge (o', sign'') \in osign \} \} \end{array} \right\}
 \end{aligned}$$

SQL join signatures can include several semantically identical columns. For instance, the following SQL query has two semantically identical columns “Person_ID”:

```
SELECT * FROM PersonTaskRel ptr, PersonProjectRel ppr
WHERE ptr.Person_ID = ppr.Person_ID AND ppr.Project_ID = 'MES',
```

while the equivalent composition “(Task–Person–Project=’MES’)” has only one column for the concept “Person” in spite of the fact that both composed associations contain the concept.

Another important property of composition operations is implicit join predicates, while, for example, SQL requires definition of join predicates in the explicit form.

Composition join predicates are constructed automatically as according to identity of role concepts. Any identical role concepts of different nested operations equal each other. For instance, the associations (Employee, Person), (Person, Phone), and (Person, Skill Level, Skill Type) were composed by the concept “Person” without an explicit predicate. Note that the query has the result signature “Employee, Person, Phone, Skill Level, Skill Type” with the single concept “Person.”

An SCQL composition operation can be outer one. In this case, all nested operations are divided into two categories: outer and non-outer. Such composition operation is executed in two stages: (a) a non-outer composition operation of all non-outer nested operations is first fulfilled; and (b) the result of the non-outer composition is then extended with all compatible instances of outer operations. The extension procedure is the following. Two instances are considered to be compatible if they have the same values for all common role concepts. Select an instance of the non-outer composition and all its compatible instances of outer-nested operations. Make a partial composition of the non-outer composition and the selected outer operations, taking into account only the selected instances. Repeating such partial composition for all instance of the non-outer composition, one creates the extension that is the result of the desired outer composition. The outer composition operation type is analogous to the SQL outer join, but it is simpler and more transparent owing to the same reasons as the non-outer composition operation type.

Outer-nested operations are marked with the plus sign right after, and a composition is outer one if it has at least one outer-nested operation. For instance, the query “((Person, Task)+, (Person–Age<30))” is the outer composition of the subquery (Person–Age<30) (see this section below for details) and the association (Person, Task). The former subquery is extended using the latter one, which is marked with the plus sign. The analogous SQL query is:

```
SELECT ptr.Task_ID, ptr.Person_ID, p.Age FROM Person p, PersonTaskRel ptr
WHERE p.Person_ID = ptr.Person_ID(+) AND p.Age < 30.
```

One can see that the SQL query uses the outer join and is more complicated than the analogous SCQL query.

Formally, the outer composition operation set *ocomp* is a subset of the general composition operation set: $ocomp \subseteq comp$; and the outer operation set *outo* is a subset of the general operation set: $outo \subseteq o$. There are several constraints imposed on outer compositions and their nested operations. First of all, each outer composition operation is to have at least one non-outer nested operation:

$$[C9] \forall ocomp' \in ocomp \exists o' \in o \{ (o', ocomp') \in oo \wedge o' \notin outo \}$$

All outer operations nested to an outer composition operation are to have common role concepts with non-outer ones:

$$[C10] \forall a \in outo \exists oc \in ocomp \exists b \in o \left\{ \begin{array}{l} (a, oc) \in oo \wedge (b, oc) \in oo \wedge \\ b \notin outo \wedge directly\_connected(a, b) \end{array} \right\}$$

where the *directly_connected* predicate is as follows:

$$\begin{aligned} \text{directly\_connected}(a \in o, b \in o) &\equiv \exists \text{sign}_a \in \text{sign} \exists \text{sign}_b \in \text{sign} \exists oc \in \text{comp} \\ &\{(a, oc) \in oo \wedge (b, oc) \in oo \wedge (a, \text{sign}_a) \in o\text{sign} \wedge (b, \text{sign}_b) \in o\text{sign} \wedge \text{sign}_a \cap \text{sign}_b \neq \emptyset\} \end{aligned}$$

The last outer composition constraint is the following: all non-outer nested operations are to form a connected graph:

$$\begin{aligned} &\forall oc \in o\text{comp} \forall a \in o \forall b \in o \\ \text{[C11]} &\{(a, oc) \in oo \wedge (b, oc) \in oo \wedge a \notin \text{outo} \wedge b \notin \text{outo} \rightarrow \text{connected}(a, b)\} \end{aligned}$$

where the *connected* predicate is as follows:

$$\begin{aligned} \text{connected}(a \in o, b \in o) &\equiv \text{directly\_connected}(a, b) \vee \exists d \in o \{ \text{directly\_connected}(a, d) \\ &\wedge \text{connected}(d, b) \} \end{aligned}$$

Also, the composition serves for filtering subqueries. For this purpose, a composition operation can have some nested operations based on logical predicates defined over role concepts. Such operations are named as logical selections. A logical selection is a leaf operation selecting all instances (combinations of role concept values) satisfying a given predicate. Consider a composition operation based on some logical selections and other operations. Derive another composition (non-filtered one) based on all nested operations of the desired composition besides logical selections. The desired composition's result contains only those instances of the non-filtered composition's result that satisfy all predicates of all the nested logical selections. So composition of logical selections and other operations results in additional filtering effect — only those instances that satisfy all predicates of the nested logical selections are kept.

Logical selections are written as logical predicates in round brackets. For instance, the query ((Task-Person-Age), (Age<30)) has the logical selection (Age<30) that contains all ages less than 30; the same query in the short notation is written as (Task-Person-Age<30). The query in both notations reads as “select all tasks being solved by persons younger than 30.” The analogous SQL query is:

```
SELECT ptr.Task_ID, ptr.Person_ID, p.Age FROM Person p, PersonTaskRel ptr
WHERE p.Person_ID = ptr.Person_ID AND p.Age < 30.
```

As one can see, the SCQL query is a lot more simple and transparent than the analogous SQL query, as the SCQL query does not apply proper association names, explicit join predicates, and has semantic signature as opposed to the SQL query.

Formally, let p be a set of predicates based on role concepts, ls be a logical selection set being subset of the general leaf operation set: $ls \subset lo$. Then each logical selection must be characterized by one predicate: $lsp : ls \rightarrow p$,

$$\text{[C12]} \quad \forall ls' \in ls \exists p' \in p \{ (ls', p') \in lsp \}$$

Any logical selection is nested to a composition operation:

$$[C13] \quad \forall ls' \in ls \exists o' \in o \{ (ls', o') \in oo \wedge o' \in comp \}$$

The logical selection is to be based on role concepts of other operations nested to the same composition and not being logical selections:

$$[C14] \quad \forall c \in comp \forall ls' \in ls \{ signature(ls') \subseteq \bigcup \{ signature(o') \mid (o', c) \in oo \wedge o' \notin ls \} \}$$

where the *signature* function is as follows:

$$signature(a \in o) \equiv \{ sign' \mid (a, sign') \in osign \}$$

Another leaf operation type is the association selection. An operation of this type is defined over a role concept set being its signature and is executed in either of two ways: (a) it selects all instances of an association based on the given concept set if the queried schema contains such association; or (b) it executes, using the current context, a shortened query that is a selection of indirect interrelations of the given concept set. A signature being a role concept set is the structure and the only parameter of the association selection operation.

The association selection was discussed in the previous section as two ways of referring to associations: directly and indirectly using contexts. The notation of the operation type is enumeration of role concepts in round brackets. For example, the enumeration (Person, Phone) is the direct selection of the appropriate association, and the enumeration (Employee, Phone) is the indirect selection of the concepts' interrelation by means of a composition-projection shortened query uncovering to (Employee–Person–Phone).(Employee, Phone) (see the projection syntax below) using the current context of the running example. See the previous section for details concerning both ways of selecting concept set interrelation.

One can see that a SCQL association selection does not use association (relation) proper names, which results in more simplicity and transparency of SCQL queries in comparison with other query languages. Formally, let *as* be an association selection set being subset of the general leaf operation set: $as \subset lo$. It is true that all leaf operations of SCQL expressions are to be either association selections or logical selections:

$$[C15] \quad \forall lo' \in lo \forall e' \in scql \{ (lo', e') \in oe \rightarrow lo' \in ls \vee lo' \in as \}$$

where the set *scql* is a subset of the general expression set: $scql \subseteq e$.

The next non-leaf SCQL operation type is the minus. An operation of this type is based on two ordered nested operations. The result of the operation consists of all instances of the first nested operation that have no compatible instances among instances of the second nested operation (see the definition of compatible instances above). The SCQL minus is more simple than the analogous operation types of other query languages since it does not require alignment of nested operation signatures. For

instance, selection of phones of persons having no “experienced” skills can be written down in SQL as:

```
SELECT Person_ID, Phone FROM Person
MINUS SELECT p.Person_ID, p.Phone FROM Person p, Skill s
WHERE p.Person_ID = s.Person_ID AND s.Level = 1 /*Experienced*/.
```

The analogous SCQL query can be written as ((Person, Phone) minus (Skill Level=”Experienced,” Skill Type, Person)). One can see that the second nested operation’s signature is not aligned to the first one as opposed to the above SQL query. The resulting signature of the minus operation is the signature of the first nested operation (Person, Phone), in spite of the fact that the second nested operation has an absolutely different signature.

Saying this formally, a minus operation has exactly two nested operations placed at one of two positions: $p \equiv \{1, 2\}$, $op : o \rightarrow p$,

[C16]

$$\forall m \in \text{minus} \exists no_1 \in o \exists no_2 \in o \left\{ \begin{array}{l} no_1 \neq no_2 \wedge (no_1, m) \in oo \wedge (no_2, m) \in oo \wedge (no_1, 1) \in op \wedge \\ (no_2, 2) \in op \wedge \{no \mid (no, m) \in oo\} = \{no_1, no_2\} \end{array} \right\}$$

The result signature of any minus operation is a signature of its first nested operation:

[C17]

$$\forall m \in \text{minus} \forall no \in o \{ (no, m) \in oo \wedge (no, 1) \in op \rightarrow \text{signature}(m) = \text{signature}(no) \}$$

The next non-leaf SCQL operation type is the union. Operations of this type have an unordered set of nested operations. The result of a union operation is calculated as follows. Select an instance of a nested operation and all its compatible instances of other nested operations (see above for instance compatibility definition). Then extend the selected instances with all other instances compatible with at least one already selected instance (the compatible instances are to be of different operations). Repeat the extension procedure while the set of selected instances is expanded. The resulting set of selected instances is a composition cluster. Calculate all existing composition clusters using all the nested operations. For each composition cluster, calculate a partial composition of nested operations included with the cluster, taking into account only those instances that are in the cluster. If the cluster includes an only nested operation, the partial composition has resulted in only that instance that is included with the cluster. The result of the union is all instances of all the partial compositions.

One can see that the union is more extensive than analogous operation types of other query languages since it does not require alignment of signatures of nested operations. Both the full outer join and the union of SQL are special cases of the SCQL union: (a) if nested operations have the same signature, the SCQL union is analogue for the SQL union; (b) if there are two nested operations with different signatures, the SCQL

union is analogous to the SQL full outer join; or (c) otherwise, SQL (and other languages) has no analogues for the SCQL union.

The SCQL union has the intuitively clear notation: the word “union” between each pair of subqueries to be united. For instance, the query “for each project select all persons being its members or solving its tasks” is written down as ((Project–Person) union (Project–Task–Person).(Project, Person)). The analogous SQL query is as follows:

```
SELECT e.Person_ID, ppr.Project_ID FROM Employee e, PersonProjectRel ppr
WHERE e.Person_ID = ppr.Person_ID
UNION SELECT e.Person_ID, t.Project_ID
FROM Employee e, Task t, PersonTaskRel ptr
WHERE e.Person_ID = ptr.Person_ID AND ptr.Task_ID = t.Task_ID.
```

It is important that SCQL union operations have simple structure since the nested operations are not to be reorganized to have the equal signatures. A union operation is executed taking into account application domain concepts completely and, as a result, it has a semantic signature being union of signatures of all nested operations:

$$[C18] \quad \forall u \in \text{union} \{ \text{signature}(u) = \{ \text{signature}(no) \mid (no, u) \in oo \} \}$$

The last non-leaf SCQL operation is the transformation. Signatures of all operations described above are completely calculated from signatures of nested operations. Transformation operations cover all cases concerned with controlled modification of a single nested operation’s signature by means of projection, grouping, and calculation of both aggregate and non-aggregate functions:

$$[C19] \quad \forall t \in \text{trans} \exists no \{ (no, t) \in oo \}$$

One defines a transformation operation by specifying a resulting signature on basis of a nested operation signature. Role concepts of the resulting signature can be of the following types: projected, calculated, and aggregated. A projected role concept is a role concept of the nested operation copied from it without any modification. A calculated role concept is a role concept that is calculated with a non-aggregate function based on role concepts of the nested operation. And an aggregated role concept is a role concept calculated with an aggregate function based on the nested operation’s role concepts that are not used as a projected role concept or for computation of a calculated role concept.

A transformation operation is calculated in the following way. Consider a transformation operation and its nested operation. For each initial instance of the nested operation, calculate a derived instance, filling it with a) the copied values of projected role concepts, and b) values of calculated role concepts computed with the specified functions from the initial instance. If the transformation operation has no aggregated role concepts, then its result is all distinct derived instances (two instances are considered to be equal if they have the same set of pairs <a role concept, its value>). Otherwise, grouping is necessary to calculate the aggregate role concepts: group initial instances by equal derived instances. For each resulting group extend its derived instance with

aggregated role concepts computed using the specified aggregate functions from all initial instances of the group. In this case, the result of the transformation operation is all the extended derived instances. One can see that a transformation operation can fulfill all the modification types: projection, calculation, and grouping. The transformation permits almost any combinations of the modification types. One can prove that any transformation operation is to have at least one role concept not being aggregated.

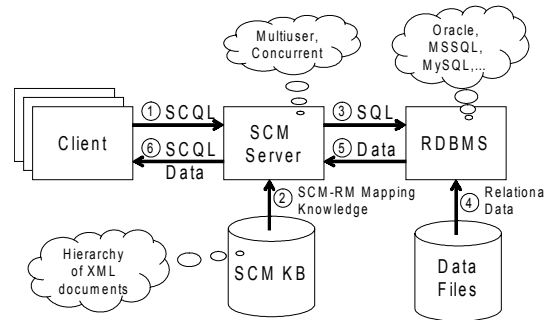
SCQL provides for two alternative notations of resulting signature definition — in round brackets after dot, and between ‘SELECT ... FROM’ words. Both ways are equivalent, but the first one is used after nested operation expression, and the second one is used before it. The way to use is determined by a user’s preferences. SCQL does not require specifying role concepts to be grouped in an explicit form. Grouping is done implicitly by role concepts not used for computing aggregate functions. For instance, the average age of persons working on a project can be calculated as (Project–Person–Age).(Project, AVG(Age)), or as (SELECT Project, AVG(Age) FROM (Project–Person–Age)). Here grouping on the concept “Project” is implicitly done because the aggregate function AVG is used. Using the context mechanism, the same query can be written down as (Project, AVG(Age)), which is very simple and transparent. The analogous SQL query is the following:

```
SELECT p.Person_ID, AVG(p.Age) FROM PersonProjectRel ppr, Person p
WHERE ppr.Person_ID = p.Person_ID GROUP BY ppr.Project_ID.
```

Table 2. Summary of SCQL operations’ structure

Operation Category	Operation Type	Nested Operations	Parameter	Resulting Signature
Non-leaf	Composition	n	(no)	Union of nested signatures
	Union	n	(no)	Union of nested signatures
	Minus	2	(no)	From the first nested operation
	Transformation	1	Resulting signature	Defined explicitly
Leaf	Logical selection	0	Predicate	Defined explicitly
	Association selection	0	Role concept set	Defined explicitly

Figure 5. SCM-based client/server technology architecture



One can see that the SQL query uses proper table names, an explicit join criterion, a grouping clause, and has no semantic signature, unlike the analogous SCQL query.

Note that signatures of all SCQL operations are unordered sets of role concepts. Signatures of all non-leaf operations, besides transformation operations, are calculated on the basis of nested operations' signatures. Only signatures of transformation operation, logical selection, and association selection operations are to be specified explicitly (see the summary of SCQL operations' structure in Table 2).

APPLICATIONS AND FUTURE TRENDS

Now SCQL is used as the foundation of SCM-based client/server technology that permits client programs to communicate with SCM servers in terms of SCQL queries. At the moment, SCM servers are backed by any existing relational database management system (RDBMS), and it will support other DBMS types in the future. The technology permits creating client/server applications interacting with end-users in terms of application domain concepts and their unique associations. Any existing RDBMS of a given version or higher can now be overbuilt with an SCM server to create SCM-based client/server applications (Ovchinnikov, 2005a).

The technology works as illustrated in Figure 5. First of all, one or several clients send some SCQL queries to an SCM server concurrently. The server processes each query in a separate thread. During processing, it uses an SCM knowledge base that is a hierarchy of XML documents describing mapping of existing relational schema to a published SCM schema. The processing results are SQL queries transmitted to an RDBMS. The RDBMS executes each SQL query and returns results back to the SCM server. The SCM server returns query execution results to appropriate clients in the SCQL form. Within the technology, clients are abstracted from data storing and can execute queries in terms of an application domain by means of SCQL.

The most emerging direction of development of the technology is creation of a data integration system that permits fusion of several heterogeneous SCM servers into one space. Any client is expected to use the space the same way as a single SCM server. The first sub-direction is finding the ways of distributed SCQL queries' execution within a

heterogeneous environment of SCM servers. The second sub-direction is application of the transaction control theory to information modification through the unified SCM interface. And another sub-direction is elaboration of methods of mapping SCM schemas to schemas of other types and SCQL queries to queries of other query languages.

The other important development trend is creation of the tool “semantic browser.” The tool is expected to solve some different tasks. The first task is building GUI applications by means of semantic-oriented components that publish its data as a derivation of a SCM schema using application domain concepts. The applications are expected to be created using mostly declarative way, and so to be suitable for end-users who are not IT specialists. The second task is integration of all information representation forms to one space with built-in support of semantic navigation between the existing forms and applications. For instance, one method of semantic navigation is the following. One selects values of a concept and requests the semantic browser to create a list of all possible transitions from the concept values. The browser analyses all existing forms, selects those to which one can move in the context of the concept, and gives the resulting list. One selects a desired form, the browser loads the form and adjusts it to show information about the selected concept values only. As a result, one has made a transition, which was not programmed in advance, within the selected concept values. A set of possible transitions is completely determined by a set of existing forms and their structures. So Semantic Browser is expected to permit semantic navigation over structured data published in the form of a SCM schema.

And the last perspective trend to be mentioned is the use of SCM as a computational independent model (CIM) of OMG model-driven architecture (OMG, 2003a). Since SCM is the most abstract model having the less possible set of implementation detail expressive means, the author believes it best suits the CIM purpose and can fill the gap of CIM models in MDA. The trend has a mass of sub-trends directed at implementation of tools serving for automation of using SCM as CIM (for instance, mapping SCM to other models, including UML (OMG, 2003b, 2004).

CONCLUSION

The Semantically Complete Query Language (SCQL) is a declarative conceptual query language not using proper relation names, based on the semantically complete model (SCM). All non-leaf operations of SCQL, besides transformations, are merely parameterized by nested operation sets and have no any additional parameters; and transformations have the only additional parameter that is a resulting signature. Signatures of all SCQL operations are unordered. Composition operations do not require join predicates in an explicit form. As a result, SCQL is more semantically transparent and simple for end-users than other conceptual and data query languages since the latter require more detailed specification of the query execution way: proper relation names, explicit join predicates, grouping fields, and other details.

As opposed to other query languages, signatures of all SCQL queries are semantic as they are sets of application domain concepts and not sequences of abstract columns not associated with concepts of an underlying application domain. Additionally, SCQL does not permit creation of some type of senseless queries. Path and star expressions are written using very intuitively clear notations.

Table 3. Summary of properties of SCQL expressions

Characteristic	Example
Application domain concept usage	(Project, Task)
Semantic nature of resulting signatures	(Person, Phone)
Concept-based query formulation (Uselessness of proper relation names)	(Person, Skill Level, Skill Type)
Capability of implicit join predicates	((Project, Task), (Task, Person))
Prohibition of senseless queries	SQL: ... WHERE a.Project_ID = b.Task_ID
Capability of queries as concept chains and stars	(Project–Task–Person) or (Task–[Project, Person])
Capability of join-like queries formulated as resulting signatures merely	Using the current context: (Employee, Phone)
Some query adaptation without rewriting	Different meaning in different contexts: (Employee, Project)

SCQL has introduced a context mechanism that permits querying some indirectly associated concepts as if they are associated directly, adopting SCQL queries to users' needs without query rewriting, and browsing a SCM schema by end-users in the more transparent way. As a result, SCQL is useful both for end-users and for IT professionals. All the properties are summarized in Table 3.

All the described properties make SCQL a truly semantic query language that is founded on application domain concepts. Its queries are easy-to-read, write, check, and translate to a natural language, and they are not concerned with implementation details at all. Other existing query languages share some of the properties to a certain extent, but none of them has all the properties in full measure.

REFERENCES

- Bakema, G., Zwart, J., & van der Lek, H. (1994). Fully communication oriented NIAM. In G. M. Nijssen & J. Sharp (Eds.), *NIAM-ISDM 1994 Conference. Working Papers*, Albuquerque, NM (pp. L1-35).
- Bloesch, A. C. & Halpin, T. A. (1997, November). Conceptual queries using ConQuer-II. In *Proceedings of ER'97: 16th International Conference on Conceptual Modeling*, Los Angeles, CA (LNCS 1331, pp. 113-126).

- Bronts, G. H. W. M., Brouwer, S. J., Martens, C. L. J., & Proper, H. A. (1995). A unifying object role modelling approach. *Information Systems*, 20(3), 213-235.
- Brouwer, S. J., Martens, C. L. J., Bronts, G. H. W. M., & Proper, H. A. (1994, July). Towards a unifying object role modelling approach. In *Proceedings of the First International Conference on Object-Role Modelling (ORM-1)*, Magnetic Island, Australia (pp. 259-273).
- Chen, P. P. S. (1976). The entity-relationship model — Towards a unified view of data. *ACM Transactions on Database Systems*, 1(1), 9-36.
- Chen, P. P. S. (1981, October 12-14). A preliminary framework for entity-relationship models. In P. P. Chen (Ed.), *Proceedings of 2nd International Conference on Entity-Relationship Approach to Information Modeling and Analysis* (pp. 19-28). North-Holland.
- Codd, E. F. (1972). *Relational completeness of data base sublanguages*. Data Base Systems, Courant Computer Science Symposia Series 6. Upper Saddle River, NJ: Prentice-Hall.
- Codd, E. F. (1979). Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems*, 4(4), 397-434.
- Dibie-Barthelemy, J., Haemmerle, O., & Loiseau, S. (2001). *Refinement of conceptual graphs* (LNAI 2120, p. 216). London: Springer-Verlag.
- Embley, D. W., Wu, H. A., Pinkston, J. S., & Czejdo, B. (1996). *OSM-QL: A calculus-based graphical query language*. Technical report. Salt Lake City, UT: Department of Computer Science, Brigham Young University.
- Halpin, T. A. (1995). *Conceptual schema and relational database design*. Sydney, Australia: Prentice-Hall.
- Halpin, T. A. (2001). *Information modeling and relational databases*. San Francisco: Morgan Kaufmann.
- Halpin, T. A. (2004). Business rule verbalization. In A. Doroshenko, T. Halpin, S. Liddle, H. Mayr (Eds.), *Proceedings of the 3rd International Conference ISTA'2004: Information Systems Technology and its Applications* (pp. 39-52). Salt Lake City, UT: GI Lecture Notes in Informatics P-48.
- Halpin, T. A., & Orlowska, M. E. (1992). Fact-oriented modelling for data analysis. *Journal of Information Systems*, 2(2), 1-23.
- ter Hofstede, A. H. M., Proper, H. A., & van der Weide. (1993). Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 18(7), 489-523.
- ter Hofstede, A. H. M., Proper, H. A., & van der Weide, Th.P. (1996). Query formulation as an information retrieval problem. *The Computer Journal*, 39, 255-274.
- ter Hofstede, A. H. M., Proper, H. A., & van der Weide, Th. P. (1997). Exploiting fact verbalisation in conceptual information modelling. *Information Systems*, 22(6/7), 349-385.
- ter Hofstede, A. H. M., & van der Weide, Th. P. (1993). Expressiveness in conceptual data modeling. *Data & Knowledge Engineering*, 10(1), 65-100.
- Nijssen, G. M., & Halpin, T. A. (1989). *Conceptual schema and relational database design: A fact oriented approach*. Sydney, Australia: Prentice-Hall.

- OMG. (2003a, June 1). *OMG Model Driven Architecture Guide V1.0.1*. [Online]. Retrieved September 1, 2005, from <http://www.omg.org/cgi-bin/doc?omg/03-06-01>
- OMG. (2003a, September 15). *OMG UML 2.0 Infrastructure Specification*. [Online]. Retrieved September 1, 2005, from <http://www.omg.org/cgi-bin/doc?ptc/2003-09-15>
- OMG. (2004). *OMG UML 2.0 Superstructure Specification*. [Online]. Retrieved September 1, 2005, from <http://www.omg.org/cgi-bin/doc?ptc/2004-10-02>
- Ovchinnikov, V. V. (2004a). A conceptual modeling technique based on semantically complete model, its applications. In A. Doroshenko, T. Halpin, S. Liddle, & H. Mayr (Eds.), *Proceedings of the 3rd International Conference ISTA'2004: Information Systems Technology and its Applications* (pp. 25-38). Salt Lake City, UT: GI Lecture Notes in Informatics P-48.
- Ovchinnikov, V. V. (2004b). A semantically complete conceptual modeling technique. *Journal of Conceptual Modeling*, 32. [Online]. Retrieved September 1, 2005, from <http://www.inconcept.com/jcm>
- Ovchinnikov, V. V. (2004c). Improving controllability of vast conceptual models. *Journal of Conceptual Modeling*, 31. [Online]. Retrieved September 1, 2005, from <http://www.inconcept.com/jcm>
- Ovchinnikov, V. V. (2005a). *SCM portal*. [Online]. Retrieved September 1, 2005, from <http://scm.lipetsk.ru>
- Ovchinnikov, V. V. (2005b). A concept-based query language not using proper relation names. In J. Castro & E. Teniente (Eds.), *Proceedings of CAiSE'05 Workshops* (Vol. 1, pp. 617-628). Porto: FEUP.
- Ovchinnikov, V. V., & Vahromeev, Y. V. (2005). A declarative concept-based query language as a mean for relational database querying. *Journal of Conceptual Modeling*, 34. [Online]. Retrieved September 1, 2005, from <http://www.inconcept.com/jcm>
- Owei, V. (2000). Natural language querying of databases: An information extraction approach in the conceptual query language. *International Journal of Human-Computer Studies*, 53(4), 439-492.
- Owei, V., & Navathe, S. (2001a). A formal basis for an abbreviated concept-based query language. *Data & Knowledge Engineering*, 36(2), 109-151.
- Owei, V., & Navathe, S. (2001b). Enriching the conceptual basis for query formulation through relationship semantics in databases. *Information Systems*, 26(6), 445-475.
- Owei, V., Navathe, S. B., & Rhee, H.-S. (2002). An abbreviated concept-based query language and its exploratory evaluation. *Journal of Systems and Software*, 63(1), 45-67.
- Seaborne, A. (2004). RDQL — A query language for RDF. *W3C Member Submission*. [Online]. Retrieved August 20, 2005, from <http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/>
- de Troyer, O. M. F. (1991). The OO-binary relationship model: A truly object-oriented conceptual model. In *Proceedings of the Third International Conference CaiSE'91 on Advanced Information Systems Engineering* (LNCS 498, pp. 561-578). Berlin: Springer-Verlag.

- van Bommel, P., ter Hofstede, A. H. M., & van der Weide, Th.P. (1991). Semantics and verification of object-role models. *Information Systems*, 16(5), 471-495.
- W3C (2004a). Resource Description Framework (RDF): Concepts and abstract Syntax. In G. Klyne, & J. J. Carroll (Eds.), *W3C Recommendation*. [Online]. Retrieved August 20, 2005, from <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>
- W3C (2004b). OWL Web Ontology Language Reference. In M. Dean, & G. Schreiber (Eds.), *W3C Recommendation*. [Online]. Retrieved August 20, 2005, from <http://www.w3.org/TR/2004/REC-owl-ref-20040210/>

ENDNOTE

- ¹ A signature implies a sequence or a set, depending on a language, of pairs <a result column, its value type>. In the context of SCQL, a signature is a set of domain concepts.

Chapter XX

Semantic Analytics in Intelligence: Applying Semantic Association Discovery to Determine Relevance of Heterogeneous Documents

Boanerges Aleman-Meza, University of Georgia, USA

Amit P. Sheth, University of Georgia, USA

Devanand Palaniswami, University of Georgia, USA

Matthew Eavenson, University of Georgia, USA

I. Budak Arpinar, University of Georgia, USA

ABSTRACT

We describe an ontological approach for determining the relevance of documents based on the underlying concept of exploiting complex semantic relationships among real-world entities. This research builds upon semantic metadata extraction and annotation, practical domain-specific ontology creation, main-memory query processing, and the notion of semantic association. A prototype application illustrates the approach by supporting the identification of insider threats for document access. In this scenario, we describe how investigative assignments performed by intelligence analysts are captured into a context of investigation by including concepts and

relationships from the ontology. A relevance measure for documents is computed using semantic analytics techniques. Additionally, a graph-based visualization component allows exploration of potential document access beyond the 'need to know'. We also discuss how a commercial product using Semantic Web technology, Semagix Freedom, is used for metadata extraction when designing and populating an ontology from heterogeneous sources.

INTRODUCTION

Creating applications that allow users to gain insightful and actionable information or mine for interesting patterns from vast amounts of heterogeneous information is one of the most exciting new areas of information systems research. This information may come from numerous sources spanning proprietary, trusted, or even open-source locations, including intranets, the deep Web and the open Web. The fast-emerging markets of business intelligence as well as national and homeland security are finding themselves in increasing need of a class of applications dealing with risk and compliance (Sheth, 2005). A representative example of this type of software is the *Insider Threat* application, which involves validating the legitimate access of sensitive documents. While physical security measures may help reduce malevolent or otherwise undesirable access to documents by employees within an organization, the development of new information-based security systems provides additional capabilities for defense against insider threat attacks. The intent of this application is to ensure that analysts who are assigned various investigative tasks access the information on a “need to know basis,” and that the system identifies access to irrelevant information in an attempt to reduce the chances that confidential information is leaked or otherwise released inappropriately.

Research into techniques for searching documents was a critical component of the first generation of the Web and has since gone from academia to mainstream. A second generation “Semantic Web” will be built by adding semantic annotations to Web content that software can utilize and from which humans can benefit. Large-scale semantic annotation of data (domain-independent or domain-specific) is now possible because of numerous advances in the areas of entity identification, automatic classification, taxonomy and ontology development, and metadata extraction (Dill et al., 2003; Hammond, Sheth, & Kochut, 2002; Shah, Finin, Joshi, Cost, & Mayfield, 2002). Relationships are at the heart of semantics (Sheth, Arpinar, & Kashyap, 2003; Woods, 1975). The next frontier, which will fundamentally change the way we acquire and use knowledge, is to automatically identify complex relationships between entities in this semantically annotated data. Instead of a search engine that merely returns documents containing terms of interest, we propose an approach that supports semantic analysis of heterogeneous content to return actionable information that gives useful insight into the connection between documents and real-world entities, thus providing superior support for important decisions and actions. Demonstrating this approach is a prototype application that supports the task of ensuring that an intelligence analyst accesses documents on a “need to know” basis, which means that only those documents relevant to the analyst’s investigative assignment should be viewed. However, this is only one of the many semantic applications needed as part of the advanced information technology pool to support homeland security.

From a research perspective, one of the challenges was to devise a framework for the formal definition and representation of meaningful and interesting relationships, which we call “semantic associations.” Semantic associations are at the core of our research in content analytics and knowledge discovery using an ontology-driven process. Other challenges arise from the enormous scale of collected metadata sets and the need for complex data structures containing entities and relationships that are used to perform queries against those sets. Finally, we utilize a notion of context to capture an analyst’s investigative assignment using an ontology. These challenges call for a fresh look at indexing, query processing, ranking, as well as tractable and scalable graph algorithms that exploit heuristics. This chapter describes a prototype application supporting the identification of insider threats for document access based on the underlying concept of exploiting semantic associations among real-world entities. Our work addresses the aforementioned challenges, building on our previous research in the following areas:

- semantic metadata extraction and annotation (Hammond, Sheth, & Kochut, 2002);
- practical domain-specific ontology creation (Aleman-Meza, Halaschek, Sheth, Arpinar, & Sannapareddy, 2004), Glyco Ontology (<http://lsdis.cs.uga.edu/Projects/Glycomics/>), and ProPreO (<http://lsdis.cs.uga.edu/projects/glycomics/propreo/>);
- semantic association definition and computation (Aleman-Meza, Halaschek, Arpinar, & Sheth, 2003; Anyanwu & Sheth, 2003; Milnor et al., 2005; Perry et al., 2005); and
- main-memory query processing (Janik & Kochut, 2005)

Extended descriptions in both technical and theoretical aspects are also provided in more detail than our previous work (Aleman-Meza, Burns, Eavenson, Palaniswami, & Sheth, 2005). In particular, we highlight the following:

- An ontological approach to capture an investigative assignment of an analyst into a *context of investigation*.
- Semantic discovery techniques that identify the relevance of documents based on the explicit relationships existing between a document and the context of investigation.
- An “inspection” visualization interface that supports exploration of the need to know of otherwise legitimate access documents.

We also discuss how a commercial product using Semantic Web technology, Semagix Freedom, which is based on SCORE technology (Sheth et al., 2002) developed in the LSDIS lab, is used for metadata extraction when designing and populating an ontology from heterogeneous sources. The ontology itself contains relevant metadata extracted from various trusted information resources including government watch-lists, sanction-lists, gazetteers, organizations, and the like.

BACKGROUND

Ontologies are at the heart of most approaches and technologies (Sheth, Arpinar, & Kashyap, 2003) that seek to realize the Semantic Web (<http://www.w3.org/2001/sw/>)

vision (Berners-Lee, Hendler, & Lassila, 2001). The Resource Description Framework (RDF) data model (Lassila & Swick, 1999) is a proposed framework to capture the meaning of an entity (or resource) by specifying how it relates to other entities (or classes of resources). In the RDF model, concepts of entities are linked together with relations (properties). The classes and/or relationships can be defined with an RDF Schema vocabulary (Brickley & Guha, 2000). The properties are denoted by arcs and labeled with the relation name. Thus, the metadata can be represented as a graph, together with a graph for the vocabulary of the classes and relationships (Karvounarakis, Alexaki, Christophides, Plexousakis, & Scholl, 2002).

A key feature needed in semantic technologies is the capability to create and maintain ontologies. Semi-automatic creation of metadata based on specific domains has been researched in the S-CREAM framework (Handschuh, Staab, & Studer, 2003), as well as other tools that have been developed (Vargas-Vera et al., 2002). An ontology populated with domain knowledge provides an important asset for applications in semantic analytics. While the schema of the ontology usually needs to be designed by a domain expert, our work shows that trusted and high-quality knowledge sources, coupled with a set of disambiguation techniques, can largely automate the process of populating domain-specific ontologies, which often have millions of instances.

Semantic annotation is referred to as both the metadata added to a document and the process of generating such metadata (Popov et al., 2003). The semantic enhancement engine (Hammond, Sheth, & Kochut, 2002) of Semagix Freedom is a tool that can provide this capability. SemTag, another annotation tool, has demonstrated a large-scale annotation capability of over one billion documents in industrial applications (Dill et al., 2003). Other annotation tools for specific domains have also been developed, such as BioAnnotator for the biomedical domain (Subramaniam et al., 2003).

Semantic Associations

The conceptual basis of the Insider Threat system is based on what we have termed semantic associations (Anyanwu & Sheth, 2003). A semantic association represents a direct or indirect relationship between two entities. “Semantics” here specifically involves those relations that are meaningful to the application and can be inferred either based on the data itself or with the help of additional knowledge. Semantic associations are meaningful and relevant complex relationships between entities, events, and concepts. They lend meaning to information, making it understandable and actionable, and provide new and possibly unexpected insights. Different entities can be related in multiple ways. For example, a professor can be related to a university, student, course, or publication; but s/he can also be related to other entities by relations not necessarily connected to the primary domain to which the entity belongs, like hobbies, religion, or politics. Relationships that span several entities may be very important in domains such as national security, because they may enable analysts to see the connections between seemingly disparate people, places, or events.

Semantic associations are based on intuitive notions such as connectivity and semantic similarity. Different semantic associations in an RDF graph have been formally defined in our previous work (Anyanwu & Sheth, 2003). Here, we present a simplified definition of semantic associations:

Definition 1 (*Semantic Association*): Two entities e_1 and e_n are *semantically associated* if there exists a sequence $e_1, p_1, e_2, p_2, e_3, \dots, e_{n-1}, p_{n-1}, e_n$ in an RDF graph where $e_i, 1 \leq i \leq n$, are entities and $p_j, 1 \leq j < n$, are properties.

Semantic associations have proven to be a foundational layer in real-world applications, being most useful in the area of homeland security such as Passenger Threat Assessment (Sheth et al., 2005). Additionally, semantic associations have been used in the retrieval of biomedical patents (Mukherjea & Bamba, 2004), knowledge discovery and composition in peer-to-peer networks (Aleman-Meza, Halaschek, & Arpinar, 2005; Perry et al., 2005), and geospatial semantic analytics (Arpinar et al., 2004). Ranking of semantic associations has also been addressed (Aleman-Meza, Halaschek-Wiener, Arpinar, Ramakrishnan, & Sheth, 2005; Aleman-Meza, Halaschek, Arpinar, & Sheth, 2003; Anyanwu, Sheth, & Maduko, 2005), as well as efficient algorithms that focus on performance, scalability, and efficiency (Janik & Kochut, 2005; Milnor et al., 2005). Measures of the credibility of semantic associations from multiple sources have also been proposed (Ding et al., 2005).

Document Access Problem of Insider Threat

In the context of the intelligence community, one of many security aspects involves that of detecting “malevolent actions by an already trusted person with access to sensitive information and information systems” (Anderson & Brackney, 2004, p. xi). For document access, the goal is to ensure that an analyst only accesses documents on a need to know basis. Typically, data about an analyst’s activities are analyzed after the fact, reactively rather than proactively. This may be due to a culture of trust, but more often it has to do with the prohibitive costs of creating/defining methods to detect malevolent actions, as well as their implementation and maintenance. There are various techniques that can be applied to determine if a collection of documents is relevant to a given domain. Some of these techniques exploit statistical, natural language-processing, machine-learning, document-clustering, or document-classification techniques. These, however, are typically referred to as implicit semantics (Sheth, Ramakrishnan, & Thomas, 2005) because they cannot or do not name specific relationships among concepts or entities.

One related approach uses a list of positive and negative examples to generate a set of weight vectors that determine the permission of each document for an analyst (Rectenwald, Lee, Seo, Giampapa, & Sycara, 2004). When an analyst selects a document, the authorization agent determines whether it is viewable to the analyst based on the generated weight vectors. However, these techniques typically do not provide reasons why a document is or is not relevant to the investigative objective of the analyst. Similarly, these techniques have either limited or nonexistent support for exploiting named relationships between concepts (e.g., an organization is located in a country). Our strategy includes the use of an ontology to capture the semantics of the domain to process-named relationships, both for identifying the relevance of documents and for providing a visualization on why and how a document is related to the investigation objective of the analyst (i.e., for auditing purposes).

The document access problem of Insider Threat also includes the “misuse” of documents. Misuse detection systems have previously been built by creating a user

profile based on legitimate queries to access information (Cathey, Ma, Goharian, & Grossman, 2003). Subsequent queries are then compared against the user profile to detect misuse.

Traditional data mining (Chen, Han, & Yu, 1996; Fayyad, Piatetsky-Shapiro, Smyth, & Uthurusamy, 1996) has mainly focused on discovering patterns and relationships out of their repetition in data. However, data mining has also been applied for misuse detection in order to eliminate manual adjustment of weights on the different levels of misuse (Ma, Goharian, & Meyers, 2005).

DESCRIPTION OF THE APPROACH

Overview of our System for Insider Threat Document

Our prototypical system demonstrates a workflow involving a supervisor and an analyst performing the following tasks:

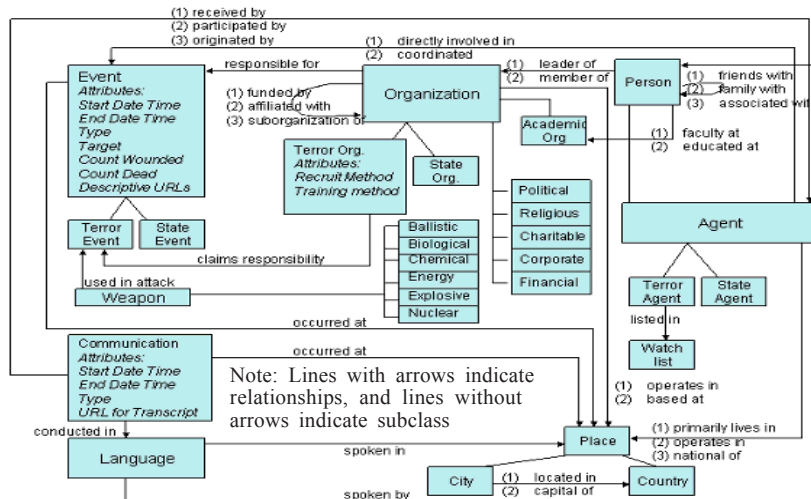
- The supervisor specifies an assignment for an analyst.
- The supervisor specifies (and modifies) a context of investigation for the assignment.
- The analyst performs tasks related to the assignment. As part of this, the analyst accesses various heterogeneous documents using a system that can keep track of the documents that were viewed.
- The supervisor can verify if the documents accessed by the analyst are within the context of investigation specified for the assignment. The system analyzes the relevance of the documents and ranks them accordingly.

Ontology Specification and Development

As part of the ongoing Semantic Discovery project at the LSDIS lab, we have created and are maintaining a test-bed ontology (SWETO) for evaluating ontological management and semantic technologies (Aleman-Meza, Halaschek, Sheth, Arpinar, & Sannapareddy, 2004). SWETO contains an ontology schema covering various domains, and is populated using factual data or knowledge from multiple sources. To serve the purposes of the document access problem of Insider Threat, we refined a part of the SWETO schema to sufficiently capture the domain of National Security and Terrorism to meet our prototyping and evaluation goals. A schematic of this part of the ontology is provided in Figure 1.

This ontology provides a conceptualization of organizations, countries, people, watch lists, terrorists, events, terrorist acts, and so on, that are all interrelated by named relationships to reflect real-world knowledge about the domain (i.e., “terrorist” belongs to “terrorist organization”). The sources used to populate the ontology were selected for their information richness, semi-structured format, and aptitude to quickly populate the ontology with a large number of entities, and, more importantly, relationships in the domain of terrorism. An example would be publicly available data maintained by intelligence agencies and international organizations, such as watch lists containing publicly declared “bad” persons and organizations. Ontology design and population

Figure 1. National security and terrorism part of SWETO ontology



was accomplished by using Semagix's Freedom, a commercial software that itself is based on technology developed at and licensed from the LSDIS lab (Sheth et al., 2002). The same technology was used to build the LSDIS Lab's Glycomics ontology (<http://lsdis.cs.uga.edu/Projects/Glycomics/>). Other large scale ontologies have also been developed elsewhere with domains outside of national security. For example, TAP (Guha & McCool, 2003) is an ontology about places, musicians, sports, movies, and so on.

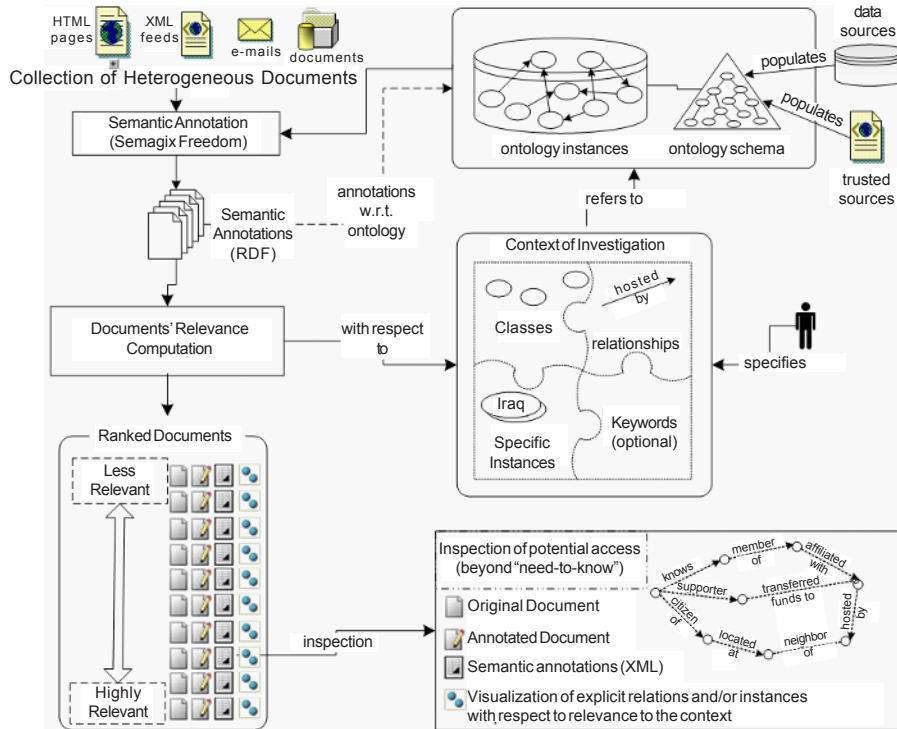
The ontology schema and populated instances were exported from Freedom and modeled in RDF. The part of SWETO used by the methods described in this chapter consists of about 40 classes in the schema part of the ontology; the instances consist of about 32,000 entities and approximately 35,000 explicit relationships.

Ontological Approach to the Document Access Problem of Insider Threat

Figure 2 provides a schematic view of our approach. The first step utilizes a large ontology populated from trusted sources to semantically annotate a collection of documents. A context of investigation can be defined by a supervisor to capture (in ontological terms) the scope of an investigation assignment given to an intelligence analyst.

The main processing involves computing a measure of the relevance of each document (using the annotations) with respect to the context. Given that a collection of documents needs to be processed, additional technical challenges include the need to compute a potentially large number of semantic associations per document and use them to measure their relevance with respect to the context.

Figure 2. Ontological approach to the legitimate document access problem



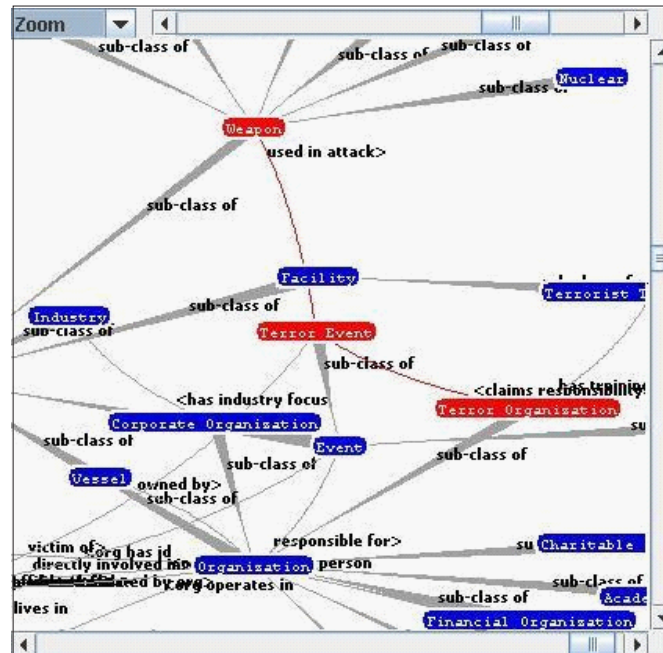
Finally, the documents are ranked according to the computed relevance measure. Each document can be viewed/inspected to gain insight on the intention of access by the analyst beyond the “need to know.” It has also been noted that “finding relationships among suspects is vital in law enforcement applications” (To Catch a Thief, 2004, p. 3). A graphical user interface (accessible in the form of a Java Applet) displays the semantic associations that interconnect entities within a document to those that form part of the context of investigation. Only the relationships regarded as relevant in the given context are displayed. Such semantic associations form the basis of identifying connections between two or more seemingly unrelated entities.

Context of Investigation

Based on our previous work (Aleman-Meza, Halaschek, Arpinar, & Sheth, 2003) we define context as follows:

Definition 2 (Context): A context is a non-empty set of entities, relationships, and/or classes from an ontology.

Figure 3. Graphical interface for defining a context of investigation

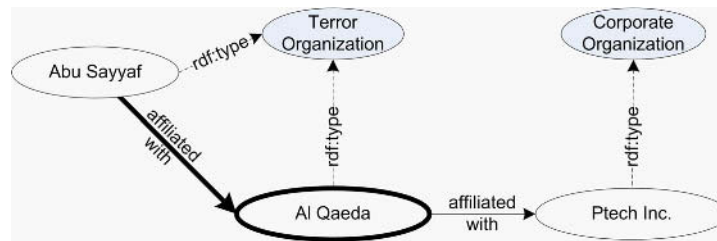


The intuition is that a context captures a set of entity types, relationships, and specific entity instances (at an ontology level) that are to be considered relevant (for example, in a query). In the case of the legitimate document access problem of Insider Threat, the context is used to capture the investigative assignment given to an intelligence analyst. As previously stated, we refer to this as the context of investigation, which is a combination of the following:

- A set of classes and/or relationships, and a negation of a set of classes and relationships;
- A set of entity instance names, and/or a negation of a set of entity instance names; and
- A set of keyword values that might appear at any attribute of the populated instance data, and/or a negation of a set of keyword values.

Related research has mentioned an “ontological foundation” for context, yet did not provide a means of expressing context with respect to an ontology (Coutaz, Crowley, Dobson, & Garlan, 2005). One of the key components of the system is a user interface providing a means for graph-based creation of a context of investigation. We expanded upon our previous prototype for capturing the context of a user’s interest with respect to an ontology (Halaschek, Aleman-Meza, Arpinar, & Sheth, 2004) and implemented a graphical user interface. This was done by extending a version of the TouchGraph (<http://www.touchgraph.com>) applet to display graphs. Figure 3 displays an example of a

Figure 4. Context constraint of a specific relation-entity combination



context of investigation where the classes “Terror Organization,” “Terror Event,” and “Weapon” have been added to the context. In addition, the context can be further defined in order to specify a more rigid set of semantic constraints. For example, it can be specified that a relation “affiliated with” is part of the context only when it is connected with an entity that belongs to a specific class, say, “Terror Organization.” Figure 4 illustrates this example by highlighting with a thick line the combination of a sample entity and a relationship that fits the context. (The gray nodes represent classes of the ontology; the “rdf:type” relation indicates the class type of an entity.)

Semantic Annotation

The documents viewed by an analyst are processed to generate a set of semantically annotated documents. As stated before, semantic annotation is referred to as both the metadata added to a document and the process of generating such metadata. We utilized Semagix’s Freedom software to semantically enhance a set of documents that an analyst could possibly access as part of an assignment. The Freedom software annotates a document by passing it through the semantic enhancement server module (Hammond, Sheth, & Kochut, 2002). Entity names or synonyms within the document that are contained in the ontology are recognized. The output of the semantic annotation is an XML document listing the identified entities; an “enhanced” version of the original document is also produced by highlighting recognized entities. A fragment of a semantically annotated document is provided in Figure 5 (both in XML and with highlighting of recognized entities in the original document).

Relevance Measure for Documents

Measuring the relevance of annotated documents with respect to the context of investigation is intended to help a supervisor determine whether the work of the analyst on a particular assignment poses an Insider Threat. At a high level, a relevance-engine module takes as input the set of semantically annotated documents (accessed by the intelligence analyst as part of his/her investigative assignment), the context of investigation for the assignment, and the ontology population and schema represented in RDF. The engine discovers semantic associations among the set of entities that were found in the annotated document and the entity classes, entity instances, and/or keywords specified in the context of investigation. The discovery algorithm traverses the RDF

Figure 5. Fragment of a semantically annotated document



Abu Musab al-Zarqawi

From Wikipedia, the free encyclopedia.

Abu Musab al-Zarqawi in one of eight photos from **Rewards for Justice**, all undated

Abu Musab al-Zarqawi (Arabic: **أبو مصعب الزرقاوي**) (possibly born on **date**(October 20), 1966) is a shadowy **Jordanian** national who is wanted as an international **terrorist**. He is from the town of **Zarqa**, a poor and crime-ridden industrial town **time**(30 minutes); northeast of **Amman**. One **alias**, **Ahmad Fadeel al-Nazal al-Khalayleh** (Arabic: **أحمد فاضل النزال الخلاله**), is believed to be his real name [1] (http://news.bbc.co.uk/1/hi/world/middle_east/2730253.stm) [2] (http://news.bbc.co.uk/2/hi/middle_east/1483089.stm) [3] (http://www.atimes.com/latimes/Middle_East/PC11Ak01.html). As a suspected **Islamic militant**, Zarqawi is believed to be violently opposed to the presence of **U.S.**, **Israelis** and allied military forces in the **Islamic** world.

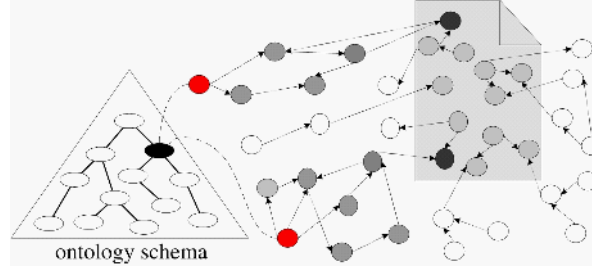
In personal accounts Zarqawi is usually described as somber and unintelligent, with a violent temper. He is alleged to be senior **al-Qaida** associate of **Osama bin Laden**. **U.S. Secretary of State Colin Powell** described Zarqawi as an "al-Qaida operative." Several U.S. military officials have described him as a "separate phantasm." Zarqawi has allegedly participated in violent actions against the **United States** military in **Iraq** and against a U.S. diplomat in Jordan. As a result, the U.S. government is offering a **US\$ money** (US\$3/25 million reward for information leading to his capture, the same amount offered for the capture of **Osama bin Laden** before **date**(March 2004)). An emerging view holds that Zarqawi now holds significantly more power than **bin Laden** because of Zarqawi's heightened visibility as a leader of the insurgency against the U.S. military and Iraq interim government. On **date**(October 21, 2004), Zarqawi officially announced his allegiance to **al-Qaida**, on **date**(December 27, 2004), **Al-Jazeera** broadcast an

```
<?xml version="1.0"?>
<entity id="167776">
  <attributes>
    <attribute name="synonym">Abu Mus</attribute>
    <attribute name="synonym">Ahmad Fadil Nazal Al-Khalayleh</attribute>
    <attribute name="date_of_birth">30 Oct. 1966</attribute>
    <attribute name="synonym">ABU AL-MUTAZ</attribute>
    <attribute name="synonym">Al-Muhojer</attribute>
    <attribute name="synonym">Abu Mus'ab Al Zarqawi</attribute>
    <attribute name="synonym">KHALAYLEH, Fedel Nazzel</attribute>
    <attribute name="entityIdFromKnowledgeModeler">167776</attribute>
    <attribute name="synonym">AL-MUHAJIR</attribute>
    <attribute name="synonym">Abu Musab al-Zarqawi</attribute>
    <attribute name="synonym">KHALAILAH, Ahmad Fadel</attribute>
    <attribute name="place_of_birth">Al Zarqa, Jordan</attribute>
    <attribute name="synonym">Ahmad Fadel al-Nazal al-Khalayleh</attribute>
    <attribute name="national_id">National ID No. 9661031030 Jordan</attribute>
    <attribute name="synonym">AL-KHALAYLAH, Ahmad Fadil Nazzel</attribute>
    <attribute name="passport_no">Z264968 Jordan</attribute>
    <attribute name="address">Whereabouts currently unknown</attribute>
    <attribute name="synonym">Abu Musab Al-Zarqawi</attribute>
    <attribute name="name">Abu Musab Al-Zarqawi</attribute>
    <attribute name="level">0</attribute>
  </attributes>
  <classifications>
    <entityClass>Terror_Agent</entityClass>
  </classifications>
  <relationships>
    <relationshipGroup>
      <name>agent_operates_in</name>
      <relationship>
        <otherEntity id="162164" position="2" />
      </relationship>
    </relationshipGroup>
    <relationshipGroup>
      <name>national_of</name>
      <relationship>
        <otherEntity id="162176" position="2" />
      </relationship>
    </relationshipGroup>
    <relationshipGroup>
      <name>member_of</name>
      <relationship>

```

graph searching for semantic associations up to a sequence of (predefined) length k (set to 9 by default). In order to perform this task of semantic analytics, we build upon previous work on discovery of semantic associations. We extend the definition of r -operators for semantic associations (Anyanwu & Sheth, 2003) to introduce a r_k operator for expressing queries for semantic associations using context.

Definition 3 (p_k -Query): A p_k -Query, expressed as $p_k(x, c)$, where x is an entity and c is a context, results in the set of all semantic associations that exist between x and c . A semantic association between x and c exists if there is a sequence $e_1, p_1, e_2, p_2, e_3, \dots$,

Figure 6. ρ_k -Query from entities within a document and a context

e_{n-1}, p_{n-1}, e_n in an RDF graph where $e_1 = x$ and $e_i, 1 \leq i \leq n$, are entities and $p_j, 1 \leq j < n$, are properties, and either $e_n \in c$ or $\text{type}(e_n) \in c$ or $p_i \in c$, where $\text{type}(e)$ is the class type (or concept) of entity e . Figure 6 illustrates an example of a ρ_k -Query.

Once the semantic associations among entities within a document and the context of investigation have been discovered, the relevance of a document d with respect to a context of investigation CI is computed as follows:

$$\text{Relevance}(d) = C_{CI} + R_{CI} + E_{CI} + K_{CI} \quad (1)$$

where, C_{CI} is the component of matching classes with respect to the context of investigation, CI . Similarly, R_{CI} , E_{CI} and K_{CI} are the components for matching relations, entities, and keywords, respectively, with respect to the context of investigation. The discovered semantic associations interconnecting a document to a context can be seen as a neighborhood of k hops — similar to the intuition of a “semantic neighborhood” (Rodriguez & Egenhofer, 2003).

Each of the components in Equation 1 is computed based on the proximity of a match of the types of the entities of the document and its neighborhood with respect to the context of investigation. The computation of C_{CI} is as follows:

$$C_{CI} = \frac{\sum_{e_j \in d} \left[\sum_{i=1}^{|ng(e_j)|} \frac{1}{\text{dist}(e_j, v_i) + 1} \right]}{|d|} \quad (2)$$

where, $ng(e)$ is the set of nodes and relations in the neighborhood of entity e ; and the function $\text{dist}(e, v)$ computes the distance between e and v . For the particular case of the component for keywords, K_{CI} , the formula also considers all attributes of each entity v_i with those keywords specified in the context of investigation. As part of future work, we plan to incorporate into the formula K_{CI} which is a simplified version of that introduced by a hybrid search approach (Rocha, Schwabe, & Aragao, 2004).

entities in the document to the context. A subset of 100 documents was carefully chosen to detect four cases of relevance: (1) directly related documents; (2) strongly related documents; (3) loosely related documents; or (4) non-related documents. Cases at the extremes were easily verified to work correctly (1,4). However, the strongly and loosely related cases required inspection and analysis by human to identify why a document has a medium to high ranking or medium to low ranking. Figure 7 illustrates the color pattern used to group documents according to their score value (instead of rank position).

We present the following examples on why documents get a high or low score with respect to a given context of investigation. A document on the terror organization Hizballah — for which the algorithm was able to establish the following three semantic associations: Hamas -*operates in* → Middle East, Al Qaeda -*operates in* → Middle East, and Hizballah -*operates in* → Middle East — received a high score of 0.915.

A document on the terror organization Jemaah Islamiyah — for which the algorithm was able to establish the single longer semantic association: Abu Sayyaf Group -*affiliated with* → Al Qaeda -*operates in* → Middle East — received a lower score of 0.735.

A document on the terror organization Liberation Tigers of Tamil Eelam (LTTE) — for which the algorithm was able to establish a few long semantic associations such as: Rajiv Gandhi -*victim of* → May 21, 1991 LTTE Suicide Bomb -*located in* → India ← -*national of* -Dawood Ibrahim -*affiliated with* → Al Qaeda -*operates in* → Middle East — received a very low score of 0.273. Additionally, an online demonstration of the application is available (<http://lsdis.cs.uga.edu/Projects/SemDis/NeedToKnow/>).

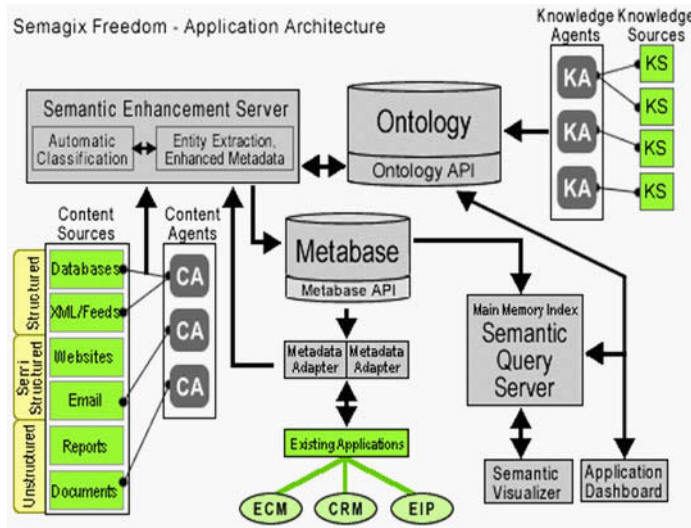
Semagix Freedom

Semagix Freedom is built around the concept of ontology-driven metadata extraction, allowing modeling of fact-based, domain-specific relationships between entities. It provides tools that enable automation in every step in the content chain — specifically ontology design, content aggregation, knowledge aggregation and creation, metadata extraction, content tagging, and querying of content and knowledge. Figure 8 shows the domain-model driven architecture of Semagix Freedom.

Semagix Freedom operates on top of a domain-specific ontology that has classes, entities, attributes, relationships, a domain vocabulary, and factual knowledge, all connected via a semantic network. The domain-specific information architecture is dynamically updated to reflect changes in the environment, while being easy to configure and maintain. The Freedom ontology maintains knowledge, which is any factual, real-world information about a domain in the form of entities, attributes, and relationships (e.g., Figure 1). The ontology forms the basis of semantic processing, including automated categorization, conceptualization, cataloging, and enhancement of content. Freedom provides a modeling tool to design the ontology schema (the assertional component of the system) based on the application requirements. Specifically, it allows flexible designing of the domain model by offering features like definition of customized entity types, relationships between entity types, entity attributes, cardinality constraints, class membership, and so on.

The ontology is automatically maintained by knowledge agents. These are software agents that traverse trusted knowledge sources and exploit structure to extract useful

Figure 8. Semagix Freedom architecture



entities and relationships for populating the ontology automatically. Once created, they can be scheduled to perform knowledge extraction automatically at any desired interval, thus keeping the ontology up-to-date.

Freedom also aggregates structured, semi-structured, and unstructured content from any source and format, by extracting syntactic and contextually relevant semantic metadata. Content agents extract useful syntactic and semantic metadata information from content and tag it automatically with pre-defined metatags. Incoming content is further “enhanced” by passing it through the semantic enhancement server module (Hammond, Sheth, & Kochut, 2002).

The Metabase stores both semantic and syntactic metadata related to content in either custom formats or one or more defined multiple metadata formats such as RDF, PRISM, Dublin Core, and SCORM. The Metabase stores content into a relational database as well as a main-memory checkpoint. At any point in time, a snapshot of the Metabase (index) resides in main memory (RAM), so that retrieval of entities is accelerated using the patented semantic query server.

The semantic query server is a main memory-based front-end query server that enables the end-user to retrieve relevant content. A variety of semantic applications that exploit this technology can be built, including anti money laundering identification and risk assessment (*Anti Money Laundering*, 2003), financial analyst workbench, homeland security, and citizen portal applications. The semantic enhancement and query servers operate on the Metabase and ontology; they yield high-quality query results because they provide the basis for in-context querying, whereas common search engines lack context and ambiguity resolution and, therefore, relevance and accuracy.

CONCLUSION

This chapter discussed a challenging problem of detecting illegitimate access of documents beyond the need-to-know, one of the problems of Insider Threat. The approach involved processing of documents to produce semantic annotations and then used the semantic annotations to measure the relevance of a document with respect to the investigative assignment of an intelligence analyst. This measure computes an aggregated score of a set of semantic associations, while the notion of context is defined to capture the scope of the assignment. A graphical representation of the ontology is used within a graphical user interface to specify the context, and a new semantic association query is introduced to query for semantic associations among an entity and a context. The prototype application is described by discussing the technical challenges involving this type of text and content analytics. The prototype also provides an interface for inspection of the explicit relations that make a document relevant to an intelligence analyst's investigative assignment. Thus, it provides insight to a supervisor of why the analyst may or may not need to know the contents of a particular document, and whether such access should be deemed illegitimate.

This effort illustrates a unique attempt of driving research from a realistic application, core research issues in semantic association discovery, and use of commercial Semantic Web technology in building a populated ontology over publicly available data. This research demonstrates an example of collaboration involving academic research, industry technology, and government priorities, to address unique and technically demanding challenges.

ACKNOWLEDGMENTS

We thank Semagix, Inc. for providing access to Freedom, which is based on the SCORE technology and related research performed at the LSDIS Lab. The work on semantic associations is funded in part by National Science Foundation (NSF) Award IIS-0325464 ("SemDis: Discovering Complex Relationships in Semantic Web"). Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF. The insider-threat prototype was developed as part of the Advanced Research Development Activity (ARDA) (<http://www.ic-arda.org>) Insider Threat Initiative, contracted through the Department of the Interior, Ft. Huachuca, Contract #NBCHC030083.

REFERENCES

- Aleman-Meza, B., Burns, P., Eavenson, M., Palaniswami, D., & Sheth, A. P. (2005, May 19-20). An ontological approach to the document access problem of insider threat. In *Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI-2005)*, Atlanta, GA (pp. 485-491). Springer.
- Aleman-Meza, B., Halaschek-Wiener, C., Arpinar, I. B., Ramakrishnan, C., & Sheth, A. P. (2005). Ranking complex relationships on the semantic Web. *IEEE Internet Computing*, 9(3), 37-44.

- Aleman-Meza, B., Halaschek, C., & Arpinar, I. B. (2005). Collective knowledge composition in a peer-to-peer network. In L. C. Rivero, J. H. Doorn, & V. E. Ferragine (Eds.), *Encyclopedia of database technologies and applications* (pp. 74-77). Hershey, PA: Idea Group Reference.
- Aleman-Meza, B., Halaschek, C., Arpinar, I. B., & Sheth, A. (2003, September 7-8). *Context-aware semantic association ranking*. Paper presented at the First International Workshop on Semantic Web and Databases, Berlin, Germany.
- Aleman-Meza, B., Halaschek, C., Sheth, A., Arpinar, I. B., & Sannapareddy, G. (2004, June 21-24). *SWETO: Large-scale semantic Web test-bed*. Paper presented at the 16th International Conference on Software Engineering and Knowledge Engineering (SEKE2004): Workshop on Ontology in Action, Banff, Canada.
- Anderson, R., & Brackney, R. (2004). *Understanding the insider threat*. Rockville, MD: RAND Corporation.
- Anti money laundering*. (2003). White paper. Semagix, Inc. Retrieved from http://www.semagix.com/documents/anti_money_laundering.pdf
- Anyanwu, K., & Sheth, A. P. (2003, May 20-24). *r-Queries: Enabling querying for semantic associations on the semantic Web*. Paper presented at the 12th International World Wide Web Conference, Budapest, Hungary.
- Anyanwu, K., Sheth, A. P., & Maduko, A. (2005, May 10-14). *SemRank: Ranking complex relationship search results on the semantic Web*. Paper presented at the 14th International World Wide Web Conference, Chiba Japan.
- Arpinar, I. B., Sheth, A. P., Ramakrishnan, C., Usery, E. L., Azami, M., & Kwan, M.-P. (2004). Geospatial ontology development and semantic analytics. In J. P. Wilson & A. S. Fotheringham (Eds.), *Handbook of geographic information science*. Cambridge, MA: Blackwell Publishing. Retrieved from <http://lsdis.cs.uga.edu/lib/download/ASRU+2004-gis.pdf>
- Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic Web — A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 284(5), 34+.
- Brickley, D., & Guha, R. V. (2000). RDF vocabulary description language 1.0: RDF schema. W3C Recommendation. Retrieved May 12, 2005, from <http://www.w3.org/TR/2003/PR-rdf-schema-20031215/>
- Cathey, R., Ma, L., Goharian, N., & Grossman, D. (2003, November 2-8). Misuse detection for information retrieval systems. In *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management*, New Orleans, LA (pp. 183-190). New York: ACM Press.
- Chen, M. S., Han, J. W., & Yu, P. S. (1996). Data mining: An overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8(6), 866-883.
- Coutaz, J., Crowley, J. L., Dobson, S., & Garlan, D. (2005). Context is key. *Communications of the ACM*, 48(3), 49-53.
- Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R. V., Jhingran, A., et al. (2003, May 20-24). *SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation*. Paper presented at the Twelfth International World Wide Web Conference, Budapest, Hungary.

- Ding, L., Kolari, P., Finin, T., Joshi, A., Peng, Y., & Yesha, Y. (2005, March 21-23). *On homeland security and the semantic Web: A provenance and trust aware inference framework*. Paper presented at the AAAI Spring Symposium on AI Technologies for Homeland Security, Stanford University, California, USA.
- Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (1996). *Advances in knowledge discovery and data mining*. Menlo Park, CA: AAAI/MIT Press.
- Guha, R. V., & McCool, R. (2003). TAP: A semantic Web test-bed. *Journal of Web Semantics*, 1(1), 81-87.
- Halaschek, C., Aleman-Meza, B., Arpinar, I. B., & Sheth, A. P. (2004, August 30-September 3). *Discovering and ranking semantic associations over a large RDF metadata*. Paper presented at the 30th International Conference on Very Large Data Bases, Toronto, Canada.
- Hammond, B., Sheth, A., & Kochut, K. (2002). Semantic enhancement engine: A modular document enhancement platform for semantic applications over heterogeneous content. In V. Kashyap & L. Shklar (Eds.), *Real world semantic Web applications* (pp. 29-49). Amsterdam, The Netherlands: IOS Press, Inc.
- Handschuh, S., Staab, S., & Studer, R. (2003). Leveraging metadata creation for the semantic web with CREAM. *Ki 2003. Advances in Artificial Intelligence*, 2821, 19-33.
- Janik, M., & Kochut, K. (2005, November 6-10). *BRAHMS: A workbench RDF store and high performance memory system for semantic association discovery*. Paper presented at the 4th International Semantic Web Conference, Galway, Ireland.
- Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D., & Scholl, M. (2002, May 7-11). *RQL: A declarative query language for RDF*. Paper presented at The Eleventh International World Wide Web Conference, Honolulu, HI, USA.
- Krebs, V. (2002). Mapping networks of terrorist cells. *Connections*, 24(3), 43-52.
- Lassila, O., & Swick, R. R. (1999). *Resource description framework (RDF) model and syntax specification*. W3C Recommendation. Retrieved May 12, 2005, from <http://www.w3.org/TR/REC-rdf-syntax/>
- Ma, L., Goharian, N., & Meyers, C. (2005, May 19-20). *Detecting misuse of information retrieval systems using data mining techniques*. Paper presented at the IEEE International Conference on Intelligence and Security Informatics (ISI-2005), Atlanta, GA, USA.
- Milnor, W. H., Ramakrishnan, C., Perry, M., Sheth, A. P., Miller, J. A., & Kochut, K. J. (2005). *Discovering informative subgraphs in RDF graphs* (Tech. Rep. No. 05-001). Athens, GA: LSDIS Lab, Computer Science, University of Georgia. Retrieved from <http://lsdis.cs.uga.edu/library/download/PID-RXSPQSVR-1114806461.pdf>
- Mukherjea, S., & Bamba, B. (2004, August 30-September 3). *BioPatentMiner: An information retrieval system for biomedical patents*. Paper presented at the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada.
- Perry, M., Janik, M., Ramakrishnan, C., Ibanez, C., Arpinar, I. B., & Sheth, A. P. (2005, July 17). *Peer-to-peer discovery of semantic associations*. Paper presented at the 2nd International Workshop on Peer-to-Peer Knowledge Management (P2PKM), La Jolla, CA.
- Popov, B., Kiryakov, A., Ognyanoff, D., Manov, D., Kirilov, A., & Goranov, M. (2003, October 20-23). *KIM — Semantic annotation platform*. Paper presented at the 2nd International Semantic Web Conference (ISWC2003), Sanibel Island, FL.

- Rectenwald, M., Lee, K., Seo, Y., Giampapa, J. A., & Sycara, K. (2004). *Proof of concept system for automatically determining need-to-know access privileges: Installation notes and user guide* (No. CMU-RI-TR-04-56). Pittsburgh, PA: Robotics Institute, Carnegie Mellon University.
- Rocha, C., Schwabe, D., & Aragao, M. P. (2004, May 17-20). *A hybrid approach for searching in the semantic Web*. Paper presented at the 13th International World Wide Web, New York.
- Rodriguez, M. A., & Egenhofer, M. J. (2003). Determining semantic similarity among entity classes from different ontologies. *IEEE Transactions on Knowledge and Data Engineering*, 15(2), 442-456.
- Shah, U., Finin, T., Joshi, A., Cost, R. S., & Mayfield, J. (2002, November 4-9). *Information retrieval on the semantic Web*. Paper presented at the 10th International Conference on Information and Knowledge Management, McLean, VA.
- Sheth, A. P. (2005, August 25-27). *Enterprise applications of semantic Web: The sweet spot of risk and compliance*. Paper presented at the IFIP International Conference on Industrial Applications of Semantic Web, Jyväskylä, Finland.
- Sheth, A. P., Aleman-Meza, B., Arpinar, I. B., Halaschek, C., Ramakrishnan, C., Bertram, C., et al. (2005). Semantic association identification and knowledge discovery for national security applications. *Journal of Database Management*, 16(1), 33-53.
- Sheth, A. P., Arpinar, I. B., & Kashyap, V. (2003). Relationships at the heart of semantic Web: Modeling, discovering and exploiting complex semantic relationships. In M. Nikravesh, B. Azvin, R. Yager & L. A. Zadeh (Eds.), *Enhancing the power of the Internet studies in fuzziness and soft computing* (pp. 63-94). Berlin: Springer-Verlag.
- Sheth, A. P., Bertram, C., Avant, D., Hammond, B., Kochut, K., & Warke, Y. (2002). Managing semantic content for the Web. *IEEE Internet Computing*, 6(4), 80-87.
- Sheth, A. P., Ramakrishnan, C., & Thomas, C. (2005). Semantics for the semantic web: The implicit, the formal and the powerful. *International Journal on Semantic Web and Information Systems*, 1(1), 1-18.
- Subramaniam, L. V., Mukherjee, S., Kankar, P., Srivastava, B., Batra, V. S., Kamesam, P. V., et al. (2003, November 2-8). *Information extraction from biomedical literature: methodology, evaluation and an application*. Paper presented at the 2003 ACM CIKM International Conference on Information and Knowledge Management, New Orleans, LA.
- To catch a thief*. (2004). Concept White Paper. Visual Analytics Inc. Retrieved September 28, 2004, from <http://www.visualanalytics.com/whitepaper/HowToCatchAThief.cfm>
- Vargas-Vera, M., Motta, E., Domingue, J., Lanzoni, M., Stutt, A., & Ciravegna, F. (2002, October 1-4). *MnM: Ontology driven semi-automatic and automatic support for semantic markup*. Paper presented at the 13th International Conference on Knowledge Engineering and Management (EKAW 2002), Sigüenza, Spain.
- Woods, W. (1975). What's in a link: Foundations for semantic networks. In D. Bobrow & A. Collins (Eds.), *Representation and Understanding*, (pp. 35-82). New York: Academic Press.

Chapter XXI

Semantic Integration in Multidatabase Systems: How Much Can We Integrate?

Te-Wei Wang, University of Illinois, USA

Kenneth E. Murphy, Willamette University, USA

ABSTRACT

This chapter reviews briefly the semantic integration issues in multidatabase development and provides a standardized representation for classifying semantic conflicts. We begin by summarizing the methods and issues in multidatabase design. From the perspective of database integration, we identify that semantic conflict is the main issue. We explore the idea further by examining semantic conflicts and propose taxonomy to classify semantic conflicts in different groups. This taxonomy is then evaluated by two different methods. Finally, we conclude by discussing the limits of database integration and how this challenge may be addressed.

INTRODUCTION

Semantic heterogeneity or semantic conflict is the main source of problems in multidatabase design. In this chapter, a brief review of previous work in semantic conflict identification is presented, which leads to the creation of a taxonomy for resolving conflicts in multidatabase design that is more inclusive when compared to existing frameworks, for example, that of Batini, Lenzerini, and Navathe (1986). A metadata

structure, based on this taxonomy, will be proposed that can be used as a point of reference (a common protocol) for semantic conflict resolution.

For the last three decades, multidatabase research has focused on resolving the problem of semantic heterogeneity or semantic conflicts. Semantic heterogeneity is often present in multidatabase systems because of the lack of global schema definition. The situation is similar to common misunderstandings that occur in everyday interpersonal communication. Misunderstandings can result between two people who speak different languages. They cannot understand one another unless interpreters are present. Even when interpreters are used, concepts that cannot be precisely translated remain. In fact, the level of shared understanding between the parties after communication depends heavily on the knowledge of the interpreters. Even if persons participating in the conversation are speaking the same language, misunderstandings can persist due to the ambiguity of language or the quality of the original information. Based on this analogy, it is apparent that not all semantic conflicts can be systematically resolved. We argue that a good conflict resolution system should have a data structure to separate resolvable and irresolvable conflicts. Given such a data structure, corresponding procedures can be created to integrate results from different data sources and to report inconsistencies that cannot be resolved.

To create multidatabases without semantic conflicts, a significant amount of research has focused on schema integration at the conceptual level (Lim & Chiang, 2000). Real-world examples such as the Cyc knowledge base (Collet, Huhns, & Shen, 1991; Deaton et al., 2005; Masters, 2002; Reed & Lenat, 2002; Singh et al., 1997) and the CORDS multidatabase (Barrowman & Martin., 1998; Martin & Powley, 1997) all use similar schema-integration concepts to provide multidatabase systems with an integrated view at the logical level. However, in practice, semantic conflicts exist not only at the logical or conceptual level, but also at the instance or run-time level. Therefore, in practice, many conflict resolutions may need to be performed at query run-time (Lonski, 1997). To facilitate run-time semantic conflict resolution, the integration engine should have the ability to construct consistent metadata at run-time. In this chapter, a data structure for the purpose of capturing the metadata generated by such a run-time integration problem is proposed. We organized this chapter by addressing the following questions in sequence:

- What is a multidatabase system?
- What are the methods currently used in multidatabase systems to resolve semantic heterogeneity?
- What are different types of semantic heterogeneity?
- Can a “better” taxonomy for classifying semantic heterogeneity be found resulting in a meta-data structure to assist in addressing semantic conflicts?
- Is the meta-data structure proposed sufficient for practice?

WHAT IS A MULTIDATABASE SYSTEM?

Typically, a database system is designed to address an organization’s needs at a fixed point in time. Organizations, however, have information needs that are dynamic. The original design of a particular database system can soon be and often is quickly outdated.

This situation is inevitable in fast-growing or multi-site organizations where different subunits have developed their own database systems. When the greater organization requires integrated data, problems emerge because database systems cannot “talk to” or query one another directly. A mediator is required to coordinate the communication and/or data exchange process. The design and implementation of such a mediator is an essential component of so-called multidatabase systems.

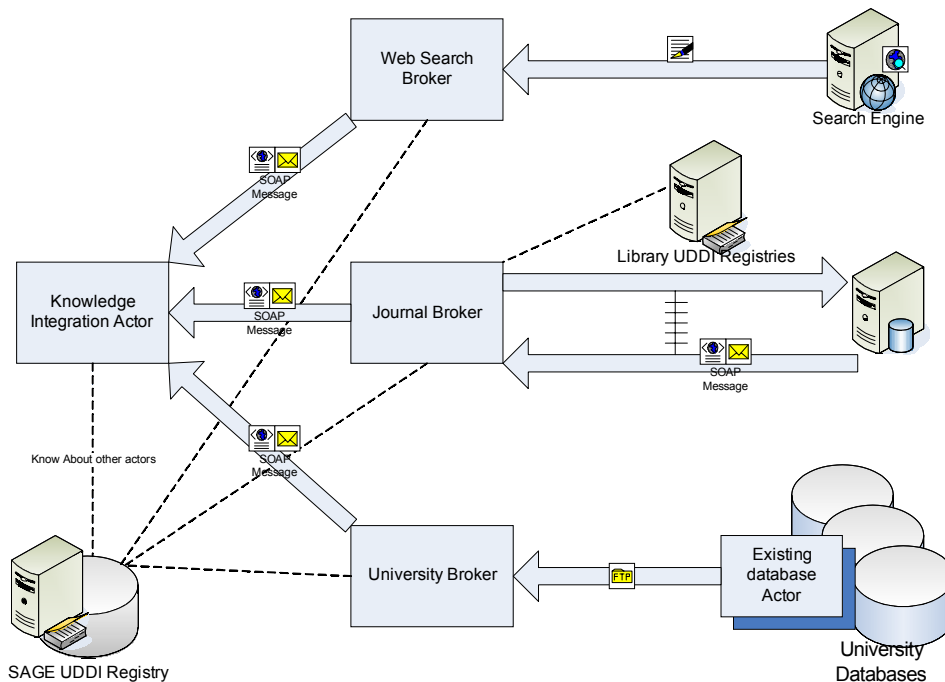
A multidatabase system provides integrated access to heterogeneous, autonomous local databases (Bright, Hurson, & Pakzad, 1994). It resides unobtrusively on top of existing database systems and presents the illusion of a single database to its users (Dogac, Dengi, & Oszu, 1998). Even though the concept of multidatabase systems has been around for several decades, the development of multidatabase systems still faces many obstacles. One obstacle is the semantic heterogeneity or semantic conflict between different database systems (Deaton et al., 2005; Lonski, 1997). The main source of these conflicts is the different data abstraction/representation mechanism used by different designers (Linthicum, 2004; Lonski, 1997). The presence of the many data modeling methodologies makes the problem even worse. Database design methodologies have evolved from those based on hierarchical models to the currently dominant relational database design. Object-oriented database models have now also appeared on the commercial database market. However, not every organization changes its database system whenever a “better” or “newer” system is available. In fact, many large organizations are still using database systems that are more than 25 years old (Miller, Yu, & Nilakanta, 2002). Database systems that are based on different designs increase the complexity of the schema integration problem.

Although there is no consensus, the term “multidatabase system” usually refers to a distributed information-sharing system. Unlike the term distributed database, a multidatabase system always implies the integration of heterogeneous database systems. In the literature, multidatabase systems are also called federated database systems, multidatabase language systems, and interoperable systems. The basic requirements/assumptions for a multidatabase system include the following (Bright, Hurson, & Pakzad, 1992; Grandi, 1998):

1. The local data base management systems (DBMSs) have independent metadata and exist before joining the multidatabase system.
2. The local DBMS should participate in the multidatabase system with little or no modifications.
3. The local DBMS retains autonomy with full control over local data and processes.

Based on the above definition, many modern knowledge management systems can be considered multidatabase systems. These systems may have been designed under the concept of service-oriented system, agent-oriented systems, or portal-oriented systems (Linthicum, 2004). Nonetheless, these terms may indicate database designers’ architecture choices but fail to signify how databases are integrated. To clarify the method of database integration, Lim and Chiang (2000) articulate the concept of coupling the local DBMSs to build a multidatabase by introducing two important dimensions of database integration: schema vs. instance and physical vs. virtual. Schema integration refers to the integration of metadata at the time the multidatabase is designed, while instance integration refers to the process of merging query results (from several local databases)

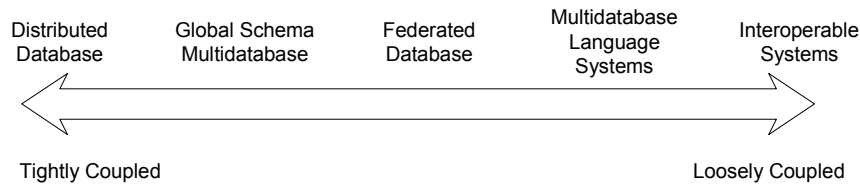
Figure 1. The SAGE expert locator system



at run-time. In addition, a database can be integrated physically or virtually. Physical integration leaves no atomicity to local systems, while virtual integration gives the local system full control of its own operations. According to Lim and Chiang, the main difference between physical integration and virtual integration is the timing of database integration at both the schema and instance levels. A physically integrated database achieves both schema and instance integration at the time of construction. Virtually integrated databases, on the other hand, leave each local database untouched and integrate only the query results. Purely physically integrated systems are rare in practice. Most multidatabase systems generate the global schema at the time of construction and attempt to resolve instance integration problems at run time (Lim & Chiang, 2000; Scheuermann, Li, & Clifton, 1998).

An example of such a "hybrid" design can be found in the National Aeronautics and Space Administration's (NASA) Searchable Answer Generated Environment (SAGE) Expert Finder system (Becerra-Fernandez, Gonzalez, & Sabherwal, 2004; Becerra-Fernandez, Wang, & Agha, 2005) (see Figure 1). The original design of SAGE creates an integrated database schema to merge expert databases from many public universities in Florida. Over time, the SAGE system increased its functionality by allowing Web search results to be used as valid information. Consequently, the global schema is not longer comprehensive enough to incorporate all search results. Therefore, the modified SAGE design adopts the concept of actor system to enable real-time search result translation.

Figure 2. Bright, Hurson, and Pakzad's (1994) continuum for multidatabase systems



The distinction between the different types of multidatabase systems can be seen clearly in the taxonomy provided by Bright, Hurson, and Pakzad (1992). Their classification scheme is shown as a continuum in Figure 2.

At one extreme is the Distributed Database system, which is the most tightly integrated, while the Interoperable system is on the other extreme (Ozsu & Valduriez, 1991). A distributed database is a homogeneous database system that is physically integrated at the schema level. In this case, the problem of semantic conflicts has been solved at the stage of global schema construction. Any new node added to the distributed system must follow the predefined terms in the global schemata. Unlike the distributed database, the semantic heterogeneity present in existing local database systems causes more problems in designing all other systems illustrated in Figure 2.

Henceforth, the multidatabase system referred in this chapter excludes the distributed database but includes global schema multidatabase, federated database (Sheth & Larson, 1990), multidatabase language systems (Bright et al., 1994), and various interoperable systems (Becerra-Fernandez et al., 2004; Bright et al., 1994; Linthicum, 2004; Liu, Eker, & Jamneck, 2004). Therefore, a working definition of multidatabase system is one in which there are local autonomous databases that require an integrated view for the display of query results submitted by end-users.

METHODS OF MULTIDATABASE SYSTEMS DESIGN

Many multidatabase integration methods have been introduced in the past 20 years (Batini et al., 1986; Lim & Chiang, 2000; Reddy, Prasad, Reddy, & Gupta, 1994). In general, there are two major categories into which these can be assigned: the global-schema (metadata) method and the multidatabase language method (Bright et al., 1992, 1994). The global-schema approach is also known as the schema-integration approach (Reddy et al., 1994), while the language system approach is also called the translation approach (Pitoura & Bukhres, 1995). The two approaches (schema-integration and language system) differ in their focus on the level of integration. In general, global schema approaches focus on schema integration, while the translation approaches focus on the integration of query results at run-time. Also, the multidatabase language method requires a certain amount of artificial intelligence to support the translation.

The global-schema approach is an extension of distributed database design that uses a schema integration process. This class of methods uses the independently

developed local schema as inputs and resolves semantic and syntactic differences by creating an integrated summary of all information from all the local schema (Bright et al., 1994). Semantic conflicts are resolved by building a global knowledge base. This knowledge base contains all the necessary definitions for terms collected from local data repositories or data dictionaries. It is not hard to envision that this process is highly labor-intensive. To complete this process, the global-schema designer must resolve all semantic problems at the schema level. Furthermore, once completed, the global schema and its associated system are rigid, complex, and inflexible. Whenever a new database joins the system, the global-schema must be extended to accommodate that database in the new system.

On the other hand, the development of multidatabase language system is a “service-oriented” method or “agent-oriented” method (Siegel et al., 2004). No global (integrated) schema is available to such systems, and hence all query results are integrated at run-time; that is, if any semantic conflict requires resolution, it will be resolved at run-time and at the instance level. A “translation agent” or a “query parser” is usually responsible to provide such service (Lim & Chiang, 2000; Ram & Ramesh, 1999). Many multidatabase language systems exist that can process imprecise query operations and handle vague meanings. In contrast to the precise semantic matches that global-schema systems try to achieve, the multidatabase language systems give users the option of finding approximate matches. For instance, the Cyc systems encourage users to engage in question and answer during the querying process (Curtis, Matthews, & Baxter, 2005). This technique relieves some of the burden of completely anticipating and solving semantic conflicts that the global schema designers face. Many concepts developed in the multidatabase language system literature have been incorporated in the Internet-based database search engines or knowledge management systems. In these systems, functions and tools to interpret conflicts are provided to the end-user at the interface level. The users need to develop their own strategy to search for the data required and for resolving the resulting semantic problems. A multidatabase language system has, in this sense, additional flexibility as compared to a global-schema system, but it persists in having the problems of data duplication, inefficiency, and imprecise data.

Most multidatabase system designs utilize the advantages of both design methods. Some of the systems are designed with an emphasis on global-schema techniques, and others are designed mostly using language system methods. For example, the resource integration project developed on the Carnot architecture used both methods but emphasized global schema techniques (Collet et al., 1991). On the other hand, the summary schemes model (Bright et al., 1994) provides the user with a tool to address the challenge of imprecise query processing.

Regardless of how a multidatabase design resolves semantic heterogeneity, a deconstruction/reconstruction process is used by almost all multidatabase designs. For instance, Sciore, Siegel, and Rosenthal (1994) suggest a two-step database integration process: (1) the building of metadata (global schema), and, (2) the conversion of concept using metadata (Becker, Gibson, & Leist, 1996). The first step in Sciore et al.’s method is to deconstruct the local database schema, followed by the reconstruction of a global schema. The second step is to present the query results at the run-time or instance level. Lim and Chiang (2000), in the study of relational database integration, have suggested a more complex integration process. This method has four major tasks: database reverse engineering (deconstruction), schema integration (reconstruction), instance exporta-

tion (deconstruction), and instance integration (reconstruction) (Lim & Chiang, 2000; Ram & Ramesh, 1999).

More recently, W3C is initiating an audacious attempt to create a framework, called Semantic Web, allowing users to integrate data across various sources. The goal of Semantic Web is to create a universal medium so that all data can be easily shared on the Internet. To achieve this, many standards, including methods of defining data, methods of sharing data, and methods of transmitting data need to be defined (Hendler, Berners-Lee, & Miller, 2002). In other words, the Semantic Web initiative is attempting to construct a global schema on how to define schema in databases, services, and other Internet resources (Hendler et al., 2002). Based on the above three examples and a method proposed by Batina et al. (1986), a generic, data structure-independent, destruction/reconstruction process is proposed as the foundation to develop the taxonomy for semantic conflicts. This approach is described in sequence below.

1. Utilizing the existing repository or data dictionary in the local database, a standard representation of local database schemata should be defined for each database. This standard representation of semantic components should be designed so that the similarity and conflict can be easily derived.
2. Compare the standardized representative formats to identify the conflicts and similarities between local databases at the schema level.
3. The conflicts should be classified into two groups: those that require resolution, and those that do not require resolution.
4. At the instance level, all the existing conversion rules should be listed. These rules will be used to resolve conflicts when they occur at the instance level.
5. Where needed, new functions, conversion rules, and/or strategies should be developed to address those conflicts for which there is no existing conversion rule.
6. Potential future conflicts should be anticipated and strategies for addressing these conflicts should be employed.

THE CLASSIFICATION OF SEMANTIC MEANINGS

An essential requirement of the above approach is a standard representation for database schema (Becker et al., 1996). Note that this standard representation is not the standard language or standard communication protocol. Languages like XML have already established the infrastructure for communication across the Internet. Most work in this area is limited to representing relational database structures (Becker et al., 1996; Lim & Chiang, 2000). For instance, Becker et al. use ISO/IEC9075 (also known as SQL-92) as their standard representation format, which is a relational database schema. Grandi (1998), in a study on temporal database integration, also utilizes the same standard. In this chapter, a standard representation based on the classification of semantic meanings is proposed, and this can be used for merging all types of data models.

In the schema integration literature, the classification method is trying to capture all the relevant information for any “concept” that is used in a database system (Bright et al., 1992). For example, when the term “Salary” is used in a local database, a multidatabase designer would have several questions: Does the term represent monthly or annual salary (naming information)? Is the unit of measurement U.S. dollar or Japanese

Yen (domain information)? Does this term represent an entity or an attribute (structure information)? Without precisely knowing the relevant context information, the multidatabase system cannot be properly designed.

Which dimensions should be identified for the component schema? The proposal here is that the categories of classification should be identified through all possible sources of semantic conflicts. Some commonly identified sources from the literature are discussed here.

Naming

Naming refers to the semantic relationship between object, attribute, or instance name. These relationships include synonyms, homonyms (Batini et al., 1986; Miller, 1995; Naiman & Ouksel, 1995) and unrelated terms (Naiman & Ouksel, 1995).

- **Synonyms:** Synonyms are terms that represent similar concepts. For example, the term “ID” may represent a student identification number in one database while the term “Student_ID” is used in another database. “ID” and “Student_ID” are known as synonyms.
- **Homonyms:** The same name is used for two different concepts. For example, “school” may be used to represent a university in one database while it may represent a high school in another database.
- **Unrelated terms:** Terms that are not related to one another directly, that is, they are neither synonyms nor homonyms. For example, the two terms, “school” and “ID,” in the previous two examples are not directly related. These are different concepts and will relate to each other only through externally defined business rules.

Structure

Structural conflicts arise as a result of a different choice of modeling constructs or integrity constraints (Batini et al., 1986). Many differences between terms were identified in the literature. These conflicts include: type differences, abstraction differences, dependency conflicts, modeling conflicts, and cardinality conflicts, among others (Batini et al., 1986; Reddy et al., 1994).

- **Type differences:** Type differences arise when the same concept is represented by different modeling constructs in different schemas. For example, the concept “Skill” may be represented as an entity in one database while being represented as an attribute in another database.
- **Abstraction differences:** Abstraction is a process of simplifying complex things (Coad & Yourdon, 1991). Consider a relational database table as an example; the selection of attributes would be an abstraction process. In this case, abstraction differences refer to different sets of attributes for the same concept selected by different database designers. For instance, in one database, a “Student” table may have three attributes. While in another database, a table may contain more than three attributes but represent exactly the same meaning.
- **Dependency (cardinality) conflicts:** When a group of concepts are related among themselves through different dependencies in different local schemas. For example, the one-to-many relationship between “Managers” and “Employees” may

apply in one organization, whereas the relationship may be one-to-one in another organization.

- **Key conflicts:** Different keys are assigned to the same concept in different schemas. For example, “Student_ID” and “Social_Security_Number” could both be utilized as the primary key for the entity “Student.”
- **Behavioral (integrity) conflicts:** When different updating rules or computation functions are placed on the same concept, behavioral conflicts arise. For example, in one database, a client record cannot exist without the assignment of an employee in one database, while in a second database this referential integrity constraint need not be enforced.

Identification and Domain

The concept of identification and domain conflicts overlap but are not encompassed by context differences (Madnick, 1995). Other terms including attribute value conflict (Lim & Chiang, 2000) have also been used to describe this class of conflicts. The term “domain conflict” will be employed here, because the terminology “context conflicts” has been used to describe other concepts (Sciore et al., 1994). Domain conflicts include data type differences, scale differences, instance identification conflicts (Madnick, 1995; Pitoura & Bukhres, 1995), and instance data value conflicts (Dayal & Hwang, 1984; Pitoura & Bukhres, 1995).

- **Data type differences:** Data type differences arise when different data types are used for the same concept. For example, the data type for “Student_ID_Number” could be Integer in one database while the same attribute could be defined as a String in another database.
- **Scale differences:** Even when the same data type is used in different databases, the same value could still mean the same thing. Madnick (1995) uses the example of currency to illustrate this conflict. The commonly seen attribute Price, for instance, has a numerical value, which could be U.S. dollars in one database but Japanese Yen in another.
- **Instance identification conflicts:** The problem of instance identification conflicts is very similar to the naming conflict. Naming conflicts exist in the level of schema integration. The instance identification conflicts, as the name suggests, exist in the instance level. Suppose that there is an object (instance name) “US” in one database that represents the United States. In a second database, an object name “USA” could represent the same concept.
- **Data value conflicts:** Data value conflicts arise when the same object instances have different values (Linthicum, 2004). For example, assume that there are two object instances both called “Employee_Phone_Number” in two different databases and that these objects both represent the same person’s phone number. In this case, the domain, naming, and structure of these two objects are exactly identical. However, the value of the phone numbers stored in these two object instances could be different. This difference at the instance level is known as a data value conflict.

Table 1. Classification of possible conflicts in multidatabase system design

Dimensions		Identifiable Components
Schema Conflicts	Naming	Homonyms, Synonyms
	Structural	Type, Dependency, Key, Behavior
Domain Conflicts		Data Type, Data Scale, Data Range
Identification Conflicts		Instance Identification, Instance Value

All the identified semantic conflicts discussed are summarized in Table 1. Similar taxonomies can be found in Pitoura and Bukhres (1995), Daul and Hwang (1984), and Kim and Seo (1991). Since Table 1 synthesizes all of the known research on semantic heterogeneity, the proposed taxonomy is more inclusive than previous models.

The naming and structural conflicts identified in the literature focus on the method of schema integration. The third and fourth dimensions (Domain and Identification) proposed here indicate that the building of a multidatabase system requires more than just schema integration when considering the challenge of integration at the instance level (Lim & Chiang, 2000). Conflicts at the instance level often require convergence rules or assertions to assist in the process of resolution.

INTER-SCHEMA PROPERTIES AND SEMANTIC RELATIONSHIPS

Understanding semantic conflicts alone is not enough to develop a representation for either schema integration or instance integration purposes. The classification of conflicts provides information only about the similarities and differences between semantic components. Additional information on the nature of relationships between different components is also necessary for constructing multidatabase systems. This information is sometimes easily identifiable, especially at the schema integration level. Batini et al. (1986) defined this type of information Inter-schema Properties, and Naiman and Ouksel (1995) termed it Abstraction Relationships.

Inter-schema properties are semantic relationships that hold between sets of objects or components in two schema (Pitoura & Bukhres, 1995). Inter-schema relationships that arise when both of the associated data models are relational have been extensively studied. The following five inter-schema relationships are most commonly seen in the literature (Miller, Yu, & Nilakanta, 2002; Naiman & Ouksel, 1995; Pitoura & Bukhres, 1995).

- **Aggregation:** Aggregation arises when a component in one database maps to a group of components in another database. For example, the entity “Car” in a relational database may be an aggregation of many different classes of cars such as subcompact, compact, full size, and so forth.
- **Generalization:** Generalization occurs when some components are subtypes of other, more general components. The ISA relationships discussed in the entity-relationship (ER) modeling technique are of this kind.
- **Inheritance:** Inheritance is the property that, when entity types or object classes are arranged in a hierarchy, each entity type or object class assumes the attributes and methods of its ancestors. The concept of “Person” is an example of an ancestor of the concept of “Employee.”
- **Functions and Methods:** Functions are used to construct relationships. A “Total_Salary” function can utilize information from “Employees,” “Monthly_Salary,” and “Deduction_Rate” to compute the net annual salary that an employee receives. The “Total_Salary” function links these three components and creates new information based on them. The function relationship has often been ignored in the literature. Due to the growing importance of object-oriented databases, the function- and method-defined relationships should be included in any inter-schema relationship taxonomy.
- **Arbitrary:** The arbitrary rule is meant to be a specification rule that can account for any type of relationship between two database objects; hence it encompasses the previous four relationships. It is meant to account for possible conflicts that may arise from business rules that are at odds in an organization. These are rules specific to a given database environment. For example, in a certain unit of an organization, a technical manual may only be used by a specific group of persons, whereas in another unit it is available to all employees.

Although the inter-schema relationships are often defined at the schema level, many of these concepts can also be applied to components at the instance level. In addition, the relationships discussed above for the schema level — semantic relationships modeled at instance level — should include specification of domain information for the purpose of domain conflict resolution. Sciore et al. (1994) presents an example of the semantic relationship at the instance level that they call semantic value. A semantic value is defined as the association of a context with a simple value, for example, the result of a specific Internet search on Google is assigned a match probability or index. The association of the context of the search and the values assigned to search results define a relationship at the instance level that did not exist a priori.

Identifying the relationships at the instance level is very difficult (Scheuermann et al., 1998). Nonetheless, it is very important for data exchange at highly complex multidatabase environments such as the “Webbed Database” (Nordbotten & Nordbotten, 2002), which is a multidatabase that is connected only virtually through the Internet and utilizes the World Wide Web as its interface. Many efforts to identify semantic relationships have been carried out by linguistic researches. As mentioned earlier, the W3C Semantic Web initiative is gaining visibility, and companies like IBM and Adobe have started the implementation effort (Miller, 2004). The concept of semantic distance,

for example, has assigned a value to each pair of words for the distance from one another. Most of this research is not directly related to that of this chapter; however, it could be fruitful in the future with respect to describing instance-level relationships.

A STANDARDIZED REPRESENTATION FOR CONFLICT RESOLUTION

In order to identify the conflicts and relationships described in this chapter, a well-defined, structured representation for the terms used in database design is indispensable. Data dictionaries present in local databases are often semantically incompatible. A well-defined representation standard is not only a tool for identifying the semantic conflicts, but also a foundation for constructing a usable semantic model in real time at the instance level for multidatabase systems. As stated previously with respect to the deconstruction/reconstruction approach, every semantic component involved in the multidatabase system should be analyzed. Following this, the analyzed results should be represented in certain ways so that all the concepts can be reconstructed to an integrated semantic model. In other words, it is desirable to build a standardized representation for the data dictionaries across all local data repositories.

The standardized representation should be compatible with the data modeling method used for semantic reconstruction. Many current representation formats have evolved from the entity-relationship (ER) modeling concept, for example, the unifying semantic model proposed by Ram (1995). The unifying semantic model defines semantic components according to object/entity types and relationship types. The clear definition of each element makes the reconstruction of the unifying semantic model possible.

Although the extended concepts of ER modeling are often used in defining the elements in multidatabase systems, it is more appealing to use object-oriented modeling to represent semantic meaning. The reason for this is that object-oriented models are more inclusive for the purpose of complexity management (Pitoura & Bukhres, 1995).

Figure 3. Standardized representation of schema

Item Name
{
Syntax Dimension:
Syntax sense: (Value Function)
Syntax Definition: Definition 1, Definition 2, Definition 3
Structure Dimension
Item Source: Name of the database
Item Type: (Object, Entity, Attribute, Instance)
Horizontal Links: Related item(s) at the same Item source and the same item type
Vertical Links: Related items from different item sources or item types (Parents, Offsprings).
Domain Dimension:
Data Type: Item's data type
Measured Unit: Unit used to measure this item
Constraint or Range: Possible range of the measure
}

Therefore, one such representation format is proposed here that is based on a format similar to an object-oriented language like C++. It is also compatible with newer data representation protocols such as XML. The format is fundamentally an extension of Cardelli's (1990) approach of defining object types. The format is shown in Figure 3.

The definition format consists of three dimensions, which xtensively cover the information needed to identify semantic meanings. Each line in this structured format provides one facet of semantic meaning for the item.

- **Item name:** Item name is the symbolic that is used in the database.
- **Syntax sense:** Syntax sense is either value or functions. Value represents the concept of a "thing." Function is an action. For example, the term "XXX_Info" could represent some information for XXX in one setting. It could also be some kind of calculation function in another setting. Usually, a function will act upon one or more values.
- **Syntax definition:** A syntax definition is the data dictionary definition of the syntax of the item. An item contained in several local databases could have more than one definition.
- **Item source:** The item source is the location from which the item was drawn. This would usually refer to a particular local database.
- **Item type:** This is the role of a specific item as defined in a local database. For example, in a relational database, an item can be a table name or an attribute name.
- **Horizontal links:** This line lists all other elements that have the same "conceptual level" in a particular setting.
- **Vertical links:** This line lists all items related to the current item but not at the same conceptual level. Vertical links could link the item to a higher, more general concept or to a lower, more specific concept. Therefore, this list can be divided into two parts: parents and offspring.
- **Data type:** The domain definition for data type. For example, the data type could be string, integer, date, and so on.
- **Measurement unit:** The unit used for the data type. For a numerical data type, it could be U.S. dollar, meters, or any other unit.
- **Constraint and range:** Any constraint that might apply on the item.

Figure 4. An example of an entity-relationship diagram

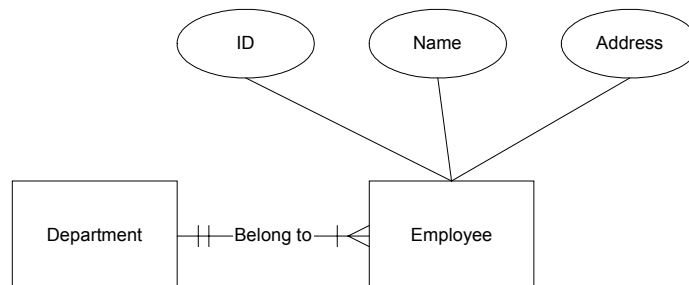


Figure 5. The standard representation for the example shown in Figure 4

```

EMPLOYEE
{
  Syntax Dimension:
    Syntax sense: Value
    Syntax Definition: Employee Record for xyz Company
  Structural Dimension:
    Item Source: xyz Company Database
    Item Type: Entity
    Horizontal Links: DEPARTMENT
    Vertical Links: Offspring-Name, Employee ID, Address
  Domain Dimension:
    Data Type: Concept
    Measured Unit: None
    Constraint or Range: None
}

Name
{
  Syntax Dimension:
    Syntax sense: Value
    Syntax Definition: Name of an Employee in xyz Company
  Structural Dimension:
    Item Source: xyz Company Relational Database
    Item Type: Attribute
    Horizontal Links: Employee ID, Address
    Vertical Links: Parents-EMPLOYEE
  Domain Dimension:
    Data Type: String
    Measured Unit: String Length in number of characters
    Constraint or Range: Smaller than 30
}

Employee_ID
{
  Syntax Dimension:
    Syntax sense: Value
    Syntax Definition: Employee ID number in xyz Company
  Structural Dimension:
    Item Source: xyz Company Relational Database
    Item Type: Attribute
    Horizontal Links: Name, Address
    Vertical Links: Parents-EMPLOYEE
  Domain Dimension:
    Data Type: String
    Measured Unit: Length of String
    Constraint or Range: Length fixed at 10 characters
}

Address
{
  Syntax Dimension:
    Syntax sense: Value
    Syntax Definition: Employee ID number in xyz Company
  Structural Dimension:
    Item Source: xyz Company Relational Database
    Item Type: Attribute
    Horizontal Links: Name, Employee_ID
    Vertical Links: Parents-EMPLOYEE
  Domain Dimension:
    Data Type: String
    Measured Unit: Length of String
    Constraint or Range: Length smaller than 100 characters
}

```


Figure 6. An example of an assertion rule representation

```

Assert(item 1, item 2)
{
  Syntax Statement;
  Structure Statement;
  Domain Statement;
}

```

An example will illustrate the use of this format. Figure 4 shows an entity EMPLOYEE that is commonly defined in a relational database. The entity EMPLOYEE can be broken into four elements: the Entity block EMPLOYEE, attribute Employee ID, and Name and Address. Each of these four elements can be written as shown in Figure 5.

In these formatted representations, the words written in normal font provide a structural format and the words written in italic font are identified by the database designer.

IDENTIFYING SEMANTIC CONFLICT/ SIMILARITY, AND RELATIONSHIPS

Once the formatted representation is at hand, the deconstruction process is complete. The relevant items have been broken down into semantic atoms and molecules. These atoms can then be used to reconstruct a global semantic model or as a foundation for instance level integration. Before this reconstruction can commence, the relationships and characteristic between these atoms and molecules should be identified; that is, the conflicts and relationships between every pair of items should be identified.

The most commonly used format to represent these conflicts and relationships is in the form of assertion statements. Following the format of the “item definition,” the format of an assertion statement is shown in Figure 6.

The detail about how to determine the three statements in the proposed assertion format is beyond the scope of this chapter. No examples or detailed illustration will be shown here. However, similar assertion formats can be seen in Naiman and Ouksel (1995). An intelligent algorithm utilizing the assertion method can be seen in Krishnan, Li, Steier, and Zhao’s work (2001).

A BRIEF EVALUATION OF PROPOSED REPRESENTATION STRUCTURE

How do we evaluate the quality of the representation presented here? Batini, Lenzerini, and Navathe (1986) suggest three criteria for evaluating a global conceptual schema. Although the application of the proposed classification method is not limited to global schema building, these three criteria are still valid in this context.

- **Criterion 1:** Completeness and correctness: The method should be able to address all concepts represented in any local database correctly. If the resulting output product is a global schema, the integrated schema must be a representation of the union of the application domains associated with the local schemas.
- **Criterion 2:** Minimality: Replications should be avoided. Local concepts that are exactly the same should be represented only once in the resulting products.
- **Criterion 3:** Understandability: Both the database designers and the end-users should easily understand the resulting product.

The data representation structure proposed here is an extension of the following five major papers on multidatabase system designs: Batini et al. (1986), Bright et al. (1992), Bright et al., Ram and Ramesh (1999), and Pitouri & Bukhres (1995). Batini et al.'s work (1986) is the earliest identifiable research in classifying semantic conflict. It emphasizes the importance of differentiating homonyms, synonyms, and unrelated terms in building global schema, and also discusses the concept of structural conflicts. Bright et al., (1992) extend Batini et al.'s work by elaborating on the four dimensions of structural conflicts: type, dependency, (Pitouri & Bukhres, 1995) key, and behavior. Ram and Ramesh (1999) summarized the above work and draw attention to domain conflict resolution. We borrow their terms and include data type, data scale, and data range in our classification. We have also adopted Pitouri & Bukhres's (1995) work to include Instance Identification as one potential source of conflicts. Finally, we extend all the work mentioned here to include "instance value" as one component of identification conflicts. The result of the above work is a list of identifiable components (as presented in Table 1) where the overlapping components have been eliminated.

In addition, all the dimensions and sub-dimensions in the proposed data structure are mutually exclusive by definition. Even so, we observe that two problems may be revealed when the classification scheme is used in practice. First, duplications may exist in the conflict identification process. It is possible that when one conflict is identified, it may relate to multiple components in our model. Second, the multi-dimensionality of semantic conflict may be the result of dependency relationship among identified components in our classification. For instance, the "instance identification" conflict may be the result of "data range" conflict. Consider an example in which one is trying to integrate two databases that both contain a data item call "middle name." In one database, middle name is stored with only an abbreviation, while in another database, the same item is saved as a string. The data range conflict in this case not only creates incompatibility between the two databases, but also raises the problem of "instance identification." Even so, we argue that the classification scheme is the minimal set by definition at the schema level. Therefore, this method fulfills the first and the second criteria. An empirical investigation will be necessary to address the question of usability the proposed method.

In addition to the above three criteria, Naiman and Ouksel (1995) proposed two additional criteria to evaluate any database integration method. These two criteria are based on the needs of a dynamic reconciliation process.

- **Criterion 4:** Dynamic not static: Many multidatabase environments have the constraint that metadata are not available. Dynamic integration requires an automatic method of determining what conflicts must be resolved, what metadata are

needed to resolve those conflicts, and which representations of the relevant real-world entities are equivalent or related, over the different databases. Only a good representation structure can enable automatic semantic heterogeneity resolution at both the schema and instance level.

- **Criterion 5:** Alternative representation: During the dynamic reconciliation process, more than one interpretation of the conflicting schematic representations may be plausible. The taxonomy must represent these plausible interpretations of the conflict.

The first three criteria can be used to evaluate both the representation structure and the integration procedure. The last two criteria are proposed only for the evaluation of the taxonomy and its corresponding representation structure. This chapter did not propose any automatic algorithm or translation rules. However, it is hoped that the proposed representation may serve as a foundation (a standard protocol) for future development.

DISCUSSION AND CONCLUSION

We started by asking the questions “What is a multidatabase?” and “What are semantic conflicts in a multidatabase system?” At the end of this chapter, we propose a data representation format that can be used to identify the conflicts and relationships. This representation format, also known as the canonical data model (Roantree, Kennedy, & Barclay, 1999), is crucial for the integration or translation process in multidatabase design.

How much can we integrate and automate in the database integration effort? From the perspective of semantic conflict resolution, semantic conflicts can all be resolved and automated. Domain conflicts can be partially resolved if an appropriate description of the data is provided through a common ontology. Finally, identification conflicts are the data quality problems. Without human intervention, quality of data can only be ranked or prioritized. Therefore, identification conflicts may be resolved, but are hard to automate using current AI technologies.

The accomplishments of this chapter are threefold. First, this work helps the reader understand what a multidatabase is, and second, possible types of semantic conflicts and relationships were explored that are useful for multidatabase design. Third, the chapter provides a feasible direction of systematically resolving semantic problems during the construction of a multidatabase. Our future work involves the following.

1. The format for the new representation should be written in a more concise way. It would be desirable to convert it into a mathematical format.
2. Comparison rules should be derived to help identify semantic conflicts and relationships.
3. Details of the assertion statement representation should be worked out. The assertion statement should satisfy the criteria stated previously, especially the first criterion, completeness.
4. After accomplishing the first three items, a reconstructed semantic model (possibly an object-oriented type model) will be required.

5. Conversion rules should be developed to accompany the assertion statement representation. A fully constructed semantic map should be generated automatically and dynamically.
6. The proposed structure should be compatible with existing standards, such as Semantic Web.

The evolving nature of database development will continuously create new problems for data integration. Due to the popularity of the Internet, search engine designers are facing the big challenge of integrating databases that are connected by Internet searches. Just as Madnick (1995, p.) predicted a decade ago, “the key challenge in effectively integrating global information and exploiting the capability of the information superhighway is our ability to tie the contexts together.” There are countless reasons for seeking a truly effective way of integrating global system. Facing the explosion of information in cyberspace — the real challenge of database integration — is just beginning to be revealed.

REFERENCES

- Barrowman, D., & Martin, P. (1998). The performance of SQL queries on an X.500 directory service. *Computer Communications*, 21(2), 133-146.
- Batini, C., Lenzerini, M., & Navathe, S. B. (1986). A comparative analysis of methodologies for database schema integration. *ACM Computing Survey*, 18(4), 323-364.
- Becerra-Fernandez, I., Gonzalez, A., & Sabherwal, R. (2004). *Knowledge management: challenges, solutions, and technologies* (1st ed.). Upper Saddle River, NJ: Prentice Hall.
- Becerra-Fernandez, I., Wang, T.-W., & Agha, G. (2005, April 10-13). Actor model and knowledge management systems: Social interaction as a framework for knowledge integration. Paper presented at *The Third Conference of Professional Knowledge Management (WM2005)*, Kaiserslautern, Germany.
- Becker, S. A., Gibson, R., & Leist, N. L. (1996). A study of a generic schema for management of multidatabase systems. *Journal of Database Management*, 7(4), 14-20.
- Bright, M. W., Hurson, A. R., & Pakzad, S. (1992). A taxonomy and current issues in multidatabase systems. *IEEE Computer*, 25(3), 50-60.
- Bright, M. W., Hurson, A. R., & Pakzad, S. (1994). Automated resolution of semantic heterogeneity in multidatabases. *ACM Transactions on Database System*, 19(2), 212-253.
- Cardelli, L. A. (1990). A semantics of multiple inheritance. In S. Zdonik & D. Maier (Eds.), *Readings in object-oriented database system*. San Francisco: Morgan Kaufmann.
- Coad, P., & Yourdon, E. (1991). *Object-oriented design*. Englewood Cliffs, NJ: Yourdon Press.
- Collet, C., Huhns, M. N., & Shen, W. M. (1991). Resource integration using a large knowledge base in Carnot. *IEEE Computer*, 24(12), 55-62.
- Curtis, J., Matthews, G., & Baxter, D. (2005). On the effective use of Cyc in a question answering system. Paper presented at the *IJCAI Workshop on Knowledge and Reasoning for Answering Questions*, Edinburgh, Scotland.

- Dayal, U., & Hwang, H. (1984). View definition and generalization for database integration in a multidatabase system. *IEEE Transaction Software Engineering*, 10(6), 628-645.
- Deaton, C., Shepard, B., Klein, C., Mayans, C., Summers, B., Brusseau, A., et al. (2005, May). The comprehensive terrorism knowledge base in Cyc. Paper presented at the *International Conference on Intelligence Analysis*, McLean, Virginia.
- Dogac, A., Dengi, C., & Oszu, M. T. (1998). Distributed object computing platforms. *Communication of the ACM*, 41(9), 95-103.
- Grandi, F. (1998). Temporal interoperability in multi-temporal databases. *Journal of Database Management*, 9(1), 14-23.
- Hendler, J., Berners-Lee, T., & Miller, E. (2002). Integrating applications on the Semantic Web. *Journal of the Institute of Electrical Engineers of Japan*, 122(10), 676-680.
- Kim, W., & Seo, J. (1991). Classifying schematic and data heterogeneity in multidatabase systems. *IEEE Computer*, 24(12), 12-17.
- Krishnan, R., Li, X., Steier, D., & Zhao, L. (2001). On heterogeneous database retrieval: A cognitively guided approach. *Information System Research*, 12(3), 286-301.
- Lim, E.-P., & Chiang, R. H. L. (2000). The integration of relationship instances from heterogeneous databases. *Decision Support Systems*, 29(2), 153-167.
- Linthicum, D. S. (2004). *Next generation application integration: From simple information to Web services*. Reading, MA: Addison Wesley Professional.
- Liu, J., Eker, J., & Jamneck, J. W. (2004). Actor-oriented control system design: A responsible framework perspective. *IEEE Transactions on Control Systems Technology*, 12(2), 250-262.
- Lonski, T. (1997). Database integration: Criteria and techniques. Paper presented at the *Geospatial Information & Technology Association (GITA) Annual Meeting and Exhibition*, Nashville, Tennessee.
- Madnick, S. E. (1995). Integrating information from global system: Dealing with the "on-and-off ramps" of the information superhighway. *Journal of Organizational Computing*, 5(2), 69-82.
- Martin, P., & Powley, W. (1997). Catalogue management for a multidatabase system using an X.500 directory system. *Distributed System Engineering*, 4(4), 203-213.
- Masters, J. (2002, July 2002). Structured knowledge source integration and its applications to information fusion. Paper presented at the *Fifth International Conference on Information Fusion*, Annapolis, MD.
- Miller, E. (2004, May 17-22). Semantic Web, Phase 2: Developments and deployment. Paper presented at *The Thirteenth International World Wide Web Conference*, New York.
- Miller, G. A. (1995). WordNet: A lexical database for English. *Communication of the ACM*, 38(11), 39-41.
- Miller, L. L., Yu, X., & Nilakanta, S. (2002). Integration of relational databases and record-based legacy systems for populating data warehouses. Paper presented at the *35th Annual Hawaii International Conference on System Sciences (HICSS'02)*, Hawaii.
- Naiman, C. F., & Ouksel, A. M. (1995). A classification of semantic conflicts in heterogeneous database systems. *Journal of Organizational Computing*, 5(2), 167-193.

- Nordbotten, J., & Nordbotten, S. (2002). Evaluation of user search in a Web database. Paper presented at the 35th Hawaii International Conference on Systems Sciences, (HICSS'02),
- Ozsu, M. T., & Valduriez, P. (1991). Distributed database systems. Where are we now? *IEEE Computer*, 24(8), 68-78.
- Pitoura, E., & Bukhres, O. E., A. (1995). Object orientation in multidatabase systems. *ACM Computing Survey*, 27(2), 141-195.
- Ram, S. (1995). Intelligent database design using the unifying semantic model. *Information & Management*, 29(2), 191-206.
- Ram, S., & Ramesh, V. (1999). Schema integration: Past, present, and future. In A. Elmagarmid, M. Rusinkiewicz, & A. Sheth (Eds.), *Management of heterogeneous and autonomous database system*, (pp. 120-155). San Francisco: Morgan Kaufmann.
- Reddy, M. P., Prasad, B. E., Reddy, P. G., & Gupta, A. (1994). A methodology for integration of heterogeneous databases. *IEEE Transactions on Knowledge and Data Engineering*, 6(6), 920-934.
- Reed, S. L., & Lenat, D. B. (2002, July). Mapping ontologies into Cyc. Paper presented at the AAAI 2002 Conference Workshop on Ontologies for the Semantic Web, Edmonton, Canada.
- Roantree, M., Kennedy, J. B., & Barclay, P. J. (1999). Providing views and closure for the object data management group object model. *Information and Software Technology*, 41(15), 1037-1044.
- Scheuermann, P., Li, W.-S., & Clifton, C. (1998). Multidatabase Query Processing with Uncertainty in Global Keys and Attribute Values. *Journal of the American Society for Information Science*, 49(3), 283-301.
- Sciore, E., Siegel, M., & Rosenthal, A. (1994). Using semantic values to facilitate interoperability among heterogeneous information system. *ACM Transactions on Database System*, 19(2), 254-290.
- Sheth, A., & Larson, J. (1990). Federated database systems for managing distributed heterogeneous and autonomous database. *ACM Computing Survey*, 23(3), 183-236.
- Siegel, N., Matthews, G., Masters, J., Kahlert, R., Witbrock, M., & Pittman, K. (2004, April 6-10). Agent architectures: Combining the strengths of software engineering and cognitive systems. Paper presented at the AAAI Workshop on Intelligent Agent Architectures: Combining the Strengths of Software Engineering and Cognitive Systems, Menlo Park, California.
- Singh, M. P., Cannata, P., Huhns, M. N., Jacobs, N., Ksiezyk, T., Ong, K., et al. (1997). The Carnot heterogeneous database project: Implemented applications. *Distributed and Parallel Databases*, 5(2), 207-225.

About the Editor

Keng Siau is a full professor of management information systems (MIS) at the University of Nebraska-Lincoln (UNL) (USA). He is currently serving as editor-in-chief of the *Journal of Database Management* and as the book series editor for the *Advanced Topics in Database Research*. He earned his PhD from the University of British Columbia (UBC), where he majored in MIS and minored in cognitive psychology. His master's and bachelor's degrees are in computer and information sciences from the National University of Singapore. Dr. Siau has over 200 academic publications. He has published more than 80 refereed journal articles, and these articles have appeared (or are forthcoming) in journals such as *Management Information Systems Quarterly*, *Communications of the ACM*, *IEEE Computer*, *Information Systems*, *ACM SIGMIS's Data Base*, *IEEE Transactions on Systems, Man, and Cybernetics*, *IEEE Transactions on Professional Communication*, *IEEE Transactions on Information Technology in Biomedicine*, *IEICE Transactions on Information and Systems*, *Data and Knowledge Engineering*, *Decision Support Systems*, *Journal of Information Technology*, *International Journal of Human-Computer Studies*, *International Journal of Human-Computer Interaction*, *Behaviour and Information Technology*, *Quarterly Journal of Electronic Commerce*, and others. In addition, he has published more than 90 refereed conference papers (including nine ICIS papers), edited/co-edited more than 10 scholarly and research-oriented books, edited/co-edited nine proceedings, and has written more than 15 scholarly book chapters. He served as the organizing and program chairs of the International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD) (1996-2005). He is also serving on the organizing committees of AMCIS 2005, ER 2006, and AMCIS 2007. For more information, please visit his Web site at <http://www.ait.unl.edu/siau/>.

About the Authors

Marzia Adorni is a PhD student in information science at Università di Milano Bicocca, Italy. Her research interests include architectural reflection in adaptive and multichannel information systems.

Pär J. Ågerfalk is a post-doctoral research fellow in software research with the University of Limerick, Ireland, and an assistant professor (Universitetslektor) in informatics at Örebro University, Sweden, where he heads the Methodology Exploration Lab. He received his PhD in information systems development from Linköping University in 2003. Dr. Ågerfalk has published more than 30 research papers in journals such as *European Journal of Information Systems* and *Information and Software Technology* and at conferences such as European Conference on Information Systems and Information Resources Management Association International Conference. His current research centers on open-source software development in the secondary software sector, globally distributed and flexible software development methods, and how information systems development approaches can be informed by language/action theory.

Boanerges Aleman-Meza is a PhD candidate in computer science at the University of Georgia (USA). His research interests include databases, and semantic technologies for search and analytics. Aleman-Meza received a master's degree in applied mathematics from the University of Georgia. His bachelor's degree in computer engineering is from the Technological Institute of Chihuahua II, Mexico. He is member of the IEEE Computer Society and the ACM.

Francesca Arcelli received an MS and a PhD in information science from the Università di Milano Bicocca, Italy. Currently, she is an associate professor at the Università di Milano Bicocca. She has taught software engineering for several years, focusing on the design of software architectures and reverse engineering. Her main research activities

are related to the following topics: object-oriented design methodologies, application of design patterns in forward engineering, and design pattern detection in the context of reverse engineering.

Danilo Ardagna is a post-doctoral fellow at Politecnico di Milano, Italy. He worked for a six month period at IBM's T.J. Watson Research Center in the System Optimization Department. His research interests include information systems architectural design, composed Web services optimization, and autonomic computing infrastructures.

Ulf Ärlig is a senior software consultant at Combitech Systems, Sweden. He received his MSc (1994) in electrical engineering and applied physics from the Technical University of Linköping. Since 1992, he has worked with development of embedded real-time systems in the automotive, gas turbine, defense, medical, and telecom industries. He has been working with mode driven development in several projects. He has also co-developed and been a lecturer in commercial education courses in UML.

I. Budak Arpinar is an assistant professor of computer science at the University of Georgia, USA. His research interests include workflow management, Semantic Web, and Web service composition. Arpinar has past research experience in multidatabase and transaction processing systems, and workflow fields, and also has industry background. He received his MS and PhD in computer science from the Middle East Technical University in Turkey.

Luciano Baresi is an associate professor at Dipartimento di Elettronica e Informazione at Politecnico di Milano, Italy, where he earned both his Laurea degree and PhD in computer science. He also had positions at Cefriel (a research consortium between technical universities and industries in the Milan area), University of Oregon at Eugene (USA), University of Paderborn (Germany), and University of Buenos Aires (Argentina).

Carlo Batini is a full professor of computer engineering at the Università di Milano Bicocca, Italy. His research interests include cooperative information systems, database modeling and design, data repositories, usability of information systems, data and information quality. He has published over 30 papers in international journals and books and about 80 papers at the international level, including *IEEE* and *ACM Transactions*, has co-edited five books, and has published 15 books. From 1995 to 2003, he was a member of the board of directors of the Authority for Information Technology in Public Administration, where he headed several large-scale projects for the modernization of public administration.

Cinzia Cappiello is a post-doctoral fellow at Politecnico di Milano, Italy, where she also completed her PhD in computer science. Her research interests include data quality issues and Web service QoS definition and management.

Marco Comerio is a graduate student at Università di Milano-Bicocca, Italy.

Marco Comuzzi is a PhD student at Politecnico di Milano, Italy. He is currently working as a visiting researcher at the University of Texas at Austin, McCombs School of

Business, Department of Information, Risk, and Operation Management. His research interests include automated negotiation algorithms applied to the fields of Web service QoS management and QoS-enabled Internet networks.

Gautier Dallons is a software engineer at DECIS SA/NV, Belgium, and a former member of PRECISE. He holds an MSc in computer science from the University of Namur where he has been doing research on the quality of conceptual models under the supervision of professor Patrick Heymans.

Jérôme Darmont received his PhD in computer science from the University of Clermont-Ferrand II, France (1999). He has been an associate professor at the University of Lyon 2, France, since then, and has been the head of the Decision Support Databases research group within the ERIC Laboratory since 2000. His current research interests mainly relate to the evaluation and optimization of database management systems and data warehouses (benchmarking, auto-administration, optimization techniques...), but also include XML and complex data warehousing and mining and medical or health-related applications.

Flavio M. De Paoli, PhD, is associate professor in computer science at the Università di Milano Bicocca, Italy. He was a visiting researcher at HP Labs (Palo Alto) in 1989, and at the University of California at Santa Barbara in 1997. His research interests are in software engineering with a focus on distributed systems architecture and languages, e-service computing, multi-agent systems, user-centred cooperative systems, and multi-user interaction. He is author of more than 50 papers, and two books as the author and three as editor. He is member of ACM and IEEE.

Matthew Eavenson is currently finishing up coursework towards a bachelor's in computer science at the University of Georgia's Computer Science Department, USA. He has been an undergraduate research assistant at the LSDIS Lab at the same institution for over a year, where he has been involved in various research projects dealing with the Semantic Web. Most recently, his work has focused on the applications and construction of a Semantic Portal, as well as the second phase of the Insider Threat project.

Liliana Favre is a professor at the Computer Science Department in the Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina. She is a researcher of CIC (Comisión de Investigaciones Científicas de la Provincia de Buenos Aires). She has been working on several national projects about formal methods, software engineering methodologies, and software reusability. Currently, she is research leader of the Software Technology Group at the Universidad Nacional del Centro de la Provincia de Buenos Aires. Her current research interests are focused on rigorous software and system engineering, mainly on the integration of formal techniques with model-driven architecture.

Chiara Francalanci is an associate professor of information systems at Politecnico di Milano, Italy. She has a master's degree in electronic engineering from Politecnico di Milano, where she has also completed her PhD in computer science. As part of her post-doctoral studies, she has worked for two years at the Harvard Business School as a

Visiting Researcher. She has authored articles on the design of information technology architectures and on feasibility analyses of IT projects, consulted in the financial industry, both in Europe and the U.S., and is a member of the editorial board of the *Journal of Information Technology*.

Manuel F. Garasi was born in Milan, Italy, in 1980. In 1999, he obtained a diploma in computer sciences. He graduated in computer sciences (first level) from the Università di Milano Bicocca, Italy, with a thesis on “IPv6: Exploration of the standard with focus on security aspect and cross-platform interactions.” As a final-year postgraduate student in computer science at University of Milano Bicocca, he is currently working with CSI Piemonte to develop a tool for automatic reconceptualization of databases and their integration-abstraction.

Des Greer is a lecturer in computer science at Queen’s University, Belfast (QUB), UK. He is a graduate of QUB and earned a master’s and doctorate at the University of Ulster. Before his career in academia at Queens and previously at Ulster, he worked in industry as an analyst programmer, and his research is inspired by real problems in software engineering. His particular research interests are in incremental and iterative software processes, software evolution planning, software risk management, software adaptivity and agile computing. Dr. Greer teaches software engineering at the undergraduate and graduate levels and is a member of the IEEE.

Simone Grega is a graduate student at Università di Milano Bicocca, Italy.

Riccardo Grosso was born in Moncalieri near Turin, Italy, in 1961. He received a technical diploma in computer sciences in 1980. He started his career in IT mainly working on operational information systems in the automotive field. From the late 1990s onwards, he began specializing in data concept modelling, metadata repositories, glossaries, and conceptual schemas. Since 2001, he has worked at CSI Piemonte (Consortium for the Information System), Italy, in the areas of data modelling, data administration, and database reverse engineering techniques.

Henrik Gustavsson has been a staff member at the University of Skövde, Sweden, since 1997. He received his MSc (1997) in computer science from the University of Skövde, and his PhD (2003) from the University of Exeter. His research is published in a variety of international conferences and workshops. He is the architect and lead developer of a prototype UML meta-modelling tool with support for OCL rule execution. The tool has been used in a variety of research projects. Recent work includes work on XMI, specifically the analysis of XMI documents for interchange.

Sari Hakkarainen received her PhD in computer and systems sciences (DSV) at Stockholm University (SU), Sweden, in 1999 and her BSc in administrative information processing at Stockholm University and Royal Institute of Technology (KTH) in 1991. Her main areas of research are information systems development, conceptual modelling, metadata, and information systems interoperability. She has participated in EU and national funded research projects in distributed requirements engineering, schema

integration, systems renewal, and document metadata at the Swedish Institute for Systems Development (1994-2000). She is associate professor at the Department of Computer and Information Science (IDI) at Norwegian University of Science and Technology (NTNU) (2004-present), after a post-doctoral study at the same department (2002-2004). Her ongoing research projects involve semantic interoperability, formal concept modelling in semantic Web applications, conceptual metadata analysis, and design in Web information systems quality assurance in ontology building.

Terry Halpin, BSc, DipEd, BA, MLitStud, PhD, is distinguished professor and vice president (Conceptual Modeling) at Neumont University, Salt Lake City, Utah, USA. His doctoral thesis formalized Object-Role Modeling (ORM/NIAM). After leaving academia to work on data-modeling technology at Asymetrix Corporation, InfoModelers Inc., Visio Corporation, and Microsoft Corporation, he returned to academia, specializing in software development using a business rules approach to informatics. With a research focus on conceptual modeling and conceptual query technology, he has authored over 100 technical publications and five books, including *Information Modeling and Relational Databases* and *Database Modeling with Microsoft Visio for Enterprise Architects*. He is a member of IFIP WG 8.1 (Information Systems) and is an editor/reviewer for several academic journals.

Zhen He received his PhD degree in computer science from the Australian National University in 2003. He then worked as a postdoctoral research associate at the University of Vermont until the end of 2004. He is currently working as a lecturer in La Trobe University in Melbourne, Australia. His current research interests include database query optimization, databases cache optimization, XML databases, data stream mining, wireless sensor networks, and time-series mining.

Lillian Hella received her master's degree in information systems from the Norwegian University of Science and Technology (NTNU) in 2004. Her master's thesis addressed a semantic transformation approach for ISO 15926. She is a doctoral candidate at the Department of Computer and Information Science at NTNU, working in the NFR-financed MOSES project. MOSES focuses on increased ability for users and service providers to express what information and services they need or provide using Semantic Web technology. Her research interests include ontology, Semantic Web technology, pervasive computing, and personalisation.

Patrick Heymans, born in 1972, is a full professor in computer science at the University of Namur, Belgium, where he is also a member of the PRECISE lab. He teaches object-oriented programming, information systems modelling, and requirements engineering. His research focuses on the semantics of modelling languages and on requirements engineering. He is the author of several international publications in these areas. He is a program committee member for the well known conferences RE and CAiSE, and a regular reviewer for international scientific journals. In 2001, he completed a PhD thesis on the subject of requirements animation under the supervision of Prof. Pierre-Yves Schobbens and Prof. Eric Dubois. During the last ten years, he has been involved in a series of regional, national, and international research projects, most notably CREWS and InterOP.

Ali R. Hurson is a member of the computer science and engineering faculty at Pennsylvania State University, USA. His research for the past 20 years has been directed toward the design and analysis of general as well as special purpose computer architectures. His research has been supported by NSF, NCR Corp., DARPA, IBM, Lockheed Martin, ONR, and the Pennsylvania State University. He has published over 200 technical papers in areas including database systems, multidatabases, global information-sharing processing, application of mobile-agent technology, object-oriented databases, mobile-computing environment, computer architecture and cache memory, parallel and distributed processing, dataflow architectures, and VLSI algorithms. Dr. Hurson served as guest editor for various technical journals and has given tutorials and technical talks on global information sharing, dataflow processing, database management systems, supercomputer technology, data/knowledge-based systems, scheduling and load balancing, and parallel computing.

Yu Jiao is a postdoctoral researcher at the Oak Ridge National Laboratory. She received the BSc degree in computer science from the Civil Aviation Institute of China in 1997. She received her MSc and PhD degrees from The Pennsylvania State University, USA, in 2002 and 2005, respectively, both in computer science. Her main research interests include software agents, pervasive computing, and secure global information system design.

John Krogstie has a PhD (1995) and an MSc (1991) in information systems, both from the Norwegian University of Science and Technology (NTNU). He is professor in information systems at IDI, NTNU, Trondheim, Norway. He is also a senior advisor at SINTEF. He was employed as a manager at Accenture 1991-2000. John Krogstie is the Norwegian representative for IFIP TC8 and vice-chair of IFIP WG 8.1 on information systems design and evaluation, where he is the initiator and leader of the task group for mobile information systems. He has published around 75 refereed papers in journals, books, and archival proceedings since 1991.

Brian Lings was awarded a doctorate in computer science from the University of East Anglia in 1975. He is currently a member of the academic staff at the University of Skövde, Sweden, and a consultant with Certus Technology Associates Ltd UK. He chairs the steering committee of the British National Conference on Databases (BNCOD). His recent publications center on the areas of tool evaluation and development, and on issues of consistency maintenance and interoperability in multi-model environments. He is a co-developer of the 2G method, and continues to be active in applications of the method.

Paolo Losi graduated in electronic engineering from the Politecnico di Torino in 1999. He has been working in the field of data networks, security and system integration for more than 10 years. After being employed for four years as chief of network and service engineering by one of the main Italian Telco operators, he started a free-lance consultancy activity focusing on network design, analysis, and performance. He teaches as a contract lecturer in data and network security at University of Insubria-Varese.

Björn Lundell has been a staff member at the University of Skövde, Sweden, since 1984. He received his MSc (1991) in computer science from the University of Skövde, and his PhD (2001) from the University of Exeter. He has been active in international standardisation on drafts to ISO 14102, as well as on the published standard ISO/IEC 14102. He is a co-developer of the 2G method, a qualitative method evolved for use in socio-technical evaluations. His research is published in a variety of international journals and conferences, and he has established active links with a number of Swedish companies.

Z. M. Ma received a Ph D degree from the City University of Hong Kong in 2001 and is currently a full professor in the College of Information Science and Engineering at Northeastern University, China. His current research interests include intelligent database systems, knowledge management, Semantic Web and XML, life science data management, e-learning systems, intelligent planning and scheduling, decision making, engineering database modeling, and enterprise information systems. Dr. Ma has published over 40 papers in international journals, conferences, edited books, and encyclopedias in these areas since 1998. He also authored and edited several scholarly books published by Springer and Idea Group Publishing, respectively.

Pat Martin is a professor and associate director of the School of Computing at Queen's University, Canada. He holds a BSc from the University of Toronto, an MSc from Queen's University, and a PhD from the University of Toronto. He is also a faculty fellow with IBM's Centre for Advanced Studies. His research interests include database system performance, Web services, and autonomic computing systems.

Anders Mattsson has a Master of Science and is a senior systems engineering consultant at Combitech Systems, Sweden. He is specialized in development methods and architecture and since 1989 has worked at a wide range of companies in the automotive, defense, telecom and automation industries, typically as a software architect or as a mentor in UML and model-driven development. He has developed commercial education courses on model-driven development and UML, and developed a method for model-driven development of embedded real-time systems, called PARTS, that has been successfully implemented at several companies. He has also authored and co-authored several articles and conference publications on model-driven development.

Andrea Maurino received his MS and PhD from Politecnico di Milano. Currently, he is assistant professor at Università di Milano Bicocca, Italy. His research interests include the design of data-intensive Web applications, the personalization of Web sites, and geographical information systems (GIS) in the field of archaeology. Within the MAIS Project, his main interest is adaptive strategies for mobile information systems and partitioning of workflow orchestration.

Stefano Modafferi is a PhD student at Politecnico di Milano, Italy. His main research is on mobile information systems, in particular its focus on partitioning of workflow orchestration.

Kenneth E. Murphy holds a PhD in operations research from Carnegie Mellon University and is currently an associate professor of information systems at the Atkinson Graduate

School of Management, Willamette University in Salem, Oregon, USA. Recently, Dr. Murphy has been investigating the implementation of integrated systems, specifically, ERP systems, in large enterprises. Dr. Murphy's work on integrated systems has followed several threads including the financial justification of large-scale systems using both tangible and intangible factors, and investigating the tools and methods for successful system implementation. He has published in *Operations Research*, *Communications of the ACM*, the *Information Systems Journal*, and in other IS journals.

Anna Gunhild Nysetvold has an MSc (2005) in information systems from IDI, NTNU. She is currently working as a system developer in Kantega.

Ratko Orlandic received his MS and PhD degrees in computer science from the University of Virginia in 1989. He is currently an associate professor of computer science at the University of Illinois at Springfield, USA. He has published over 30 refereed journal and conference papers and has served as a reviewer for a number of scientific journals and conferences. The research of Dr. Orlandic focuses on the problems of scale in scientific data management. His major research interests include data storage, data mining, access methods, data clustering, and software architecture.

Vladimir Ovchinnikov, PhD in computer science, is a chief specialist of NISCo (Russia, Lipetsk), and assistant professor of Lipetsk State Technical University (LSTU). He has been participating in implementation of manufacture execution and planning systems of NISCo as a system architect, project leader, and project manager. He lectures on object-oriented designing and programming at LSTU (Russia, Lipetsk). His PhD thesis was devoted to theoretical and practical issues of query and data structure optimization within relational scheme, as applied to creating effective information systems of continuous (non-discrete) production. At present, his research activity is concentrated on conceptual modeling, knowledge management, and semantic data integration.

Devanand Palaniswami is a research scientist with the LSDIS Lab at the University of Georgia, USA. He has eight years of software industry experience with Capgemini, Asea Brown Boveri, and Taalee (now Semagix). He received his bachelor in engineering in mechanical engineering from Anna University, India, and an MS in computer science from the University of Georgia.

Barbara Pernici is full professor of computer engineering at Politecnico di Milano, Italy. Her research interests include cooperative information systems, workflow management systems, information systems modeling and design, temporal databases, and applications of database technology. She has published 35 papers in international journals, including *IEEE and ACM Transactions*, co-edited 10 books, and published about 130 papers at the international level. She is an editor of the *Requirements Engineering Journal*. She is chief scientist of the Italian FIRB MAIS (Multichannel Adaptive Information Systems), 2002-2005. She is chair of Working Group 8.1 Information Systems Design of IFIP (International Federation for Information Processing).

Anna Persson is a staff member at the University of Skövde, Sweden. She received her MSc (2004) in computer science from the University of Skövde. She has conducted

theoretical and empirical studies on UML tools with a focus on XMI interchange in heterogeneous tool environments. Her work has been published in academic and industrial conferences.

Isabelle Pollet, born in 1973, is a business analyst at SmalS-MvM/Egov, Belgium. She holds a PhD in computer science from the Catholic University of Louvain, Belgium, for her thesis “Towards a generic framework for the abstract interpretation of Java.” supervised by professor Baudouin Lecharlier. She is the author of several international publications in the area of programming language analysis. She has been a teaching assistant and researcher at the University of Namur, where she participated to the InterOP project.

Wendy Powley is a research associate and adjunct lecturer in the School of Computing at Queen’s University, Canada. She has worked on various database-related projects in conjunction with the Center of Advanced Studies at IBM since 1992. Most recently, her research has focused on the addition of autonomic features to database management systems and Web services environments. Wendy received a BA in psychology in 1984, a BED in 1985, and an MSc in computer science in 1990 from Queen’s University.

Claudia Raibulet received her MS degree in computer science from Politehnica University of Bucharest, Romania, in July, 1997. From 1999 to 2002, she attended the PhD course in computer and system engineering at Politecnico di Torino, Italy, receiving her degree in 2002. Currently, she is assistant professor at Università di Milano Bicocca, Italy. Her main research activities are related to the software engineering domain, including software architectures, object-oriented methodologies, design patterns, and reverse engineering.

Sudha Ram is professor of management information systems in the Eller College of Business and Public Administration at the University of Arizona, USA. She received a BS degree in mathematics, physics and chemistry from the University of Madras in 1979, PGDM from the Indian Institute of Management, Calcutta in 1981, and a PhD from the University of Illinois at Urbana-Champaign, in 1985. Dr. Ram has published articles in such journals as *Communications of the ACM*, *IEEE Expert*, *IEEE Transactions on Knowledge and Data Engineering*, *Information Systems*, *Information Systems Research*, *Management Science*, and *MIS Quarterly*. Her research deals with issues related to enterprise data management. Her research has been funded by organizations such as, IBM, Intel Corporation, Raytheon, US ARMY, NIST, NSF, NASA, and the Office of Research and Development of the CIA. Specifically, her research deals with Interoperability among heterogeneous database systems, semantic modeling, bioinformatics and spatio-temporal semantics, business rules modeling, Web services discovery and selection, and automated software tools for database design. Dr. Ram serves on the editorial board for such journals as *Decision Support Systems*, *Information Systems Frontiers*, *Journal of Information Technology and Management*, and as an associate editor for *Information Systems Research*, *Journal of Database Management*, and the *Journal of Systems and Software*. She has chaired several workshops and conferences supported by ACM, IEEE, and AIS. She is a cofounder of the Workshop on Information Technology and Systems (WITS) and serves on the steering committee of many workshops and conferences

including the Entity Relationship (ER) Conference. Dr. Ram is a member of ACM, IEEE Computer Society, INFORMS, and Association for Information Systems (AIS). She is also the director of the Advanced Database Research Group based at the University of Arizona.

Germain Saval, born in 1979, is a PhD student in computer science at the University of Namur, Belgium, and a member of PRECISE. He is currently investigating semantic issues of requirements engineering languages under the supervision of Prof. Patrick Heymans. He holds an MSc in computer science from the Université Pierre et Marie Curie (Paris VI).

Amit P. Sheth is a professor in the computer science department at the University of Georgia, USA, and the director of the LSDIS Lab (<http://lsdis.cs.uga.edu>). He's also the editor-in-chief of the *International Journal on Semantic Web and Information Systems*, and a co-founder/CTO of Semagix (<http://www.semagix.com>), a Semantic Web technology company. His research interests include Semantic Web and semantic interoperability, rich-media content management, workflow and collaboration systems, and semantic applications in financial, national security, and health care. He received his BE from B.I.T.S., Pilani, India, and his MS and PhD in computer science from Ohio State University. <http://lsdis.cs.uga.edu/~amit/>

Guttorm Sindre (b.1964) holds a PhD in computer science from the Norwegian Institute of Technology (NTH), University of Trondheim, 1990. From 1992-95, he was associate professor in software engineering and from 1999-2003 associate professor in information systems at IDI, NTNU. In 2003, he was promoted to full professor. He has advised several PhD students. He spent a sabbatical at the University of Auckland, New Zealand, in 2002/2003. He is currently manager of the 5 MNOK Research Project WISEMOD (2004-2007), funded by the Norwegian Research Council over the IKT-2010 programme, and is also involved in the EU IST 6FP NoE project INTEROP.

Seok Il Song received his BS, MS, and PhD degrees in computer and communication engineering from Chungbuk National University in 1998, 2000, and 2003, respectively. Currently, he is an assistant professor in the School of Electronic and Information Engineering at Chungju National University, Korea. His research interests include database management systems, concurrency control, high-dimensional index structures, storage systems, sensor networks, and XML index structures.

Darijus Strasunskas is a PhD candidate in the information system group of the Computer and Information Science Department of the Norwegian University of Science and Technology (NTNU). He graduated in information science and received an MS from Vilnius University (2000). From 1997 to 2001, he was employed by Kraft Foods International subsidiary in Lithuania as a finance analyst and key user of ERP system manufacturing module. His research interests include (but are not restricted to) management and traceability of developed artifacts in geographically distributed cooperative systems development, semantic interoperability of information (model) fragments, semantic Web, and design repository.

Francesco Tisato is full professor of computer science at the Università di Milano Bicocca, Dipartimento di Informatica, Sistemistica e Comunicazione, Italy, and is coordinator of the course degree of computer science. His research and technology transfer activities are in the following areas: computational reflection of object-oriented design of monitoring and control systems, environment and urban traffic, production planning system based on autonomous agents, software architectures for real-time embedded systems, Web-based architectures for ERP systems, and real-time computer-supported co-operative work.

Stine Tuxen received a bachelor's degree in computer engineering from Agder University College in Grimstad in 2002, and a master of science in computer science from the Norwegian University of Technology and Science (NTNU) in Trondheim in 2004. Her master's thesis addressed a semantic transformation approach for ISO 15926. She has been a consultant at Bekk Consulting, Norway, since August 2004.

Terje Wahl (b. 1977) holds a Master of Science in computer science from the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway, 2001. One year of the studies was undertaken as an exchange student at Queen's University in Kingston, Canada. From 2001-2004, he worked as an IT consultant and systems developer in Norway and Latvia, before returning to NTNU in 2004, where he is currently a PhD student at the Department of Computer and Information Science. His current research interests include methods for enabling Web information service engineering on the semantic Web.

Te-Wei Wang is an assistant professor in the MIS Department at the University of Illinois at Springfield, USA. He received his PhD in business administration from the Southern Illinois University at Carbondale. He holds a master of science in mechanical engineering from University of Missouri-Rolla. His research interests include e-commerce assurance service, information systems analysis and design, requirement engineering and multi-agent simulation. Dr. Wang is well trained on several vendor-specific software engineering methodologies. His publications can be seen in many national/international conferences and IS journals.

Liang Xiao is undertaking a PhD in the school of computer science at Queen's University, Belfast, UK. He obtained his BSc at Huazhong University of Science & Technology (HUST) and his MSc at the University of Edinburgh. He has worked in the telecommunications industry as a software designer and programmer. His experience on solving real-domain problems has stimulated his interests in the area of software engineering. Specifically, his research work focuses on software adaptivity, agent-oriented modeling, and requirements engineering. The results of his research have been presented and published at several international conferences.

Jae Soo Yoo received a BS degree in computer engineering from Chonbuk National University, Korea, in 1989, and an MS and PhD degrees in computer science from Korea Advanced Institute of Science and Technology (KAIST) in Korea. He is currently an associate professor in the School of Electronic and Computer Engineering at Chungbuk National University. His research interests include database management systems, XML

repository systems, distributed object computing, storage area network, bioinformatics, and information retrieval systems.

Byunggu Yu received his PhD in computer science from the Illinois Institute of Technology in 2000. He is the Gold Prize winner of the 1st Human-Tech Research Paper Competition of Samsung Electronics. He is currently an assistant professor of computer science at the University of Wyoming, USA. He has published 30 refereed computer science research papers in the area of databases and has served as a reviewer for a number of journals and conferences. He has served as a program committee member for two international conferences and organized an invited conference session on spatiotemporal databases. His major research interests include spatial databases and spatiotemporal databases.

Huimin Zhao is an assistant professor of management information systems at the School of Business Administration, University of Wisconsin-Milwaukee, USA. He received his BE and ME in Automation from Tsinghua University, China (1990 and 1993, respectively), and a PhD in management information systems from the University of Arizona (2002). His current research interests include data integration, data mining, and Web services. He has published in such journals as *Communications of the ACM*; *Journal of Management Information Systems*; *IEEE Transactions on Knowledge and Data Engineering*; *IEEE Transactions on Systems, Man, and Cybernetics*; *Information Systems*; and *Journal of Database Management*.

Min Zheng received a BS degree in computer science and technology from the Xidian University, Xian, China, in 1991, and an MS in computer science from Queen's University, Kingston, Canada, in 1999. In 2000, he joined Nortel Networks, Ottawa, where he works in research and development. His interests include database applications for network management.

Index

A

- AAM (see adaptive agent model)
- ABRL (see Agent Behaviour Representation Language)
- abstract schema 176
- abstraction 166, 174
- accountability 64
- accuracy 219
- action 64, 66
- action knowledge 64
- activity-decision flow (ADF) diagram 96
- adaptive agent model (AAM) 149, 152
- adaptivity 149, 188
- ADF diagram (see activity-decision flow diagram)
- agent 149, 150, 151, 152, 157, 165, 323, 326, 343
- agent behaviour 152
- Agent Behaviour Representation Language (ABRL) 150
- agent communication diagram 157
- agent message 165
- agent pattern 150
- agent system 149
- agent UML (AUML) 151
- agent-based computation model 343
- agent-object-relationship (AOR) 151
- agent-oriented system 149
- agent-oriented systems development 149
- agent-view reorganization algorithm (AVRA) 326
- agent/rule/class hierarchy 155
- agile method 64
- AGG (see attributed graph grammar)
- AHP (see analytical hierarchy process)
- analytical hierarchy process (AHP) 91
- AOR (see agent-object-relationship)
- API (see application program interface)
- application program interfaces (API) 265
- architecture independence 157
- ArgoUML 32
- ARIES 307
- artefacts/artifacts 88, 96
- association classes 106
- association selection 391
- ATHENA 86
- attributed graph grammar (AGG) 196
- attributes 178
- AUML (see agent UML)
- auto-adaptivity 167
- autonomic computing 206
- AVRA (see agent-view reorganization algorithm)

B

basic schema 176
 behavioural model 152
 benchmark for object-relational databases (BORD) 296
 binding 166
 BORD (see benchmark for object-relational databases)
 BPD (see business process diagram)
 BPMI (see Business Process Management Initiative)
 BPML (see Business Process Modeling Language)
 BPMN (see business process modeling notation)
 buffer pool 206, 209, 211
 buffer pool tuning algorithm 211
 bulk-loading 267
 Bunge-Wand-Weber (BWW) ontology 44, 94, 102, 124
 business classes 162
 business objects 161
 business process diagram (BPD) 95, 88, 101
 Business Process Executive Language for Web Services 97, 189
 Business Process Management Initiative (BPMI) 95
 Business Process Modeling Language (BPML) 80
 business process modeling notation (BPMN) 88, 95
 business rules 109, 149
 business user 101
 BWW ontology (see Bunge-Wand-Weber ontology)

C

case study 32
 CASE tool 161
 CASL (see Common Algebraic Specification Language)
 CDM (see common data model)
 CG (see conceptual graphs)
 changing requirements 149

CIM (see computation-independent model/computational independent model)
 CIS (see cooperative information system) 171
 classification framework 45
 cluster analysis 227, 230
 CML 43
 Combitech Systems AB 32
 Common Algebraic Specification Language (CASL) 2, 16
 common data model (CDM) 322
 common warehouse metamodel (CWM) 4
 compatible matrix 251
 completeness of the conceptual schema 182
 composition operation 382
 computation-independent model/computational independent model (CIM) 1, 2, 3, 396
 conceptual data modeling 290
 conceptual graphs (CG) 375
 conceptual model 43
 conceptual modelling 44
 conceptual schemas 171
 concurrency control 250
 connecting objects 88, 96
 constraints 109
 context 403
 context mechanism 383
 cooperative information system (CIS) 171
 correctness of the conceptual schema 181
 coverage in process 46
 coverage in product 46
 covtype 364
 CPLEX, 197
 cross-validate 230
 CWM (see common warehouse metamodel) 4
 CycL 43

D

Daemon program 327
 DAML+OIL 43
 DAML+OIL Tutorial 49
 data flow diagram (DFD) 100
 data search GUI 327

data-partitioning 250
 database administrator (DBA) 206, 228
 database design 275
 database integration 171, 422
 database interoperability 177
 database management system (DBMS)
 206, 228, 250, 295
 DataSearchMaster 329
 DB2 Universal Database 209
 DBMA-aglet framework 325
 DBMS (see database management system)
 DBMS-aglet framework 325
 deadlock 251
 DEF (see dynamic evaluation framework)
 DFD (see data flow diagram)
 derivation rules 109
 detection and reclustering of objects (DRO)
 306
 development situations 66
 Document Access Problem of Insider
 Threat 405
 DoEF (see dynamic object evaluation
 framework)
 Domain appropriateness 82
 DRO (see detection and reclustering of
 objects)
 DSTC algorithm (see dynamic, statistical,
 and tuneable clustering algorithm)
 dynamic evaluation framework (DEF)
 294, 295, 297
 dynamic GP algorithm (see dynamic graph
 partitioning algorithm)
 dynamic graph partitioning (GP) algorithm
 306
 dynamic object evaluation framework
 (DoEF) 294, 295
 dynamic probability ranking principle (PRP)
 algorithm 306
 dynamic PRP algorithm (see dynamic
 probability ranking principle algo-
 rithm)
 dynamic reconfiguration algorithm (DRF)
 206, 213
 dynamic, statistical, and tuneable cluster-
 ing (DSTC) algorithm 306

E

EDI (see engaging, dynamic innovation)
 EDI CASE 55
 EEML (see Extended Enterprise Modeling
 Language)
 EER model (see enhanced entity-rela-
 tion-ship model)
 efficiency 183
 engaging, dynamic innovation (EDI) 55
 enhanced entity-relationship (EER) model
 275
 enterprise modelling 124
 entity-relationship (ER) 375, 377
 EPC (see event-process chain)
 equality constraint 115
 ER (see entity-relationship)
 evaluation 43, 95
 evaluation of languages 95
 event-process chain (EPC) 96
 executable requirements 167
 export/import functionality 33
 Extended Enterprise Modeling Language
 (EEML) 86
 eXtensible Agent Behaviour Specification
 Language (XABSL) 150
 eXtensible Markup Language (XML)
 43, 149
 external uniqueness constraint 113
 externalisation 149, 167
 eXtreme Programming (XP) 72

F

F-logic 43
 fact type 110
 FCO-IM (see fully communication oriented
 information modeling)
 FDBS (see federated database system)
 feature vectors 249
 federated database system (FDBS) 322
 FEER model (see fuzzy extended entity-
 relationship model)
 FIPA 151
 flow objects 88, 96
 framework 80

FRDB (see fuzzy relational database)
 fully communication oriented information
 modeling (FCO-IM) 375
 functional requirement 152, 153
 fuzzy data 273, 275, 276, 278
 fuzzy extended entity-relationship (FEER)
 model 275
 fuzzy IFO data model (IF₂O) 273
 fuzzy object-relational databases 275
 fuzzy relational database (FRDB) 275, 278

G

generalization hierarchies 179
 goal modelling 124
 gradual moving window of change 310
 granular-locking 257
 Graph eXchange Language (GXL) 196
 graphical user interface (GUI) 327
 GUI (see graphical user interface)
 GXL (see Graph eXchange Language)

H

heterogeneous data 322
 heterogeneous databases 227
 heuristics 177
 HostMaster 329

I

i* 126
 i-Logix Rhapsody® tool 30
 IB (see information broker)
 IBM Rose® tool 30
 ICM (see implementation component
 model)
 ICT (see information and communication
 technology)
 ideal typical method 67
 IF₂O (see fuzzy IFO data model)
 IFO data model 273
 implementation component model (ICM) 11
 implicit semantics 405
 index structures 249
 information and communication technology
 (ICT) 171
 information broker (IB) 326

information retrieval 325
 information security 321
 information systems 125
 information technology (IT) 375
 information-modeling approach 106
 Insider Threat 402
 integrated schema 174
 integration 174
 integration/abstraction 174
 interpretation 65
 interschema relationship identification (IRI)
 227
 IRI (see interschema relationship identifica-
 tion)
 IT (see information technology)

J

JADE 149
 Java database connectivity (JDBC) 228
 JavaBeans 163
 JDBC (see Java database connectivity)

K

K-NN query 265
 key-range locking 256
 knowledge externalizability 101
 Knowledge Query and Manipulation
 Language (KQML) 325
 KQML (see Knowledge Query and
 Manipulation Language)

L

language quality 80, 82
 large-scale models 37
 latches 251
 link techniques 250
 lock-coupling techniques 250
 locks 251
 logical consistency 251
 logical selection 390
 logical sequence number (LSN) 253
 LOOM 43
 LOVeM 96
 LSN (see logical sequence number)

M

- MAIS (multichannel adaptive information systems)
- MAMDAS framework (see mobile agent-based secure mobile data access system framework)
- mandatory role constraints 113
- maturity 48
- MBR (see minimal bounding rectangle)
- MDA (see model-driven architecture)
- MDA-based tools 24
- MDAS (see mobile data access system)
- MDBAS (see mobile database agent system)
- MDBS (see multidatabase systems)
- MDD (see model-driven development)
- message-oriented middleware (MOM) 155
- metadata extraction 402
- metadata interchange format 30
- metamodel 6, 97, 124
- metamodeling patterns 6
- method configuration 65, 68
- method creator 65
- method description 65
- method engineering 64
- method engineering approach 74
- method fragments 66
- method rationale 64
- method tailoring 64
- method-in-action 64
- method-in-concept 64
- method-ism 74
- methodological framework 187
- methodology 45, 171
- methods 64
- MILP (see mixed integer linear programming)
- minimal bounding rectangle (MBR) 349
- minimum bounding region (MBR) 255
- minus 391
- misuse detection 405
- mixed integer linear programming (MILP) 197
- mobile agent 321, 323
- mobile agent-based secure mobile data access system (MAMDAS) framework 320
- mobile data access system (MDAS) 321
- mobile database agent system (MDBAS) 325
- mobile information systems 188
- model interchange 28
- model-based development 1, 28
- model-driven approaches 6
- model-driven architecture (MDA) 1, 2, 189
- model-driven communication architecture 165
- model-driven development (MDD) 1
- modelling tools 29
- modifiability 166
- MOM (see message-oriented middleware)
- moving window of change 310
- MPL (see multi programming levels)
- multi programming levels (MPL) 266
- multichannel 188
- multichannel adaptive information systems (MAIS) 188
- multidatabase 321
- multidatabase system (MDBS) 322

N

- need to know 402
- NEREUS 7
- next-key locking 256
- NFR (see non-functional requirement)
- NIAM 44
- node sequence number (NSN) 253
- node splits 252
- NodeManager 329
- nominalization 106
- non-functional requirement (NFR) 126
- NSN (see node sequence number)

O

- object clustering benchmark (OCB) 296, 301
- Object Constraint Language (OCL) 2

object database management system (ODBMS) 295
 object frequency 306
 object linking and embedding for databases (OLE DB) 228
 Object Management Group (OMG) 2, 30
 object type 110
 object-oriented database management system (OODBMS) 297
 object-role modeling (ORM) 106, 375
 objectification 106
 OCB (see object clustering benchmark)
 OCL (see Object Constraint Language)
 OCML 43
 ODBC (see open database connectivity)
 ODBMS (see object database management system)
 OIL 43
 OLE DB (see object linking and embedding for databases)
 OLTP application (see on-line transaction processing application)
 OMG (see Object Management Group)
 on-line transaction processing (OLTP) application 213
 Ontolingua 43
 ontological discrepancies 102
 ontology 43, 44, 48, 101, 189, 403
 ontology building 48
 ontology development 101
 ontology specification 44
 ontology specification languages 43
 OODBMS (see object-oriented database management system)
 OPCF (see opportunistic prioritised clustering framework)
 open database connectivity (ODBC) 228
 open interchange standard 30
 open standards 30
 open-source 29
 open-source tool 30, 33
 opportunistic prioritised clustering framework (OPCF) 306
 ORM (see object-role modeling)
 OWL (see Web Ontology Language)
 OWL-Tutorial 49

P

page usage 306
 PAM (see point access method)
 partial lock-coupling (PLC) 255
 partitioning 196
 path notation 382
 path stack 263
 path-loss 259
 performance 206
 PGP protocol (see pretty good privacy protocol)
 phantom protection 250
 physical consistency 251
 PICM (see platform-independent component model)
 PIM (see platform-independent model)
 PIR protocol (see private information retrieval protocol)
 PKI (see public key infrastructure)
 platform independent model (PIM) 189
 platform specific model (PSM) 1, 189
 platform-independent component model (PICM) 11
 platform-independent model (PIM) 1
 platform-specific component model (PSCM) 11
 platform-specific model (PSM) 1, 3
 Platypus 296
 PLC (see partial lock-coupling)
 point access method (PAM) 349, 359
 predicate locking 256
 pretty good privacy (PGP) protocol 337
 primary keys 178
 private information retrieval (PIR) protocol 337
 process modeling 80
 process partitioning 196
 propositional nominalization 107
 propositions 108
 protégé 43
 PSCM (see platform-specific component model)
 PSM (see platform-specific model)
 public key infrastructure (PKI) 336

Q

QoS (see quality of service)
 QoS trees 193
 quality 42, 80, 98, 189, 193, 337
 quality evaluation framework 125
 quality of conceptual modelling languages 98
 quality of models 80
 quality of service (QoS) 189, 193, 337
 query server 326
 query simplification 376
 query, view, transformations (QVT) metamodel 5
 QVT metamodel (see query, view, transformations metamodel)

R

RAAM (see reflective and adaptive agent model)
 range search 265
 range search operations 265
 rationale 68
 rationality resonance 68
 RDF (see resource description framework)
 RDF/XML 49
 recovery 250
 referential integrity constraints 178
 reflective 192
 reflective and adaptive agent model (RAAM) 167
 reification 106
 reinsert operations 250
 relational database schemas 178
 relevance of documents 403
 remote procedure call (RPC) 325
 repository 171
 representation of product 48
 requirements 163
 requirements transformation 154
 resource description framework (RDF) 43, 376, 404
 resource management 206
 retrieval server 326
 reuse 171
 reuse of product and process 47

Rhapsody 32
 ripple effect 166
 risk and compliance 402
 RM 377
 robustness 220
 role concept 383
 Rose Realtime 32
 RosettaNet 96
 round-trip interchange 33
 RPC (see remote procedure call)
 rule model 154
 RUP 45

S

S-reference dependency protocol 310
 SAM (see spatial access method)
 schema correspondences 229
 SCM (see semantically complete model)
 SCM-based client/server technology 395
 SCQL (see Semantically Complete Query Language)
 SCRUM 72
 secure socket layer (SSL) protocol 337
 self-adaptivity 167
 self-organizing map (SOM) 227, 228
 semantic analysis 402
 semantic annotations 402
 semantic associations 403
 semantic completeness 375
 semantic conflict 420, 421
 semantic discovery 403
 Semantic Web 42, 402
 semantically complete model (SCM) 375, 380
 Semantically Complete Query Language (SCQL) 375, 378, 382, 396
 semiotic framework 98
 semiotic quality framework 95
 SHOE 43
 SHORE 296, 307
 simple QSF-trees (sQSF-trees) 369
 situational method 67
 situational nominalization 107
 social action 65
 software architecture 157
 software processes 65

space-partitioning 250
 spatial access method (SAM) 349
 sQSF trees (see simple QSF-trees)
 SSL protocol (see secure socket layer protocol)
 SSM (see summary-schemas model)
 SSM administration server 326
 SSM prototype 326
 stakeholder participation 47
 star notation 383
 state machine 150
 states of affairs 108
 structural model, agent diagram 155
 structuring primitives 173
 summary-schemas model (SSM) 321, 322, 326
 SWETO (see test-bed ontology)
 swimlane 88, 96
 systems development 28

T

TAU 32
 TDIM (see top-down index region modification)
 Telos 43
 template 124
 test-bed ontology (SWETO) 406
 thesaurus server 326
 tool 171
 tool lock-in 28
 top-down index region modification (TDIM) 254
 TPC (see Transaction Processing Performance Council)
 Transaction Processing Performance Council (TPC) 295
 transformation 393
 two-phase locking 256
 two-way encapsulation 165

U

UEML (see Unified Enterprise Modelling Language)
 UML (see Unified Modeling Language)

UML class diagram 32
 UML diagram 193
 UML model 30
 Unified Enterprise Modelling Language (UEML) 86, 125
 Unified Modeling Language (UML) 2, 17, 30, 32, 44, 87, 95, 106, 149, 193
 union 392
 uniqueness constraint 111
 UoD 44

V

values 65
 view 174
 virtual object-oriented database (VOODB) simulator 307
 VOOB simulator (see virtual object-oriented database simulator)

W

Web Ontology Language (OWL) 43, 54
 Web Service Definition Language (WSDL) 189
 Web services 97, 188
 Web Services Business Process Execution Language (WS-BPEL) 95
 Web-based ontology specification language 48
 weltanschauung 45
 workflow processes 150
 WS-BPEL (see Web Services Business Process Execution Language)
 WSDL (see Web Service Definition Language)

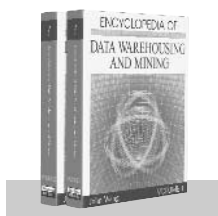
X

XABSL (see eXtensible Agent Behaviour Specification Language)
 XMI (see XML metadata interchange)
 XMI interchange 36
 XML (see eXtensible Markup Language)
 XML metadata interchange (XMI) 5, 28, 30, 36

New Releases from Idea Group Reference

**Idea Group
REFERENCE**

The Premier Reference Source for Information Science and Technology Research

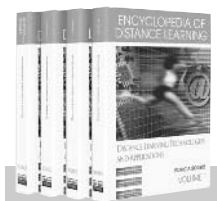


ENCYCLOPEDIA OF DATA WAREHOUSING AND MINING

Edited by: John Wang,
Montclair State University, USA

Two-Volume Set • April 2005 • 1700 pp
ISBN: 1-59140-557-2; US \$495.00 h/c
Pre-Publication Price: US \$425.00*
*Pre-pub price is good through one month
after the publication date

- Provides a comprehensive, critical and descriptive examination of concepts, issues, trends, and challenges in this rapidly expanding field of data warehousing and mining
- A single source of knowledge and latest discoveries in the field, consisting of more than 350 contributors from 32 countries
- Offers in-depth coverage of evolutions, theories, methodologies, functionalities, and applications of DWM in such interdisciplinary industries as healthcare informatics, artificial intelligence, financial modeling, and applied statistics
- Supplies over 1,300 terms and definitions, and more than 3,200 references



ENCYCLOPEDIA OF DISTANCE LEARNING

Four-Volume Set • April 2005 • 2500+ pp
ISBN: 1-59140-555-6; US \$995.00 h/c
Pre-Pub Price: US \$850.00*
*Pre-pub price is good through one
month after the publication date

- More than 450 international contributors provide extensive coverage of topics such as workforce training, accessing education, digital divide, and the evolution of distance and online education into a multibillion dollar enterprise
- Offers over 3,000 terms and definitions and more than 6,000 references in the field of distance learning
- Excellent source of comprehensive knowledge and literature on the topic of distance learning programs
- Provides the most comprehensive coverage of the issues, concepts, trends, and technologies of distance learning

ENCYCLOPEDIA OF INFORMATION SCIENCE AND TECHNOLOGY AVAILABLE NOW!



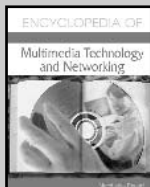
Five-Volume Set • January 2005 • 3807 pp
ISBN: 1-59140-553-X; US \$1125.00 h/c

ENCYCLOPEDIA OF DATABASE TECHNOLOGIES AND APPLICATIONS



April 2005 • 650 pp
ISBN: 1-59140-560-2; US \$275.00 h/c
Pre-Publication Price: US \$235.00*
*Pre-publication price good through
one month after publication date

ENCYCLOPEDIA OF MULTIMEDIA TECHNOLOGY AND NETWORKING



April 2005 • 650 pp
ISBN: 1-59140-561-0; US \$275.00 h/c
Pre-Publication Price: US \$235.00*
*Pre-pub price is good through
one month after publication date

www.idea-group-ref.com

Idea Group Reference is pleased to offer complimentary access to the electronic version
for the life of edition when your library purchases a print copy of an encyclopedia

For a complete catalog of our new & upcoming encyclopedias, please contact:
701 E. Chocolate Ave., Suite 200 • Hershey PA 17033, USA • 1-866-342-6657 (toll free) • cust@idea-group.com



Experience the latest full-text research in the fields
of Information Science, Technology & Management

InfoSci-Online

InfoSci-Online is available to libraries to help keep students, faculty and researchers up-to-date with the latest research in the ever-growing field of information science, technology, and management.

The InfoSci-Online collection includes:

- Scholarly and scientific book chapters
- Peer-reviewed journal articles
- Comprehensive teaching cases
- Conference proceeding papers
- All entries have abstracts and citation information
- The full text of every entry is downloadable in .pdf format

Some topics covered:

- Business Management
- Computer Science
- Education Technologies
- Electronic Commerce
- Environmental IS
- Healthcare Information Systems
- Information Systems
- Library Science
- Multimedia Information Systems
- Public Information Systems
- Social Science and Technologies

**InfoSci-Online
features:**

- Easy-to-use
- 6,000+ full-text entries
- Aggregated
- Multi-user access

"...The theoretical bent of many of the titles covered, and the ease of adding chapters to reading lists, makes it particularly good for institutions with strong information science curricula."

— Issues in Science and
Technology Librarianship



To receive your free 30-day trial access subscription contact:

Andrew Bundy

Email: abundy@idea-group.com • Phone: 717/533-8845 x29

Web Address: www.infosci-online.com

InfoSci-Online

Full Text • Cutting Edge • Easy Access

A PRODUCT OF  **IDEA GROUP INC.**

Publishers of Idea Group Publishing, Information Science Publishing, CyberTech Publishing, and IIRM Press

infosci-online.com