ZONGMIN MA

Database Modeling for Industrial Data Management

Emerging Technologies and Applications

Database Modeling for Industrial Data Management: Emerging Technologies and Applications

Zongmin Ma
Northeastern University, China

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:	Susanna Svidunovich
Typesetter:	Amanda Kirlin
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
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

Library of Congress Cataloging-in-Publication Data

Database modeling for industrial data management : emerging technologies
and applications / Zongmin Ma, editor.
p. cm.
Summary: "This book covers industrial databases and applications and
offers generic database modeling techniques"--Provided by publisher.
Includes bibliographical references and index.
ISBN 1-59140-684-6 (hardcover) -- ISBN 1-59140-685-4 (softcover)
-- ISBN 1-59140-686-2 (ebook)
1. Industrial management--Technological innovations. 2. Relational
databases. 3. Database design. I. Ma, Zongmin, 1965- .
HD45.D327 2005
005.75'6--dc22
2005023883

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

Database Modeling for Industrial Data Management: Emerging Technologies and Applications

Table of Contents

Preface

Computer-based information technologies have been extensively used to help industries manage their processes, and information systems hereby become their nervous center. More specifically, databases are designed to support the data storage, processing, and retrieval activities related to data management in information systems. Database management systems provide efficient task support, and database systems are the key to implementing industrial data management. Industrial data management requires database technical support. Industrial applications, however, are typically data- and knowledge-intensive and have some unique characteristics (e.g., large volumes of data with complex structures) that make them difficult to manage. Some new techniques such as the Web, artificial intelligence, and so forth have been introduced into industrial applications. These unique characteristics and the usage of new technologies have put many potential requirements on industrial data management, which challenges today's database systems and promotes their evolvement.

Viewed from database technology, information modeling in databases (database modeling for short) can be identified at two levels: conceptual data modeling and database modeling. This results in conceptual (semantic) data model and logical database model. Generally, a conceptual data model is designed, then the designed conceptual data model will be transformed into a chosen logical database schema. Database systems based on logical database models are used to build information systems for data management. Much attention has been directed at conceptual data modeling of industrial information systems. Product data models, for example, can be viewed as a class of semantic data models (i.e., conceptual data models) that take into account the needs of engineering data. Recently, conceptual data modeling of enterprises has received increasing attention. Generally speaking, traditional ER/EER or

UML models in database areas can be used for industrial data modeling at the conceptual level. But, limited by their power in industrial data modeling, some new conceptual data models such as IDEF1X and STEP/EXPRESS have been developed. In particular, to implement share and exchange of industrial data, the Standard for the Exchange of Product Model Data (STEP) is being developed by the International Organization for Standardization (ISO). EXPRESS is the description methods of STEP and a conceptual schema language, which can model product design, manufacturing, and production data. EXPRESS model hereby becomes a major one of conceptual data models for industrial data modeling. Many research works have been reported on the database implementation of the EXPRESS model in context of STEP, and some software packages and tools are available in the marketplace. For industrial data modeling in database systems, the generic logical database models such as relational, nested relational, and object-oriented databases have been used. However, these generic logical database models do not always satisfy the requirements of industrial data management. In non-transaction processing such as CAD/CAM, knowledge-based system, multimedia and Internet systems, for example, most of these data-intensive application systems suffer from the same limitations of relational databases. Some non-traditional database models based on special, hybrid, and/or the extended database models above have been proposed accordingly.

Database technology is typically application-oriented. With advances and in-depth applications of computer technologies in industry, database modeling for industrial data management is emerging as a new discipline. The research and development of industrial databases is receiving increasing attention. By means of database technology, large volumes of industrial data with complex structures can be modeled in conceptual data models and further stored in databases. Industrial information systems based the databases can handle and retrieve these data to support various industrial activities. Therefore, database modeling for industrial data management is a field which must be investigated by academic researchers, together with developers and users both from database and industry areas.

Introduction

This book, which consists of 11 chapters, is organized into two major sections. The first section discusses the issues of industrial databases and applications in the first nine chapters. The next two chapters covering the data modeling issue in generic databases comprise the second section.

First of all, we take a look at the problems of the industrial databases and applications.

Databases are designed to support data storage, processing, and retrieval activities related to data management, and database systems are the key to implementing engineering information modeling. But some engineering requirements challenge current mainstream databases, which are mainly used for business applications, and promote their evolvement. Ma tries to identify the requirements for engineering information modeling and then investigates the satisfactions of current database models to these requirements at two levels: conceptual data models and logical database models. Also, the relationships among the conceptual data models and the logical database models for engineering information modeling are presented as viewed from database conceptual design.

ASSO is a database design methodology defined for achieving conceptual schema consistency, logical schema correctness, flexibility in reflecting the real-life changes on the schema, and efficiency in accessing and storing information. B is an industrial formal method for specifying, designing, and coding software systems. Locuratolo investigates the integration of the ASSO features in B. Starting from a B specification of the data structure and of the transactions allowed on a database, two model transformations are designed: The resulting model *Structured Database Schema* integrates static and dynamics, exploiting the novel concepts of *Class-Machines* and *Specialized Class-Machines*. Formal details which must be specified if the conceptual model of ASSO is directly constructed in B are avoided; the costs of the consistency obligations are minimized. Class-Machines supported by semantic data models can be correctly linked with Class-Machines supported by object models.

Carnduff and Goonetillake present research aimed at determining the requirements of a database software tool that supports integrity validation of versioned design artifacts through effective management of evolving constraints. It results in the design and development of a constraint management model, which allows constraint evolution through representing constraints within versioned objects called Constraint Versions Objects (CVOs). This model operates around a version model that uses a well-defined configuration management strategy to manage the versions of complex artifacts. Internal and interdependency constraints are modeled in CVOs. They develop a model which has been implemented in a prototype database tool with an intuitive user interface.

The user interface allows designers to manage design constraints without the need to program. Also, they introduce the innovative concepts developed using an ongoing example of a simple bicycle design.

Similarity search in database systems is an important task in modern application domains such as multimedia, molecular biology, medical imaging and many others. Especially for CAD (Computer-Aided Design), suitable similarity models and a clear representation of the results can help to reduce the cost of developing and producing new parts by maximizing the reuse of existing parts. Kriegel, Kröger, Pfeifle, Brecheisen, Pötke, Schubert, and Seidl present different similarity models for voxelized CAD data based on space partitioning and data partitioning. Based on these similarity models, they introduce an industrial prototype, called BOSS, which helps the user to get an overview over a set of CAD objects. BOSS allows the user to easily browse large data collections by graphically displaying the results of a hierarchical clustering algorithm.

STEP-NC is an emerging ISO standard, which defines a new generation of NC programming language and is fully compliant with STEP. There is a whole suite of implementation methods one may utilize for development purposes. STEP-NC brings richer information to the numerically-controlled machine tools; hence intelligent machining and control are made possible. Its Web-enabled feature gives itself an additional dimension in that e-manufacturing can be readily supported. Xu addresses the issue of product development chain from the perspective of data modeling and streamlining. The focus is on STEP-NC, and how it may close the gap between design and manufacturing for a complete, integrated product development environment. A case study is given to demonstrate a STEP compliant, Web-enabled manufacturing system.

Yuan shares his experience of enabling semantic-based dynamic information integration across multiple heterogeneous information sources. While data is physically stored in existing legacy data systems across the networks, the information is integrated based upon its semantic meanings. Ontology is used to describe the semantics of global information content, and semantic enhancement is achieved by mapping the local metadata onto the ontology. For better system reliability, a unique mechanism is introduced to perform appropriate adjustments upon detecting environmental changes.

Panagis, Sakkopoulos, Sioutas, and Tsakalidis present the Web Service architecture and propose Web Service integration and management strategies for large-scale datasets. They mainly present the elements of Web Service architecture, the challenges in implementing Web Services whenever large-scale data are involved, and the design decisions and business process re-engineering steps to integrate Web Services in an enterprise information system. Then they provide a case study involving the largest private-sector telephony provider in Greece, where the provider's billing system datasets is utilized. Moreover, they present the scientific work on Web Service discovery along with experiments on implementing an elaborate discovery strategy over real-world, large-scale data.

Bose, Chun, Yue, Ines, and Helen describe the planning and implementation of the Wal-Mart data warehouse and discuss its integration with the operational systems. They also highlight some of the problems encountered in the developmental process of the data warehouse. The implications of the recent advances in technologies such as RFID, which is likely to play an important role in the Wal-Mart data warehouse in future, is also detailed.

Content-based image retrieval (CBIR) can be used to locate medical images in large databases using image features, such as color and texture, to index images with minimal human intervention. Wei, Li, and Wilson introduce a content-based approach to medical image retrieval. First, they introduce the fundamentals of the key components of content-based image retrieval systems are to give an overview of this area. Then they present a case study, which describes the methodology of a CBIR system for retrieving digital mammogram database.

In the second section, we see the generic database modeling.

A strong design phase is involved in most current application development processes (e.g., ER design for relational databases). But conceptual design for XML has not been explored significantly in literature or in practice. Most XML design processes start by directly marking up data in XML, and the metadata is typically designed at the time of encoding the documents. So Mohan and Sengupta introduce the existing methodologies for modeling XML. A discussion is presented comparing and contrasting their capabilities and deficiencies, and delineating the future trend in conceptual design for XML applications.

Ravat, Teste, and Zurfluh focus on constraint-based multi-dimensional modeling. The defined model integrates a constellation of facts and dimensions. Along each dimension, various hierarchies are possibly defined and the model supports multiple instantiations of dimensions. To facilitate data querying, they also define a multi-dimensional query algebra, which integrates the main multi-dimensional operators. These operators support the constraint-based multi-dimensional modeling. Finally, they present two implementations of this algebra, which are OLAP-SQL and a graphical query language. The former is a textual language integrating multi-dimensional concepts (fact, dimension, hierarchy), but it is based on classical SQL syntax. This language is dedicated to specialists such as multi-dimensional database administrators. The latter consists in a graphical representation of multi-dimensional databases and users specify directly their queries over this graph. This approach is dedicated to non-computer scientist users.

Acknowledgments

The editor wishes to thank all of the authors for their insights and excellent contributions to this book, and would like to acknowledge the help of all involved in the collation and review process of the book, without whose support the project could not have been satisfactorily completed. Most of the authors of chapters included in this book also served as referees for papers written by other authors. Thanks go to all those who provided constructive and comprehensive reviews.

A further special note of thanks goes also to all the staff at Idea Group Inc., whose contributions throughout the whole process from inception of the initial idea to final publication have been invaluable. Special thanks also go to the publishing team at Idea Group Inc. — in particular to Mehdi Khosrow-Pour, whose enthusiasm motivated me to initially accept his invitation for taking on this project, and to Michele Rossi, who continuously prodded via e-mail for keeping the project on schedule. This book would not have been possible without the ongoing professional support from Mehdi Khosrow-Pour and Jan Travers at Idea Group Inc.

The idea of editing this volume stems from the initial research work that the editor did in the past several years. The assistances and facilities of University of Saskatchewan and Université de Sherbrooke, Canada, Oakland University and Wayne State University, USA, and City University of Hong Kong and Northeastern University, China, are deemed important, and are highly appreciated.

Finally, the editor wishes to thank his family for their patience, understanding, encouragement, and support when the editor needed to devote many time in the edition of this book. This book will not be completed without their love.

Zongmin Ma, PhD
Shenyang, China
May 2005

Section I: Industrial Databases and Applications

Chapter I

Databases Modeling of Engineering Information

Z. M. Ma, Northeastern University, China

Abstract

Information systems have become the nerve center of current computer-based engineering applications, which hereby put the requirements on engineering information modeling. Databases are designed to support data storage, processing, and retrieval activities related to data management, and database systems are the key to implementing engineering information modeling. It should be noted that, however, the current mainstream databases are mainly used for business applications. Some new engineering requirements challenge today's database technologies and promote their evolvement. Database modeling can be classified into two levels: conceptual data modeling and logical database modeling. In this chapter, we try to identify the requirements for engineering information modeling and then investigate the satisfactions of current database models to these requirements at two levels: conceptual data models and logical database models. In addition, the relationships among the conceptual data models and the logical database models for engineering information modeling are presented in the chapter viewed from database conceptual design.

Introduction

To increase product competitiveness, current manufacturing enterprises have to deliver their products at reduced cost and high quality in a short time. The change from sellers' market to buyers' market results in a steady decrease in the product life cycle time and the demands for tailor-made and small-batch products. All these changes require that manufacturing enterprises quickly respond to market changes. Traditional production patterns and manufacturing technologies may find it difficult to satisfy the requirements of current product development. Many types of advanced manufacturing techniques, such as Computer Integrated Manufacturing (CIM), Agile Manufacturing (AM), Concurrent Engineering (CE), and Virtual Enterprise (VE) based on global manufacturing have been proposed to meet these requirements. One of the foundational supporting strategies is the computer-based information technology. Information systems have become the nerve center of current manufacturing systems. So some new requirements on information modeling are introduced.

Database systems are the key to implementing information modeling. Engineering information modeling requires database support. Engineering applications, however, are data- and knowledge- intensive applications. Some unique characteristics and usage of new technologies have put many potential requirements on engineering information modeling, which challenge today's database systems and promote their evolvement. Database systems have gone through the development from hierarchical and network databases to relational databases. But in non-transaction processing such as CAD/CAPP/CAM (computer-aided design/computer-aided process planning/computer-aided manufacturing), knowledge-based system, multimedia and Internet systems, most of these data-intensive application systems suffer from the same limitations of relational databases. Therefore, some non-traditional data models have been proposed. These data models are fundamental tools for modeling databases or the potential database models. Incorporation between additional semantics and data models has been a major goal for database research and development.

Focusing on engineering applications of databases, in this chapter, we identify the requirements for engineering information modeling and investigate the satisfactions of current database models to these requirements. Here we differentiate two levels of database models: conceptual data models and logical database models. Constructions of database models for engineering information modeling are hereby proposed.

The remainder of the chapter is organized as follows: The next section identifies the generic requirements of engineering information modeling. The issues that current databases satisfy these requirements are then investigated in the third section. The fourth section proposes the constructions of database models. The final section concludes this chapter.

Needs for Engineering Information Modeling

Complex Objects and Relationships

Engineering data have complex structures and are usually large in volume. But engineering design objects and their components are not independent. In particular, they are generally organized into taxonomical hierarchies. The specialization association is the well-known association. Also the part-whole association, which relates components to the compound of which they are part, is another key association in engineering settings.

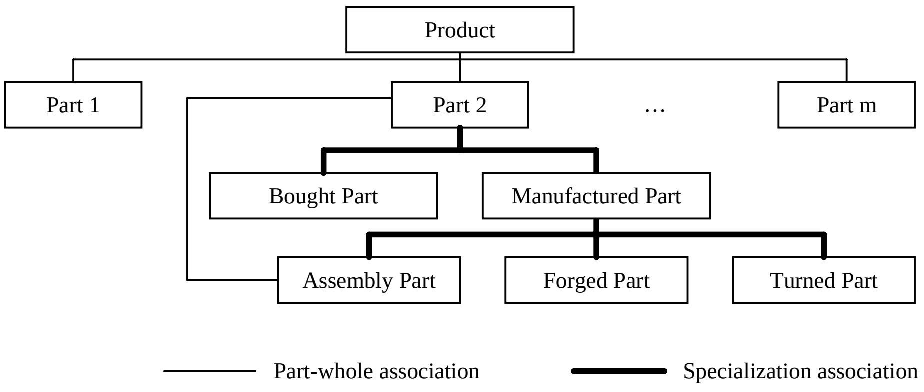
In addition, the position relationships between the components of design objects and the configuration information are typically multi-dimensional. Also, the information of version evolution is obviously time-related. All these kinds of information should be stored. It is clear that spatio-temporal data modeling is essential in engineering design (Manwaring, Jones, & Glagowski, 1996).

Typically, product modeling for product family and product variants has resulted in product data models, which define the form and content of product data generated through the product lifecycle from specification through design to manufacturing. Products are generally complex (see Figure 1, which shows a simple example of product structure) and product data models should hereby have advanced modeling abilities for unstructured objects, relationships, abstractions, and so on (Shaw, Bloor, & de Pennington, 1989).

Data Exchange and Share

Engineering activities are generally performed across departmental and organization boundaries. Product development based on virtual enterprises, for

Figure 1. An example illustration of product structure



example, is generally performed by several independent member companies that are physically located at different places. Information exchange and share among them is necessary. It is also true in different departments or even in different groups within a member company. Enterprise information systems (EISs) in manufacturing industry, for example, typically consist of supply chain management (SCM), enterprise resource planning (ERP) (Ho, Wu, & Tai, 2004), and CAD/CAPP/CAM. These individual software systems need to share and exchange product and production information in order to effectively organize production activities of enterprise. However, they are generally developed independently. In such an environment of distributed and heterogeneous computer-based systems, exchanging and sharing data across units are very difficult. An effective means must be provided so that the data can be exchanged and shared among deferent applications and enterprises. Recently, the PDM (product data management) system (CIMdata, 1997) is being extensively used to integrate both the engineering data and the product development process throughout the product lifecycle, although the PDM system also has the problem of exchanging data with ERP.

Web-Based Applications

Information systems in today's manufacturing enterprises are distributed. Data exchange and share can be performed by computer network systems. The Internet is a large and connected network of computers, and the World Wide Web (WWW) is the fastest growing segment of the Internet. Enterprise operations go increasingly global, and Web-based manufacturing enterprises can not only obtain online information but also organize production activities. Web technology facilitates cross-enterprise information sharing through interconnectivity and integration, which can connect enterprises to their strategic partners as well as to their customers. So Web-based virtual enterprises (Zhang, Zhang, & Wang, 2000), Web-based PDM (Chu & Fan, 1999; Liu & Xu, 2001), Web-based concurrent engineering (Xue & Xu, 2003), Web-based supply chain management, and Web-based B2B e-commerce for manufacturing (Fensel et al., 2001; Shaw, 2000a, 2000b; Soliman & Youssef, 2003; Tan, Shaw, & Fulkerson, 2000) are emerging. A comprehensive review was given of recent research on developing Web-based manufacturing systems in Yang and Xue (2003).

The data resources stored on the Web are very rich. In addition to common types of data, there are many special types of data such as multimedia data and hypertext link, which are referred to as semi-structured data. With the recent popularity of the WWW and informative manufacturing enterprises, how to model and manipulate semi-structured data coming from various sources in manufacturing databases is becoming more and more important. Web-based applications, including Web-based supply chain management, B2B e-commerce, and PDM systems, have been evolved from information publication to information share and exchange. HTML-based Web application cannot satisfy such requirements.

Intelligence for Engineering

Artificial intelligence and expert systems have extensively been used in many engineering activities such as product design, manufacturing, assembly, fault diagnosis, and production management. Five artificial intelligence tools that are most applicable to engineering problems were reviewed in Pham and Pham (1999), which are *knowledge-based systems*, *fuzzy logic*, *inductive learning*, *neural networks*, and *genetic algorithms*. Each of these tools was outlined in the paper together with examples of their use in different branches of engineering. In Issa, Shen, and Chew (1994), an expert system that applies analogical reasoning to mechanism design was developed. Based on fuzzy logic, an integration of financial and strategic justification approaches was proposed for manufacturing in Chiadamrong (1999).

Imprecision and Uncertainty

Imprecision is most notable in the early phase of the design process and has been defined as the choice between alternatives (Antonsoon & Otto, 1995). Four sources of imprecision found in engineering design were classified as *relationship imprecision*, *data imprecision*, *linguistic imprecision*, and *inconsistency imprecision* in Giachetti et al. (1997). In addition to engineering design, imprecise and uncertain information can be found in many engineering activities. The imprecision and uncertainty in activity control for product development was investigated in Grabot and Geneste (1998). To manage the uncertainty occurring in industrial firms, the various types of buffers were provided in Caputo (1996) according to different types of uncertainty faced and to the characteristics of the production system. Buffers are used as alternative and complementary factors to attain technological flexibility when a firm is unable to achieve the desired level of flexibility and faces uncertainty. Nine types of flexibility (*machine*, *routing*, *material handling system*, *product*, *operation*, *process*, *volume*, *expansion*, and *labor*) in manufacturing were summarized in Tsourveloudis and Phillis (1998).

Concerning the representation of imprecision and uncertainty, attempts have been made to address the issue of imprecision and inconsistency in design by way of intervals (Kim et al., 1995). Other approaches to representing imprecision in design include using utility theory, implicit representations using optimization methods, matrix methods such as Quality Function Deployment, probability methods, and necessity methods. An extensive review of these approaches was provided in Antonsoon and Otto (1995). These methods have all had limited success in solving design problems with imprecision. It is believed that fuzzy reorientation of imprecision will play an increasingly important role in design systems (Zimmermann, 1999).

Fuzzy set theory (Zadeh, 1965) is a generalization of classical set theory. In normal set theory, an object may or may not be a member of a set. There are only two states. Fuzzy sets contain elements to a certain degree. Thus, it is possible to represent an object that has partial membership in a set. The membership value of element u in a fuzzy set is represented by $\mu(u)$ and is normalized such that $\mu(u)$ is in [0, 1]. Formally, let F be a fuzzy set in a universe of discourse U and $\mu_F: U \rightarrow [0, 1]$ be the membership function for the fuzzy set F. Then the fuzzy set F is described as:

$F = \{\mu(u_1)/u_1, \mu(u_2)/u_2, ..., \mu(u_n)/u_n\}$, where $u_i \in U$(i = 1, 2, …, n).

Fuzzy sets can represent linguistic terms and imprecise quantities and make systems more flexible and robust. So fuzzy set theory has been used in some engineering applications (e.g., engineering/product design and manufacturing, production management, manufacturing flexibility, e-manufacturing, etc.), where, either crisp information is not available or information flexible processing is necessary.

1. Concerning engineering/product design and manufacturing, the needs for fuzzy logic in the development of CAD systems were identified and how fuzzy logic could be used to model aesthetic factors was discussed in Pham (1998). The development of an expert system with production rules and the integration of fuzzy techniques (fuzzy rules and fuzzy data calculus) was described for the preliminary design in Francois and Bigeon (1995). Integrating knowledge-based methods with multi-criteria decision-making and fuzzy logic, an approach to engineering design and configuration problems was developed in order to enrich existing design and configuration support systems with more intelligent abilities in Muller and Sebastian (1997). A methodology for making the transition from imprecise goals and requirements to the precise specifications needed to manufacture the product was introduced using fuzzy set theory in Giachetti et al. (1997). In Jones and Hua (1998), an approach to engineering design in which fuzzy sets were used to represent the range of variants on existing mechanisms was described so that novel requirements of engineering design could be met. A method for design candidate evaluation and identification using neural network-based fuzzy reasoning was presented in Sun, Kalenchuk, Xue, and Gu (2000).
2. In production management, the potential applications of fuzzy set theory to new product development; facility location and layout; production scheduling and control; inventory management; and quality and cost-benefit analysis were identified in Karwowski and Evans (1986). A comprehensive literature survey on fuzzy set applications in product management research was given in Guiffrida and Nagi (1998). A classification scheme for fuzzy applications in product management research was defined in their paper, including job shop scheduling; quality management; project scheduling; facilities location and layout; aggregate planning; production and inventory planning; and forecasting.

3. In manufacturing domain, flexibility is an inherently vague notion. So fuzzy logic was introduced and a fuzzy knowledge-based approach was used to measure manufacturing flexibility (Tsourveloudis & Phillis, 1998).
4. More recently, the research on supply chain management and electronic commerce have also shown that fuzzy set can be used in customer demand, supply deliveries along the supply chain, external or market supply, targeted marketing, and product category description (Petrovic, Roy, & Petrovic, 1998, 1999; Yager, 2000; Yager & Pasi, 2001).

It is believed that fuzzy set theory has considerable potential for intelligent manufacturing systems and will be employed in more and more engineering applications.

Knowledge Management

Engineering application is a knowledge-intensive application. Knowledge-based managements have covered the whole activities of current enterprises (O'Leary, 1998; Maedche et al., 2003; Wong, 2005), including manufacturing enterprises (Michael & Khemani, 2002). In Tan and Platts (2004), the use of the connectance concept for managing manufacturing knowledge was proposed. A software tool called Tool for Action Plan Selection (TAPS) has been developed based on the connectance concept, which enables managers to sketch and visualize their knowledge of how variables interact in a connectance network. Based on the computer-integrated manufacturing open-system architecture reference model (CIMOSA), a formalism was presented in de Souza, Ying, and Yang (1998) to specify the business processes and enterprise activities at the knowledge level. The formalism used an integration of multiple types of knowledge, including precise, muddy, and random symbolic and numerical knowledge to systematically represent enterprise behavior and functionality. Instead of focusing on individual human knowledge, as in Thannhuber, Tseng, and Bullinger (2001), the ability of an enterprise to dynamically derive processes to meet the external needs and internal stability was identified as the organizational knowledge. On the basis, a knowledge management system has been developed.

The management of engineering knowledge entails its modeling, maintenance, integration, and use (Ma & Mili, 2003; Mili et al., 2001). Knowledge modeling consists of representing the knowledge in some selected language or notation. Knowledge maintenance encompasses all activities related to the validation, growth, and evolution of the knowledge. Knowledge integration is the synthesis of knowledge from related sources. The use of the knowledge requires bridging the gap between the objective expressed by the knowledge and the directives needed to support engineering activities.

It should be noticed that Web-based engineering knowledge management has emerged because of Web-based engineering applications (Caldwell et al., 2000). In addition, engineering knowledge is closely related to engineering data, although they are different. Engineering knowledge is generally embedded in engineering data. So it is necessary to synthetically manage engineering knowledge and data in bases (Xue, Yadav, & Norrie, 1999; Zhang & Xue, 2002). Finally, the field of artificial intelligence (AI) is usually concerned with the problems caused by imprecise and uncertain information (Parsons, 1996). Knowledge representation is one of the most basic and active research areas of AI. The conventional approaches to knowledge representation, however, only support exact rather than approximate reasoning, and fuzzy logic is apt for knowledge representation (Zadeh, 1989). Fuzzy rules (Dubois & Prade, 1996) and fuzzy constraints (Dubois, Fargier, & Prade, 1996) have been advocated and employed as a key tool for expressing pieces of knowledge in fuzzy logic. In particular, fuzzy constraint satisfaction problem (FCSP) has been used in many engineering activities such as design and optimization (Dzbor, 1999; Kapadia & Fromherz, 1997; Young, Giachetti, & Ress, 1996) as well as planning and scheduling (Dubois, Fargier, & Prade, 1995; Fargier & Thierry, 1999; Johtela et al., 1999).

Data Mining and Knowledge Discovery

Engineering knowledge plays a crucial role in engineering activities. But engineering knowledge is not always represented explicitly. Data mining and knowledge discovery from databases (KDD) can extract information characterized as "knowledge" from data that can be very complex and in large quantities. So the field of data mining and knowledge discovery from databases has emerged as a new discipline in engineering (Gertosio & Dussauchoy, 2004) and now is extensively studied and applied in many industrial processes. In Ben-Arieh, Chopra, and Bleyberg (1998), data mining application for real-time distributed shop-floor control was presented. With a data mining approach, the prediction problem encountered in engineering design was solved in Kusiak and Tseng (2000). Furthermore, the data mining issues and requirements within an enterprise were examined in Kleissner (1998).

With the huge amount of information available online, the World Wide Web is a fertile area for data mining research. The Web mining research is at the crossroads of research from several research communities such as database, information retrieval, and within AI, especially the sub-areas of machine learning and natural language processing (Kosala & Blockeel, 2000). In addition, soft computing methodologies (involving fuzzy sets, neural networks, genetic algorithms, and rough sets) are most widely applied in the data mining step of the overall KDD process (Mitra, Pal, & Mitra, 2002). Fuzzy sets provide a natural framework for the process in dealing with uncertainty. Neural networks and rough sets are widely used for classification and rule generation. Genetic algorithms (GAs) are involved in various optimization and search processes, like query optimization and template selection. Particularly, a review of Web Mining in Soft Computing Framework was given in Pal, Talwar, and Mitra (2002).

Current Database Models

Engineering information modeling in databases can be carried out at two different levels: conceptual data modeling and logical database modeling. Therefore, we have conceptual data models and logical database models for engineering information modeling, respectively. In this chapter, database models for engineering information modeling refer to conceptual data models and logical database models simultaneously. Table 1 gives some conceptual data models and logical database models that may be applied for engineering information modeling. The following two sub-sections give the more detailed explanations about these models.

Conceptual Data Models

Much attention has been directed at conceptual data modeling of engineering information (Mannisto et al., 2001; McKay, Bloor, & de Pennington, 1996). Product data models, for example, can be viewed as a class of semantic data models (i.e., conceptual data models) that take into account the needs of engineering data (Shaw, Bloor, & de Pennington, 1989). Recently, conceptual information modeling of enterprises such as virtual enterprises has received increasing attention (Zhang & Li, 1999). Generally speaking, traditional ER

Table 1. Database models for engineering information modeling

Database Models					
Conceptual Data Models		***Logical Database Models***			
Generic Conceptual Data Models	Specific Conceptual Data Models for Engineering	Classical Logical Database Models	XML Databases	Specific & Hybrid Database Models	Extended Database Models
• ER data model • EER data model • UML data model • XML data model	• IDEF1X data model • EXPRESS data model	• Relational databases • Nested relational databases • Object-oriented databases • Object-relational databases	• Classical logical databases • Native XML databases	• Active databases • Deductive databases • Constraint databases • Spatio-temporal databases • Object-oriented active databases • Deductive object-relational databases …	• Fuzzy relational databases • Fuzzy nested relational databases • Fuzzy object-oriented databases • Deductive fuzzy relational databases …

(entity-relationship) and EER (extended entity-relationship) can be used for engineering information modeling at conceptual level (Chen, 1976). But limited by their power in engineering modeling, some improved conceptual data models have been developed.

IDEF1X is a method for designing relational databases with a syntax designed to support the semantic constructs necessary in developing a conceptual schema. Some research has focused on the IDEF1X methodology. A thorough treatment of the IDEF1X method can be found in Wizdom Systems Inc. (1985). The use of the IDEF1X methodology to build a database for multiple applications was addressed in Kusiak, Letsche, and Zakarian (1997).

In order to share and exchange product data, the Standard for the Exchange of Product Model Data (STEP) is being developed by the International Organization for Standardization (ISO). STEP provides a means to describe a product model throughout its life cycle and to exchange data between different units. STEP consists of four major categories, which are *description methods*, *implementation methods*, *conformance testing methodology and framework*, and *standardized application data models/schemata,* respectively. EXPRESS (Schenck & Wilson, 1994), as the description methods of STEP and a conceptual schema language, can model product design, manufacturing, and production data. EXPRESS model hereby becomes one of the major conceptual data models for engineering information modeling.

With regard to CAD/CAM development for product modeling, a review was conducted in Eastman and Fereshetian (1994), and five information models used in product modeling, namely, ER, NAIM, IDEF1X, EXPRESS and EDM, were studied. Compared with IDEF1X, EXPRESS can model complex semantics in engineering application, including engineering objects and their relationships. Based on EXPRESS model, it is easy to implement share and exchange engineering information.

It should be noted that ER/EER, IDEF1X and EXPRESS could model neither knowledge nor fuzzy information. The first effort was done in Zvieli and Chen (1996) to extend ER model to represent three levels of fuzziness. The first level refers to the set of semantic objects, resulting in fuzzy entity sets, fuzzy relationship sets and fuzzy attribute sets. The second level concerns the occurrences of entities and relationships. The third level is related to the fuzziness in attribute values of entities and relationships. Consequently, ER algebra was fuzzily extended to manipulate fuzzy data. In Chen and Kerre (1998), several major notions in EER model were extended, including fuzzy extension to generalization/specialization, and shared subclass/category as well as fuzzy multiple inheritance, fuzzy selective inheritance, and fuzzy inheritance for derived attributes. More recently, using fuzzy sets and possibility distribution (Zadeh, 1978), fuzzy extensions to IDEF1X and EXPRESS were proposed in Ma, Zhang, and Ma (2002) and Ma (in press), respectively.

UML (Unified Modeling Language) (Booch, Rumbaugh, & Jacobson, 1998; OMG, 2003), being standardized by the Object Management Group (OMG), is a set of OO modeling notations. UML provides a collection of models to capture many aspects of a software system. From the information modeling point of view, the most relevant model is the class model. The building blocks in this class model are those of classes and relationships. The class model of UML encompasses the concepts used in ER, as well as other OO concepts. In addition, it also presents the advantage of being open and extensible, allowing its adaptation to the specific needs of the application such as workflow modeling of e-commerce (Chang et al., 2000) and product structure mapping (Oh, Hana, & Suhb, 2001). In particular, the class model of UML is extended for the representation of class constraints and the introduction of stereotype associations (Mili et al., 2001).

With the popularity of Web-based design, manufacturing, and business activities, the requirement has been put on the exchange and share of engineering information over the Web. XML (eXtensible Markup Language), created by the World Wide Web Consortium, lets information publishers invent their own tags for particular applications or work with other organizations to define shared sets of tags that promote interoperability and that clearly separate content and presentation. XML provides a Web-friendly and well-understood syntax for the exchange of data. Because XML impacts on data definition and share on the Web (Seligman & Rosenthal, 2001), XML technology has been increasingly studied, and more and more Web tools and Web servers are capable of supporting XML. In Bourret (2004), product data markup language, the XML for product data exchange and integration, has been developed. As to XML modeling at concept level, UML was used for designing XML DTD (document- type definition) in Conrad, Scheffner, and Freytag (2000). In Xiao et al. (2001), an object-oriented conceptual model was developed to design XML schema. ER model was used for conceptual design of semi-structured databases in Lee et al. (2001). But XML does not support imprecise and uncertain information modeling and knowledge modeling. Introducing imprecision and uncertainty into XML has increasingly become a topic of research (Abiteboul, Segoufin, & Vianu, 2001; Damiani, Oliboni, & Tanca, 2001; Ma, 2005).

Logical Database Models

Classical Logical Database Models

As to engineering information modeling in database systems, the generic logical database models such relational databases, nested relational databases, and object-oriented databases can be used. Also, some hybrid logical database models such as object-relational databases are very useful for this purpose.

In Ahmed (2004), the KSS (Kraftwerk Kennzeichen System) identification and classification system was used to develop database system for plant maintenance and management. On top of a relational DBMS, an EXPRESS-oriented information system was built in Arnalte and Scala (1997) for supporting information integration in a computer-integrated manufacturing environment. In this case, the conceptual model of the information was built in EXPRESS and then parsed and translated to the corresponding relational constructs. Relational databases for STEP/EXPRESS were also discussed in Krebs and Lührsen (1995). In addition, an object-oriented layer was developed in Barsalou and Wiederhold (1990) to model complex entities on top of a relational database. This domain-independent architecture permits object-oriented access to information stored in relational format-information that can be shared among applications.

Object-oriented databases provide an approach for expressing and manipulating complex objects. A prototype object-oriented database system, called ORION, was thus designed and implemented to support CAD (Kim et al., 1990). Object-oriented databases for STEP/EXPRESS have been studied in Goh et al. (1994, 1997). In addition, an object-oriented active database was also designed for STEP/EXPRESS models in Dong, Y. et al. (1997). According to the characteristics of engineering design, a framework for the classification of queries in object-oriented engineering databases was provided in Samaras, Spooner, and Hardwick (1994), where the strategy for query evaluation is different from traditional relational databases. Based on the comparison with relational databases, the selections and characteristics of the object-oriented database and database management systems (OODBMS) in manufacturing were discussed in Zhang (2001). The current studies and applications were also summarized.

XML Databases

It is crucial for Web-based applications to model, store, manipulate, and manage XML data documents. XML documents can be classified into data-centric documents and document-centric documents (Bourret, 2004). Data-centric documents are characterized by fairly regular structure, fine-grained data (i.e., the smallest independent unit of data is at the level of a PCDATA-only element or an attribute), and little or no mixed content. The order in which sibling elements and PCDATA occurs is generally not significant, except when validating the document. Data-centric documents are documents that use XML as a data transport. They are designed for machine consumption and the fact that XML is used at all is usually superfluous. That is, it is not important to the application or the database that the data is, for some length of time, stored in an XML document. As a general rule, the data in data-centric documents is stored in a traditional database, such as a relational, object-oriented, or hierarchical database. The data can also be transferred from a database to a XML document. For the transfers between XML documents and databases, the mapping relationships between their architectures as well as their data should be created (Lee & Chu, 2000; Surjanto, Ritter, & Loeser, 2000). Note that it is possible to discard some information such as the document and its physical structure when transferring data between them. It must be pointed out, however, that the data in data-centric documents such as semi-structured data can also be stored in a native XML database, in which a document-centric document is usually stored. Document-centric documents are characterized by less regular or irregular structure, larger-grained data (that is, the smallest independent unit of data might be at the level of an element with mixed content or the entire document itself), and lots of mixed content. The order in which sibling elements and PCDATA occurs is almost always significant. Document-centric documents are usually documents that are designed for human consumption. As a general rule, the documents in document-centric documents are stored in a native XML database or a content management system (an application designed to manage documents and built on top of a native XML database). Native XML databases are databases designed especially for storing XML documents. The only difference of native XML databases from other databases is that their internal model is based on XML and not something else, such as the relational model.

In practice, however, the distinction between data-centric and document-centric documents is not always clear. So the previously-mentioned rules are not of a certainty. Data, especially semi-structured data, can be stored in native XML databases, and documents can be stored in traditional databases when few XML-specific features are needed. Furthermore, the boundaries between traditional databases and native XML databases are beginning to blur, as traditional databases add native XML capabilities and native XML databases support the storage of document fragments in external databases.

In Seng, Lin, Wang, and Yu (2003), a technical review of XML and XML database technology, including storage method, mapping technique, and transformation paradigm, was provided and an analytic and comparative framework was developed. By collecting and compiling the IBM, Oracle, Sybase, and Microsoft XML database products, the framework was used and each of these XML database techniques was analyzed.

Special, Hybrid, and Extended Logical Database Models

It should be pointed out that, however, the generic logical database models such as relational databases, nested relational databases, and object-oriented databases do not always satisfy the requirements of engineering modeling. As pointed out in Liu (1999), relational databases do not describe the complex structure relationship of data naturally, and separate relations may result in data inconsistencies when updating the data. In addition, the problem of inconsistent data still exists in nested relational databases, and the mechanism of sharing and reusing CAD objects is not fully effective in object-oriented databases. In particular, these database models cannot handle engineering knowledge. Some special databases based on relational or object-oriented models are hereby introduced. In Dong and Goh (1998), an object-oriented active database for engineering application was developed to support intelligent activities in engineering applications. In Liu (1999), deductive databases were considered as the preferable database models for CAD databases, and deductive object-relational databases for CAD were introduced in Liu and Katragadda (2001). Constraint databases based on the generic logical database models are used to represent large or even infinite sets in a compact way and are suitable hereby for modeling spatial and temporal data (Belussi, Bertino, & Catania, 1998; Kuper, Libkin, & Paredaens, 2000). Also, it is well established that engineering design is a constraint-based activity (Dzbor, 1999; Guiffrida, & Nagi, 1998; Young, Giachetti, & Ress, 1996). So constraint databases are promising as a technology for modeling engineering information that can be characterized by large data in volume, complex relationships (structure, spatial and/or temporal semantics), intensive knowledge and so forth. In Posselt and Hillebrand (2002), the issue about constraint database support for evolving data in product design was investigated.

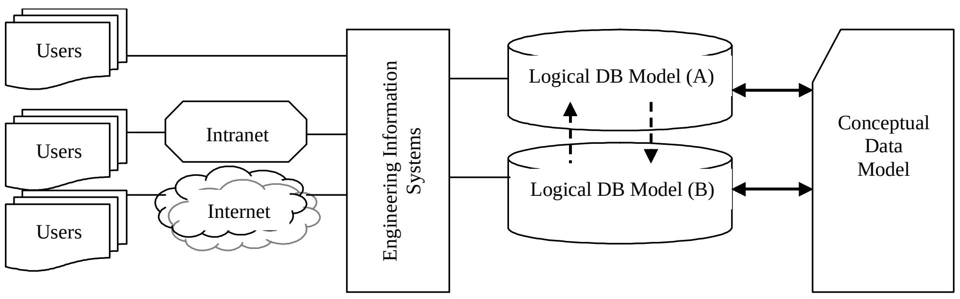
It should be noted that fuzzy databases have been proposed to capture fuzzy information in engineering (Sebastian & Antonsson, 1996; Zimmermann, 1999). Fuzzy databases may be based on the generic logical database models such as relational databases (Buckles & Petry, 1982; Prade & Testemale, 1984), nested relational databases (Yazici et al., 1999), and object-oriented databases (Bordogna, Pasi, & Lucarella, 1999; George et al., 1996; van Gyseghem & de Caluwe, 1998). Also, some special databases are extended for fuzzy information handling. In Medina et al. (1997), the architecture for deductive fuzzy relational database was presented, and a fuzzy deductive object-oriented data model was proposed in Bostan and Yazici (1998). More recently, how to construct fuzzy event sets automatically and apply it to active databases was investigated in Saygin and Ulusoy (2001).

Constructions of Database Models

Depending on data abstract levels and actual applications, different database models have their advantages and disadvantages. This is the reason why there exist a lot of database models, conceptual ones and logical ones. It is not appropriate to state that one database model is always better than the others. Conceptual data models are generally used for engineering information modeling at a high level of abstraction. However, engineering information systems are constructed based on logical database models. So at the level of data manipulation, that is, a low level of abstraction, the logical database model is used for engineering information modeling. Here, logical database models are often created through mapping conceptual data models into logical database models. This conversion is called *conceptual design of databases*. The relationships among conceptual data models, logical database models, and engineering information systems are shown in Figure 2.

In this figure, *Logical DB Model (A)* and *Logical DB Model (B)* are different database systems. That means that they may have different logical database models, say relational database and object-oriented database, or they may be different database products, say *Oracle*™ and *DB2*, although they have the same logical database model. It can be seen from the figure that a developed conceptual data model can be mapped into different logical database models. Besides, it can also be seen that a logical database model can be mapped into a conceptual data model. This conversion is called *database reverse engineering*. It is clear that it is possible that different logical database models can be converted one another through database reverse engineering.

Figure 2. Relationships among conceptual data model, logical database model, and engineering information systems



Development of Conceptual Data Models

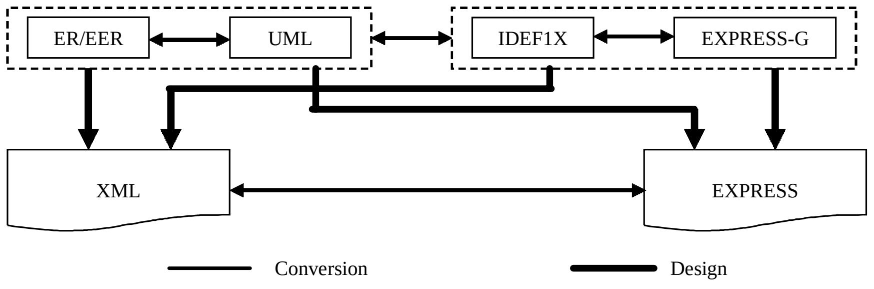
It has been shown that database modeling of engineering information generally starts from conceptual data models, and then the developed conceptual data models are mapped into logical database models. First of all, let us focus on the choice, design, conversion, and extension of conceptual data models in database modeling of engineering information.

Generally speaking, ER and IDEF1X data models are good candidates for business process in engineering applications. But for design and manufacturing, object-oriented conceptual data models such EER, UML, and EXPRESS are powerful. Being the description methods of STEP and a conceptual schema language, EXPRESS is extensively accepted in industrial applications. However, EXPRESS is not a graphical schema language, unlike EER and UML. In order to construct EXPRESS data model at a higher level of abstract, EXPRESS-G, being the graphical representation of EXPRESS, is introduced. Note that EXPRESS-G can only express a subset of the full language of EXPRESS. EXPESS-G provides supports for the notions of entity, type, relationship, cardinality, and schema. The functions, procedures, and rules in EXPRESS language are not supported by EXPRESS-G. So EER and UML should be used to design EXPRESS data model conceptually, and then such EER and UML data models can be translated into EXPRESS data model.

It should be pointed out that, however, for Web-based engineering applications, XML should be used for conceptual data modeling. Just like EXPRESS, XML is not a graphical schema language, either. EER and UML can be used to design XML data model conceptually, and then such EER and UML data models can be translated into XML data model.

That multiple graphical data models can be employed facilitates the designers with different background to design their conceptual models easily by using one of the graphical data models with which they are familiar. However, a complex conceptual data model is generally completed cooperatively by a design group, in which each member may use a different graphical data model. All these graphical data models, designed by different members, should be converted into a union data model finally. Furthermore, the EXPRESS schema can be turned into XML DTD. So far, the data model conversions among EXPRESS-G, IDEF1X, ER/EER, and UML only receive few attentions although such conversions are crucial in engineering information modeling. In (Cherfi, Akoka, and Comyn-Wattiau, 2002), the conceptual modeling quality between EER and UML was investigated. In Arnold and Podehl (1999), a mapping from

Figure 3. Relationships among conceptual data models



EXPRESS-G to UML was introduced in order to define a linking bridge and bring the best of the worlds of product data technology and software engineering together. Also, the formal transformation of EER and EXPRESS-G was developed in Ma et al. (2003). In addition, the comparison of UML and IDEF was given in Noran (2000).

Figure 3 shows the design and conversion relationships among conceptual data models.

In order to model fuzzy engineering information in a conceptual data model, it is necessary to extend its modeling capability. As we know, most database models make use of three levels of abstraction, namely, the data dictionary, the database schema, and the database contents (Erens, McKay, & Bloor, 1994). The fuzzy extensions of conceptual data models should be conducted at all three levels of abstraction. Of course, the constructs of conceptual data models should accordingly be extended to support fuzzy information modeling at these three levels of abstraction. In Zvieli and Chen (1996), for example, three levels of fuzziness were captured in the extended ER model. The first level is concerned with the schema and refers to the set of semantic objects, resulting in fuzzy entity sets, fuzzy relationship sets and fuzzy attribute sets. The second level is concerned with the schema/instance and refers to the set of instances, resulting in fuzzy occurrences of entities and relationships. The third level is concerned with the content and refers to the set of values, resulting in fuzzy attribute values of entities and relationships.

EXPRESS permits null values in array data types and role names by utilizing the keyword *Optional* and used three-valued logic (*False*, *Unknown*, and *True*). In addition, the select data type in EXPRESS defines one kind of imprecise and uncertain data type which actual type is unknown at present. So EXPRESS indeed supports imprecise information modeling but very weakly. Further fuzzy extension to EXPRESS is needed. Just like fuzzy ER, fuzzy EXPRESS should capture three levels of fuzziness and its constructs such as the basic elements (reserved words and literals), the data types, the entities, the expressions and so on, should hereby be extended.

Development of Logical Database Models

It should be noticed that there might be semantic incompatibility between conceptual data models and logical database models. So when a conceptual data model is mapped into a logical database model, we should adopt such a logical database model which expressive power is close to the conceptual data model so that the original information and semantics in the conceptual data model can be preserved and supported furthest. Table 2 shows how relational and object-oriented databases fair against various conceptual data models. Here, *CDM* and *LDBM* denote conceptual data model and logical database model, respectively.

It is clear from the table that relational databases support ER and IDEF1X well. So, when an ER or IDEF1X data model is converted, relational databases should be used. Of course, the target relational databases should be fuzzy ones if ER or IDEF1X data model is a fuzzy one. It is also seen that EER, UML, or EXPRESS data model should be mapped into object-oriented databases. EXPRESS is extensively accepted in industrial application area. EER and UML, being graphical conceptual data models, can be used to design EXPRESS data model conceptually, and then EER and UML data models can be translated into EXPRESS data model (Oh, Hana, & Suhb, 2001). In addition, the EXPRESS schema can be turned into XML DTD (Burkett, 2001). So, in the following, we focus on logical database implementation of EXPRESS data model.

In order to construct a logical database around an EXPRESS data model, the following tasks must be performed: (1) defining the database structures from EXPRESS data model and (2) providing SDAI (STEP Standard Data Access

Table 2. Match of logical database models to conceptual data models

LDBM / CDM	Relational Databases	Object-Oriented Databases
ER	*good*	*bad*
IDEF1X	*good*	*bad*
EER	*fair*	*good*
UML	*fair*	*good*
EXPRESS	*fair*	*good*

Interface) access to the database. Users define their databases using EXPRESS, manipulate the databases using SDAI, and exchange data with other applications through the database systems.

Relational and Object-Oriented Database Support for EXPRESS Data Model

In EXPRESS data models, entity instances are identified by their unique identifiers. Entity instances can be represented as tuples in relational databases, where the tuples are identified by their keys. To manipulate the data of entity instances in relational databases, the problem that entity instances are identified in relational databases must be resolved. As we know, in EXPRESS, there are attributes with UNIQUE constraints. When an entity type is mapped into a relation and each entity instance is mapped into a tuple, it is clear that such attributes can be viewed as the key of the tuples to identify instances. So an EXPRESS data model must contain such an attribute with UNIQUE constraints at least when relational databases are used to model EXPRESS data model. In addition, inverse clause and where clause can be implemented in relational databases as the constraints of foreign key and domain, respectively. Complex entities and subtype/superclass in EXPRESS data models can be implemented in relational databases via the reference relationships between relations. Such organizations, however, do not naturally represent the structural relationships among the objects described. When users make a query, some join operations must be used. Therefore, object-oriented databases should be used for the EXPRESS data model.

Unlike the relational databases, there is no widely accepted definition as to what constitutes an object-oriented database, although object-oriented database standards have been released by ODMG (2000). Not only is it true that not all features in one object-oriented database can be found in another, but the interpretation of similar features may also differ. But some features are in common with object-oriented databases, including object identity, complex objects, encapsulation, types, and inheritance. EXPRESS is object-oriented in nature, which supports these common features in object-oriented databases. Therefore, there should be a more direct way to mapping EXPRESS data model into object-oriented databases. It should be noted that there is incompatibility between the EXPRESS data model and object-oriented databases. No widely accepted definition of object-oriented database model results in the fact that there is not a common set of incompatibilities between EXPRESS and object-oriented databases. Some possible incompatibilities can be found in Goh et al. (1997).

Now let us focus on fuzzy relational and object-oriented databases. As mentioned previously, the fuzzy EXPRESS should capture three levels of fuzziness: the schema level, the schema/instance, and the content. Depending on the modeling capability, however, fuzzy relational databases only support the last two levels of fuzziness, namely, the schema/instance and the content. It is possible that object-oriented databases are extended to support all three levels of fuzziness in fuzzy EXPRESS.

Requirements and Implementation of SDAI Functions

The goal of SDAI is to provide the users with uniform manipulation interfaces and reduce the cost of integrated product databases. When EXPRESS data models are mapped into databases, users will face databases. As a data access interface, SDAI falls into the category of the application users who access and manipulate the data. So the requirements of SDAI functions are decided by the requirements of the application users of databases. However, SDAI itself is in a state of evolution. Considering the enormity of the task and the difficulty for achieving agreement as to what functions are to be included and the viability of implementing the suggestions, only some basic requirements such as data query, data update, structure query, and validation are catered for. Furthermore, under fuzzy information environment, the requirements of SDAI functions needed for manipulating the fuzzy EXPRESS data model must consider the fuzzy information processing such as flexible data query.

Using SDAI operations, the SDAI applications can access EXPRESS data model. However, only the specifications of SDAI operations are given in STEP Part 23 and Part 24. The implementation of these operations is empty, which should be developed utilizing the special binding language according to database systems. One will meet two difficulties when implementing SDAI in the databases. First, the SDAI specifications are still in a state of evolution. Second, the implementation of SDAI functions is product-related. In addition, object-oriented databases are not standardized. It is extremely true for the database implementation of the SDAI functions needed for manipulating the fuzzy EXPRESS data model, because there are no commercial fuzzy relational database management systems, and little research is done on fuzzy object-oriented databases so far.

It should be pointed out that, however, there exists a higher-level implementation of EXPRESS data model than database implementation, which is knowledge-based. Knowledge-based implementation has the features of database implementations, plus full support for EXPRESS constraint validation. A knowledge-based system should read and write exchange files, make product data available to applications in structures defined by EXPRESS, work on data stored in a central database, and should be able to reason about the contents of the database. Knowledge-based systems encode rules using techniques such as frames, semantic nets, and various logic systems, and then use inference techniques such as forward and backward chaining to reason about the contents of a database. Although some interesting preliminary work was done, knowledge-based implementations do not exist. Deductive databases and constraint databases based on relational and/or object-oriented database models are useful in knowledge-intensive engineering applications for this purpose. In deductive databases, rules can be modeled and knowledge bases are hereby constituted. In constraint databases, complex spatial and/or temporal data can be modeled. In particular, constraint databases can handle a wealth of constraints in engineering design.

Conclusion

Manufacturing enterprises obtain increasing product varieties and products with lower price, high quality and shorter lead time by using enterprise information systems. The enterprise information systems have become the nerve center of current computer-based manufacturing enterprises. Manufacturing engineering is typically a data- and knowledge-intensive application area and engineering information modeling is hereby one of the crucial tasks to implement engineering information systems. Databases are designed to support data storage, processing, and retrieval activities related to data management, and database systems are the key to implementing engineering information modeling. But the current mainstream databases are mainly designed for business applications. There are some unique requirements from engineering information modeling, which impose a challenge to databases technologies and promote their evolvement. It is especially true for contemporary engineering applications, where some new techniques have been increasingly applied and their operational patterns are hereby evolved (e.g., e-manufacturing, Web-based PDM, etc.). One can find many researches in literature that focus on using database techniques for engineering information modeling to support various engineering activities. It should be noted that, however, most of these papers only discuss some of the issues according to the different viewpoints and application requirements. Engineering information modeling is complex because it should cover product life cycle times. On the other hand, databases cover wide variety of topics and evolve quickly. Currently, few papers provide comprehensive discussions about how current engineering information modeling can be supported by database technologies. This chapter tries to fill this gap.

In this chapter, we first identify some requirements for engineering information modeling, which include complex objects and relationships, data exchange and share, Web-based applications, imprecision and uncertainty, and knowledge management. Since the current mainstream databases are mainly designed for business applications, and the database models can be classified into conceptual data models and logical database models, we then investigate how current conceptual data models and logical database models satisfy the requirements of engineering information modeling in databases. The purpose of engineering information modeling in databases is to construct the logical database models, which are the foundation of the engineering information systems. Generally the constructions of logical database models start from the constructions of conceptual data models and then the developed conceptual data models are converted into the logical database models. So the chapter presents not only the development of some conceptual data models for engineering information modeling, but also the development of the relational and object-oriented databases which are used to implement EXPRESS/STEP. The contribution of the chapter is to identify the direction of database study viewed from engineering applications and provide a guidance of information modeling for engineering design, manufacturing, and production management. It can be believed that some more powerful database models will be developed to satisfy engineering information modeling.

References

Abiteboul, S., Segoufin, L., & Vianu, V. (2001). Representing and querying XML with incomplete information, In *Proceedings of the 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems,* California (pp. 150-161).

Ahmed, S. (2004). Classification standard in large process plants for integration with robust database. *Industrial Management & Data Systems, 104*(8), 667-673.

Antonsoon, E. K., & Otto, K. N. (1995). Imprecision in engineering design. *ASME Journal of Mechanical Design, 117*(B), 25-32.

Arnalte, S., & Scala, R. M. (1997). An information system for computer-integrated manufacturing systems. *Robotics and Computer-Integrated Manufacturing, 13*(3), 217-228.

Arnold, F., & Podehl, G. (1999). Best of both worlds — A mapping from EXPRESS-G to UML. *Lecture Notes in Computer Science, Vol. 1618*, 49-63.

Barsalou, T., & Wiederhold, G. (1990). Complex objects for relational databases. *Computer-Aided Design, 22*(8), 458-468.

Belussi, A., Bertino, E., & Catania, B. (1998). An extended algebra for constraint databases. *IEEE Transactions on Knowledge and Data Engineering, 10*(5), 686-705.

Ben-Arieh, D., Chopra, M., & Bleyberg, M. Z. (1998). Data mining application for real-time distributed shop floor control. In *Proceedings of 1998 IEEE International Conference on Systems, Man, and Cybernetics,* San Diego, California (pp. 2738-2743).

Booch, G., Rumbaugh, J., & Jacobson, I. (1998). *The unified modeling language user guide*. Reading, MA: Addison-Welsley Longman.

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.

Bostan, B., & Yazici, A. (1998). A fuzzy deductive object-oriented data model. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, Alaska (vol. 2, pp. 1361-1366). IEEE.

Bourret, R. (2004). *XML and databases*. Retrieved October 2004, from http://www.rpbourret.com/xml/XMLAndDatabases.htm

Buckles, B. P., & Petry, F. E. (1982). A fuzzy representation of data for relational database. *Fuzzy Sets and Systems, 7*(3), 213-226.

Burkett, W. C. (2001). Product data markup language: A new paradigm for product data exchange and integration. *Computer Aided Design, 33*(7), 489-500.

Caldwell, N. H. M., Clarkson, Rodgers, & Huxor (2000). Web-based knowledge management for distributed design. *IEEE Intelligent Systems, 15*(3), 40-47.

Caputo, M. (1996). Uncertainty, flexibility and buffers in the management of the firm operating system. *Production Planning & Control, 7*(5), 518-528.

Chang, Y. L., et al. (2000). Workflow process definition and their applications in e-commerce. In *Proceedings of International Symposium on Multimedia Software Engineering,* Taiwan (pp. 193-200). IEEE Computer Society.

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,* Alaska (vol. 2, pp. 1320-1325). IEEE.

Chen, P. P. (1976). The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems, 1*(1), 9-36.

Cherfi, S. S. S., Akoka, J., & Comyn-Wattiau, I. (2002). Conceptual modeling quality — From EER to UML schemas evaluation. *Lecture Notes in Computer Science, Vol. 250*, 414-428.

Chiadamrong, N. (1999). An integrated fuzzy multi-criteria decision-making method for manufacturing strategy selection. *Computers and Industrial Engineering, 37*, 433-436.

Chu, X. J., & Fan, Y. Q. (1999). Product data management based on Web technology. *Integrated Manufacturing Systems, 10*(2), 84-88.

CIMdata. (1997). *Product data management: The definition*. Retrieved September 1997, from http://www.cimdata.com

Conrad, R., Scheffner, D., & Freytag, J. C. (2000). XML conceptual modeling using UML. *Lecture Notes in Computer Science, Vol. 1920*, 558-571.

Damiani, E., Oliboni, B., & Tanca, L. (2001). Fuzzy techniques for XML data smushing. *Lecture Notes in Computer Science, Vol. 2206*, 637-652.

de Souza, R., Ying, Z. Z., & Yang, L. C. (1998). Modeling business processes and enterprise activities at the knowledge level. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM, 12*(1), 29-42.

Dong, Y., & Goh, A. (1998). An intelligent database for engineering applications. *Artificial Intelligence in Engineering, 12*, 1-14.

Dong, Y., et al. (1997). Active database support for STEP/EXPRESS models. *Journal of Intelligent Manufacturing, 8*(4), 251-261.

Dubois, D., Fargier, H., & Prade, H. (1995). Fuzzy constraints in job-shop scheduling. *Journal of Intelligent Manufacturing, 6*(4), 215-234.

Dubois, D., Fargier, H., & Prade, H. (1996). Possibility theory in constraint satisfaction problems: Handling priority, preference, and uncertainty. *Applied Intelligence, 6*, 287-309.

Dubois, D., & Prade, H. (1996). What are fuzzy rules and how to use them. *Fuzzy Sets and Systems, 84*, 169-185.

Dzbor, M. (1999). Intelligent support for problem formalization in design. In *Proceedings of the 3rd IEEE Conference on Intelligent Engineering Systems,* Stara Lesna, Slovakia (pp. 279-284). IEEE.

Eastman, C. M., & Fereshetian, N. (1994). Information models for use in product design: A comparison. *Computer-Aide Design, 26*(7), 551-572.

Erens, F., McKay, A., & Bloor, S. (1994). Product modeling using multiple levels of abstract: Instances as types. *Computers in Industry, 24*, 17-28.

Fargier, H., & Thierry, C. (1999). The use of qualitative decision theory in manufacturing planning and control: Recent results in fuzzy master production scheduling. In R. Slowinski, & M. Hapke (Eds.), *Advances in scheduling and sequencing under fuzziness* (pp. 45-59). Heidelberg: Physica-Verlag.

Fensel, D., Ding, & Omelayenko (2001). Product data integration in B2B e-commerce. *IEEE Intelligent Systems and Their Applications, 16*(4), 54-59.

Francois, F., & Bigeon, J. (1995). Integration of fuzzy techniques in a CAD-CAM system. *IEEE Transactions on Magnetics, 31*(3), 1996-1999.

George, R., et al. (1996). Uncertainty management issues in the object-oriented data model. *IEEE Transactions on Fuzzy Systems, 4*(2), 179-192.

Gertosio, C., & Dussauchoy, A. (2004). Knowledge discovery form industrial databases. *Journal of Intelligent Manufacturing, 15*, 29-37.

Giachetti, R. E., Young, Roggatz, Eversheim, & Perrone (1997). A methodology for the reduction of imprecision in the engineering design process. *European Journal of Operations Research, 100*(2), 277-292.

Goh, A., et al. (1994). A study of SDAI implementation on object-oriented databases. *Computer Standards & Interfaces, 16*, 33-43.

Goh, A., et al. (1997). A STEP/EXPRESS to object-oriented databases translator. *International Journal of Computer Applications in Technology, 10*(1-2), 90-96.

Grabot, B., & Geneste, L. (1998). Management of imprecision and uncertainty for production activity control. *Journal of Intelligent Manufacturing, 9*, 431-446.

Guiffrida, A., & Nagi, R. (1998). Fuzzy set theory applications in production management research: A literature survey. *Journal of Intelligent Manufacturing, 9*, 39-56.

Ho, C. F., Wu, W. H., & Tai, Y. M. (2004). Strategies for the adaptation of ERP systems. *Industrial Management & Data Systems, 104*(3), 234-251.

Issa, G., Shen, S., & Chew, M. S. (1994). Using analogical reasoning for mechanism design. *IEEE Expert, 9*(3), 60-69.

Johtela, T., Smed, Johnsson, & Nevalainen (1999) A fuzzy approach for modeling multiple criteria in the job grouping problem. In *Proceedings of the 25th International Conference on Computers & Industrial Engineering,* New Orleans, LA (pp. 447-50).

Jones, J. D., & Hua, Y. (1998). A fuzzy knowledge base to support routine engineering design. *Fuzzy Sets and Systems, 98*, 267-278.

Kapadia, R., & Fromherz, M. P. J. (1997). Design optimization with uncertain application knowledge. In *Proceedings of the 10th International Conference on Industrial and Engineering Application of Artificial Intelligence and Expert Systems,* Atlanta, GA (pp. 421-430). Gordon and Breach Science Publishers.

Karwowski, W., & Evans, G. W. (1986). Fuzzy concepts in production management research: A review. *International Journal of Production Research, 24*(1), 129-147.

Kim, K., Cormier, O'Grady, & Young (1995). A system for design and concurrent engineering under imprecision. *Journal of Intelligent Manufacturing, 6*(1), 11-27.

Kim, W., et al. (1990). Object-oriented database support for CAD. *Computer-Aided Design, 22*(8), 521-550.

Kleissner, C. (1998). Data mining for the enterprise. In *Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences,* Hawaii (vol. 7, pp. 295-304). IEEE Computer Society.

Kosala, R., & Blockeel, H. (2000). Web mining research: A survey. *SIGKDD Explorations, 2*(1), 1-15.

Krebs, T., & Lührsen, H. (1995). STEP databases as integration platform for concurrent engineering. In *Proceedings of the 2nd International Conference on Concurrent Engineering,* Virginia (pp. 131-142). Johnstown, PA: Concurrent Technologies.

Kuper, G., Libkin, L., & Paredaens, J. (2000). *Constraint databases*. Springer Verlag.

Kusiak, A., Letsche, T., & Zakarian, A. (1997). Data modeling with IDEF1X. *International Journal of Computer Integrated Manufacturing, 10*(6), 470-486.

Kusiak, A., & Tseng, T. L. (2000). Data mining in engineering design: A case study. In *Proceedings of the IEEE Conference on Robotics and Automation,* San Francisco (pp. 206-211). IEEE.

Lee, D. W., & Chu, W. W. (2000). Constraints-preserving transformation from XML document type definition to relational schema. *Lecture Notes in Computer Science,* Utah (vol. 1920, pp. 323-338).

Lee, M. L., et al. (2001). Designing semi-structured databases: A conceptual approach. *Lecture Notes in Computer Science, Vol. 2113* (pp. 12-21).

Liu, D. T., & Xu, X. W. (2001). A review of Web-based product data management systems. *Computers in Industry, 44*(3), 251-262.

Liu, M. C. (1999). On CAD databases. In *Proceedings of the 1999 IEEE Canadian Conference on Electrical and Computer Engineering,* Edmonton, Canada (pp. 325-330). IEEE Computer Society.

Liu, M. C., & Katragadda, S. (2001). DrawCAD: Using deductive object-relational databases in CAD. *Lecture Notes in Artificial Intelligence* (vol. 2113, pp. 481-490). Munich, Germany: Springer.

Ma, Z. M. (2005). *Fuzzy database modeling with XML*. Springer.

Ma, Z. M. (in press). Extending EXPRESS for imprecise and uncertain engineering information modeling. *Journal of Intelligent Manufacturing*.

Ma, Z. M., & Mili, F. (2003). Knowledge comparison in design repositories. *Engineering Applications of Artificial Intelligence, 16*(3), 203-211.

Ma, Z. M., et al. (2003). Conceptual data models for engineering information modeling and formal transformation of EER and EXPRESS-G. *Lecture Notes in Computer Science, Vol. 2813* (pp. 573-575). Springer Verlag.

Ma, Z. M., Zhang, W. J., & Ma, W. Y. (2002). Extending IDEF1X to model fuzzy data. *Journal of Intelligent Manufacturing, 13*(4), 295-307.

Maedche, A., Motik, Stojanovic, Studer, & Volz (2003). Ontologies for enterprise knowledge management. *IEEE Intelligent Systems, 18*(2), 2-9.

Mannisto, T., Peltonen, Soininen, & Sulonen (2001). Multiple abstraction levels in modeling product structures. *Date and Knowledge Engineering, 36*(1), 55-78.

Manwaring, M. L., Jones, K. L., & Glagowski, T. G. (1996). An engineering design process supported by knowledge retrieval from a spatial database. In *Proceedings of Second IEEE International Conference on Engineering of Complex Computer Systems,* Montreal, Canada (pp. 395-398). IEEE Computer Society.

McKay, A., Bloor, M. S., & de Pennington, A. (1996). A framework for product data. *IEEE Transactions on Knowledge and Data Engineering, 8*(5), 825-837.

Medina, J. M., et al. (1997). FREDDI: A fuzzy relational deductive database interface. *International Journal of Intelligent Systems, 12*(8), 597-613.

Michael, S. M., & Khemani, D. (2002). Knowledge management in manufacturing technology: An A.I. application in the industry. In *Proceedings of the 2002 International Conference on Enterprise Information Systems,* Ciudad Real, Spain (pp. 506-511).

Mili, F., Shen, Martinez, Noel, Ram, & Zouras (2001). Knowledge modeling for design decisions. *Artificial Intelligence in Engineering, 15*, 153-164.

Mitra, S., Pal, S. K., & Mitra, P. (2002). Data mining in soft computing framework: A survey. *IEEE Transactions on Neural Networks, 13*(1), 3-14.

Muller, K., & Sebastian, H. J. (1997). Intelligent systems for engineering design and configuration problems. *European Journal of Operational Research, 100*, 315-326.

Noran, O. (2000). *Business Modeling: UML vs. IDEF* (Report/Slides). Griffith University, School of CIT. Retrieved February 2000, from http://www.cit.gu.edu.au/~noran

O'Leary, D. E. (1998). Enterprise knowledge management. *IEEE Computer, 31*(3), 54-61.

ODMG. (2000). *Object Data Management Group*. Retrieved November 2000, from http://www.odmg.org/

Oh, Y., Hana, S. H., & Suhb, H. (2001). Mapping product structures between CAD and PDM systems using UML. *Computer-Aided Design, 33*, 521-529.

OMG. (2003). *Unified Modeling Language (UML)*. Retrieved December 2003, from http://www.omg.org/technology/documents/formal/uml.htm

Pal, S. K., Talwar, V., & Mitra, P. (2002). Web mining in soft computing framework: Relevance, state of the art and future directions. *IEEE Transactions on Neural Networks, 13*(5), 1163-1177.

Parsons, S. (1996). Current approaches to handling imperfect information in data and knowledge bases. *IEEE Transactions on Knowledge and Data Engineering, 8*(2), 353-372.

Petrovic, D., Roy, R., & Petrovic, R. (1998). Modeling and simulation of a supply chain in an uncertain environment. *European Journal of Operational Research, 109*, 299-309.

Petrovic, D., Roy, R., & Petrovic, R. (1999). Supply chain modeling using fuzzy sets. *International Journal of Production Economics, 59*, 443-453.

Pham, B. (1998). Fuzzy logic applications in computer-aided design. *Fuzzy Systems Design*. In L. Reznik, V. Dimitrov, & J. Kacprzyk (Eds.), *Studies in Fuzziness and Soft Computing, 17*, 73-85.

Pham, D. T., & Pham, P. T. N. (1999). Artificial intelligence in engineering. *International Journal of Machine Tools & Manufacture, 39*(6), 937-949.

Posselt, D., & Hillebrand, G. (2002). Database support for evolving data in product design. *Computers in Industry, 48*(1), 59-69.

Prade, H., & Testemale, C. (1984). Generalizing database relational algebra for the treatment of incomplete or uncertain information. *Information Sciences, 34*, 115-143.

Samaras, G., Spooner, D., & Hardwick, M. (1994). Query classification in object-oriented engineering design systems. *Computer-Aided Design, 26*(2), 127-136.

Saygin, Y., & Ulusoy, O. (2001). Automated construction of fuzzy event sets and its application to active databases. *IEEE Transactions on Fuzzy Systems, 9*(3), 450-460.

Schenck, D. A., & Wilson, P. R. (1994). *Information modeling: The EXPRESS way*. Oxford University Press.

Sebastian, H. J., & Antonsson, E. K. (1996). *Fuzzy sets in engineering design and configuration*. Boston: Kluwer Academic Publishers.

Seligman, L., & Rosenthal, A. (2001). XML's impact on databases and data sharing. *IEEE Computer, 34*(6), 59-67.

Seng, J. L., Lin, Y., Wang, J., & Yu, J. (2003). An analytic study of XML database techniques. *Industrial Management & Data Systems, 103*(2), 111-120.

Shaw, M. J. (2000a). Information-based manufacturing with the Web. *The International Journal of Flexible Manufacturing Systems, 12*, 115-129.

Shaw, M. J. (2000b). Building an e-business from enterprise systems. *Information Systems Frontiers, 2*(1), 7-17.

Shaw, N. K., Bloor, M. S., & de Pennington, A. (1989). Product data models. *Research in Engineering Design, 1*, 43-50.

Soliman, F., & Youssef, M. A. (2003). Internet-based e-commerce and its impact on manufacturing and business operations. *Industrial Management & Data Systems, 103*(8), 546-552.

Sun, J., Kalenchuk, D. K., Xue, D., & Gu, P. (2000). Design candidate identification using neural network-based fuzzy reasoning. *Robotics and Computer Integrated Manufacturing, 16*, 383-396.

Surjanto, B., Ritter, N., & Loeser, H. (2000). XML content management based on object-relational database technology. In *Proceedings of the First International Conference on Web Information Systems Engineering,* Hong Kong (vol. 1, pp. 70-79).

Tan, G. W., Shaw, M. J., & Fulkerson, B. (2000). Web-based supply chain management. *Information Systems Frontiers, 2*(1), 41-55.

Tan, K. H., & Platts, K. (2004). A connectance-based approach for managing manufacturing knowledge. *Industrial Management & Data Systems, 104*(2), 158-168.

Thannhuber, M., Tseng, M. M., & Bullinger, H. J. (2001). An autopoietic approach for building knowledge management systems in manufacturing enterprises. *CIRP Annals — Manufacturing Technology, 50*(1), 313-318.

Tsourveloudis, N. G., & Phillis, Y. A. (1998). Manufacturing flexibility measurement: A fuzzy logic framework. *IEEE Transactions on Robotics and Automation, 14*(4), 513-524.

van Gyseghem, N., & de Caluwe, R. (1998). Imprecision and uncertainty in UFO database model. *Journal of the American Society for Information Science, 49*(3), 236-252.

Wizdom Systems Inc. (1985). *U.S. Air Force ICAM Manual: IDEF1X*. Naperville, IL.

Wong, K. Y. (2005). Critical success factors for implementing knowledge management in small and medium enterprises. *Industrial Management & Data Systems, 105*(3), 261-279.

Xiao, R. G., et al. (2001). Modeling and transformation of object-oriented conceptual models into XML schema. *Lecture Notes in Computer Science, Vol. 2113* (pp. 795-804).

Xue, D., & Xu, Y. (2003). Web-based distributed systems and database modeling for concurrent design. *Computer-Aided Design, 35*, 433-452.

Xue, D., Yadav, S., & Norrie, D. H. (1999). Knowledge base and database representation for intelligent concurrent design. *Computer-Aided Design, 31*, 131-145.

Yager, R. R. (2000). Targeted e-commerce marketing using fuzzy intelligent agents. *IEEE Intelligent Systems, 15*(6), 42-45.

Yager, R. R., & Pasi, G. (2001). Product category description for Web-shopping in e-commerce. *International Journal of Intelligent Systems, 16*, 1009-1021.

Yang, H., & Xue, D. (2003). Recent research on developing Web-based manufacturing systems: A review. *International Journal of Product Research, 41*(15), 3601-3629.

Yazici, A., et al. (1999). Uncertainty in a nested relational database model. *Data & Knowledge Engineering, 30*, 275-301.

Young, R. E., Giachetti, R., & Ress, D. A. (1996). A fuzzy constraint satisfaction system for design and manufacturing. In *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems,* New Orleans, LA (vol. 2, pp. 1106-1112).

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.

Zadeh, L. A. (1989). Knowledge representation in fuzzy logic. *IEEE Transactions on Knowledge and Data Engineering, 1*(1), 89-100.

Zhang, F., & Xue, D. (2002). Distributed database and knowledge base modeling for intelligent concurrent design. *Computer-Aided Design, 34*, 27-40.

Zhang, Q. Y. (2001). Object-oriented database systems in manufacturing: selection and applications. *Industrial Management & Data Systems, 101*(3), 97-105.

Zhang, W. J., & Li, Q. (1999). Information modeling for made-to-order virtual enterprise manufacturing systems. *Computer-Aided Design, 31*(10), 611-619.

Zhang, Y. P., Zhang, C. C., and Wang, H. P. B. (2000). An Internet-based STEP data exchange framework for virtual enterprises. *Computers in Industry, 41*, 51-63.

Zimmermann, H. J. (1999). *Practical applications of fuzzy technologies*. Boston: Kluwer Academic Publishers.

Zvieli, A., & Chen, P. P. (1996). Entity-relationship modeling and fuzzy databases. In *Proceedings of the 1986 IEEE International Conference on Data Engineering* (pp. 320-327).

Chapter II

Database Design Based on B

Elvira Locuratolo, ISTI, Consiglio Nazionale delle Ricerche, Italy

Abstract

This chapter is devoted to the integration of the ASSO features in B. ASSO is a database design methodology defined for achieving conceptual schema consistency, logical schema correctness, flexibility in reflecting the real-life changes on the schema and efficiency in accessing and storing information. B is an industrial formal method for specifying, designing, and coding software systems. Starting from a B specification of the data structures and of the transactions allowed on a database, two model transformations are designed: The resulting model, called Structured Database Schema*, integrates static and dynamics exploiting the novel concepts of* Class-Machine *and* Specialized Class-Machine. *Formal details which must be specified if the conceptual model of ASSO is directly constructed in B are avoided; the costs of the consistency obligations are minimized. Class-Machines supported by semantic data models can be correctly linked with Class-Machines supported by object Models.*

Introduction

The B Method (Abrial, 1996) is an industrial general-purpose formal method which uses a model, the *Abstract Machine*, to encapsulate a fragment of the state within a provably consistent specification and a refinement theory to derive correct programs. The direct use of B for developing database applications can lead to several advantages including the possibility of guaranteeing the correctness of the design process; however, it presents some shortcomings: The B Method lacks the abstraction mechanisms supported by the database conceptual languages, and its refinement has not been designed to obtain efficient database implementations. Specifying a database application with the B notation is a tedious process, since many properties implicitly declared within the database conceptual schemas must be explicated. Further, the consistency proofs are too expensive, since they must be performed with respect not only to the application constraints, but also to the conceptual schema constraints.

ASSO (Locuratolo, 1997, 2002, 2004; Locuratolo & Matthews, 1999a, b, c) is an innovative database design methodology defined for achieving conceptual schema consistency, logical schema correctness, flexibility in reflecting the real-life changes on the schema and efficiency in accessing and storing information. This makes it possible to overcome some inadequacies of existing informal methodologies (Batini, Ceri, & Navathe, 1992; Booch, 1994; Rumbaugh, Booch, & Jacobson, 1999) such as to guarantee the conceptual schema consistency and the logical schema correctness. Background information on ASSO can be found in initially disjointed approaches of work: A former approach aimed at establishing formal relationships between classes of objects based on semantic data models and classes of objects based on object models. The objective was to achieve the flexibility of semantic data models and the efficiency of the object-oriented database systems. This approach, called *Partitioning Method*, was proposed as a static method in 1992 (Locuratolo & Rabitti, 1998). A latter approach aimed at integrating features from conceptual modeling and abstract machines in order to guarantee the conceptual schema consistency (Castelli & Locuratolo, 1994). ASSO (Castelli & Locuratolo, 1995) tried to integrate these two approaches; however, the proposed model was not suitable to the Partitioning Method applicability. Approaches of study to design the conceptual model of ASSO, called *Structured Database Schema* and the *ASSO refinement* can be found in Andolina and Locuratolo (1997) and Locuratolo (1997). The Structured

Database Schema integrates the specification of both structural and behavioral information at a high abstraction level. It extends the model on which the Partitioning works by designing elementary operations that add objects, remove objects, modify attributes, or let the class unchanged in order to still allow the Partitioning applicability. Approaches of translation from ASSO to B have been proposed in papers by Locuratolo and Matthews (1999, a, b, c). The approach employed to define ASSO, called MetaASSO has been described in Locuratolo (2002). The unitary element in the definition of ASSO is that of correct model transformation, as evidenced in Locuratolo (2004). Differently from a recent approach to the specification and development of database applications based on B (Mammar & Laleau, 2003), where the database application refinements have been implemented using the relational database model, in ASSO classes of objects supported by semantic data models and classes of objects supported by object systems are linked together.

This chapter aims at raising the abstraction level of B exploiting features of ASSO. Starting from a B specification of the data structures and of the transactions allowed on a database, two model transformations are designed. The former transformation, called *Database Schema Model*, restricts the state of the model supported by B in order to represent aspects which are typical of database applications. The latter transformation, called *Structured Database Schema*, reduces the possible B operations on the state of the Database Schema Model. The *Structured Database Schema* is a Conceptual/Semantic model based on a graph which integrates static and dynamics. The Structured Database Schema specifications are written using a formal notation which exploits the concepts of *Class-Machine* and *Specialized Class-Machine*, two concepts which enrich the corresponding concepts supported by the database conceptual languages with transactions and application constraints. In the Structured Database Schema specifications, many formal details are avoided with respect to the B specifications and only the state transformations, which satisfy the class and the specialization constraints, are allowed.

Two different forms of refinement are applied to a conceptual schema: *behavioral refinement* and *data refinement*. The behavioural refinement is defined by steps of B refinements which leave the state unchanged while modifying transactions, whereas the data refinement is based on an algorithm of schema transformations which generates a logical schema. The consistency obligations of any two behaviorally refined schemas are related by the logical implication, whereas graph transformations preserving the schema equivalence establish a formal link between the last behaviorally refined conceptual schema and an object-oriented logical schema. Various approaches have been developed to capture object-oriented modeling in B (Facon, Laleau, & Nguyen, 1996; Shore, 1996); however, differently from them, the proposed approach links together conceptual and logical schemas of databases, in order to gain the benefits of both conceptual/semantic data modeling and object-oriented modeling.

The emphasis of this chapter is on database application modeling and correct model transformations. This approach can have various practical implications:

- To minimize the number of proof obligations and the costs to guarantee the Structured Database Schema consistency.
- To interpret the specialization hierarchies of the Structured Database Schemas as an approach to eliminate redundancy of not necessarily executable code.
- To acquire knowledge for specializing general purpose methods employed at industrial level for specific application areas.

The chapter is organized as follows: The description of B is first given; the abstraction level of B is then raised; conceptual and logical schemas of databases are then linked together; conclusion and further developments are enclosed in a final section.

B-Method

The B-Method (Abrial, 1989, 1996) is a formal method of software development which reduces the distinction between the two traditional activities of specification and programming, thus ensuring the systematic passage from specification to implementation. Two important phases can be distinguished in the software development: *specification* and *refinement*. Specification models the application requirements by means of not necessarily executable statements, whereas refinement is a step-wise approach which, starting from a specification, reaches an implementation. Modeling is essential in the B-Method definition, since this can be proposed in terms of models and model transformations. Figure 1 illustrates the approach: S represents a specification, I an implementation; the intermediate steps are models, whereas the directed

Figure 1. B-Method

$$\boxed{S \to M_1 \to .... \to M_n \to I}$$

arrows are model transformations. The whole process is carried out by using the same formal model: the *Abstract Machine* (AM). The specification is a particular AM, the intermediate models are AM's, and also the implementation is a special AM.

An AM is defined by means of a mathematical data model and a set of operations which animate the data model. The data model describes the static aspects of the system, *the state*, whereas the operations describe the dynamic aspects. In the following, first the main features of both the data model and the model animation will be outlined, and then a paragraph on the B refinement will be provided.

Data Model

The data model is given by listing a set of variables and writing the relevant properties of the variables, that is, the *invariant*. The invariant is formalized using the full notation of the first order predicate logic and a restricted version of the set theory notation. Each of these notations is defined in terms of a basis and a set of constructors as can be seen in Table 1.

These constructors suffice to define other concepts such as binary relations and functions, mathematical objects such as natural numbers, finite sequences and

Table 1.

First Order Notation		
Basis:	$\in$	set membership
Constructors:	$\Rightarrow$	Implication
	$\wedge$	disjunction
	$\vee$	conjunction
	$\neg$	negation
	$\exists, \forall$	quantification

Set Theory Notation		
Basis:	A, B, C	given sets
Constructors:	$A \times B$	Cartesian product
	$\mathbb{P}(A)$	power set
	$\{x \in A \cdot P(x)\}$	set comprehension

trees, and common predicates on sets such as set inclusion and set equality. A theory of recursive functions on mathematical objects is also given. Any rigorous definitions can thus be formalized using these two elementary notations. The following are examples of invariants: $x \subseteq A$, $x \in \mathsf{P}(A)$ where variables are denoted with lower-case letters, and sets with upper-case letters.

Model Animation

An animated model provides a common formal framework to specify both static and dynamic aspects of applications. The dynamic aspects, that is, the operations, are represented as state transitions using the *Generalized Substitution Language* (GSL).

A *substitution*, denoted by "*x:=E*", is a function which transforms the generic predicate R into the predicate obtained, replacing all the free occurrences of x in R by the expression E. Functions that transform predicates into predicates are called *predicate transformers*. An example of predicate transformer is the following: $[x := x + 1]\ (x \in N) \equiv (\ x + 1 \in N)$ where variable x is substituted in the predicate $(x \in N)$ with the expression $x + 1$, thus obtaining the predicate $(x + 1 \in N)$. The substitution $x := E$ specifies only total and deterministic state transformations. The generalized substitutions allow representing also partial and non-deterministic state transformations.

GSL is defined by means of the bases and constructors shown in Table 2, where P is a predicate, S and T are generalized substitutions and n is a variable distinct from those of the machine state. A generalized substitution S defines a predicate transformer S: $R \rightarrow [S]R$ which associates the weakest pre-conditions $[S]R$ with any post-condition R , ensuring that R holds just after the operation has taken place. The axioms shown in Table 3 define the semantics of the GLS in terms of weakest pre-condition predicate transformers.

In a *pre-conditioned substitution*, the pre-condition expresses the indispensable condition under which the operation can be invoked. When the predicate P does not hold, the substitution cannot establish anything; whereas in the *guarded substitution*, when P does not hold, the substitution is able to establish anything. Let us observe that for the pre-conditioned substitution, in order to establish a post-condition, you must prove P, whereas in the guarded substitutions, in order to establish a post-condition, you may assume P. A *bounded-choice substitution* is a non-deterministic choice between the two substitutions S and T. This construct means that the future implementer has the

Table 2.

Generalized Substitution Language		
Bases:	$x:=E$ *skip*	simple substitution empty substitution
Constructors:	**pre** P **then** S **end**	pre-conditioning
	$P \Longrightarrow S$	guarding
	choice S **or else** T **end**	bounded-choice
	var n **in** S **end**	unbounded-choice

Table 3.

[**pre** P **then** S **end**] R	$\Leftrightarrow$	$P \wedge [S]\,R$
$[P \Longrightarrow S]\,R$	$\Leftrightarrow$	$P \Rightarrow [S]\,R$
[**choice** S **or else** T **end**] R	$\Leftrightarrow$	$[S]\,R \wedge [T]R$
[**var** n **in** S **end**] R	$\Leftrightarrow$	$\forall\, n \cdot [S]R$

freedom to implement either the operation corresponding to S or that corresponding to T. The *unbounded-choice substitution* generalizes the conjunction appearing in the choice operator to a universal quantifier.

The operations specified in an AM are state transformations described in terms of properties that the variable modifications and the results of computations must enjoy. Such operations will be later realized by means of programs, that is, by giving the precise description of how to modify the values of the variables and how to compute results. A machine operation is defined by giving a name and a definition. The operation can also be parametric; in this case, the operation defines a "family" of state transformations. An instantiation of the parameter yields a different substitution. In the following, the parametric operation *add.employees(emp)* which modifies the machine state is presented.

```
add.employees(emp) =
    Pre $emp \in$ EMPLOYEES
    then $employees := employees \cup \{emp\}$
    end
```

Model Consistency

The axiomatic definition of the AM permits properties of the model, such as its *consistency*, to be proved. A set of formulas, *consistency proof obligations*, must be proved to determine the AM consistency. If the AM has invariant I and operation **pre** P **then** S **end**, then the interesting consistency obligation is the following: $P \wedge I \Rightarrow [S]\, I$. This consistency obligation ensures that under both the assumptions I and P, the operation S establishes I. However, before proving this obligation, it must be proved that the animated model is not empty. For this purpose, a special operation, called *initialization*, belongs to the machine operations. In particular, the initialization is a substitution with true pre-conditions whose final states are starting states of the machine. In order to guarantee consistency, also the initialization must be proved to establish the invariant.

Abstract Machines can be composed to build complete systems allowing modular design and development of systems. These modularity mechanisms have been designed to support the composition of proofs; once an AM has been proven consistent, then it need not be proven again.

Refinement

Refinement is intended as a sequence of AM transformations: Starting from an initial AM which is a consistent specification, step by step the designer proposes new AMs, each of which with more implementation details than the previous one. At each step, formulas of first-order logic are proved in order to ensure the step correctness. As refinement is a transitive relation, the whole process results to be correct. Refinement is conducted in three different ways: abolishment of pre-conditions and choices; introduction of sequencing and loop; and transformation of the mathematical data structures (sets, relations, functions, sequences and trees) into structures that might be programmable (simple variables, arrays or files). In order to consider a step of refinement, let us consider a consistent AM; for example, the **Abstract Machine** M. The elements proposed to refine M are presented in the Abstract Machine **Refinement** N (see Table 4).

The consistent Machine M is refined into machine N if they are related via a coupling condition in the invariant of N, and each operation of M has a counterpart in N. The following proof obligation of correctness establishes that the behavior of the refined operation T respects that of the original operation S:

$$P(\mathrm{x}) \wedge I(\mathrm{x}) \wedge J(\mathrm{x,y}) \Rightarrow [T]\, \neg\, [S]\, \neg\, J(\mathrm{x,y})$$

Table 4.

Abstract Machine M	**Refinement** N
Variables x	**Variables** y
Invariant *I*(x)	**Invariant** *J*(x,y)
Operation *S* (with pre-condition P(x))	**Operation** *T*
End	**End**

The double negation guarantees that the new operation *T* is less non-deterministic than the corresponding old one *S*. The Abstract Machine N can in turn be refined in a similar way.

Support tools for B have been realized. In particular, the B-Toolkit is an integrated suite of tools which provides syntax and type checkers, support for structuring, support for animation, proof obligation generation, proof tools, support for refinement and implementation, documentation and version control tools, and code generation.

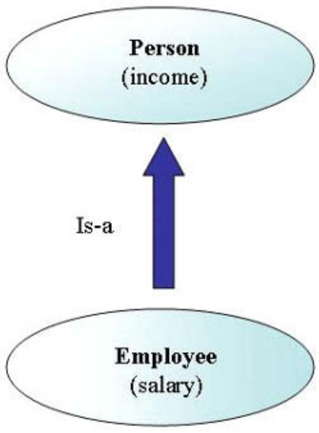
Despite the properties of specification *consistency* and refinement *correctness*, which are desired quality features, and the *set of tools* provided to support the engineering task, the B-Method cannot be considered as a formal method of database design; in fact, it lacks abstraction mechanisms and features typical of the database area. As a consequence, both specifications and proofs are expensive.

Raising the Abstraction Level of B

In order to illustrate the weaknesses of the AM in specifying a database application, let us refer to the specialization hierarchy of a semantic data model, which presents two important mechanisms used in the database area, that is, the mechanisms of *classification* and *specialization*. Figure 2 provides an example.

The top node of the hierarchy represents the *person* class, that is, a set of *person* objects in the database having the *income* attribute, whereas the bottom node represents the *employee* class. The directed arrow, called the *is-a relationship* of the *employee* class with respect to the *person* class, indicates that the *employee* class is a specialization of the *person* class, that is, it is defined through the following properties, called *attribute inheritance* and *object inclusion*, respectively:

Figure 2. Specialization hierarchy



- The *employee* class inherits the *income* attribute from the person class and has also the specific *salary* attribute; and
- The objects of the *employee* class are a subset of the objects of the *person* class.

The formal definition of the hierarchy in Figure 2 can be provided specifying the state of an AM, that is, listing the *person, income, employee* and *salary* variables and writing in the invariant the properties which define the specialization hierarchy. Let us call these properties *implicit constraints*. In the following, the implicit constraints which must be enclosed into the AM invariant are presented.

Invariant

$person \subseteq \mathrm{PERSON} \wedge income \in person \rightarrow \mathrm{N} \wedge$

$\wedge\ employee \subseteq person \wedge income.employee \in employee \rightarrow \mathrm{N} \wedge$

$\wedge\ salary \in employee \rightarrow \mathrm{N}$

In the database area, the same activity of formalization can be performed more easily: As an example, let us consider the database conceptual languages. These languages provide the following two syntactic forms to specify a specialization hierarchy:

class *variable* **of** *given-set* **with** (*attr-list*), and

class *variable* **is-a** *variable* **with** (*attr-list*),

where attr-list is a sequence of specifications of the form: variable: set-esp. Variable and given sets are identifiers and set-expr is an expression built in terms of sets and set operators.

These constructs implicitly specify the properties defining the abstraction mechanism of classification, called *class constraints*, and the two properties of attribute inheritance and object inclusion, globally called *specialization constraints*. The specification of the model represented in Figure 2 is then the following:

class *person* **of** PERSON **with** (*income*:N), and

class *employee* **is-a** *person* **with** (*salary*:N),

where the implicit constraints are represented by the conjunction of the classification and the specialization constraints.

Modeling a database application using an AM requires the specification in the invariant not only of the *implicit constraints*, but also of the *explicit constraints,* that is, of the application constraints.

As far as the dynamic aspects are concerned, that is, the *database transactions*, these are specified using the GSL. The *consistency proofs* of the AM must be performed not only with respect to the *explicit constraints*, but also with respect to the *implicit constraints*. This makes the proof process for ensuring the schema consistency too expensive.

In order to reduce specifications and proofs, a particular AM, called *Database Schema Model,* has been defined. The next section describes the Database Schema Model.

Database Schema Model

The Database Schema Model avoids the formal definition of the implicit constraints using the database conceptual languages constructs. The database transactions are defined on a state, called *database schema state,* which is reduced with respect to the AM state: The *initialization* is defined as a substitution establishing the implicit constraints, whereas, a new building block, not the simple substitution, is considered as the basis for the transaction specifications. The building block, denoted by *op-hierarchy*(*par_list*) has the same semantics of the simplest B substitution preserving the implicit constraints. To specify a transaction, the GSL constructors are recursively applied to the *op-hierarchy*(*par_list*). In this way, all and only the B transformations preserving the implicit constraints are specified.

The axiomatic semantics of the building block is the following:

$$[\textit{op-hierarchy}\ (\textit{par\_list})\ ]\ R \Leftrightarrow (\textit{implicit} \Rightarrow \textit{implicit'}) \wedge R'$$

where *op-hierarchy*(*par_list*) is the parametric basic substitution, *R* is a predicate on the state variables, *implicit* is the predicate which formalize the state and finally *implicit'* and *R'* are the predicates *implicit* and *R* respectively after the multiple substitutions specified through *par_list*. We further consider as a basic substitution a substitution which does not specify any state transformation. This is defined in the following: [*skip-hierarchy*] $R \Leftrightarrow R$.

In order to improve the transaction specification readability, the *op-hierarchy* (*par_list*) substitution has been specialized with five basic transactions. The basic transactions associated with the state in Figure 2 are shown in Table 5.

The former *add-hierarchy* requires the parameters associated with all the attributes of the top class to be enclosed in *par_list*, that is, the parameter list; the latter *add-hierarchy* requires the parameters associated with both the attributes of the top class and the specific attributes of the bottom class to be enclosed in *par_list*. The former *mig-hierarchy* requires only the parameters associated with the specific attributes of the bottom class to be enclosed in *par_list*, whereas both the latter *mig-hierarchy* and the *rem-hierarchy* require only one parameter to be enclosed in *par_list*.

The Database Schema Model is a particular Abstract Machine, since not all the AM substitutions are database transactions, but only those preserving the implicit constraints.

The state of a Database Schema can be interpreted as a restricted state with respect to the corresponding Abrial's AM. This allows avoiding the consistency proofs of the specialization hierarchy with respect to the implicit constraints, whereas new consistency obligations can be derived from the Abrial's consistency obligations.

Table 5.

add-hierarchy(*pers,i)*	inserts the *pers* object only into the person class of the hierarchy
add-hierarchy(*pers,i,s)*	inserts the *pers* object into both the classes of the hierarchy
mig-hierarchy(*pers,s*)	permits the *pers* object to migrate from the person class to the employee class of the hierarchy
mig-hierarchy (*pers*)	permits the *pers* object to migrate from the employee to the person class of the hierarchy
rem-hierarchy(*pers)*	removes the *pers* object from the hierarchy

In the following, an example of database application (Andolina & Locuratolo, 1997) is given: First the application requirements are listed, then the conceptual schema supported by the Database Schema Model is specified.

A Database Application

1. The database maintains information about a set of *persons*, their *income*, a subset of *persons* in employment and their *salary*;
2. The *income* of each *person* is greater or equal to 1000; the *salary* of each *employee* is greater or equal to 500; the *income* of each *employee* is equal to the *employee salary* incremented by a value;
3. An operation adds information to the database when a *person* stored in the database is unrolled;
4. An operation partially or completely removes information when a *person* goes away from the company; and
5. An operation removes information related with the employees.

Figure 3 illustrates the conceptual schema specification.

The *company.schema* specification is described listing a clause for each component of the database schema model. In the **classes** clause, there are two class constructors which specify respectively the top class and the bottom class of the specialization hierarchy in Figure 3. The **constraints** clause specifies the *explicit constraints* described in point 2 of the application requirements. Both the **initialization** and the **transaction** clauses specify properties which must be satisfied by the database schema state transformations. The initialization describes a special transaction which associates the empty set with each state variable simultaneously. The new.employee(*pers, s*) transaction specifies a

Figure 3. Conceptual schema

```
database schema
     company schema

classes
     class person of PERSON with (income:N)
     class employee is-a person with (salary:N)

constraints
    ∀p (p∈person  ⇒  income(p)≥1000)
    ∀e (e∈employee  ⇒  salary(e)≥500)
    ∀e (e∈employee  ⇒  ∃x (x∈N ∧ income(e) = salary(e) + x))

initialization
    person, income, employee , salary:=∅, ∅, ∅, ∅
transactions
    new employee(pers, s)=
                  PRE
                    pers∈person ∧ s≥500
                  THEN
                    mig-hierarchy(pers, s)
                  END;

    delete(pers) =
              PRE
                 pers∈person
              THEN
                 CHOICE rem-hierarchy(pers) ORELSE mig-hierarchy  (pers)
              END;

    pers(pers) =
             pers∈employee  ⇒  mig-hierarchy (pers)

end.
```

parametric operation which inserts the *pers* object, belonging to the *person* class, into the *employee* class while associating the *s* value with his *salary*. This operation is not able to establish anything when the *pre-conditions* are false, that is, when the value of the *pers* parameter is not a member of the set person and/ or the value of the input variable *s* is less then 500. The delete(*pers*) specifies a parametric transaction which can be described either by the basic transaction *rem-hierarchy* (*pers*), which removes the object from the whole hierarchy, or by the basic transaction *mig-hierarchy*(*pers*), which removes the object only from the *employee* class. Finally, the pers(*pers*) specifies a parametric transaction only defined on the employee class. In order to ensure the conceptual schema consistency, the following *proof obligations* must be proved:

- The initialization satisfies both the implicit and explicit constraints;
- Each operation preserves the explicit constraints.

The former proof obligation guarantees the *initialization correctness* and the latter the *operation correctness*. The correctness proof of an operation is performed by demonstrating that the following formula is true:

$$\textit{implicit} \wedge \textit{explicit} \wedge \textit{pre-conditions} \Rightarrow [\text{op-name}(\textit{par\_list})]\textit{explicit}$$

The main features of the Database Schema Model are summarized in the following:

- It is an AM;
- Its state expressivity is comparable with that of a semantic data model;
- It ensures that only transactions which are correct with respect to the implicit constraints can be specified;
- The structural aspects of the model are defined through specialization, whereas the dynamic aspects are defined as operations on a restricted AM state; and
- No specialization mechanisms allow the transaction specification.

The last feature of the Database Schema Model lets us understand that the transaction specification is still a tedious process, whereas the correctness proof of transactions is still an expensive process. In order to overcome this shortcoming of the Database Schema Model, the abstraction mechanism of classification has been extended with basic transactions to insert objects into a class, to remove objects from a class, and to modify attributes. A notion of behavioural specialization, which allows both structural and behavioural aspects of modeling to be specified in a similar way, has been introduced and a new model called *Structured Database Schema* which is based on a specialization hierarchy, has been defined. Similarly to attributes, the Structured Database Schema transactions are inherited through specialization, so that specifications and proofs of inherited transactions are avoided. The nodes of the hierarchy are database schemas, in the following called *Class-Machines*, whereas the is-a relationship has been extended in order to support not only attribute specialization, but also transaction specialization. The extended is-a relationship is called *is-a** relationship. In the following, the Structured Database Schema is proposed.

Structured Database Schema

The Structured Database Schema Model results as a composition of small Database Schema Models, called *Class-Machines*, formalizing the nodes of an extended specialization hierarchy. The links of the hierarchy represent relationships between Class-Machines which extends the traditional is-a relationship with a notion of transaction specialization given in Andolina and Locuratolo (1997).

Definition (Class-Machine)

A **Class-Machine** is an Abstract Machine whose state variables are constrained to satisfy the class constraints.

Let us observe that, as the Class-Machine state is enclosed in the Abstract Machine state, not all the Abrial's state transformations are Class-Machine transactions, but only those preserving the class constraints.

Property (Class-Machine)

A **Class-Machine** is an Abstract Machine whose initialization establishes the class constraints and whose transactions preserve the class constraints.

Let us observe that, if the initialization establishes the class constraints and the operations preserve the class constraints, the state of the Abstract Machine can be considered as restricted to the Class-Machine state.

Relation (Class-Machine and Abstract Machine)

The Class-Machine definition and the respective property are two equivalent propositions, which establish the relationship between *Class-Machine and Abstract-Machine*. As a consequence, the *Class-Machine formalization* can be given exploiting any model used to formalize the Abstract

Machine restricted to capture all and only those Abstract Machines which are Class-Machines.

Definition (Consistency)

A Class-Machine is *consistent* if the initialization establishes the implicit and the explicit constraints and each of the remaining operations preserves the explicit constraints.

Definition (Specialized Class-Machine)

If **Class-Machine** *name* is in **is-a*** *relationship* **with Class-Machine** *root*, then an AM called **Specialized Class-Machine** *name* is defined as follows:

- The objects of the **Specialized Class-Machine** *name* are those of **Class-Machine** *name*;
- The attributes of the **Specialized Class-Machine** *name* are both the attributes of the **Class-Machine** *root* restricted to the class name and the specific attributes of the **Class-Machine** *name*;
- The explicit constraints of the **Specialized Class-Machine** *name* are defined by the conjunction of the explicit constraints of the **Class-Machine** *root* (restricted to corresponding variables of the **Class-Machine** *name*), the explicit constraints of the **Class-Machine** *name* and the explicit constraints involving variables of both the **Class-Machine** *root* and the **Class-Machine** *name*;
- Each transaction on the **Class-Machine** *root* is specialized on the **Class-Machine** *name* with a corresponding trasaction caled **specialization** (Andolina & Locuratolo, 1997). The initialization (resp. a transaction which inserts objects) is explicitly specialized; the remaining transactions can be implicitly specialized;
- The initialization (resp. an inherited transaction) of the **Specialized Class-Machine** *name* is the parallel composition (Abrial, 1996) of the initialization (resp. a transaction) of the **Class-Machine** *root* restricted to the **Class-Machine** *name* with the corresponding **specialization**;
- The specific transactions of the **Class-Machine** *name* belong to the **Specialized Class-Machine** *name*; and

- Transactions involving variables of both the **Class-Machine** *root* and the **Class-Machine** *name* belong to the **Specialized Class-Machine** *name*.

Property (Specialized Class-Machine)

A **Specialized Class-Machine** is a Class-Machine whose initialization establishes the specialization constraints and whose transactions preserve the specialization constraints.

Definition (Structured Database Schema)

A **Structured Database Schema** is a connected acyclic graph whose nodes are Class-Machines and whose links are **is-a*** relationships between Class-Machines.

Property (Structured Database Schema)

A **Structured Database Schema** is a set composed by a Class-Machine, called *Root Class-Machine*, and by a finite number of Specialized Class-Machines.

The definition of Structured Database Schema and the respective property are two equivalent propositions; however, while the definition is exploited in the process for linking the database conceptual schema with the logical schema, the property permits to see the Structured Database Schema as a set of independent Class-Machines, that is, the Root Class-Machine and the Specialized Class-Machines. As a consequence, the model consistency can be defined as follows.

Definition (Structured Database Schema Consistency)

A **Structured Database Schema** is *consistent* if the Root Class-Machine is consistent and each Specialized Class-Machine is consistent.

This definition allows the decomposition of large consistency obligations into a set of independent small obligations. The proof of the inherited operations of Specialized Class-Machines can be avoided. Thus, in the case of operations and explicit constraints involving only Class-Machines, the Specialized Class-Machine consistency is reduced to the Class-Machine consistency, whereas, in the case of operations and application constraints involving variables of Specialized Class-Machines, the consistency proof of the Specialized Class-Machine can be optimized (Locuratolo, 2001).

The following syntactic forms extending those given by the database conceptual languages are provided to specify the **Class-Machines** of a Structured Database Schema:

class *variable* **of** *given-set* **with** *(attr-list; init; oper-list)*, and

class *variable* **is-a*** *variable* **with** *(attr-list; init, oper-list)*,

where *init and oper-list* denote an initialization and a list of database transactions on the specified Class-Machines state, respectively.

The Structured Database Schema in Figure 4 comprises two clauses: the **Class-Machines** clause and the **Specialized-Constraints** clause. Within the former, the Class-Machine constructors specify small Database Schemas. The *is-a** relationship of the employee Class-Machine with respect to the person Class-Machine extends the original *is-a* relationship to the behavior. Within the **Specialized-Constraints** clause, the explicit constraints involving variables of both the *person* Class-Machine and the *employee* Class-Machine are specified. The Structured Database Schema structure can be usefully exploited for optimizing the proof process. Specifically, in our example, the specified transactions involve variables of Class-Machines, and no transaction involves variables of Specialized Class-Machine; thus the model consistency can be defined by performing two consistency proofs: the *person* Class-Machine proof and the *employee* Class-Machine proof.

The Structured Database Schema can be seen as a model at a higher abstraction level with respect to the B model. In fact, formal details which must be specified in an Abstract Machine are implicitly specified in a Structured Database Schema, whereas the consistency obligations required to be proved in B are reduced to a set of small obligations, since only transactions correct with respect to the specialization constraints can be specified, and since the proofs of the inherited transactions can be avoided.

The *Behavioral Refinement* of a Class-Machine (a Specialized Class-Machine) is a step-wise approach, which let the state unchanged, weakening the pre-conditions and/or reducing the non-determinism. The *Behavioral Refine-*

Figure 4. The Structured Database Schema

```
Structured Database Schema
company. schema
Class-Machines
class person of PERSON with (income:N;

                                 ∀p (p∈person  ⇒  income(p)≥1000)

                                 init.person ()=person, income:=∅, ∅;
                                 new.person(pers)=
                                      PRE
                                       pers∈person
                                      THEN SKIP person END

                                 del.person  (pers)=
                                    PRE
                                      pers∈PERSON
                                    THEN
                                      CHOICE REM person(pers) OR ELSE SKIP person
                                    END )

class employee is-a* person with (salary: N;

                                   ∀e (e∈employee  ⇒  salary(e)≥500)

                                   init.employee()=employee, salary:=∅, ∅;

                                   new.employee(pers, s) =
                                     PRE
                                       s≥500
                                     THEN
                                       ADD employee(pers, s)
                                     END

                                  del.employee(pers)= REM employee(pers)

                                   pers(pers) =  pers∈employee  ⇒ REM employee(pers, i)

Specialized - Constraints

employee  ∀e (e∈employee  ⇒  ∃x (x∈N ∧ income(e) = salary(e) + x))
```

ment of a Class-Machine (a Specialized Class-Machine) guarantees the Class-Machine (the specialized Class-Machine) consistency.

Within a Structured Database Schema, transactions and explicit constraints can be specified that involve variables of Class-Machines, Specialized Class-Machines and specialization Sub-Hierarchies. The Structured Database Schema consistency can thus be partitioned into the consistency proofs of Class-Machines, those of Specialized Class-Machines and those of Database Schemas corresponding to the Specialization Sub-Hierarchies. Similarly, the Behavioral Refinement of the Structured Database Schema can be reduced to the Behavioral Refinement of Class-Machines, Specialized Class-Machines and Database Schemas corresponding to the Specialization Sub-Hierarchies.

Linking Conceptual and Logical Database Schemas

The specification of database applications supported by the Structured Database Schema provides two interpretations of the conceptual schema. The former interpretation sees a conceptual schema as a collection of independent Class-Machines; this interpretation is used for performing both the consistency proofs and the behavioral refinement proofs. The latter interpretation sees a conceptual schema as a connected acyclic graph of Class-Machines, in the following called *Conceptual Class-Machines*; this interpretation is used within the data refinement, which links *Class-Machines supported by semantic data models with Class-Machines supported by object systems*. Semantic data models (Cardenas & McLeod, 1990) are the most appropriate models for conceptual database design, since they allow both a representation of database objects close to real word objects and a flexible reflection on the schema of the changes occurring in real life; however, they have never been implemented efficiently (Nixon & Mylopoulos, 1990). On the contrary, object models (Abitebul, Hull, & Vianu, 1995), which have abstraction mechanisms similar to those of semantic data models, have reached a remarkable level of efficiency. The better results in the engineering of object database systems can be explained by observing that in semantic data models each object instance can belong to any class of a specialization hierarchy, thus enhancing flexibility, whereas in object data models each object instance belongs to one and only one class, thus limiting flexibility while enhancing efficiency. Semantic data models have always been translated into relational models; an approach proposed in literature to map semantic data models into object models is the Partitioning Method (Locuratolo & Rabitti, 1998).

This section gives a procedure for mapping a directed acyclic graph of classes supported by semantic data models, in the following called *conceptual* *classes,* into classes supported by object models, in the following called *logical classes.* Let us suppose that no direct descendant of the root has further descendants. We discuss an approach to designing the graph of logical classes G_L starting from a graph of conceptual classes G_C.

Let us distinguish the following two cases: a_1 and a_2.

a_1) *Elementary case:* The graph of Conceptual Classes is only composed by a class in is-a relationship with the root.

a_2) *General case:* The graph of Conceptual Classes is composed by more classes in is-a relationship with the root.

Case a_1 can be solved directly as follows:

$$G_C = < son > is - a < root >$$

$$\Updownarrow$$

$$G_L = < son > is - a_L < root - son >$$

The two nodes $<son>$ and $< root - son >$ define a partition of the original root, whereas $is - a_L$ is the logical specialization, or an oriented link from the node $<son>$ to the node $< root - son >$. Each object belongs to one and only one logical class. The logical inclusion property is indirect, that is, it holds in terms of attribute inheritance.

Case a_2 can be solved by decomposing the problem into two independent problems of the same type and by merging the respective solutions.

$$G_C = \{< son_1 > i - sa < root >,......< soon_p > is - a < root >\}$$

The simplest situation illustrating the case a_2 is composed by a root and two direct descendants of the root. The graph G_C is then decomposed as follows:

$$G_{C1} = \{< son_2 > is - a < root - son_1 >\}$$
$$G_{C2} = \{< son_2 > is - a < root \cap son_1 >\}$$

These two conceptual graphs belong to the previous solved elementary case and thus can be transformed into corresponding logical graphs.

The Logical graphs G_{L1} and G_{L2} are then merged by linking their respective roots through the $is - a_L$ relationship.

The described procedure is based on the classic Divide and Conquer technique. This technique has been adapted in order to be applied to a graph of conceptual classes as follows:

1. Direct solution of the problem when applied to the elementary case.
2. Decomposition of the problem into two independent sub-problems of the same type. The Graph G_C is decomposed into two graphs whose root defines a partition of the G_C root. The partition is obtained by set difference and set intersection between the G_C root and the most left son.
3. Recursive solution of each a_2 sub-problem.
4. Composition of the two sub-problem solutions in order to obtain the global solution.

Attributes and transactions are associated with the nodes of the graphs as in Locuratolo and Matthews (1999, a, b, c).

Conclusion and Further Developments

The use of formal methods in the design of database applications can lead to several advantages, including the possibility of guaranteeing the specification consistency and the design correctness. However, formal methods can be too difficult and expensive if compared with existing informal methods. Specifying a database application with the abstract machine notation is a tedious process, since many properties implicitly declared with the database conceptual languages must be explicated. Further, the direct use of B makes the consistency proofs too expensive, since these must be performed not only with respect to the explicit constraints, but also with respect to the class and the specialization constraints. This chapter is devoted to the integration of the ASSO abstraction mechanisms in B. The Abstract Machine state has been restricted with respect to B, since only the state transformations which satisfy the class and the specialization constraints are allowed.

If the model for ASSO is directly constructed in B:

1. More detail is added to the Abstract Machines than is necessary in the form of implicit constraints;
2. More proof obligations are generated and proved than is necessary, to show that transactions respect the implicit constraints; and
3. More Abstract Machines are designed than is necessary.

According to the described proposal, only those Class-Machine and Specialized Class-Machines are designed which are necessary to prove the model consistency.

By designing only those machines, we have raised the abstraction level of B by imposing the implicit assumptions of ASSO, and we have thus produced a more efficient way of generating designs in the database field, while adding the quality of higher design.

Consistency and behavioral refinements of Structured Database Schema specifications can be proved exploiting tools for B (B-Core); the partitioning tool could be designed exploiting the features of the proposed algorithm with the associated properties.

The emphasis of this chapter is on database application modeling and correct model transformations. This approach can have various practical implications:

- To minimize the number of proof obligations and the costs to guarantee the Structured Database Schema consistency.
- To interpret the specialization hierarchies of the Structured Database Schemas as an approach to eliminate redundancy of not necessarily executable code.
- To acquire knowledge for specializing general purpose methods employed at industrial level for specific application areas.

Acknowledgments

The author would like to thank her husband, Antonio Canonico, and her children, Paola and Floriano, for their help and advice.

Figure 3 has been reprinted from "ASSO: Behavioral Specialization Modelling" by Rosanna Andolina and Elvira Locuratolo in *Information Modelling and Knowledge Bases VIII*, edited by H. Kangassalo, J. Nilson, H. Jaakkola and S. Ohsuga, pages 241-259, copyright 1997, with kind permission from IOS Press.

References

Abiteboul, S., & Hull, R., & Vianu, V. (1995). *Foundations of databases*. Addison-Wesley.

Abrial, J. R. (1989). A formal approach to large software construction. *Lecture Notes in Computer Science, Vol. 375*.

Abrial, J. R. (1996). *The B-book: Assigning programs to meanings*. Cambridge University Press.

Andolina, R., & Locuratolo, E. (1997). ASSO: Behavioural Specialisation Modelling. In H. Kangassalo (Ed.), *Information modelling and knowledge bases VIII* (pp. 241-259). IOS Press.

Batini, C., Ceri, S., & Navathe, S. B. (1992). *Conceptual database design: An entity-relationship approach*. Redwood City, CA: Benjamin Cummings.

B-Core. *B-toolkit* (Online Manual). Oxford, UK. Retrieved from http://www.b-core.com

Booch, G. (1994). *Object-oriented analysis and design with applications*. Redwood City, CA: Beniamin Cummings.

Cardenas, A. F., & McLeod. (1990). *Research foundations in object-oriented and semantic database systems*. Englewood Cliffs, NJ: Prentice Hall.

Castelli, D., & Locuratolo, E. (1994). A formal notation for database conceptual schema specifications. In H. Jaakkola, H. Kangassalo, T. Kitahashi, & A. Markus (Eds.), *Information modelling and knowledge bases V*. IOS Press.

Castelli, D., & Locuratolo, E. (1995). ASSO: A formal database design methodology. In H. Kangassalo, H. Jaakkola, S. Ohsuga, & B. Wangler (Eds.), *Information modelling and knowledge bases VI*. IOS Press.

Facon, P., Laleau, R., & Nguyen, H. P. (1996). Mapping object diagrams into B specifications. In A. Bryant, & L. T. Semmens (Eds.), *Proceedings of the Methods Integration Workshop. Electronics Workshops in Computing, BCS*.

Locuratolo, E. (1997, August 12-13). ASSO: Evolution of a formal database design methodology. In *Proceedings of Symposium on Software Technology (SoST'97)*, Buenos Aires.

Locuratolo, E., & Rabitti, F. (1998). Conceptual classes and system classes in object databases. *Acta Informatica, 35*(3), 181-210.

Locuratolo, E., & Matthews, B. M. (1999a). On the relationship between ASSO and B. In H. Jaakkola, H. Kangassalo, & E. Kawaguchi (Eds.), *Information modelling and knowledge bases X* (235-253). IOS Press.

Locuratolo, E., & Matthews, B. M. (1999b). ASSO: A formal methodology of conceptual database design. In S. Gnesi & D. Latella (Eds.), *Proceedings of the Federated Logic Conference, 4th International ERCIM Workshop on Formal Methods for Industrial Critical Systems* (pp. 205-224).

Locuratolo, E., & Matthews, B. M. (1999c). Formal development of databases in ASSO and B. In J. Wing, Woodcock, & J. Davies (Eds.), *Lecture Notes in Computer Science, Vol. 1708, FM 99 — Formal methods* (pp. 388-410). Berlin-Heidelberg: Springer-Verlag.

Locuratolo, E. (2001). *MetaASSO: An approach for quality*. IEI, TR 41.

Locuratolo, E. (2002). Designing methods for quality. In H. Kangassalo, H. Jaakkola, E. Kawaguchi, & T. Welzer (Eds.) *Information modelling and knowledge bases XIII* (pp. 279-295). IOS Press.

Locuratolo, E. (2004). Model transformations in designing the ASSO methodology. In P. van Bommel (Ed.), *Transformation of knowledge, information and data: Theory and applications* (pp. 283-302). Hershey, PA: Information Science Publishing.

Mammar, A., & Laleau, R. (2003). Design of an automatic prover dedicated to the refinement of database applications. In K. Araki, S. Gnesi, & D. Mandrioli (Eds.), *CEDRIC-IIE (CNAM), Lecture Notes in Computer Science, Vol. 2805, FME 2003: Formal Methods* (pp. 835-853), Evry, France.

Nixon, B., & Mylopoulos, J. (1990). Integration issues in implementing semantic data models. *Advances in database programming languages* (pp. 187-217). ACM Press.

Rumbaugh, J., & Booch, G., & Jacobson, I. (1999). *The unified modelling language reference manual*. Reading, MA: Addison-Wesley.

Shore, R. (1996). An object-oriented approach to B. In H. Habrias (Ed.), *Proceedings of the 1st Conference on the B-Method*, Nantes, France.

Chapter III

The Management of Evolving Engineering Design Constraints

T. W. Carnduff, University of Glamorgan, UK
J. S. Goonetillake, University of Colombo, Sri Lanka

Abstract

This chapter presents research aimed at determining the requirements of a database software tool that supports integrity validation of versioned-design artefacts through effective management of evolving constraints. It has resulted in the design and development of a constraint management model, which allows constraint evolution through representing constraints within versioned objects called Constraint Versions Objects (CVOs). This model operates around a version model that uses a well-defined configuration management strategy to manage the versions of complex artefacts. Internal and inter-dependency constraints are modelled in CVOs. The combination of our versioning, configuration management, and constraint management approaches has produced a unique model which has been implemented in a prototype database tool with an intuitive user interface. The user interface *allows designers to manage design constraints without the need to program. The innovative concepts developed in this chapter are introduced using an ongoing example of a simple bicycle design.*

Introduction

Artefacts in engineering design are structurally complex and may be represented in software as recursively composite objects. In all but the most trivial of projects, it is impossible for a single designer to perform all of the design effort alone, and therefore engineering design requires teamwork within a cooperative design environment. Due to the evolutionary nature of the design process, design constraints evolve and the design solution of each complex artefact and its components progress through a series of versions. To produce consistent versions, versioning software systems should be augmented with an integrity validation management system. While working within a software environment in which there are evolving constraints which are applied to complex structural artefacts, designers require designer-friendly features to enable them to effectively deal with the design task, without the necessity to program the software.

This chapter presents research aimed at developing a database software tool that supports integrity validation of versioned-design artefacts through effective management of evolving constraints. It has resulted in the design and development of a constraint management model, which allows constraint evolution through representing constraints within versioned objects called Constraint Versions Objects (CVOs). This model operates around a version model that uses a well-defined configuration management strategy to manage the versions of complex artefacts. Internal and inter-dependency constraints are modelled in CVOs. Inter-dependency constraints are used to express the consistency semantics necessary to combine the validated component versions into useful configurations. The combination of our versioning, configuration management and constraint management approaches has produced a unique model which has been implemented in a prototype database tool. This model utilises object-oriented technology and operates in a cooperative and distributed design environment. The innovative concepts developed in this paper are introduced using an ongoing example of a simple bicycle design. The prototype is evaluated using this design which demonstrates the feasibility and viability of our approach.

The remainder of the chapter is structured in the following way. We begin by describing the practical engineering design environment and give a brief analysis of the issues relating to evolutionary design constraint management and the management of evolving design artefacts. Following a brief treatment of the limitations of current approaches to managing evolving design constraints, we present our conceptual framework for integrity constraint management and the system we have developed for the management of evolving constraints by end users. We conclude with a brief description of our system implementation and the identification of future work

Practical Engineering Design Environment

Engineering design is normally performed according to a product design specification (PDS) which contains a detailed listing of the requirements to be met by the designed artefact (Dieter, 1991). Initially these design requirements may often come from the customer representing his/her wishes with respect to the functional, structural, or physical aspects, or cost of the artefact. Design constraints can be defined as design requirements that must be satisfied by the artefact (Lin, Fox, & Bilgic, 1996a, 1996b). Design constraints impose restrictions, which may include restrictions on individual data values of the artefacts, relationships between data values from different artefacts, or existence and other dependencies between artefacts. The product design specification, when possible, expresses constraints in quantitative terms, and we address them as formal constraints.

During the design process, designers have to find values for design parameters of the design artefact (represented as a design object in the software environment) satisfying the constraints applied on them (Chandrasekaran, 1989). It is necessary to make sure that the artefacts adhere to the given set of design constraints to minimise design errors. The process of checking data values of a design object for their adherence to the corresponding set of constraints is known as *integrity validation*. We consider integrity here as the accuracy or adherence of object data to its design constraints. Integrity validation is a cumbersome process to perform manually, and it can also be prone to errors. Furthermore, the design constraints tend to change during the design process for a number of reasons such as changes in customer requirements/market demand, changes in the technology/production cost, and improvements to the performance of the product (Chandrasekaran, 1989; Dieter, 1991; Kott, & Peasant, 1995; Otto & Antonsson, 1991; Thompson, Tomski, Ellacott, & Kuczora, 1995; Ram, Vivekananda, Rao, & Mohan, 1997). For these reasons the manual integrity validation process is difficult. Consequently, it is necessary that the designer is supported with an automatic integrity validation mechanism to reduce the burden imposed on him or her. This mechanism should identify the constraint violations and should bring them to the designer's attention, allowing the designer to correct the violations. In this way, early identification of inconsistent data values will enable the designers to produce consistent design solutions with less rework and with fewer design errors (Twari & Franklin, 1994). It will also make them more confident that the designed solution is consistent. Formal representation of constraints is particularly important for the employment of computer automation in the integrity validation process. We outline the key issues that should be dealt with by a constraint management system that manages evolving constraints in a versioning environment:

1. Flexibility in manipulating evolutionary constraints.
2. Easy capture of design constraints taking cognizance that the designer is not a programmer.
3. Support for the re-use of unchanged constraints while avoiding their re-definition.
4. Validation of primitive versions and configurations on their creation on explicit user request:
   a. Enforcing the currently active set of constraints through coupling them automatically to each version;
   b. Providing informative messages to report violations, enabling the designers to correct them appropriately;
   c. Assisting the designer in selecting eligible component versions to form a configuration; and
   d. Reducing the validation cost.
5. Validation of existing versions against a newly applicable set of integrity constraints in the event of constraint changes, and report to the designer whether, or to what extent, the existing versions comply with the new constraints.

6. Managing constraint evolution histories, which facilitates:
   a. Understanding the designer of the association between constraint evolution and object evolution;
   b. Retrieval of constraints relevant to each version and thus helps the designer to view the different consistency states throughout the design process; and
   c. Change to a previous stage of constraint evolution if one line of constraint evolution does not work out.

According to our observation, the issues 4b and 4d on validation mechanism are required for any constraint management system irrespective of whether constraints are evolving or static. Other issues are crucial for a system with evolutionary constraints and, in particular, the issues defined in 4c, 5, and 6 are important for a constraint management system that deals with both evolving constraints and versions.

Issues Relating to Evolutionary Design Constraint Management

Evolutionary Nature of Design Constraints

In managing the design constraints and thus providing an automatic integrity validation mechanism, a common problem encountered is that of constraint evolution during the design process.

Constraint evolution may result in:

1. Identification of new constraints,
2. Modification of existing constraints,
3. Omission of existing constraints, or
4. Any combination of the above 1-3.

A PDS is a dynamic document that evolves and changes with the progress of the design reflecting these constraint modifications (Dieter, 1991). In investigations with a group of designers, it was revealed that this type of an evolutionary environment requires the constraint changes to be agreed upon through communication and discussion with other designers to avoid design conflicts. Supporting automatic integrity validation whilst catering for constraint evolution requires a number of issues to be dealt with as stated in the previous section, including incorporation of constraint modifications and the enforcement of the currently active set of constraints for each validation.

Constraint Categories in Design Environment

To simplify discussions on artefact design, we introduce a running example, a bicycle design, which will be used throughout this chapter. The structure of the bicycle is indicated in Figure 1.

There are a number of different constraint classification schemes in the literature with respect to the design environment (Lin et al., 1996a, 1996b; Ram et al., 1997; Twari & Franklin, 1994). However, in this chapter, we only consider design constraints imposed on the properties of artefacts (Chandrasekaran, 1989) which can be divided into:

1. Physical constraints (Lin et al., 1996a, 1996b), which refer to “physical” properties of the artefact. For example, in a bicycle frame there may be constraints on the frame size, seat angle, head angle, material, and colour.
2. Structural constraints (Lin et al., 1996a, 1996b), which enforce the composition of sub-component objects to form a complex object. For example, a bicycle may consist of a frame, two wheels, saddle, and a handle; or consider the formation features of the complex artefact. For example, to form a bicycle the frame size should be > 40 cm if the diameter of the wheel is > 60 cm.

It is acknowledged that evolution in structural constraints in the “composition” category, that is, changes relevant to the composition of sub-components, affects the schema/class definition of the database. This leads to schema/class versioning, which is beyond the scope of this research, as our main concern is on object versioning. Therefore, in our research, we consider only the value-based constraints relevant to physical and structural constraints that enforce the accuracy of the data values assigned to the corresponding attributes of the object. To this end, we consider the following value-based constraint classification:

Range constraints (e.g., `35 cm` ≤ `frame_size` ≤ `58 cm`) which check the validity of an attribute value specified by the designer by returning true if the value is within the specified range and false otherwise.

Enumeration constraints (e.g., `{steel, stainless steel, titanium, aluminium}` ⊇ `material`) which may also return a Boolean after validation.

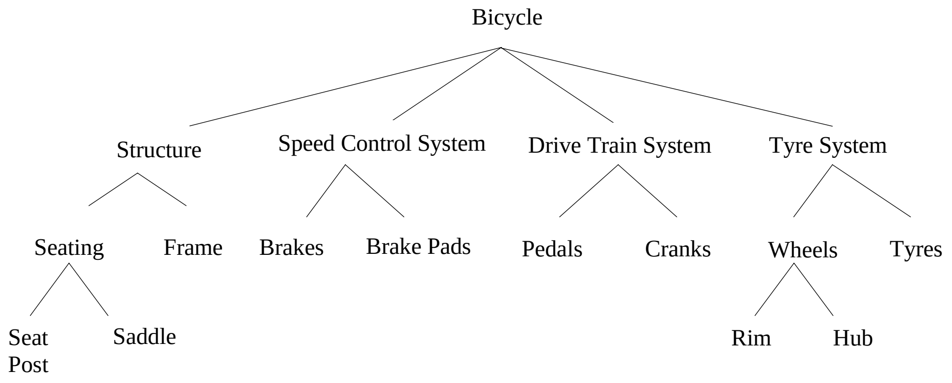
Relationship constraints are used to establish a relationship between two or more attributes that may belong to either the same or different objects. The validation of this constraint may either return a derived value (e.g., `Spoke anchor angle = Hub spoke angle * Wheel cross number`[1]) or a Boolean (e.g., checking whether `frame_size` > `40 cm =>` `wheel_diameter` > `60 cm`).

It is possible for the constraints to change from one constraint type to another constraint type when they evolve. Some of these can be, for example:

enumeration** to **range (e.g., `{40,44,45,50}` ⊇ `frame_size may change to 45 cm` ≤ `frame_size` ≤ `55 cm`)

range to enumeration (e.g., `65` ≤ `head_angle` ≤ `75 in frame design may change to {68,71,72}` ⊇ `head_angle`).

Figure 1. Artefact decomposition for bicycle design



relationship *to* ***enumeration*** *or* ***range*** (e.g., `brakeSidediameter = driveSidediameter` constraint in hub may change to a range constraint, as in `60 mm` $\leq$ `brakeSidediameter` $\leq$ `65mm` or an enumeration constraint `{62,65}` $\supseteq$ `brakeSidediameter`).

In engineering design, constraints can be categorised as hard and soft (Friesen, Gauthier-Villars, Lefebvre, & Vieille, 1994). Hard constraints are restrictive and cannot be violated, while soft constraints are not restrictive and can be violated if required.

Artefact Decomposition

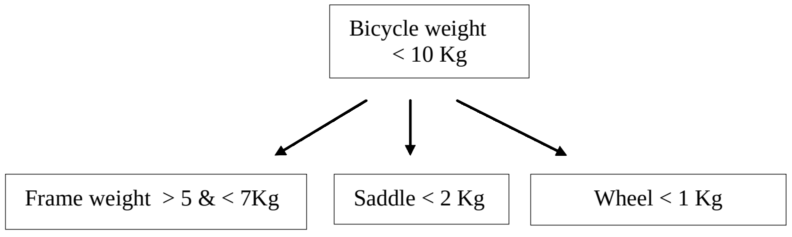
An artefact or a design object can be a complex one consisting of a number of component artefacts (which are also referred to as component objects or constituent objects in this chapter). As a result, they are too extensive to be analysed as a whole, and solutions based on local approaches often allow them to be solved more quickly. Artefact decomposition is an important aspect in this regard in engineering design. It is based on the assumption that any object, which is not treated as a single element, can be decomposed into other objects (Rosenman, 1993). In the decomposition process, the complex design artefact is often decomposed into its component parts, which in turn may be decomposed recursively in such a way as to allow the individual/group of designers to address them separately (Baldwin & Chung, 1995; Dieter, 1991; Sauce, Martini, & Powell, 1992). Consequently, if D is the artefact to be designed, the decomposition would derive $D \rightarrow \{D_1, D_2, \ldots.. D_n\}$ where D_is are components of D. These components may also be complex which can in turn be decomposed into a number of other components (Chandrasekaran, 1989). Therefore, the decomposition process usually goes down to the deepest possible level until an indivisible component is encountered, yielding a component hierarchy. We use the term component in general to address both indivisible and complex components. To distinguish between the two component types when necessary, the terms *primitive component* and *complex component* are used. For some design problems, there may be a number of possible decompositions available. An example for decomposition is illustrated in Figure 1 for the bicycle design.

Constraint Decomposition

Due to artefact decomposition, design constraints can be identified as local and global in relation to component and complex objects respectively. Local constraints, also known as intra-object constraints, belong entirely to the local object and are valid only in that particular object space. Global constraints, also known as inter-object constraints, are those constraints that belong to a complex object/configuration and whose validity may span multiple objects. However, it is necessary to understand that general constraints at one level of the composition hierarchy will be local constraints to its upper level. In the CAD environment, there is also a possibility to split some (global) constraints on complex objects, into a set of local constraints on components or leaf objects (Freeman-Benson & Borning, 1992; Gehani & Jagadish, 1991; Jagadish & Qian, 1992; Lin et al., 1996a). For example, the weight constraint of an artefact is often decomposed into local constraints on the components of that artefact. This is shown in Figure 2 using the bicycle example. The decomposition of global constraints to a set of local constraints is a manual process performed by the designers. However, defining strategies for constraint decomposition is beyond the scope of this chapter.

Later in the design process, the resultant designed components from the decomposition are integrated (re-composition) to form the complex design object. Consequently, as with the overall design, component designs should be handled systematically. Otherwise, vital factors will be omitted and the overall product will be affected (Pugh, 1991). To carry out successful component design and thus to make the final design a success, it is necessary to consider the constraints or limitations that should be imposed on each component. Hence, the main emphasis in component design will be local performance, local environment, and local constraints (Pugh, 1991). These aspects are defined in the relevant *component design specification* (CDS) for each component within the envelope of the total PDS (Pugh, 1991). Although the aim of the decomposition is

Figure 2. Constraint decomposition



to make the component objects as independent as possible, they may not always be truly independent (Chandrasekaran, 1989; Prasad, 1996).

Issues Relating to Managing Evolving Design Artefacts

Versioning and Configuration Management

There has been a lot of research over the years into version management in the areas of CASE (Computer-Aided Software Engineering) and CAD (Computer-Aided Design). CASE systems support the development and maintenance of software, and versioning is aimed at providing facilities for managing evolving source code modules (Loomis, 1992). SCCS (Rochkind, 1975) and RCS (Tichy, 1985) are early file-based version systems that have influenced other later version models as described by Choi and Kwon (1997) and Plaice and Wadge (1993). Even though there are some similarities between CAD and CASE version models, they have evolved almost independently due to their different requirements (Estublier, Favre, & Morat, 1998; Westfechtel & Conradi, 1998). The fundamental difference between the two areas is due to the kinds of objects to be managed. CASE has focused mainly on the management of software objects — mainly programs — represented as text files. In contrast, objects in CAD applications are non-textual and are defined as database objects. Some of the differences between the two areas CAD and CASE are highlighted in Table 1 (Estublier et al., 1998).

We will not consider versioning and configuration of software design any further in this chapter.

As indicated earlier, a composite design artefact can be logically decomposed into its component parts, which in turn may be decomposed recursively in such a way to allow the individual/group of designers to address them separately (Sauce et al., 1992). Subsequently during the design process, this decomposed complex artefact is recomposed by combining its constituent component artefacts. In a versioning environment, each composite and component design object may have a number of versions, making this composition task cumbersome. For example, if a composite object/artefact is composed of m objects, each one in n versions, there can be up to n^m different combinations to be used for the construction of configurations out of which only a few may be actually

Table 1. Differences between CAD and CASE

	CAD	CASE
Data Model	Object modelling	Weak data models
Main issue	Object handling	File handling
Decomposition	Based on the real objects exist in the structure	Governed by aspects like ease of use, efficiency, functionality
Components	Parts/assemblies	Modules, files
Relation	Composition, generalisation, specialisation	Dependence relationship, file hierarchy
Representation	Database	File system

consistent or relevant. A configuration is defined as a structurally composite object, that is, an object composed of other objects, formed by combining other configurations (known as sub-configurations) and versions of different primitive and/or composite objects. Useful configurations are formed from versions of constituent objects that are consistent together (Cellary & Jomier, 1990). Since different configurations may exist due to differing constituent version combinations, it would be useful if the designer had the facility to store meaningful configurations and to keep track of configuration evolution. This can be achieved if configurations can be managed as versions. Another benefit in this situation is that objects and configurations may be freely combined to construct higher-level configurations (Golendziner & Santos, 1995).

To produce a consistent configuration, all of the constituent versions participating in that configuration should satisfy the inter-dependency constraints imposed on them. Inter-dependency constraints are known as global constraints since their validity spans multiple objects. Successful validation of inter-dependency constraints ensures that a configuration is consistent with its constituent component versions. Consequently, it is important to augment configuration management with an integrity mechanism that checks each configuration for data correctness and consistency. However, due to frequent constraint changes, different configurations of the same composite object may have to satisfy different sets of inter-dependency constraints at different times. We have not located in the literature, a version model that deals with the consistency of design configurations, through managing constraint evolution.

Configuration Management Requirements

In practical terms, the designer should be able to construct configurations by selecting and combining component versions without the need for programming skill. In terms of the designer's role in managing the data validation aspects regarding the configuration consistency, he/she should be freed from the need of:

- Any manual checking of the inter-dependency constraints to ensure the consistency of configurations, or
- Typing the currently applicable inter-dependency constraints every time a new configuration is constructed, or
- Changing and compiling application programs when constraints change.

Checking the consistency of the component versions that are selected to form a configuration should be handled automatically by the configuration management system based on the inter-dependency constraints active in the system at the time. Our configuration management model handles configurations as versions. It is possible that a constituent version in a previous configuration version should not be part of the new configuration version, if that constituent version does not adhere to the new set of constraints applicable on its artefact. Therefore, we suggest that each new configuration should be created from scratch by selecting a combination of constituent object versions that are consistent together. The derivation of new configuration versions from preceding configurations (and therefore the parent-child relationship) is not applicable in this situation. Consequently, instead of using a version derivation graph, a configuration version set is represented as an ordered list using the temporal order of creation. This ordering does not contain the logical inference that the previous configuration version in the list is the parent of its successor version. Full details of our configuration management model are given in Carnduff and Goonetillake (2004).

It is necessary to clarify the terminology that will be used in this chapter. Configuration versions are used to handle versions of the composite objects. The term "primitive object version" is used to refer to versions of primitive objects (i.e., objects without any constituent/component objects) when it is necessary to distinguish them from the configuration versions. Otherwise, the word "version" or "object version" is used in general in the rest of this chapter for both primitive and configuration versions. The versions that form a configuration version are called "constituent" or "component versions" which can themselves be either primitive or configuration versions. The "object version model" or "version model" includes the management of both "primitive versions" (of the primitive objects) and "configuration versions" (of the composite objects). The terms "primitive version model" and "configuration version model" are used if it is necessary to differentiate between these management mechanisms.

Representing Evolving Design Constraints

The management of integrity constraints in an object versioning environment is non-trivial. The design requirements of artefacts are represented as constraints both at configuration and sub- component levels. These constraints specify the characteristics of individual components and the way in which they fit together (Pugh, 1991). Therefore, integrity validation is required both at configuration and sub-component levels. These constraints are distributed amongst the designers of each of the configurations and sub-components. In the database literature, constraints have usually been defined either as active rules (Ceri & Fraternali, 1997; Diaz, Paton, & Gray, 1991; Urban, Karadimce, & Nannapaneni, 1992) attached to classes or within class/schema definitions (Ceri & Fraternali, 1997). It is difficult to have each of several different objects belonging to the same class complying with its own set of constraints, if the means of specifying these constraints relates to the class rather than its instances. The unsatisfactory solution is to introduce a new class for each group of versions that shares common constraints. This type of schema/class evolution is undesirable as it imposes significant complexity on the versioning environment with an undesirable proliferation of sub-types. This greatly complicates the type lattice and makes it difficult for the designer (at the time of version creation) to understand to which sub-type a new version should become a member (Katz & Chang, 1992).

Limitations of Current Approaches to Managing Evolving Design Constraints

It does not appear that existing version models (Cellary & Jomier, 1990; Dittrich & Lorie, 1988; Katz, 1990; Krishnamurthy & Law, 1995) provide any computer-supported integrity validation mechanism for sub-component versions or configurations.

Buchmann, Carrera, and Vazquez-Galindo, (1992) and Ram, et al. (1997) describe constraint-handling mechanisms for engineering design environments. These mechanisms provide a more flexible environment in which constraints can be inserted, deleted, and modified due to constraint evolution, without modification of the application program. This flexibility is achieved by separating constraints from the class definition and treating constraints themselves as constraint objects in the database (Buchmann et al., 1992) or aggregating constraints into a separate class called the constraint meta-object (Ram et al., 1997). Constraint objects (Buchmann et al., 1992) or constraint meta-objects (Ram et al., 1997) are associated with design objects. In this way, different objects belonging to the same class can have their own constraints. In spite of the advantages offered, there are also a number of drawbacks associated with each of the two approaches. One major drawback is that associating constraints with design objects is not automatic, and both approaches require the designer to associate currently active constraints with the corresponding design objects. The approach proposed by Ram, et al. (1997) requires the designer to change and compile programs written in C++ to attach and detach each constraint to and from the design object. We do not consider this to be appropriate for designer users who are not programmers. Data validation is cumbersome in both approaches since:

- Constraints are normal database objects and data validation requires retrieving properties of each constraint stored as attribute values in the corresponding database object (Buchmann et al., 1992) and
- Constraints defined in the constraint meta-object class require interpretation by another program (Ram et al., 1997).

Moreover, these mechanisms have not been considered against a versioning environment, in particular how evolving constraints should be managed in presence of object versions.

The systems proposed in (Twari & Franklin, 1994; Yoo & Suh, 1999) are concerned with managing design consistencies of configuration objects and sub-component objects in a co-operative design environment. These mechanisms mainly consider reducing the integrity validation overhead in a distributed environment. Consequently, the integrity constraints are separated into local and global constraints and distributed into corresponding local and global workspaces. This eliminates unnecessary interaction with remote databases.

However, these approaches do not consider the connection between evolving constraints and object versions, so they do not cater for them.

Doucet, Gancarski, Jomier, and Monties (1996) propose a framework to deal with both versions and integrity constraints. The version model used in this framework is based on the Database Version (DBV) approach (Cellary & Jomier, 1990). Each database version represents a configuration composed of one version of each constituent object. This approach deals with constraint evolution through the production of DBVs. Consistency is confined to each database version and consequent configuration level. The checking of some constraints spans multiple database versions, which adds to the complexity. It is unlikely that this model can be applied to a co-operative and distributed design environment due to the very high unit of versioning granularity. Furthermore, in a co-operative design environment, the system may end up with a large number of database versions, which will impose a considerable storage overhead. Users of the framework must define constraints and their modifications in programs written in a logic language, an approach with which designers are unlikely to be comfortable. Nevertheless, to our knowledge this is the only work that deals with evolving integrity constraints in the presence of object versions.

Constraint Validation

Constraint validation is performed to check whether the attribute values of the design object version adhere to the given set of constraints (Thompson et al., 1995). The validation is successful if attribute values satisfy the applicable set of constraints. Normally, constraint validation is associated with a transaction, which is considered to bring a database from one consistent state to another after a successful validation. Consequently, constraints are normally checked in one of two modes (Ceri & Fraternali, 1997).

- Deferred constraints are evaluated at transaction commit; the commit fails if the transaction violates any one of them.
- Immediate constraints are evaluated during the course of transaction so as to detect violations as soon as they occur. Their effect is to abort the transaction immediately after a violation.

In a design environment, it is desirable for the designer to work without constraint checking until he or she wants to verify consistency (Buchmann et al., 1992). Consequently, the constraint validation process should be invoked through explicit user request. Therefore, in CAD the deferred constraint checking mechanism is preferred to the immediate constraint checking mechanism. Moreover, Buchmann, et al. (1992) suggest that with the deferred checking mechanism, it is useful for the designer to be provided with checking granularities other than end-of-transaction level. These granularities may include constraint validation on a specific object or on one or more selected attributes. A transaction that takes a DB from one consistent state to another consistent state mainly deals with maintaining the consistency of the whole DB. This is important for commercial DBs. However, the notion of consistency in the design environment differs from this, and confines consistency to each version object. Therefore, in our opinion, constraint checking on an object basis, as suggested by Buchmann, et al. (1992), is more appropriate for engineering design applications. In addition, a transaction is normally aborted in the event of a constraint violation which causes the designers to lose the work done so far (Doucet & Monties, 1997). In engineering design, this is an inappropriate constraint validation mechanism, and some other options should be made available. For example, whenever constraints are violated, the designers should be warned of the violations in a manner that enables them to correct these violations appropriately. It should allow the designer to change and re-specify the values in such a way that the constraints are satisfied (Thompson et al., 1995).

Some research has focused on automating the violation correction process using active constraints (Ceri, Fraternali, Paraboschi, & Tance, 1994; Ceri & Widom, 1990; Urban et al., 1992), which are fired by a triggering mechanism. This involves either automatic replacement or deletion of data in order to keep the database consistent. We believe that in a CAD environment the designer should be given the facility to correct violations. To this end, there should be an informative messaging system to specify the violated constraints and the reasons for violations. This helps the end user/designer to take necessary actions to correct the violations, based on the information he or she is sent. The validation should also highlight the distinction between hard and soft constraints where:

- Violation of soft integrity constraints are merely flagged to warn the designer whilst enabling him/her to proceed in spite of the violation.

- Violation of hard integrity constraints should warn the designer and at the same time prevent designers from proceeding any further without correcting violations.

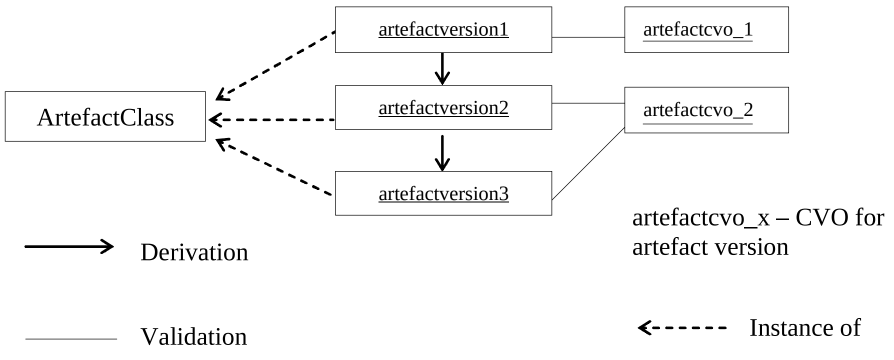
When there are multiple constraints associated with an object, there will also be a question regarding whether the order of execution is important. According to Bassiliades and Vlahavas (1994) the order of execution is not important when constraints do not perform any updates on the database.

A Conceptual Framework to Integrity Constraint Management

As indicated earlier in this chapter, it is evident that the existing version models do not meet the constraint management issues required for a co-operative and distributed versioning environment. We present our framework as a general solution to a common problem in engineering design, irrespective of the design area. In proposing our framework, we take the view that evolving integrity constraints in an engineering design environment exhibit similar characteristics to that of object version data. Embedding this type of dynamic constraint information in a class definition clearly requires a re-definition of that class whenever a change occurs. It is also necessary to support the features discussed in the section on issues relating to evolutionary design constraint management. Our framework is based on the concept of the CVO. Each CVO contains a set of integrity constraints that needs to be satisfied for an object version of a particular artefact at some instant in time. For a bicycle frame artefact, the CVO may contain the currently-applicable constraints for its frame size, tube diameter, material, and seat tube length.

Maintaining CVOs provides the flexibility to enable changed constraints to be captured independently from artefact class definitions. Constraint evolution is managed by producing new CVOs. CVOs govern the validity of data in object versions. At the time of version creation, each new design object version is automatically coupled with the current active CVO for data validation. This mechanism facilitates the validation of different object versions of an artefact, on different sets of constraints. The initial framework is depicted in Figure 3. As shown in the diagram, the object versions (artefactversion1, artefactversion2

Figure 3. Initial framework



and artefactversion3) are instances of the same artefact class but have been validated under different CVOs (artefactCVO_1 and artefactCVO_2). Any number of object versions can be associated with one CVO. Grouping a set of constraints in a CVO also facilitates the retrieval of the set of constraints that apply to a particular version. It is important to impose a uniform CVO naming scheme for their unique identification within the design environment.

CVOs are not split into different categories to hold different constraint types. A CVO may contain any combination of currently-active constraints of any category (e.g., range, enumeration, relationship, hard and soft). However, as a result of the co-operative nature of design and the resultant hierarchical database architecture that separates private and shared work, CVOs can be divided conceptually into two categories (Goonetillake, Carnduff, & Gray, 2001): local CVOs and global CVOs. Local CVOs containing local constraints applicable to the sub-components, may reside in local workspaces. Global CVOs containing global constraints applicable to the complex object, may reside in global workspaces. However, the terms *local* and *global* are relative in relation to CVOs and are dependent on the number of levels in the design database hierarchy. A global object at one level may be a local object at the next level up in the hierarchy. The separation of local constraints and global constraints makes it easier:

- To verify data consistency both at sub-component level and configuration level in a distributed environment. It eliminates the complexity of data validation that would otherwise have been incurred due to remote database access;

- To manage constraint changes in a distributed environment; for example, when there are constraint changes, designers are only required to deal with the constraints applicable to their design components;
- For designers to understand and be familiar with the constraints local to the sub-components assigned to them.

The idea of a default CVO is important in maintaining consistency within the design environment. In a cooperative and distributed design environment, the default CVO enables designers to consistently identify the current active CVO for a particular artefact. Usually, the last CVO created in the system will become the default CVO. The default object version should generally be linked with the default CVO (Goonetillake et al., 2001).

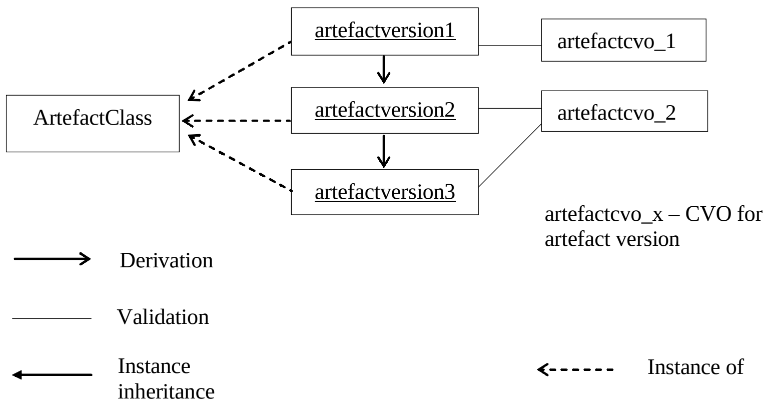
Managing Constraint Evolution

A CVO is partially or wholly invalidated for the following reasons, reiterating an earlier point:

1. A modification of existing constraints. For example, the tube diameter constraint of bicycle frame may change from (`25mm ≤ tube_diameter ≤ 28 mm`) to (`29mm ≤ tube_diameter ≤ 31 mm`);
2. Introduction of new constraints (e.g., the head angle of a bicycle frame which was not restricted before can be constrained to a particular set of values);
3. Omission of previously used constraints. For example, moving on to free-size bicycle frames means the size constraint will no longer be applicable; or
4. Any combination of 1-3.

A new CVO has to be created in a situation like this. It is important to consider how the relationship between CVOs is maintained. The important aspects that should be taken into account at this stage are constraint evolution representation and constraint reuse (avoiding constraint re-definition). Consequently, a new CVO contains only the changes to its parent CVO constraint set, and the means of referring to its parent for the unchanged constraints. When invoking constraints for data validation from the child CVO, there should be means to:

Figure 4. Finalised framework



- Delegate constraints to the parent object if not declared within the child object;
- Override constraints in the parent when they are redefined in the child object; and
- Omit constraints defined in the parent frame/object when they are no longer applicable.

Since CVOs are objects, we consider that instance inheritance (Wilkes, 1988) with an overriding inheritance mechanism is the means to best provide these CVO features. Classical-type inheritance differs from instance inheritance and does not cater for these requirements. For example, using type inheritance there are problems with constraint omission and overriding in a child object (Bassiliades & Vlahavas, 1994). Type inheritance may also cause class evolution (Katz & Chang, 1992). Within CVOs each constraint is a rule. A child CVO only contains the refined constraints (as rules) and inherits the unchanged ones from its parent. This refined framework is depicted in Figure 4. For reason of simplicity, it is considered that the CVO evolution graph takes the form of a tree rather than a directed acyclic graph.

Management of CVO Creation in Practical Design Environment

It is assumed that the initial CVOs for each component of the complex artefact are based on the initial design constraints. The initial CVOs are also assumed to be located in each designer's local workspace. As a result of design decomposition, individual designers are only required to deal with the CVOs applicable to their own design components. Designers start working in their own local workspace with the initial CVO assigned to them. New CVOs may have to be created and accommodated in the framework as the design progresses and constraints evolve. This co-operative environment requires a good information flow between the designers, especially when a change occurs. Each change is governed by the appropriate project leader with the agreement of subordinate individual designers in his/her project group. Modification to some constraints is not under the control of the designer and requires higher-level approval (e.g., negotiation with the customer). However, all modifications should be discussed with the design team and checked manually to find out whether the suggested modifications have an impact on other constraints. All designers have to agree to modifications before they come into effect. The agreed modifications should then be communicated to the appropriate designers by the project/team leader. This procedure will avoid conflict situations where:

1. Different designers make conflicting recommendations regarding attribute values in a constraint;
2. A change made by one designer has an impact on some other constraints, making it impossible to offer consistent values for other attribute values; or
3. A change in one constraint will adversely affect optimality of other constraints.

Because of design decomposition, designers are only required to know the constraint changes applicable to their own design component/s. These changes will lead to the creation of new CVOs. Ideally, the proposed framework should be managed so that design engineers themselves can easily create the executable form of CVOs. It is unrealistic to expect designers to write and compile programs when they have little idea of the inner workings of the computer system.

Data Validation for Object Versions

CVOs are used to check the consistency or adherence of a particular object version to a given set of design requirements before making them persistent in the database. The framework is responsible for communicating with the corresponding (normally default) CVO and invoking the relevant constraints for validation. Data validation is normally associated with a database transaction, when a change or update in attribute values directly affects the consistent state of the database (Doucet & Monties, 1997). One would expect this transaction to be aborted in the event of a constraint violation. This would cause designers to lose the potentially large amounts of design work. Consequently, rather than aborting the transaction that causes the violation, a constraint validation mechanism should make some other options available. Some reported research has focused on automating the violation correction process using active constraints (Twari & Franklin, 1994), which are fired by a trigger mechanism. This involves either automatic replacement or deletion of data in order to keep the database consistent. Nevertheless, we believe that in a CAD environment designers should be given the facility to correct violations in order to maintain the autonomy of participating disciplines. Accordingly in our framework, when a new object version is derived, data validation is invoked by explicit user request and performed on an object-by-object basis. At the sub-component level, the designer should be given the option to proceed by changing the attribute value in order to satisfy the constraint. At the configuration level, validation is partially concerned with selecting valid sub-components. An informative messaging system is required to specify the violated constraints and to give the reasons for violations. This would help the designer to take the necessary actions to correct the violations.

Changing the values of a parent object version should not affect the database state until a new version is formed. This new version incorporates these changes and is made persistent in the database after successful validation (may still have soft constraint violations). Forming a new version after the successful completion of the validation process ensures the design integrity of object versions stored in the database. A bottom-up approach is used in validating data for complex objects. To this end, the sub-components are the first to be validated against their local constraints. The successfully-validated sub-components are then checked-in to the global workspace/database for validation against global constraints to form a configuration/complex object.

The validation of data integrity is an inherently-expensive and time-consuming process. We have explored mechanisms for reducing the evaluation cost involved in data validation. When a new object version is derived based on a parent version, it may not be necessary to verify all of the constraints. If the state of the parent object version is acceptable, it may only be necessary to select and validate some of the attribute values. These attributes are those whose values have been changed in the new object version or where modification of the constraints imposed on them has resulted in a change of value. We call this delta validation, details of which have been presented in Goonetillake, et al. (2001). The integrity validation overhead in a distributed environment (due to remote database access) is reduced by separating integrity constraints into local and global constraints and distributing them into corresponding local and global workspaces.

Validation of an Existing Object Version

When a new CVO is created it is reasonable to check some of the existing object versions against the integrity constraints within the newly-created default CVO. Validation of existing object versions against the new CVO is too tedious to be performed manually. The proposed framework automates this task with designer's consent after a new CVO is created. In this way, the designer finds out whether any existing versions comply with the new constraints. This is also beneficial from the point of view that if an existing object version (or versions) has been validated successfully, the necessity for the creation of a new object version can be eliminated. To optimise system efficiency, only a selected set of design object versions is validated in this situation. The object versions selected for this type of validation are those created under the parent CVO of the newly created CVO. For example, the system will only pick the shaded design versions (v1, v2, v3, v4) for automatic validation when the CVO3 is created (Figure 5). The designer will be notified on the outcome of the validation.

A successful validation is feasible, for example, if the child CVO contains constraints with relaxed or intersected domains. The user is notified when a successfully-validated design version is encountered. The designer has the option to allow the system to proceed with the other versions (if there are any) or to stop the validation at that point. If there are a number of successfully-validated design versions the designer has the option to set one of the versions to be the new default object version. This also requires that an object version is associated with more than one CVO as depicted in Figure 6.

If there are no successful validations from the existing set of design versions, the framework is capable of proposing to the designer those versions which are closest to successful validation, that is, the version which has the least number

Figure 5. Validation of existing design versions

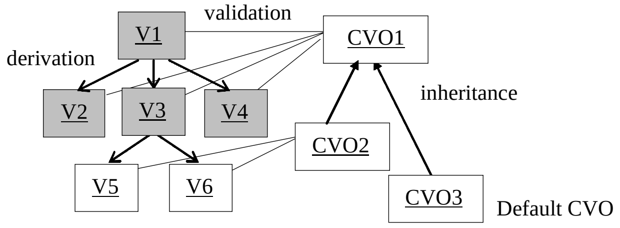
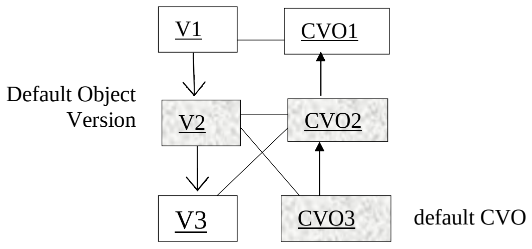


Figure 6. Successful validation of an existing object version



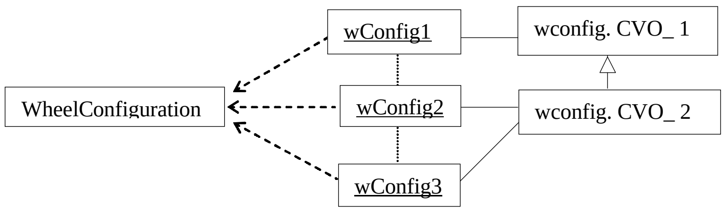
of constraint violations. In proposing the closest versions, the designer is provided with a description of the constraint violations. This will assist the designer in deciding upon the most suitable design version from which to derive the next new design object version. However, it should be noted that the designer does not have to stick to the design versions proposed by the system and is able to enforce validation against any existing design object version he or she wants.

Configuration Versions and CVOs

It is necessary to check whether the selected versions constitute a consistent configuration through satisfying a given set of constraints. The constraints that are used to validate configurations are known as *inter-dependent* or *global* constraints. The scope of these constraints spans multiple objects. At the configuration level, validation is mainly concerned with selecting valid sub-components. However, a set of design constraints will evolve over time with modifications, the addition of new constraints, and the omission of existing constraints.

Previous sections described the management of CVOs in the presence of versions in general. However, this section provides an explanation of the means

Figure 7. Integrity validation for wheel configuration versions



by which the integrity validation system validates configuration versions. With respect to configuration versions, the sub-components are first validated against their local constraints. As a result, the constituent component versions of a configuration are assumed to be individually consistent with their local constraints. Consequently, at the configuration level a configuration version will only have to be validated against its global constraints that express the inter-dependency constraints between the selected component object versions of that configuration. For example, for a wheel configuration, the inter-dependencies could be `(number of spoke holes in hub = number of holes in rim) and if (wheel finish = blue) then (rim finish = black and hub finish = blue)`. Each configuration version is automatically associated with the default CVO and validated against its constraints at the time of the configuration version creation (Figure 7).

In setting the default configuration version, it is necessary for the selected configuration version to adhere to the default CVO applicable at that time. The constituent versions of the default configuration version do not necessarily have to be default versions so long as they adhere to the CVOs currently imposed on them.

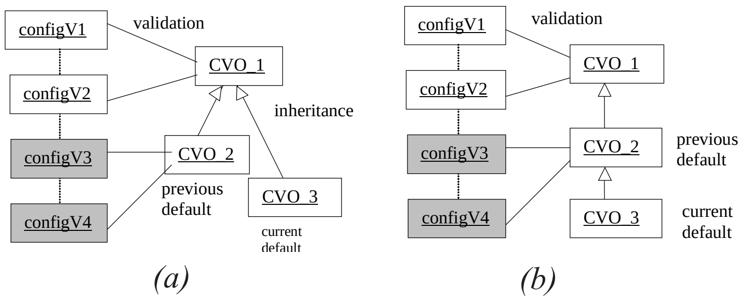
Data Validation for Configuration Versions

Configuration construction is concerned with selecting the versions that are consistent together. As the bottom-up approach is used in validating data for configuration versions, each selected object version participating in the configuration first complies with the local design constraints currently imposed on their objects (i.e., adhere to the constraints in the default CVO of each constituent object). At the configuration level, the validation process mainly checks the inter-dependency constraints that exist among constituent versions and is invoked when a new configuration version is created or when a new CVO is created. Upon the creation of a new configuration version, data validation is invoked by explicit user request. Any constraint violations are reported to the designer, which enables him or her to repair the violation by selecting the correct constituent versions. Each new configuration version is made persistent in the database after successful validation. Successful validation implies that component/constituent versions in the configuration are consistent together. However, the delta validation mechanism defined in Goonetillake, et al. (2001), will not be applicable at the configuration level, as each configuration version is created from scratch and is not derived from a parent version, as is the case with primitive versions.

When a new CVO is created, on the other hand, the existing configuration versions are validated against the newly created CVO. This is beneficial from the point of view that if an existing configuration version (or versions) has been validated successfully, the necessity for the creation of a new configuration version can be eliminated. In this situation the system invokes the integrity validation process on the creation of a new CVO. Only a selected set of configuration versions is validated to optimise system efficiency. The configuration versions selected for this type of validation are those created under the previous default CVO of the configuration. These configuration versions share the high probability of having constituent versions that adhere to their corresponding default CVOs. Nevertheless, the constituent versions of the selected configuration versions should be further checked before proceeding with the validation. For example, the system will only pick the shaded configuration versions (configV3 and configV4) for automatic validation when the CVO_3 is created (Figures 8a and 8b) provided that their constituent versions adhere to the default CVO applicable on each of the corresponding constituent objects. It is possible to impose delta validation on configuration versions, by validating existing configuration versions only if the newly created CVO is directly connected to the previous default CVO as parent and child, as depicted in Figure 8b. This delta validation is based on the assumption that if the state of the configuration version is consistent with the constraints in the parent CVO, it may only be necessary to perform validation against the changed constraints in the new (or the child) CVO, thus reducing validation costs further. If there are no successful validations, a new configuration version will be created from scratch.

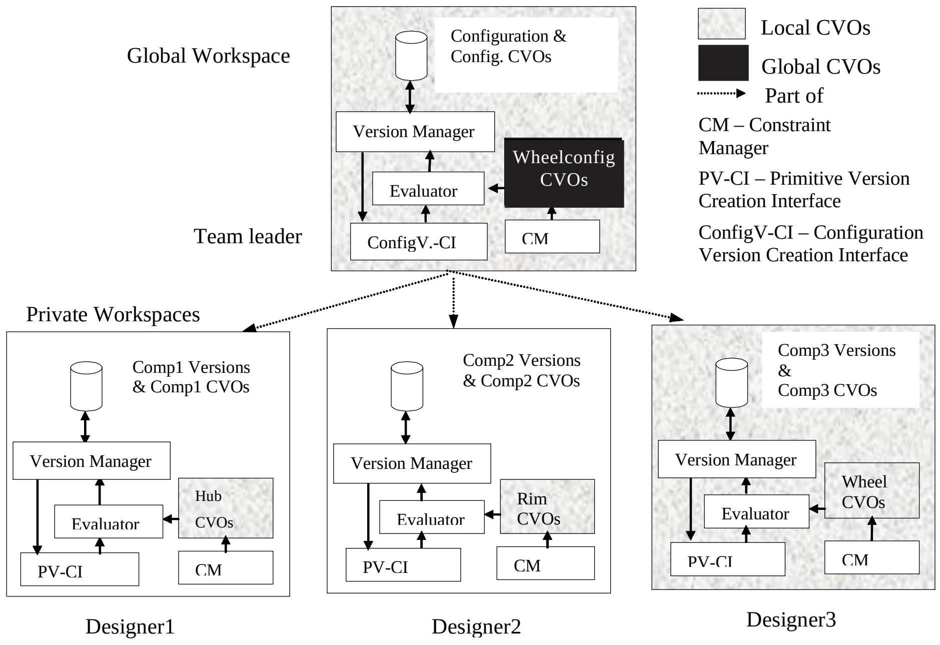
To illustrate how the proposed model would operate in a co-operative and distributed environment, we again turn to the wheel artefact, which consists of the two components, hub and rim. These components are assigned to individual designers who work on them separately. The designers use different worksta-

Figure 8. Validation of existing configuration version



tions (known as workspaces), which belong to the same network. Each workspace consists of a version manager, which also handles configuration versions, a database and a constraint manager (Figure 9). Although depicted separately for illustrative purposes, the constraint evaluator and the corresponding CVOs are components of the constraint manager. The primitive and configuration version creation interfaces are components of the constraint manager. The successfully-validated component versions are checked into the next level up in the hierarchy to form wheel configuration versions. At this level, the global constraints are validated and are used by the team/project leader to form a

Figure 9. Integrity validation model in a practical design environment



configuration version by selecting valid component versions. This activity is carried out using the user interface indicated in the next section of the chapter.

Management of Evolving Constraints by End Users

An important focus of this work was to develop the means of representing evolving design constraints and providing a framework for validating design objects against these constraints. As interesting as this may be, it would not be of great use to designers if they were unable to use these facilities without writing programming code. We recognise that engineering designers are not normally computer programmers, so we have developed user interfaces to allow designers to perform their work with much of the complexity of our versioning/configuration management and integrity constraint management systems being hidden from the user. This facility is best described by illustrating, in outline detail, a sequence of design steps through the various components of our user interface, where the designer is able to communicate his/her decisions through the interface in a form-filling manner. This illustration is largely made through a series of screen dumps showing various aspects of the system user interface. For reasons of space, we have not included every stage in the design development, nor detailed explanation of the underlying processes.

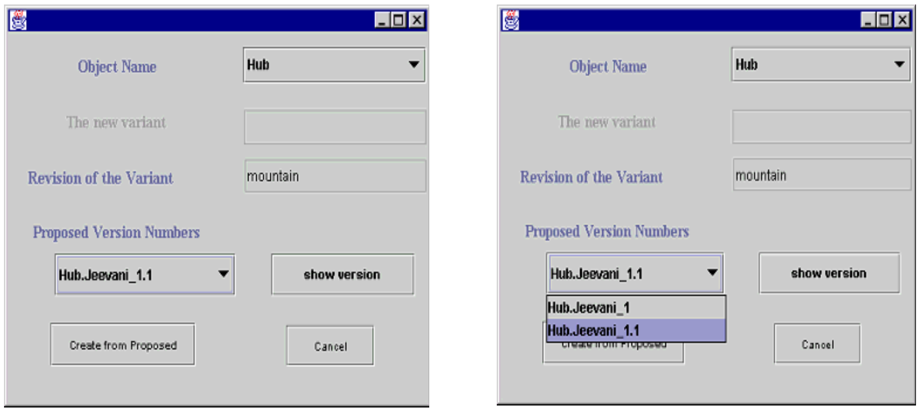
Validation of a Primitive Version on its Creation

This is demonstrated using a primitive artefact of the bicycle design, a hub artefact. To carry out the demonstration of primitive version validation on its creation, it is assumed that the following are already in place in the system:

1. *hubCVO_1*, which is the default CVO for the hub artefact; and
2. *Hub.Jeevani_1* and *Hub.Jeevani_1.1* which are active versions created for the hub artefact. These versions were validated against the default CVO (*hubCVO_1*) applicable on the hub artefact at their creation time.

The prototype enables the designer to invoke the primitive version creation/ derivation option from the main menu. On selection of this option, the designer

Figure 10. User interface that enables the designer to select the required parameters for a new (primitive) version derivation



Object Name: Hub
The new variant:
Revision of the Variant: mountain
Proposed Version Numbers: Hub.Jeevani_1.1

Object Name: Hub
The new variant:
Revision of the Variant: mountain
Proposed Version Numbers: Hub.Jeevani_1.1
Hub.Jeevani_1
Hub.Jeevani_1.1

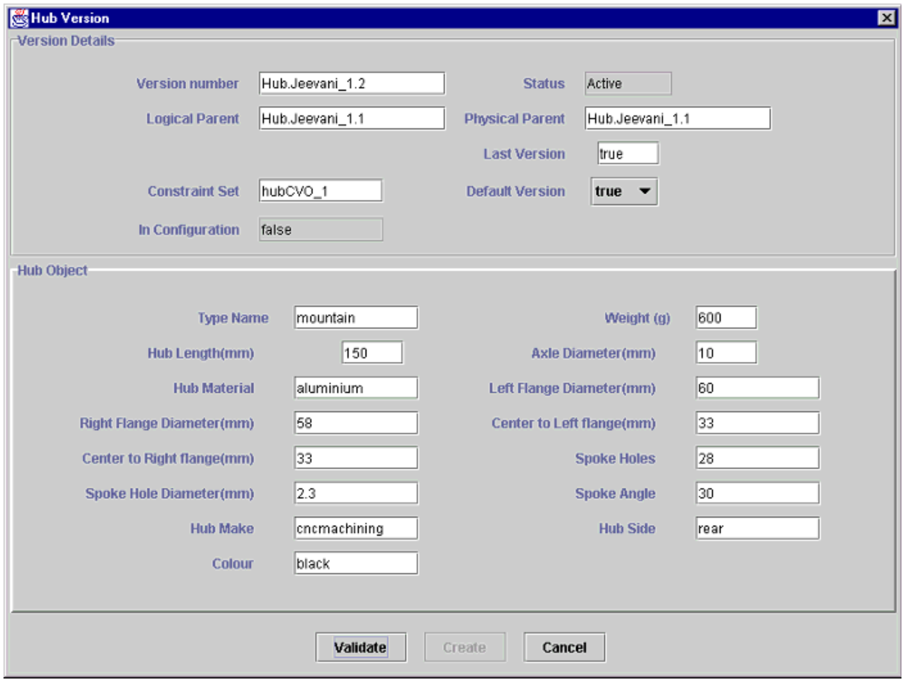
(a) *(b)*

will be prompted to select the artefact name first, that is, hub in this case and the parent version number, see Figures 10a and 10b. The eligible parent version numbers are selected by the prototype on selection of the artefact name, as described previously. In this situation *Hub.Jeevani_1* and *Hub.Jeevani_1.1* are both eligible to be parent versions, as depicted in the Figure 10b.

On selection of these parameters, with *Hub.Jeevani_1.1* as the parent version, the prototype will bring up the corresponding GUI for hub version creation/derivation. A new object version is derived by changing the attribute values in the parent object version. In deriving the new version *Hub.Jeevani_1.2*, the values relevant to weight, hub length, left and right flange diameters and spoke angle were changed; see Figure 11a. Data validation is invoked by explicit user request by pressing the validate button; see Figure 11a. The validation will be performed against the default CVO for hub artefact *hubCVO_1*, which is already associated with the new hub version by the prototype; see the value of the *Constraint Set* attribute in Figure 11a. The data validation results for *Hub.Jeevani_1.2* are represented in Figure 11b.

As depicted in the Figure 11b, the designer is provided with informative messages showing constraint violations. The constraints such as *spokeAngle* can be defined to return either the calculated value or a true/false value in a violation. The version *Hub.Jeevani_1.2* with the corrected values is shown in Figure 12a, with its data validation results in Figure 12b. The validation process can be repeated until a successful validation situation is obtained. In iterative validation cycles, the number of constraints used for validation can reduce further, except the constraints that always have to be validated, as the

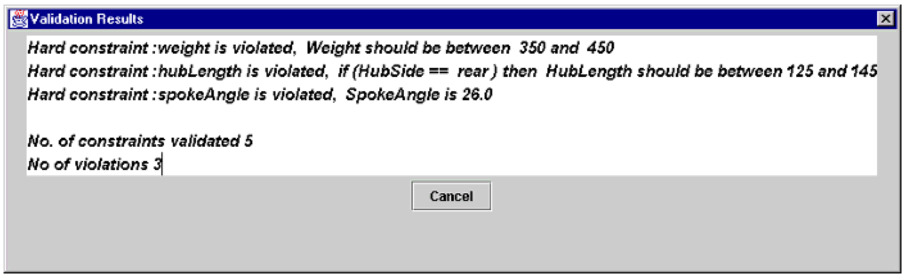
Figure 11a. The version Hub.Jeevani_1.2 before data validation



Version number	Hub.Jeevani_1.2	Status	Active
Logical Parent	Hub.Jeevani_1.1	Physical Parent	Hub.Jeevani_1.1
		Last Version	true
Constraint Set	hubCVO_1	Default Version	true
In Configuration	false		

Type Name	mountain	Weight (g)	600
Hub Length(mm)	150	Axle Diameter(mm)	10
Hub Material	aluminium	Left Flange Diameter(mm)	60
Right Flange Diameter(mm)	58	Center to Left flange(mm)	33
Center to Right flange(mm)	33	Spoke Holes	28
Spoke Hole Diameter(mm)	2.3	Spoke Angle	30
Hub Make	cncmachining	Hub Side	rear
Colour	black		

Figure 11b. The data validation results for Hub.Jeevani_1.2



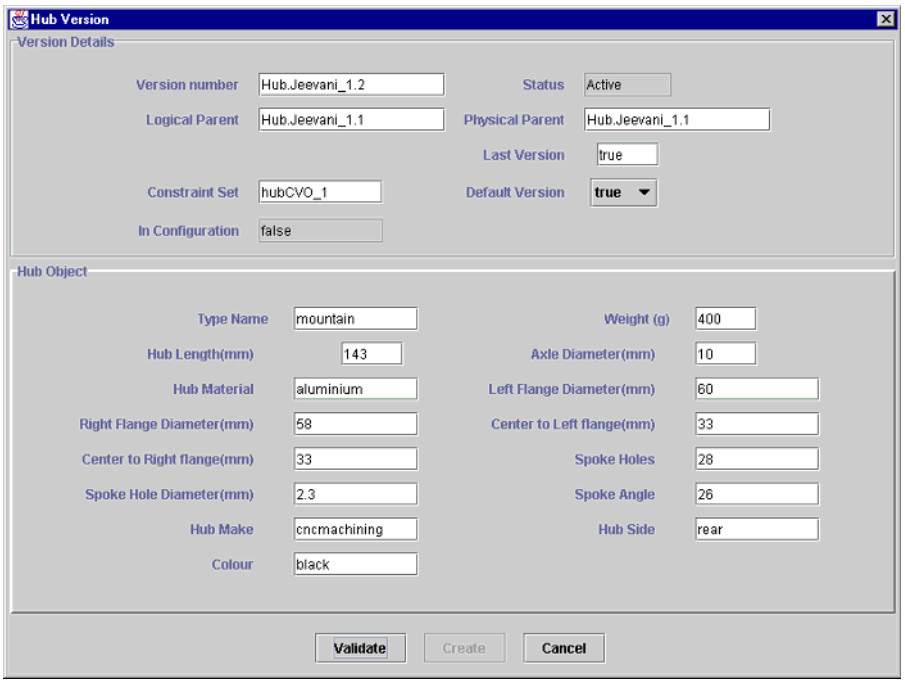
```
Hard constraint :weight is violated,  Weight should be between  350 and  450
Hard constraint :hubLength is violated,  if (HubSide ==  rear ) then  HubLength should be between 125 and 145
Hard constraint :spokeAngle is violated,  SpokeAngle is 26.0

No. of constraints validated 5
No of violations 3
```

constraints relevant to successful validation can be omitted. The new version cannot be made persistent in the database until its design values are successfully validated, although there may still be soft constraint violations. Consequently, the "create" button, see Figures 11a and 12a, that adds the new version to the database will be enabled only after a successful validation.

The outcome of these results demonstrates that the system retrieves the default CVO dynamically to invoke the rules relevant to its design constraints. This frees the designer from having to manually select and attach the currently-applicable constraints to the new object version for validation.

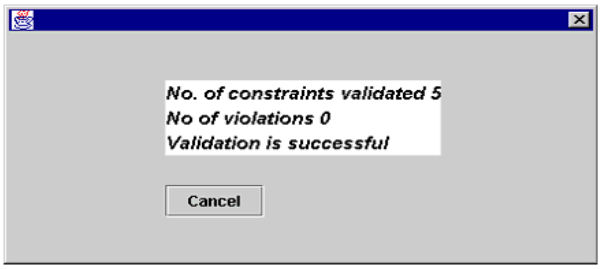
Figure 12a. The version Hub.Jeevani_1.2 with the corrected attribute values



Version number	Hub.Jeevani_1.2	Status	Active
Logical Parent	Hub.Jeevani_1.1	Physical Parent	Hub.Jeevani_1.1
		Last Version	true
Constraint Set	hubCVO_1	Default Version	true
In Configuration	false		

Type Name	mountain	Weight (g)	400
Hub Length(mm)	143	Axle Diameter(mm)	10
Hub Material	aluminium	Left Flange Diameter(mm)	60
Right Flange Diameter(mm)	58	Center to Left flange(mm)	33
Center to Right flange(mm)	33	Spoke Holes	28
Spoke Hole Diameter(mm)	2.3	Spoke Angle	26
Hub Make	cncmachining	Hub Side	rear
Colour	black		

Figure 12b. The successful data validation for Hub.Jeevani_1.2



No. of constraints validated 5
No of violations 0
Validation is successful

Capturing of the New/Modified Constraints Relevant to Primitive Artefacts

The designer is provided with a graphical user interface to capture evolving constraints in a form-filling manner. The constraints local to a primitive artefact, are fed/created through the constraint manipulation interface for sub-components, and the designer is able to invoke this user interface through the main menu. To demonstrate how it works at the primitive artefact level, we use the constraints relevant to the same primitive component of the bicycle, the hub artefact. The currently-active default set of constraints (default CVO) for hub is *hubCVO_1*, as explained in the previous section. It is assumed that a new set of constraints with the name *hubCVO_2* is agreed upon for the hub artefact when the design evolves, and affected designers are informed accordingly. The name of the new constraint set, for example, *hubCVO_2*, the parent CVO name, in this case *hubCVO_1*, and the details of the constraints (constraint specification, name of the constraint, constraint category/categories), are assumed to be stated in the project/team leader's message. Creating this new constraint set involves adding new constraints and modifying/omitting existing constraints. Figure 13 illustrates how the constraint:

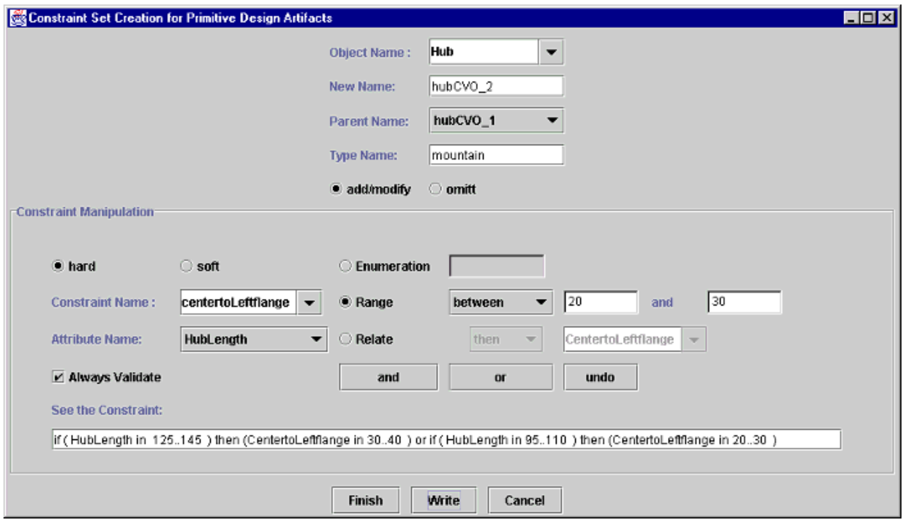
```
centertoLeftflange – (if hubLength between 125 and 145
then centertoLeftflange between 30 mm to 40 mm)  or if
hubLength between 95 and 110 then centertoLeftflange
between 20 mm to 30 mm) – hard
```

belonging to *hubCVO_2* is defined using the user interface. Each constraint specification is displayed in the text box at the bottom, which helps the designer to check and make corrections, if there are any mistakes, before writing them into the system. On completion of each constraint specification, the "write" button is pressed to convert it into executable form. The creation of the new executable CVO will be completed within the system, when the "finish" button is pressed. The process of creating executable CVOs is transparent to the designer.

Actions Taken by the Prototype on Creation of a New CVO

On completion of a new CVO creation using the user interface in Figure 13, the prototype displays the message in Figure 14a to proceed with the existing version validation. If the designer presses the OK button, the integrity validation process is invoked to validate existing hub versions against the new default CVO *hubCVO_2*. The existing versions *Hub.Jeevani_1*, *Hub.Jeevani_1.1* and *Hub.Jeevani_1.2* are all proposed by the prototype for this validation since they are versions created under the parent of the newly created CVO. As depicted in Figure 14b, the designer is able to select some or all of the proposed

Figure 13. The user interface to enter constraint specifications for primitive artefacts


Constraint Set Creation for Primitive Design Artifacts

Object Name : Hub
New Name: hubCVO_2
Parent Name: hubCVO_1
Type Name: mountain
[✓ add/modify] omitt

Constraint Manipulation

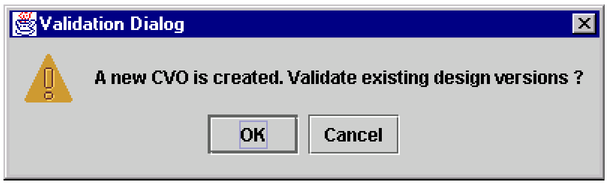
[✓ hard] soft
Enumeration
Constraint Name : centertoLeftflange
[✓ Range] between 20 and 30
Attribute Name: HubLength
Relate then CentertoLeftflange
[✓ Always Validate]
See the Constraint:
if (HubLength in 125..145) then (CentertoLeftflange in 30..40) or if (HubLength in 95..110) then (CentertoLeftflange in 20..30)

versions for validation. In this situation, it is assumed that the designer wants all the proposed versions to be validated. The designer will be notified of the outcome of the validation against the new default CVO *hubCVO_2*, as illustrated in Figure 14c. According to the reported outcome, there are no successful validations from the existing set of design versions. Therefore the prototype proposes the versions which are closest to successful validation, that is, *Hub.Jeevani_1.2*. The designer is able to proceed from there with a new version derivation, selecting one of the closest versions as the parent version, by pressing the 'create version' button in Figure 14c. A new version *Hub.Jeevani_1.3*, which is derived, from *Hub.Jeevani_1.2* and validated against the *hubCVO_2*, will become the new default object version for the hub artefact. If there are successfully validated versions, there is no need to proceed with version derivation, and one of those versions can be set as the default version at the designer's discretion. The test program proves the feasibility of our concept of validating existing versions against newly-introduced constraints, and the advantages thereof. It further shows the relative ease of performing this process from the designer's point of view.

Validation of a Configuration Version on its Creation

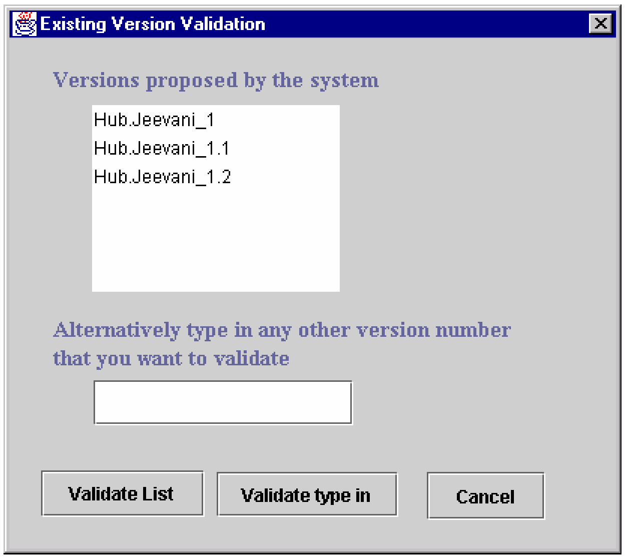
This is demonstrated using the wheel configuration of the bicycle design, which has hub and rim as its components.

Figure 14a. The message that will be displayed on completion of a new CVO creation



A new CVO is created. Validate existing design versions ?

Figure 14b. Proposed hub versions to be validated against the new CVO hubCVO_2

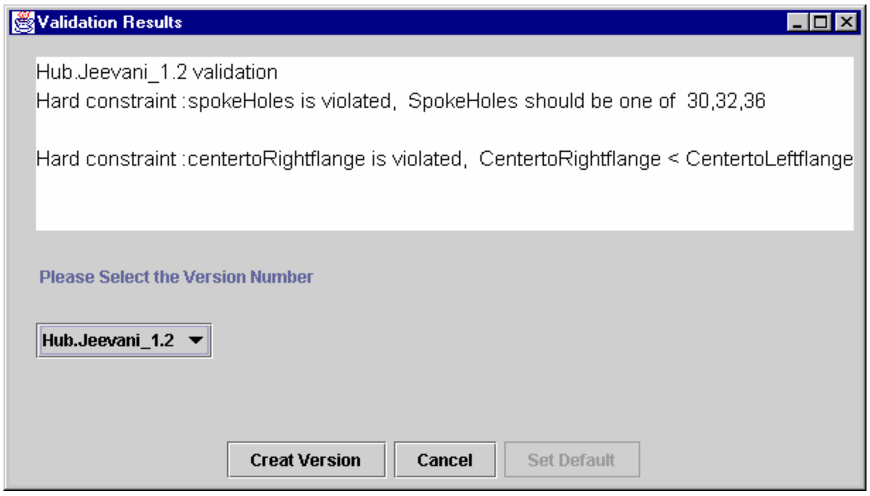


Versions proposed by the system

Hub.Jeevani_1
Hub.Jeevani_1.1
Hub.Jeevani_1.2

Alternatively type in any other version number that you want to validate

Figure 14c. The outcome of the existing version validation



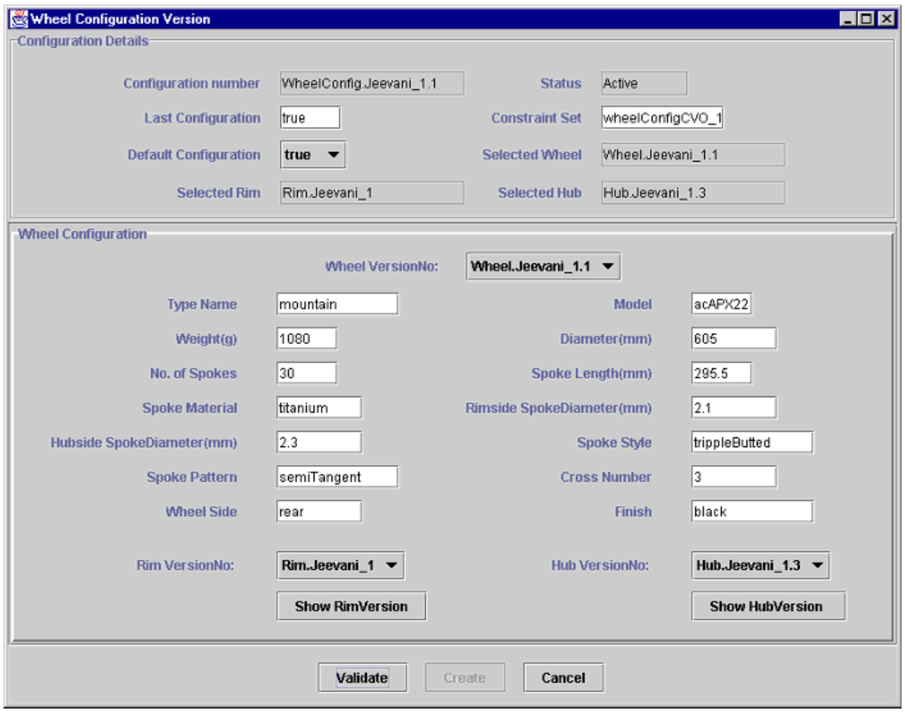
```
Hub.Jeevani_1.2 validation
Hard constraint :spokeHoles is violated,  SpokeHoles should be one of  30,32,36

Hard constraint :centertoRightflange is violated,  CentertoRightflange < CentertoLeftflange
```

Please Select the Version Number

Hub.Jeevani_1.2

Figure 15a. Configuration creation interface for wheel configuration

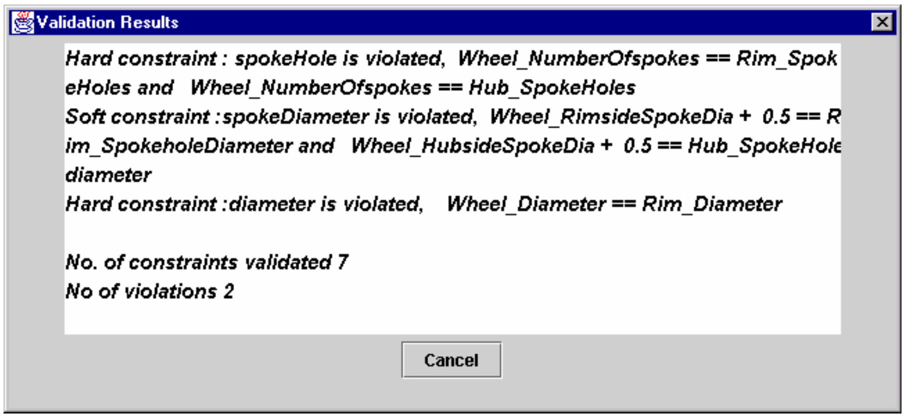


Configuration number	WheelConfig.Jeevani_1.1	Status	Active
Last Configuration	true	Constraint Set	wheelConfigCVO_1
Default Configuration	true	Selected Wheel	Wheel.Jeevani_1.1
Selected Rim	Rim.Jeevani_1	Selected Hub	Hub.Jeevani_1.3

Wheel VersionNo: Wheel.Jeevani_1.1

Type Name	mountain	Model	acAPX22
Weight(g)	1080	Diameter(mm)	605
No. of Spokes	30	Spoke Length(mm)	295.5
Spoke Material	titanium	Rimside SpokeDiameter(mm)	2.1
Hubside SpokeDiameter(mm)	2.3	Spoke Style	trippleButted
Spoke Pattern	semiTangent	Cross Number	3
Wheel Side	rear	Finish	black
Rim VersionNo:	Rim.Jeevani_1	Hub VersionNo:	Hub.Jeevani_1.3

Figure 15b. The validation result for the WheelConfig.Jeevani_1.1 for the selected constituent version combination



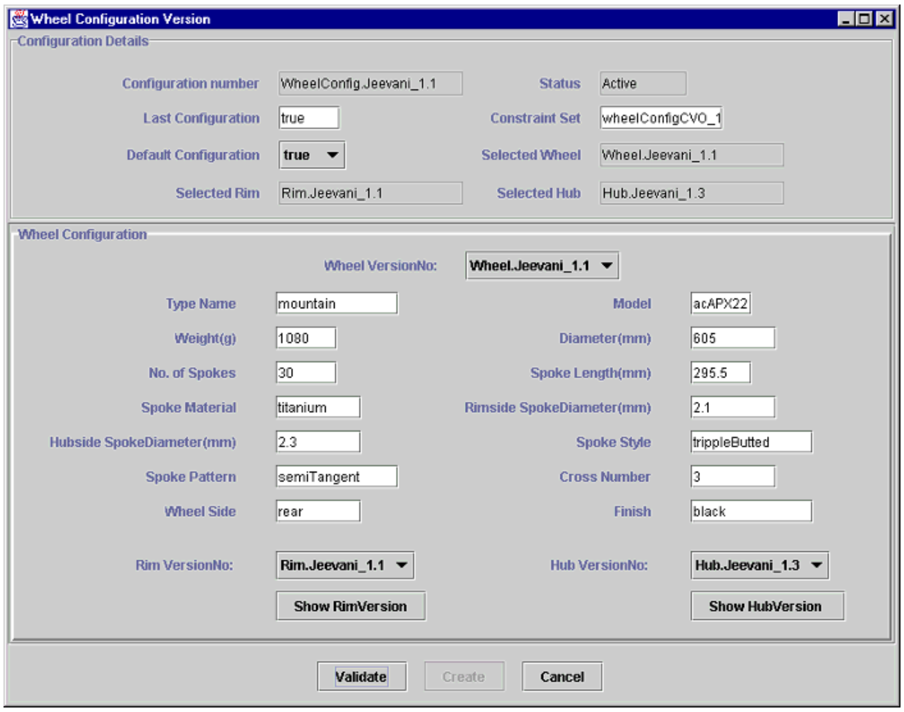
Hard constraint : spokeHole is violated, Wheel_NumberOfspokes == Rim_SpokeHoles and Wheel_NumberOfspokes == Hub_SpokeHoles
Soft constraint :spokeDiameter is violated, Wheel_RimsideSpokeDia + 0.5 == Rim_SpokeholeDiameter and Wheel_HubsideSpokeDia + 0.5 == Hub_SpokeHolediameter
Hard constraint :diameter is violated, Wheel_Diameter == Rim_Diameter

No. of constraints validated 7
No of violations 2

When the designer invokes configuration creation from the main menu he or she will be prompted to select the configuration name first, and in this case it is the wheel configuration. This will bring up the configuration creation interface for the wheel configuration; see Figure 15a. As depicted in Figure 15a, the designer is presented with lists of version/sub-configuration numbers, one list for each of the constituent objects. These version numbers correspond to the active versions, including the default version, that satisfy the set of constraints currently applicable on its artefact, and therefore become eligible to be part of a configuration version. For example, although there are four versions for the hub artefact, the list corresponding to hub artefact contains only *Hub.Jeevani_1.3* as the eligible version. This is due to the fact that it is the only version that adheres to *hubCVO_2*, which contains the currently-applicable constraint set for the hub. The designer is able to select one version/configuration number for each constituent version from the corresponding list to construct a configuration version. As can be seen, the designer does not require any programming knowledge to create a configuration. The version- specific information of the new configuration version is also shown in Figure 15a, with its system-generated configuration ID W*heelConfig.Jeevani_1.1* which illustrates how the configurations are managed as versions.

Data validation for the new configuration is invoked by explicit user request, by pressing the validate button, see Figure 15a, after selecting the constituent version numbers. Validation will be performed against the default CVO *wheelConfigCVO_1* which is already associated with the new configuration version by the system, see the *Constraint Set* attribute in Figure 15a. The validated results are represented in Figure 15b by means of informative messages that enable the designer to repair the violations by selecting a different constituent version combination. For example, the selected version combination first, as shown in Figure 15a, is not consistent according to the validated results reported in Figure 15b, with two hard constraint violations. However, by changing *Rim.Jeevani_1* to *Rim.Jeevani_1.1* as shown in Figure 16a, a consistent wheel configuration, that is, W*heelConfig.Jeevani_1.1*, can be formed, although with soft constraint violations. The validated results for this situation are depicted in Figure 16b. In this way, the validation process can be repeated until a successful validation, that is, a consistent version combination, is obtained. This further demonstrates that the designer is able to experiment with any version combination from the proposed versions at his or her discretion, until a consistent configuration is constructed. Delta validation will not be applicable at the configuration level since each configuration version is created from scratch. As with primitive versions, each new configuration version is made persistent in the database after successful validation, using the "*create*" button in Figure 16a.

Figure 16a. WheelConfig.Jeevani_1.1 after selecting a different constituent version combination

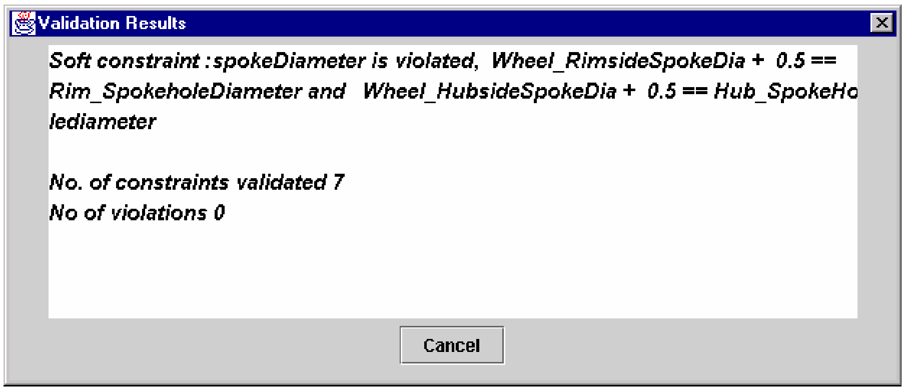


Configuration number	WheelConfig.Jeevani_1.1	Status	Active
Last Configuration	true	Constraint Set	wheelConfigCVO_1
Default Configuration	true	Selected Wheel	Wheel.Jeevani_1.1
Selected Rim	Rim.Jeevani_1.1	Selected Hub	Hub.Jeevani_1.3

Wheel VersionNo: Wheel.Jeevani_1.1

Type Name	mountain	Model	acAPX22
Weight(g)	1080	Diameter(mm)	605
No. of Spokes	30	Spoke Length(mm)	295.5
Spoke Material	titanium	Rimside SpokeDiameter(mm)	2.1
Hubside SpokeDiameter(mm)	2.3	Spoke Style	trippleButted
Spoke Pattern	semiTangent	Cross Number	3
Wheel Side	rear	Finish	black
Rim VersionNo:	Rim.Jeevani_1.1	Hub VersionNo:	Hub.Jeevani_1.3

Figure 16b. A successful validation (although with a soft constraint violation) for WheelConfig.Jeevani_1.1 for the version combination selected in Figure 16a



```
Soft constraint :spokeDiameter is violated,  Wheel_RimsideSpokeDia +  0.5 ==
Rim_SpokeholeDiameter and   Wheel_HubsideSpokeDia +  0.5 == Hub_SpokeHo
lediameter

No. of constraints validated 7
No of violations 0
```

Capturing the New/Modified Constraints Relevant to Configurations

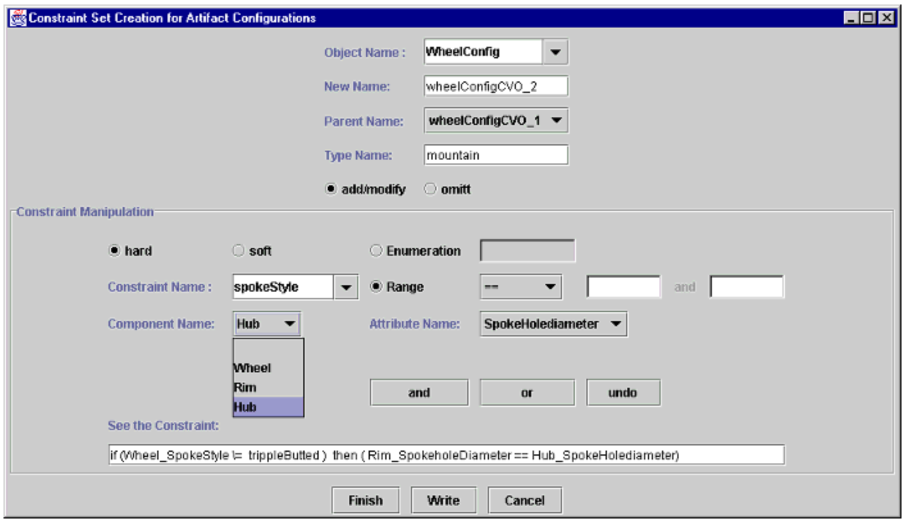
The inter-dependency constraints, or global constraints, in configurations are created through the constraint manipulation interface for configurations. The designer is able to invoke this interface through the main menu. To demonstrate how the user interface works at the configuration level, we use the constraints relevant to the wheel configuration. The currently-active, or default, set of constraints for the wheel configuration is *wheelConfigCVO_1*, as explained in the previous section. It is assumed that a new set of constraints with the name *wheelConfigCVO_2* is agreed upon for the wheel configuration, when the design progresses and is communicated to the appropriate designers.

As explained earlier, the name of the new constraint set, for example, *wheelConfigCVO_2*, the parent CVO name, in this case *wheelConfigCVO_1*, and the details of the constraints, such as constraint specification, name of each constraint, are assumed to be stated in the project/team leader's message. The user interface used to enter constraint specifications relevant to configurations is shown in Figure 17, and the constraints are displayed in a plain language form. However, the inter-dependency constraints, or global constraints in configurations, are based around establishing the dependencies between attribute values coming from the corresponding constituent artefacts. Consequently, the user interface provides facilities to select different constituent artefacts and their attributes, in defining a constraint. For example, Figure 17 illustrates how the constraint:

```
spokestyle    : if Wheel_spokeStyle != triple_butted then
Rim_spokeholeDiameter  =  Hub_spokeholeDiameter
```

belonging to *wheelConfigCVO_2,* is defined using the user interface. However, as explained before, the main intention behind the creation of this user interface is to prove the feasibility of capturing evolving constraints, without requiring any programming skills from designers. Therefore, only minimal facilities were incorporated in this user interface to demonstrate our concept. However, there is scope to improve this interface further by incorporating facilities to capture more complicated constraints than the ones given in here, with more error-checking facilities to assist the designer. The creation of the new executable CVO will be completed within the system in a manner, which is transparent to the designer, when the "*finish*" button is pressed; see Figure 17.

Figure 17. The user interface to enter constraint specifications for configurations


Constraint Set Creation for Artifact Configurations

Object Name : WheelConfig
New Name: wheelConfigCVO_2
Parent Name: wheelConfigCVO_1
Type Name: mountain
[✓ add/modify] omitt

Constraint Manipulation
[✓ hard] soft
Enumeration
Constraint Name : spokeStyle
[✓ Range] == and
Component Name: Hub
Wheel
Rim
Hub
Attribute Name: SpokeHolediameter
See the Constraint:
if (Wheel_SpokeStyle != trippleButted) then (Rim_SpokeholeDiameter == Hub_SpokeHolediameter)

Actions Taken by the Prototype on Creation of a New CVO for a Configuration

On completion of a new CVO for the configuration, the prototype displays the same message as in Figure 14a to get the designer's consent to proceed with the validation of existing configuration versions against the new default CVO *wheelConfigCVO_2*. The existing version W*heelConfig.Jeevani_1.1* is proposed by the prototype (see Figure 18a) as the eligible version for this validation. The designer will be notified on the outcome, as illustrated in Figure 18b, after validation against the new default CVO w*heelConfigCVO_2*. In this situation, the version W*heelConfig.Jeevani_1.1* is also successfully validated against the *wheelConfigCVO_2*. Consequently, the necessity for the creation of a new configuration version can be avoided since the version *WheelConfig.Jeevani_1.1* can be set as the default version.

Easy Retrieval of Constraints Relevant to Primitive/Configuration Versions

The prototype enables the designer to see the constraints in the default CVO by selecting the *default CVO* option or in any other CVO by specifying the

Figure 18a. The eligible configuration versions proposed by the prototype for existing configuration validation

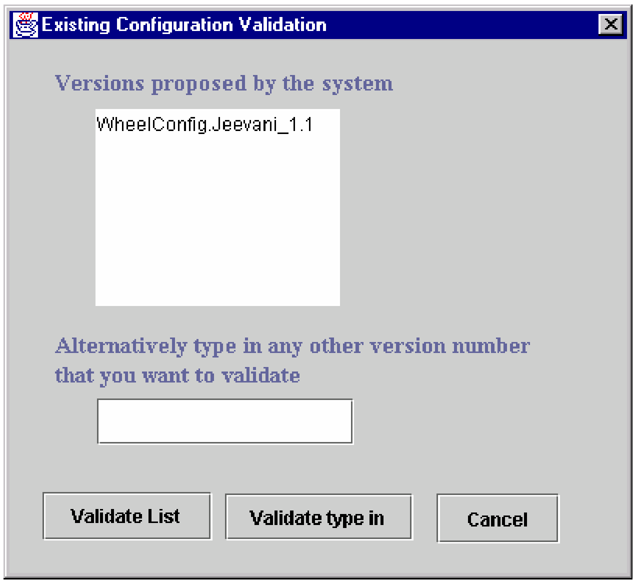
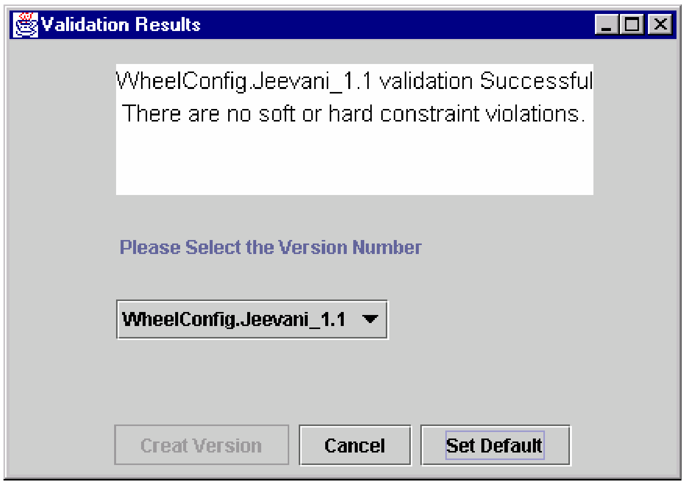

Existing Configuration Validation
Versions proposed by the system
WheelConfig.Jeevani_1.1
Alternatively type in any other version number that you want to validate
Validate List
Validate type in
Cancel

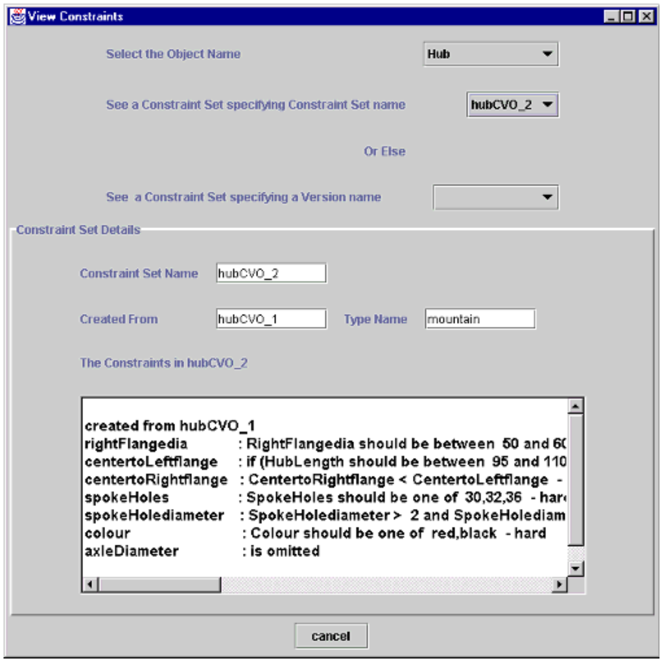
Figure 18b. A successful existing configuration validation



WheelConfig.Jeevani_1.1 validation Successful
There are no soft or hard constraint violations.

CVO name (or the constraint set). The way in which the constraints in *hubCVO_2* are displayed to the designer is depicted in Figure 19a, specifying the CVO name. The designer is also able to retrieve the constraint set applicable to each object version by specifying the required object version name/ID. This is demonstrated in Figure 19b, which displays the CVO pertinent to version *Hub.Jeevani_1.3*. The constraints and the CVO details

Figure 19a. Retrieval of constraints relevant to a CVO



View Constraints

Select the Object Name: Hub

See a Constraint Set specifying Constraint Set name: hubCVO_2

Or Else

See a Constraint Set specifying a Version name:

Constraint Set Details

Constraint Set Name: hubCVO_2

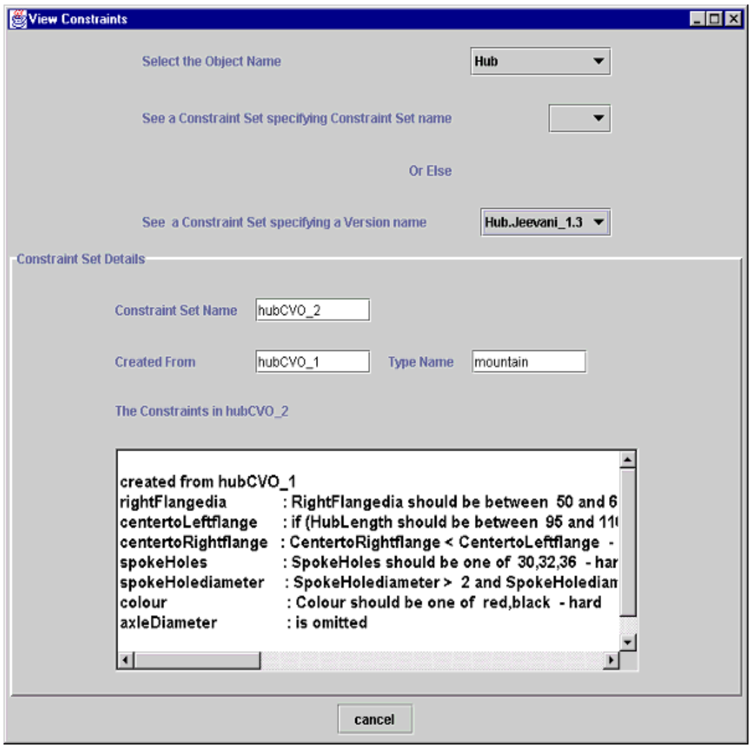
Created From: hubCVO_1 Type Name: mountain

The Constraints in hubCVO_2

```
created from hubCVO_1
rightFlangedia        : RightFlangedia should be between  50 and 6(
centertoLeftflange    : if (HubLength should be between  95 and 110
centertoRightflange   : CentertoRightflange < CentertoLeftflange  -
spokeHoles            : SpokeHoles should be one of  30,32,36  - har
spokeHolediameter     : SpokeHolediameter >  2 and SpokeHolediam
colour                : Colour should be one of  red,black  - hard
axleDiameter          : is omitted
```

cancel

Figure 19b. Retrieval of constraints relevant to a specified version



View Constraints

Select the Object Name: Hub

See a Constraint Set specifying Constraint Set name:

Or Else

See a Constraint Set specifying a Version name: Hub.Jeevani_1.3

Constraint Set Details

Constraint Set Name: hubCVO_2

Created From: hubCVO_1 Type Name: mountain

The Constraints in hubCVO_2

```
created from hubCVO_1
rightFlangedia        : RightFlangedia should be between  50 and 6
centertoLeftflange    : if (HubLength should be between  95 and 11(
centertoRightflange   : CentertoRightflange < CentertoLeftflange  -
spokeHoles            : SpokeHoles should be one of  30,32,36  - har
spokeHolediameter     : SpokeHolediameter >  2 and SpokeHoledian
colour                : Colour should be one of  red,black  - hard
axleDiameter          : is omitted
```

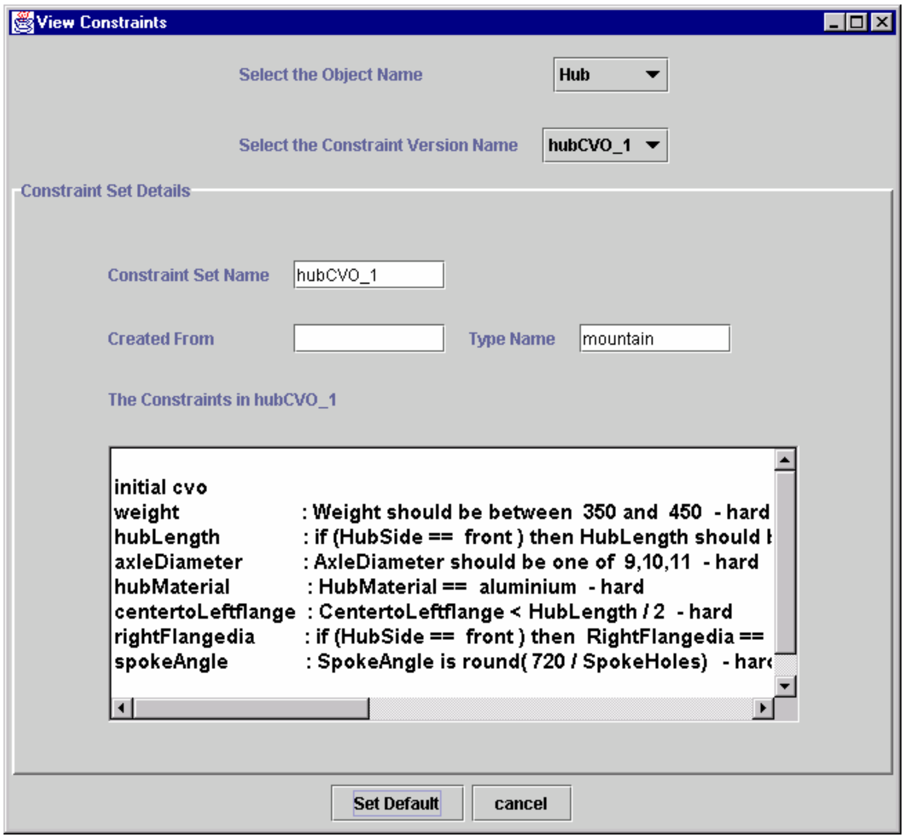
cancel

are displayed in a form close to natural language. However, the prototype displays only the modified or new constraints defined for the CVO. As the parent CVO name is defined in the statement "*created from <CVO name>*" the inherited constraints can easily be examined through accessing the corresponding parent CVO. Listing all the constraints within a CVO otherwise requires navigating up in the inheritance hierarchy, and selecting the relevant constraints by scanning the parent CVOs. Although the test program described here was based around a primitive artefact, the procedure for constraint retrieval is the same for CVOs created for configuration objects. Consequently, it is not illustrated separately here.

Ability to Change to a Previous Stage of Constraint Evolution

Change to a previous stage of constraint evolution can be achieved by setting the current default CVO to a previous CVO, and this should be decided by the project leader. As our prototype does not support an effective communication mechanism, it is assumed that the designers receive a message from the project/group leader with the CVO name to be set as the default. For the test program, we consider that the default hub CVO should be changed from *hubCVO_2* to *hubCVO_1*, and from w*heelConfigCVO_2* to w*heelConfigCVO_1* for the wheel configuration (WheelConfig). On issue of the command to set the default CVO, the designer is provided with a list of CVO names created in the system for that artefact. From this list the designer is able to select one CVO name to be the new default CVO, which in this case is *hubCVO_1* (depicted in Figures 20a and 25b). As the newly-selected CVO name is different to the existing default CVO, the prototype instructs the designer to set a new default object version for that artefact, which adheres to the constraints in the new default CVO. However, at the primitive version level, this step is imposed only if the existing default version does not adhere to the new default CVO. The prototype supports the designer in setting a new default version by selecting and presenting the designer, through the version manager, with the list of version numbers that are eligible to be the default. This is trivial for primitive artefacts such as a hub. As shown in Figure 20d, one of the versions from *Hub.Jeevani_1, Hub.Jeevani_1.1* or *Hub.Jeevani_1.2* would qualify to be the default version of the hub since they all adhere to *hubCVO_1.* Nevertheless, selecting the qualifying version numbers to be the default is non-trivial with configuration versions. The configuration numbers selected for this should correspond to the

Figure 20a. HubCVO_1 is selected as the default CVO



Constraint Set Name: hubCVO_1

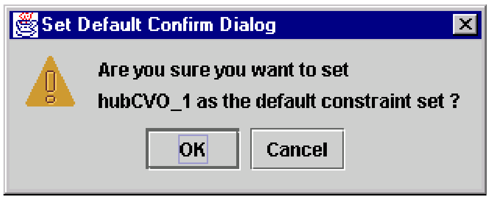
Created From:

Type Name: mountain

The Constraints in hubCVO_1

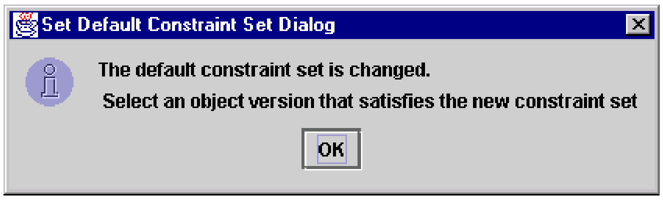
```
initial cvo
weight              : Weight should be between  350 and  450  - hard
hubLength           : if (HubSide ==  front ) then HubLength should b
axleDiameter        : AxleDiameter should be one of  9,10,11  - hard
hubMaterial         : HubMaterial ==  aluminium  - hard
centertoLeftflange  : CentertoLeftflange < HubLength / 2  - hard
rightFlangedia      : if (HubSide ==  front ) then  RightFlangedia ==
spokeAngle          : SpokeAngle is round( 720 / SpokeHoles)  - hard
```

Figure 20b. Confirmation message before changing the default CVO



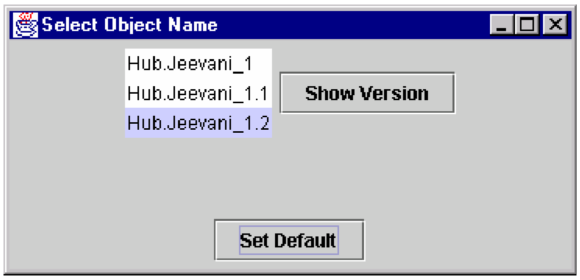
Are you sure you want to set
hubCVO_1 as the default constraint set ?

Figure 20c. The notification message to the designer to set a new default object version



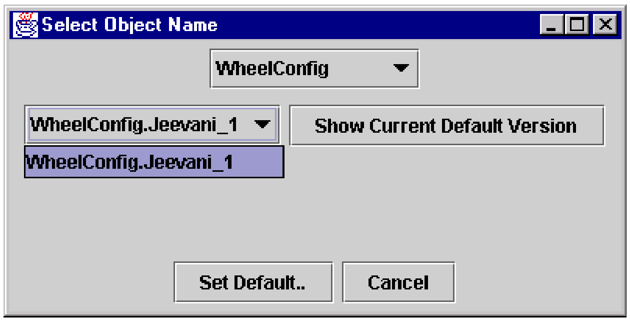
The default constraint set is changed.
Select an object version that satisfies the new constraint set

Figure 20d. The qualifying version numbers to be the default for the hub artefact



configurations that adhere to the current set of inter-dependency constraints applicable, and their constituent object versions should also adhere to their set of currently-imposed constraints. For this reason, although there are two configuration versions that are consistent with *wheelConfigCVO_1,* only *WheelConfig.Jeevani_1* qualifies to be the default as can be seen from Figure 21. On clicking the "*set default*" button (see Figure 20d and Figure 20) after selecting a version from the list, the version manager updates the versioning system to set the new default version. If there are no qualifying configuration versions, a new configuration version will have to be created to satisfy the requirements. In conclusion, this test program shows that there is no practical difficulty in changing back to a previous stage of constraint evolution.

Figure 21. The qualifying version numbers to be the default for the wheel configuration



Implementation

There are two components to the implementation of the prototype software. The first component is the version model; the second component enhances this versioning model with the integrity mechanism. Java2 and the Objectivity database system are being used to implement the prototype, and the representation of CVOs has been achieved using Prolog. Easy capture of constraints to produce the new CVO is allowed through the constraint manipulation interfaces which were developed using Java2. Constraints are represented as Prolog rules and CVOs as Prolog objects. Prolog allows rules to be defined in a declarative style and provides the inheritance mechanism required for CVOs. Significantly, no compilation is required whenever a new CVO is specified. SICStus Prolog (SICStus, 2000) has been chosen for its ability to define objects containing named collections of predicates (rules) and for its inheritance feature. The prototype system automatically converts each constraint defined by a designer into a corresponding Prolog rule. Each Prolog rule contains a head and body. The head defines the name of the constraint, and the body defines the constraint and the message to display following constraint violation. The constraint is invoked using the rule header. Each new Prolog object (or new CVO) is given a name by the designer and is specified as a child of a particular parent for inheritance purposes. For example, the CVO for *hub2* inherits from *hub1* and is defined as follows:

```
hub2 :: {
              super(hub1)  &
         rule_1 &
             rule-2  &
         .
              rule-n
             }.
```

The generation of this code is automatic. The syntax of CVOs for sub-components (local CVOs) and configurations (global CVOs) is the same. The CVO management information for each artefact (e.g., default CVO, parent CVO, name of each CVO, and the constraint names introduced/modified in each CVO) is maintained in a separate Java *properties* text file for easy retrieval. Upon creation of a new CVO, this *properties file* is automatically updated with the new information. A package called *jasper* in SICStus provides a bi-directional interface between Java and Prolog that enables rule execution for data validation. As explained earlier, the constraint validation process provides designers with informative messages when constraints are violated, as is evidenced by Figures 15, 16, and 18. The designer is able to correct the violations based on the given messages.

Easy retrieval of integrity validation constraints is also an important aspect from the designer's point of view, and constraints and CVO details should be displayed in a natural language form in this regard. Implementation of this feature can be achieved using a parser, which converts Prolog rules to natural language. However, for sake of simplicity, we have implemented this feature by writing constraint specifications to a separate text file directly from the constraint manipulation interface as they are entered. On retrieval of a particular CVO, the system seeks this text file for the given CVO name and displays the relevant constraints.

Future Work and Conclusion

Future Work

There are several topics related to our work, which merit further investigation as described in subsequent paragraphs.

Incorporation of a Communication Mechanism

We indicated that an effective communication system is a key factor in a co-operative design environment. This would ensure that designers are supplied with required information such as design changes, precisely and unambiguously. In proposing our model, we assumed a communication protocol where the project/team leader officially notifies the relevant designers with changes to design constraints. However, our prototype does not implement a full communication system. This is mainly due to the focus of our research and its demands on time. However, communication systems have been investigated in the literature to a great extent, in relation to co-operative design environments.

Therefore, enhancing our prototype with an effective communication system would not be difficult. We also suggest that it is worth investigating the use of agents, which would have the ability to check the receipt of the project leader's messages and to bring these messages up on the designer's computer screen.

Improving the Validation Mechanism

Earlier in this chapter we mentioned the delta validation concept, which aims to reduce validation costs. We consider it worthwhile to further investigate the use of the delta validation mechanism with a view to improving its effectiveness.

One difficulty in validating inter-dependency constraints for configuration versions in general is that they may be totally independent of the hierarchical artefact composition graph. For example, in constructing a higher-level version such as a tyre system configuration, one might expect that the constraint dependencies would only exist between the objects that are immediately connected to each other on the composition graph. However, it is evident that inter-dependencies can also exist between higher-level objects and lower-level objects. In a bicycle configuration, for example, one of the constraints could be frame size > 40 cm $\rightarrow$ wheel diameter > 60 cm, where frame size is from the frame object in the structure configuration and wheel diameter is from the wheel object in the tyre system configuration. Since constraints evolve dynamically during the design process, it is not possible to predict in advance the new inter-dependencies that may arise between the attribute values of the component artefacts. One solution is to make all the objects and their attribute values available in advance from the top to the very bottom of the composition hierarchy for each configuration validation. However, this would gradually become too cumbersome when validation moved from lower- to higher-level configurations. For example, data validation at the bicycle configuration level might also require access to all of the objects from the top to the very bottom of the composition hierarchy, including frames, saddles, hubs, and rims. The greater the number of levels in the composition graph, the more complicated this would become.

Configuration Management

Currently, our model considers that all the configuration versions are in a complete state. However, it may be necessary to allow the creation of partial configurations if one or more constituent versions have not been decided. A partial configuration could become active only after all of its constituent object versions have been decided. As a result, the active status flag of a partial configuration should be set to false until its completion. Nevertheless, it is not possible to check some of the global constraints applicable on the configuration version, until all the constituent versions are decided. Consequently, the validation of partial configurations should be deferred until it is completed. This requires the database to maintain both validated configurations and invalidated configurations. Another issue that should also be considered with respect to partial configurations is that there can be situations where the default CVO at its initial creation time is different to the current default CVO at its validation time. In this case the completed configuration could be associated and validated against the current default CVO. It is therefore necessary to explore in detail a strategy for managing partial configurations.

Improving the Graphical User-Interface

The main intention behind the creation of the graphical interfaces depicted in Figures 10 to 21 was to prove the feasibility and the practicality of our concepts for capturing evolving constraints without requiring any programming skills from the designer. For this reason, our interfaces have only been incorporated with the facilities that achieve this goal. For example, they are limited in capturing constraints only up to one conjunction (and) or one disjunction (or). However, there is scope to further improve these graphical interfaces. For example, they could be improved to capture more complicated constraints than the ones given in here. They could also be made more designer-friendly by incorporating further error checking facilities, for example, an indication that an opening bracket is not closed. Incorporating a mechanism to generate the new CVO name rather than expecting the designer to enter the name manually is also desirable.

Making a software environment user-friendly to the designer with graphical user-interfaces involves lot of programming effort on the programmer/software developer's part. In general terms, the more user-friendly the environment, the more programming effort is required. To perform some operations, particularly operations such as version creation and version display, it is necessary to develop separate user interfaces for each of the design artefacts. To achieve this we specifically developed separate classes, which corresponded to the user interfaces for each design artefact. However, incorporation of a new design artefact to the system under this method is a difficult task. This problem could be solved by using a mechanism similar to templates. Such a template approach should enable the programmer to automatically generate artifact-specific user interfaces, by supplying the corresponding parameters.

Conclusion

This chapter has shown how we have modelled evolving design constraints in a computer-aided design prototype built around an object-oriented database. We have shown how integrity validation is handled, and the graphical interface that is presented to the designer users. The combination of our approaches to versioning, configuration management, and constraint management represents a significant contribution to the domain of engineering design management.

References

Baldwin, R. A., & Chung, M. J. (1995). Managing engineering data for complex products. *Research in Engineering Design, 7*(4), 215-231.

Bassiliades, N., & Vlahavas, I. (1994). Modelling constraints with exceptions in object-oriented databases. In P. Loucopoulos (Ed.), *Proceedings of the 13th International Conference on the Entity-Relationship Approach (ER'94)* (pp. 189-204). Berlin: Springer.

Buchmann, A. P., Carrera, R. S., & Vazquez-Galindo, M. A. (1992). Handling constraints and their exceptions: An attached constraint handler for object-oriented CAD databases. In K. R. Dittrich, U. Dayal, & P. Buchmann (Eds.), *On object-oriented database systems* (pp. 65-83). Berlin: Springer-Verlag.

Carnduff, T. W., & Goonetillake, J. S. (2004). Configuration management in evolutionary engineering design using versioning and integrity constraints. *Advances in Engineering Software, 35*, 161-177.

Cellary, W., & Jomier, G. (1990). Consistency of versions in object oriented databases. In *Proceedings of the 16th International Conference on Very Large Databases (VLDB)* (pp. 432-441). San Francisco: Morgan Kaufamann.

Ceri, S., & Fraternali, P. (1997). *Database applications with objects and rules*. New York: Addison-Wesley.

Ceri, S., & Widom, J. (1990). Deriving production rules for constraint maintenance. In *Proceedings of the 16th Very Large Database Conference* (pp. 566-577). San Francisco: Morgan Kaufman.

Ceri, S., Fraternali, P., Paraboschi, S., & Tance, L. (1994). Automatic generation of production rules for integrity maintenance. *ACM Computing Surveys, 19*(3), 367-422.

Chandrasekaran, B. (1989). A framework for design problem solving. *Research in Engineering Design, 1*(2), 75-86.

Choi, E. J., & Kwon, Y. R. (1997, July). An efficient method for version control of a tree data structure. *Software Practice and Experience, 27*(7), 797-811.

Diaz, O., Paton, N., & Gray, P. (1991). Rule management in object oriented databases: A uniform approach. In *Proceedings of the 17th National Conference on Very Large Databases (VLDB 91)* (pp. 317-326). San Francisco: Morgan Kaufman.

Dieter, E. G. (1991). *Engineering design: A materials and processing approach* (2nd ed.). New York: McGraw-Hill.

Dittrich, K. R., & Lorie, R. (1988). Version support for engineering database systems. *IEEE Transactions on Software Engineering, 14*(4), 429-437.

Doucet, A, & Monties, S. (1997). Versions of integrity constraints in multiversion databases. In *Proceedings of the 8th International Conference on Database and Expert System Applications (DEXA'97)* (pp. 252-261). London: Springer.

Doucet, A., Gancarski, S., Jomier, G., & Monties, S. (1996). Integrity constraints in multiversion databases. In *Proceedings of the 14th British National Conference on Databases (BNCOD 14), Advances in Databases* (pp. 184-199). London: Springer.

Estublier, J., Favre, J., & Morat, P. (1998). Towards SCM/PDM integration. In B. Magnusson (Ed.), *Proceedings of the European Conference on Object-Oriented Programming (ECOOP 98) SCM-8 Symposium* (pp. 75-94). London: Springer.

Freeman-Benson, B. N., & Borning, A. (1992). Integrating constraints with an object-oriented language. In *Proceedings of the European Conference* *on Object-Oriented Programming (ECOOP'92)* (pp. 268-286). London: Springer.

Friesen, O, Gauthier-Villars, G., Lefebvre, A., & Vieille, L. (1994). Applications of deductive object-oriented databases using DEL. In R. Ramakrishnan (Ed.), *Applications of Logic Databases* (pp. 1-22). Berlin: Kluwer Academic.

Gehani, N., & Jagadish, H. V. (1991). Ode as an active database: Constraints and triggers. In *Proceedings of the 17th International Conference on Very Large Databases (VLDB'91)* (pp. 327-336). San Francisco: Morgan Kaufmann.

Golendziner, L. G., & Santos, C. S. (1995). Versions and configurations in object-oriented database systems: A uniform treatment. In *Proceedings of the 7th International Conference on Management of Data* (pp. 18-37). NJ: World Scientific.

Goonetillake, J. S., Carnduff, T. W., & Gray, W. A. (2001). Integrity validation for object versions in a co-operative design environment. In W. Shen, Z. Lin, J. Barthes, & M. Kamel (Eds.), *Proceedings of the 6th International Conference on Computer Supported Cooperative Work in Design (CSCWD'01)* (pp. 89-94).

Jagadish, H. V., & Qian, X. (1992). Integrity maintenance in an object-oriented database. In *Proceedings of the 18th International Conference on Very Large Databases (VLDB'92)* (pp. 469-480). San Francisco: Morgan Kaufmann.

Katz, R. H. (1990). Towards a unifying framework for version modeling in engineering databases. *ACM Computing Surveys, 22*(4), 376-408.

Katz, R. H., & Chang, E. (1992). Inheritance issues in computer-aided design databases. In K. R. Dittrich, U. Dayal, & A. P. Buchmann (Eds.), *On object-oriented database systems* (pp. 45-52). Berlin: Springer-Verlag.

Kott, A., & Peasant, J. L. (1995). Representation and management of requirements: The RAPID-WS project. *Concurrent Engineering: Research and Application, 3*, 93-106.

Krishnamurthy, K., & Law, K. (1995). A Data Management Model for Design Change Control. *Concurrent Engineering: Research and Applications, 3*(4), 329-343.

Lin, J., Fox, M. S., & Bilgic, T. (1996a). A requirement ontology for engineering design. In M. Sobolewski & M. Fox (Eds.), *Proceedings of*

Advances in Concurrent Engineering (CE'96) (pp. 343-351). Boca Raton: CRC Press.

Lin, J., Fox, M. S., & Bilgic, T. (1996b). A requirement ontology for engineering design. *Concurrent Engineering Research and Applications, 4*(3), 279-291.

Loomis, M. E. S. (1992, January). Object versioning. *Journal of Object-Oriented Programming*, 40-43.

Otto, K. N., & Antonsson, E. K. (1991). Trade-off strategies in engineering design. *Research in Engineering Design, 3*, 87-103.

Plaice, J., & Wadge,W. (1993). A new approach to version control. *IEEE Transaction on Software Engineering, 19*(3), 268-275.

Prasad, B. (1996). *Concurrent engineering fundamentals. Integrated product development, 2*. NJ: Prentice Hall.

Pugh, S. (1991). *Total design — Integrated methods for successful product engineering*. UK: Addison-Wesley Publishing.

Ram, D. J., Vivekananda, N., Rao, C. S., & Mohan., N. K. (1997). Constraint meta-object: A new object model for distributed collaborative designing. *IEEE Transactions on Systems, Man and Cybernetics, 27*(2), 208-220.

Rochkind, M. J. (1975). The source code control system. *IEEE Transactions on Software Engineering, 1*(4), 364-370.

Rosenman, M. A. (1993). Dynamic decomposition strategies in the conceptual modelling of design objects. *Concurrent Engineering Research and Applications, 1*(1), 31-38.

Sauce, R., Martini, K., & Powell, G. (1992). Object oriented approaches for integrated engineering design systems. *ASCE Journal of Computing in Civil Engineering, 6*(3), 248-265.

SICStus Prolog User's Manual (2000). *Release 3.8.5*. Sweden: Swedish Institute of Computer Science, Kistan.

Thompson, D. R., Tomski, T., Ellacott, S. W., & Kuczora, P. (1995). An expert system for preliminary design of timber roofs. In B. H. Topping & I. Khan (Eds.), *Information technology for civil and structural engineers* (pp. 187-196). Stirling, Scotland: Civil-Comp Press.

Tichy, W. F. (1985). RCS — A system for version control. *Software Practice and Experience, 15*(7), 637-654.

Twari, S., & Franklin, H. (1994). Automated configuration management in concurrent engineering projects. *Concurrent Engineering Research and Applications, 2*(3), 149-161.

Urban, S., Karadimce, P., & Nannapaneni, R. (1992). The implementation and evaluation of integrity maintenance rules in an object-oriented database. In *Proceedings of 8th International Conference on Data Engineering* (pp. 565-572). Washington, DC: IEEE.

Westfechtel, B., & Conradi, R. (1998). Software configuration management and engineering data management: Differences and similarities. In B. Magnusson (Ed.), *Proceedings of the European Conference on Object-Oriented Programming (ECOOP 98) SCM-8 Symposium* (pp. 95-106). Berlin: Springer.

Wilkes, W. (1988). Instance inheritance mechanism for OODBS. In *Object Oriented Database Systems (OODBS 88)* (pp. 274-279).

Yoo, S. B., & Suh, H. W. (1999). Integrity validation of product data in a distributed concurrent engineering environment in concurrent engineering projects. *Concurrent Engineering Research and Applications, 7*(3), 201-213.

Endnote

1 Cross number is a technical word relevant to bicycle design which represents the lacing or the binding pattern of spokes in a wheel.

Chapter IV

Similarity Search for Voxelized CAD Objects

Hans-Peter Kriegel, University of Munich, Germany
Peer Kröger, University of Munich, Germany
Martin Pfeifle, University of Munich, Germany
Stefan Brecheisen, University of Munich, Germany
Marco Pötke, software design & management AG, Germany
Matthias Schubert, University of Munich, Germany
Thomas Seidl, RWTH Aachen, Germany

Abstract

Similarity search in database systems is becoming an increasingly important task in modern application domains such as multimedia, molecular biology, medical imaging, and many others. Especially for CAD (Computer-Aided Design), suitable similarity models and a clear representation of the results can help to reduce the cost of developing and producing new parts by maximizing the reuse of existing parts. In this chapter, we present different similarity models for voxelized CAD data based on space partitioning and data partitioning. Based on these similarity models, we introduce an *industrial prototype, called BOSS, which helps the user to get an overview over a set of CAD objects. BOSS allows the user to easily browse large data collections by graphically displaying the results of a hierarchical clustering algorithm. This representation is well suited for the evaluation of similarity models and to aid an industrial user searching for similar parts.*

Introduction

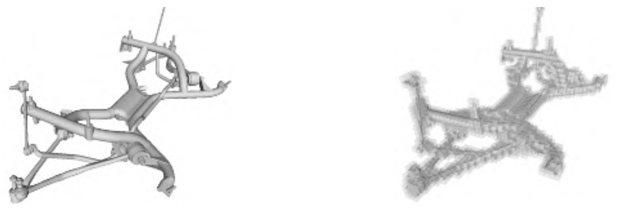
In the last ten years, an increasing number of database applications have emerged for which efficient and effective support for similarity search is substantial. The importance of similarity search grows in application areas such as multimedia, medical imaging, molecular biology, computer-aided engineering, marketing and purchasing assistance, and so forth. Particularly, the task of finding similar shapes in 2D and 3D becomes more and more important. Examples for new applications that require the retrieval of similar three-dimensional objects include databases for molecular biology, medical imaging, and virtual engineering.

Especially in the area of modern engineering, the development, design, manufacturing, and maintenance of products is a very expensive and complex task. Shorter product cycles and a greater diversity of models are becoming decisive competitive factors in the hard-fought automobile and plane market. To cope with this rapidly growing amount of data, effective and efficient similarity models are required for two- and three-dimensional CAD applications.

Accurate representations of CAD surfaces are typically implemented by parametric bi-cubic surfaces, including Hermite, Bézier, and B-spline patches. For many operations, such as graphical display or the efficient computation of surface intersections, these parametric representations are too complex (Möller & Haines, 1999). As a solution, approximative polygon (e.g., triangle) meshes can be derived from the accurate surface representation. These triangle meshes allow for an efficient and interactive display of complex objects, for instance, by means of VRML-encoded files, and serve as an ideal input for the computation of spatial interference.

By means of a uniform three-dimensional voxel grid covering the global product space, the geometry of the CAD parts is often converted into a set of voxels (cf. Figure 1). The voxelization of polygon meshes is a major research topic in the field of computer graphics and CAD. Voxelization techniques and applica-

Figure 1. Scan conversion on a triangulated surface



(a) Triangulated object *(b) Voxelized object*

tions have been proposed, for instance, for interactive volume visualization (Huang, Yagel, Filippov, & Kurzion, 1998) and haptic rendering (McNeely, Puterbaugh, & Troy, 1999). A basic algorithm for the 3D scan-conversion of polygons into a voxel-based occupancy map has been proposed by Kaufmann (1987). By applying this conversion to the given triangle mesh of a CAD object (cf. Figure 1a), a conservative approximation of the object's surface is produced (cf. Figure 1b). In this chapter, we will present similarity models, suitable for these voxelized spatial objects.

The remainder of the chapter is organized as follows: First, we review the existing spatial similarity models and provide a classification of the techniques into feature-based models and direct geometric models. Then we introduce three space-partitioning similarity models and two data-partitioning models. Next, we describe an industrial prototype which depicts the cluster hierarchy of the CAD objects in a user-friendly way and which can be used to evaluate the quality of the presented similarity models. Finally, the chapter concludes with a short summary.

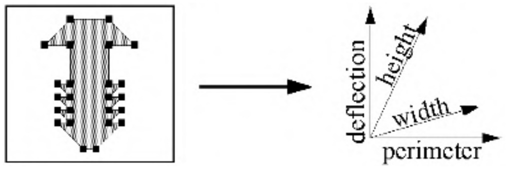
Related Work

In this section, we discuss some of the approaches presented in the literature to establish similarity measures for spatial objects. We provide a classification of the techniques into feature-based models and direct geometric models.

Feature-Based Similarity Search

Feature Transformation. As distance functions form the foundation of similarity search, we need object representations which allow efficient and

Figure 2. Feature transformation



meaningful distance computations. A common approach is to represent an object by a numerical vector, resulting in straightforward distance functions. In this case, a feature transformation extracts distinguishable spatial characteristics which are represented by numerical values and grouped together in a feature vector (cf. Figure 2).

Using a feature transformation, the objects are mapped onto a feature vector in an appropriate multi-dimensional feature space. The similarity of two objects is then defined as the proximity of their feature vectors in the feature space: The closer their feature vectors are located, the more similar two objects are considered. Most applications use the Euclidean metric (L_2) to evaluate the feature distance, but there are several other metrics commonly used, for example, the Manhattan metric (L_1), and the Maximum metric (L_∞).

Feature-Based Similarity Models. Several reasons lead to the wide use of feature-based similarity models: First, the more complex the objects are, the more difficult it may be to find an appropriate similarity distance function. A second reason wherefore feature-based similarity models are quite popular is that they may be easily tuned to fit to specific applications. In general, this task is performed in close cooperation with domain experts who specify appropriate features and adapt them to the specific requirements. Since the existing techniques for query processing are independent from the particular definition of the features, efficient support may be provided without an in-depth insight into the application domain.

Examples where the paradigm of feature-based similarity has been successfully applied to the retrieval of similar spatial objects include structural features of two-dimensional contours (Mehrotra & Gary, 1993), angular profiles of polygons (Badel, Mornon, & Hazout, 1992), rectangular covers of regions (Jagadish, 1991), algebraic moment invariants (Taubin & Cooper, 1991), and 2D section coding (Berchtold, Keim, & Kriegel, 1997). Non-geometric applications include similarity search on time series (Agrawal, Faloutsos, & Swami, 1993), and on color histograms in image databases (Hafner, Sawhney, Equitz, Flickner, & Niblack, 1995; Niblack, Barber, Equitz, Flickner, Glasmann, Petkovic, Yanker, Faloutsos, & Taubin, 1993), among several others.

Agrawal, Faloutsos, and Swami (1993) presented a method for similarity search in a sequence database of one-dimensional data. The sequences are mapped onto points of a low-dimensional feature space using a Discrete Fourier Transform, and then a Point Access Method (PAM) is used for efficient retrieval. This technique was later generalized for sub-sequence matching (Faloutsos, Ranganathan, & Manolopoulos, 1994), and searching in the presence of noise, scaling, and translation (Agrawal, Lin, Sawhney, & Shim, 1995). However, it remains restricted to one-dimensional sequence data.

Mehrotra and Gary (1993) suggested the use of boundary features for the retrieval of shapes. Here, a two-dimensional shape is represented by an ordered set of surface points, and fixed-sized subsets of this representation are extracted as shape features. All of these features are mapped to points in multi-dimensional space which are stored using a PAM. This method is essentially limited to two dimensions.

Jagadish (1991) proposed a technique for the retrieval of similar shapes in two dimensions. He derives an appropriate object description from a rectilinear cover of an object, that is, a cover consisting of axis-parallel rectangles. The rectangles belonging to a single object are sorted by size, and the largest ones serve as a retrieval key for the shape of the object. This method can be generalized to three dimensions by using covers of hyper-rectangles, as we will see later in this chapter.

Histograms as Feature Vectors. Histograms represent a quite general class of feature vectors which have been successfully applied to several applications. For any arbitrary distribution of objects, a histogram represents a more or less fine-grained aggregation of the information. The general idea is to completely partition the space of interest into disjointed regions which are called cells, and to map every object onto a single bin or to distribute an object among a set of bins of the corresponding histogram. Then a histogram can be transformed directly into a feature vector by mapping each bin of the histogram onto one dimension (attribute) of the feature vector. The histogram approach applies to geometric spaces as well as to non-geometric spaces.

A popular example for the use of histograms to define the similarity of complex objects is the color histogram approach which is a core component of the QBIC system (Niblack et al., 1993). Among other techniques, color histograms were

Figure 3. Section coding of 2D regions: (a) original object, (b) corresponding histogram, and (c) corresponding feature vector

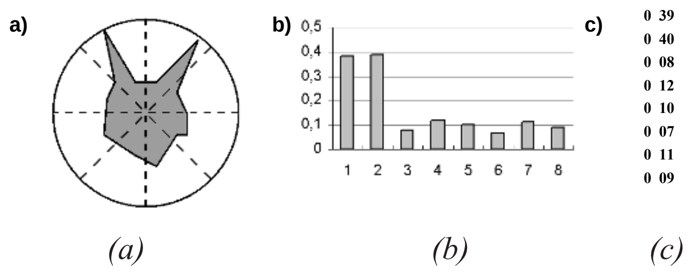
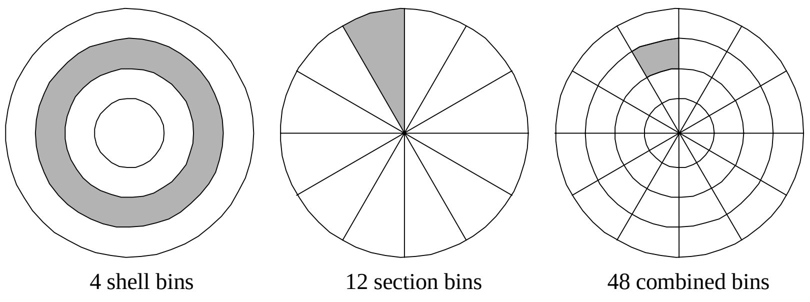


Figure 4. Shells and sections as basic models for shape histograms (a single bin is marked in each of the 2D examples)



used to encode the percentage of colors in an image (Hafner et al., 1995). Our second example is taken from a spatial database application: The 2D section coding approach (Berchtold, Keim, & Kriegel, 1997) represents a particular histogram technique that was used in the S3 system (Berchtold & Kriegel, 1997) for the retrieval of similar mechanical parts. For each object, the circumscribing circle is decomposed into a fixed number of sectors around the center point. For each sector, the fraction of the area is determined that is overlapped by the object. Altogether, the resulting feature vector is a histogram, whose bins represent the corresponding two-dimensional sectors. Figure 3 illustrates the technique by an example with eight sectors.

Ankerst, Kastenmüller, Kriegel, and Seidl (1998) investigated the retrieval of similar three-dimensional objects from a biomolecular database. The introduced models are based on shape histograms, where three different approaches were used for space partitioning: shell bins, section bins and combined bins (cf. Figure 4). Unfortunately, these models are not inherently suitable for voxelized data which are axis-parallel.

Geometry-Based Similarity Search

A class of models that are to be distinguished from the feature-based techniques are the similarity models that are defined by directly using the geometry. Two objects are considered similar if they minimize a distance criterion that is purely defined by the geometry of the objects. Examples include the similarity retrieval of mechanical parts, the difference volume approach, and the approximation-based similarity model for three-dimensional surface segments:

Rotational Symmetric Mechanical Parts. Schneider, Kriegel, Seeger, and Heep (1989) presented a method to retrieve similar mechanical parts from a database. The similarity criterion is defined in terms of tolerance areas which are specified around the query object. All objects that fit into the tolerance area count for being similar. Although the parts are three-dimensional, only their two-dimensional contour is taken into account for the retrieval technique.

Difference Volume Approach. The difference volume or error volume of spatial objects is a promising approach which has been already successfully applied to medical images (e.g., Higgins, 1990; Vincent, 1991). Furthermore, extensions such as the combination with methods from mathematical morphology have been investigated on a tumor database (Korn, Sidiropoulos, Faloutsos, Siegel, & Protopapas, 1996). However, they considered only two-dimensional images. A competing approach is based on a new geometric index structure as suggested by Keim (1999). The basic idea of this solution is to use the concept of hierarchical approximations of the three-dimensional objects to speed up the search process.

Approximation-Based Similarity of Three-Dimensional Surface Segments. The retrieval of similar three-dimensional surface segments is a task that supports the docking search for proteins in biomolecular databases. Following the approximation-based model, the similarity of surface segments is measured by their mutual approximation error with respect to a given multi-parametric surface function which serves as the underlying approximation model. To state it in a simple way, two segments are the more similar, the better they fit to the approximation of the partner segment (Kriegel, Schmidt, & Seidl, 1997).

Table 1. Classification of complex similarity models

Class	**Definition of Similarity**	**Examples**
feature-based similarity	similarity is proximity in the feature space	• rectangular cover of regions • algebraic moment invariants • 2D contour features • angular profiles of polygons • section coding • time series • color histograms • 3D shape histograms
geometric similarity	similarity is directly defined by geometry	• symmetric mechanical parts • difference volume • 3D surface segments

Summary

Table 1 summarizes our classification of similarity models into feature-based approaches and direct geometry-based proposals. The list of examples is by no means complete, but provides an impression of the potentials of both paradigms. In this chapter, effective similarity models for CAD objects were introduced, which rely on the feature based histogram approach.

Similarity Models for Voxelized CAD Objects

In this section, we concentrate on similarity models for voxelized CAD data. We start with introducing object similarity functions, emphasizing in-variance properties which are required for effective similarity search in the area of virtual engineering. Afterwards, three different *space partitioning* similarity models for voxelized CAD data are presented, namely the *volume model,* the *solid-angle model* and the *eigenvalue model*. Then we turn our attention to *data partitioning* similarity models. We first discuss the *cover-sequence model* which serves as a starting point for the *vector set model*. In contrast to the other four models, the vector set model uses sets of feature vectors for representing an object instead of single feature vectors.

Object Similarity

The degree of similarity between two objects heavily depends on the chosen distance function. Ideally, a distance measure has the properties of a metric.

Definition 1 (Metric). Let M be an arbitrary data space. A metric is a mapping $dist: M \times M \to IR$ such that for all $x, y, z \in M$ the following statements hold:

- $dist(x, y) = 0 \Leftrightarrow x = y$ (definiteness)
- $dist(x, y) = dist(y, x)$ (symmetry)
- $dist(x, z) \leq dist(x, y) + dist(y, z)$ (triangle inequality)

Based on *metric distance* functions, we can define *metric object similarity*.

Definition 2 (Metric Object Similarity). Let O be the domain of the objects and $F: O \to M$ be a mapping of the objects into a metric data space M. Furthermore, let $dist: M \times M \to IR$ be a metric distance function. Then a metric object similarity function $simdist: O \times O \to IR$ is defined as follows:

$$simdist(o_1, o_2) = dist(F(o_1), F(o_2)).$$

Often, the d-dimensional vector space IR^d is used. By means of a suitable feature transformation distinguishable spatial characteristics are extracted and grouped together in a numerical feature vector (cf. Figure 2). In this important special case, the similarity of two objects can be defined as follows.

Definition 3 (Feature-Based Object Similarity). Let O be the domain of the objects and $F: O \to IR^d$ be a mapping of the objects into the d-dimensional feature space. Furthermore, let $dist: IR^d \times IR^d \to IR$ be a distance function between two d-dimensional feature vectors. Then a feature-based object similarity function $simdist: O \times O \to IR$ is defined as follows:

$$simdist(o_1, o_2) = dist(F(o_1), F(o_2)).$$

There exist a lot of distance functions which are suitable for similarity search. In the literature, often an L_p-distance is used, for example, the Euclidean distance ($p = 2$).

$$d_{euclid}(\vec{x}, \vec{y}) = \|\vec{x} - \vec{y}\|_2 = \sqrt[2]{\sum_{i=1}^{d} (x_i - y_i)^2}.$$

Normalization of CAD Data

For effective similarity search, it is often required to meet invariance properties with respect to a certain class of transformations, that is, applying a transformation from this class to an object should have no influence on the result of the similarity function. This leads to the following definition.

Definition 4 (Invariance). Let O be the domain of the objects and *simdist:* $O \times O \rightarrow IR$ be a metric object similarity function. *simdist* is invariant with respect to a class of transformations C, iff for all objects $o_1, o_2 \in O$ and all transformations $T \in C$ holds:

$$simdist(o_1, o_2) = simdist(T(o_1), o_2) = simdist(o_1, T(o_2)).$$

Invariance can be achieved by applying appropriate transformations to the objects in the database. This is called the normalization of data. Invariance properties relevant for similarity search in CAD databases are *scaling*, *translation*, *rotation* and *reflection* invariances. It depends on the user as well as on the chosen similarity model which invariances have to be considered for a particular application. The desired normalization of the data leads to the following extended similarity definition.

Definition 5 (Extended Metric Object Similarity). Let O be the domain of the objects and F: $O \rightarrow M$ be a mapping of the objects into a metric data space M. Furthermore, let *dist*: $M \times M \rightarrow IR$ be a metric distance function, and let C be the set of all user-dependent combinations of scaling, translation, rotation, and reflection transformations. Then an extended metric object similarity function *simdist*: $O \times O \rightarrow IR$ is defined as follows:

$$simdist(o_1, o_2) = min_{T \in C}\{dist(F(o_1), F(T(o_2)))\}.$$

Invariance is achieved by taking the minimum of the distances between object o_1 and all transformations in C applied to the object o_2.

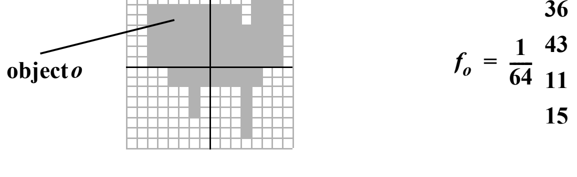
Space Partitioning Similarity Models

In this section, we discuss three different *space partitioning* similarity models suitable for voxelized CAD data, namely the *volume model*, the *solid-angle model* and the *eigenvalue model* (Kriegel, Kröger, Mashael, Pfeifle, Pötke, & Seidl, 2003). Each of the models is based on shape histograms.

Shape Histograms for Voxelized CAD Data

Shape histograms are based on a complete partitioning of the data space into disjointed cells which correspond to the bins of the histograms. We divide the data space into axis parallel, equi-sized partitions (cf. Figure 5). This kind of space partitioning is especially suitable for voxelized data, as cells and voxels are of the same shape, that is, cells can be regarded as coarse voxels. The data space is partitioned in each dimension into p grid cells. Thus, our histogram will consist of $k \cdot p^3$ bins where $k \in IN$ depends on the model specifying the kind and number of features extracted from each cell. For a given object o, let $V^o = \{v \in V_i^o \mid 1 \le i \le p^3\}$ be the set of voxels that represent o where V_i^o are the voxels covered by o in cell i. $\bar{V}^o \subseteq V^o$ denotes the set of voxels at the surface of the objects and $\dot{V}^o \subseteq V^o$ denotes the set of the voxels inside the object, such that $\bar{V}^o \cup \dot{V}^o = V^o$ and $\bar{V}^o \cap \dot{V}^o = \varnothing$ holds. Let r be the number of voxels of the data space in each dimension. In order to ensure a unique assignment of the voxels to a grid cell, we assume that $\frac{r}{p} \in IN$.

Figure 5. 2D space partitioning with 4 cells (the feature vector generated by the volume model is depicted on the right hand side)



After partitioning the data space, we have to determine the spatial features of the objects for each grid cell depending on the chosen model. By scaling the number of partitions, the number of dimensions of the feature vector can be regulated (cf. Figure 5). Obviously, for a growing number of partitions, smaller differences between the objects become decisive. Let f_o be the computed feature vector of an object o. The i-th value of the feature vector of the object o is denoted by $f_o^{(i)}$.

The Volume Model

A simple and established approach to compare two objects is based on the number of the object voxels V_i^o in each cell i of the partitioning. In the following, this model is referred to as the *volume model*. Each cell represents one dimension in the feature vector of the object. The i-th dimension ($1 \leq i \leq p^3$) of the feature vector of object o can be computed by the normalized number of voxels of o lying in cell i. Formally,

$$f_o^{(i)} = \frac{|V_i^o|}{K}, \text{ where } K = \left(\frac{r}{p}\right)^3.$$

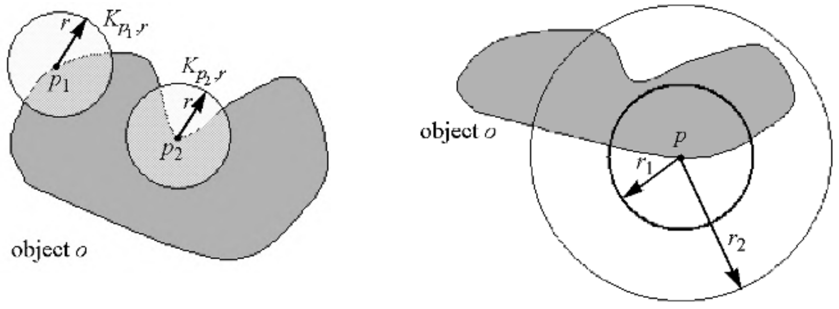
Figure 5 illustrates the volume model for the two-dimensional case.

The Solid-Angle Model

The *solid-angle* method (Connolly, 1986) measures the concavity and the convexity of geometric surfaces. It is, therefore, a good candidate for adequately modelling geometric shapes and has been used in different approaches to spatial similarity modelling. In the following, a model is described that combines the solid-angle approach with our axis-parallel partitioning.

Let $K_{c,r}$ be a set of voxels that describes a three-dimensional voxelized sphere with central voxel c and radius r. For each surface-voxel $\bar{v}$ of an object o the so called solid-angle value is computed as follows: The voxels of o which are inside $K_{\bar{v},r}$ are counted and divided by the size of $K_{\bar{v},r}$, i.e. the number of voxels in $K_{\bar{v},r}$. The resulting measure is called the solid-angle value $Sa(\bar{v}, r)$. Formally,

Figure 6. The solid-angle model: (a) different shapes at different surface points and (b) effect of the radius



(a) *(b)*

$$Sa(\bar{v}, r) = \frac{K_{\bar{v},r} \cap V^o}{\left|K_{\bar{v},r}\right|}.$$

where $K_{\bar{v},r} \cap V^o = \{w \in K_{\bar{v},r} \mid \exists v \in V^o : w.x = v.x \wedge w.y = v.y \wedge w.z = v.z\}$.

A small solid-angle value $Sa(\bar{v}, r)$ indicates that an object is convex at voxel $\bar{v}$ (cf. point p_1 in Figure 6a). Otherwise, a high value of $Sa(\bar{v}, r)$ denotes a concave shape of an object at voxel $\bar{v}$ (cf. point p_2 in Figure 6a). The choice of the radius of the measurement sphere is a crucial parameter. A particular radius could approximate a given object very well (cf. radius r_1 in Figure 6b), whereas another radius might be inept (cf. radius r_2 in Figure 6b).

The solid-angle values of the cells are transferred into the according histogram bins as described in the following. We distinguish between three different types of cells:

- Cell i contains surface-voxels of object o, in other words $\bar{V}_i^o \neq \emptyset$. The mean of all Sa-values of the surface-voxels is computed as the feature value of this cell:

$$f_o^{(i)} = \frac{1}{m}\sum_{j=1}^{m} Sa(\bar{v}_{i_j}, r), \text{ where } \bar{V}_i^o = \{\bar{v}_{i_1}, ..., \bar{v}_{i_m}\}.$$

- Cell i contains only inside-voxels of object o, in other words $\bar{V}_i^o \neq \varnothing$ and $V_i^o \neq \varnothing$. The feature value of this cell is set to 1 (i.e., $f_o^{(i)} = 0$).
- Cell i contains no voxels of object o, in other words $V_i^o \neq \varnothing$. The value of the according bin of the histogram is 0 (i.e., $f_o^{(i)} = 0$).

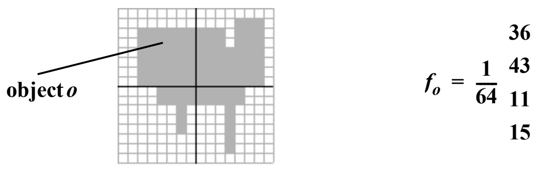
The Eigenvalue Model

In the following, we introduce an approach to extract local features which is based on eigenvalues. The set of voxels of an object can be considered as a set of points in the three-dimensional data space following a particular scattering. The *eigenvalue model* uses this scattering of the voxel sets to distinguish the objects by computing the minimum-bounding ellipsoid of the voxel set in each cell of the partitioning independently (cf. Figure 7).

A minimum-bounding ellipsoid in the three-dimensional space can be described by three vectors. In order to compute these vectors, we consider each voxel v of the object o as a Euclidian vector $\vec{v}^o = (x, y, z)$ in the data space and apply principal axis transformation. To determine the principal axis of the vectors in cell i, we first compute their centroid .

$$\bar{C}^o = \begin{pmatrix} x_C \\ y_C \\ z_C \end{pmatrix} = \frac{1}{|V_i^o|} \sum_{j=1}^{|V_i^o|} \begin{pmatrix} x_j \\ y_j \\ z_j \end{pmatrix}.$$

Figure 7. A 2D example for the eigenvalue model



After that, for each vector $\vec{v}^o$ in cell i, the following translation is carried out:

$$\vec{v}^o \mapsto \vec{v}^o - \vec{C}^o.$$

Based on these transformed vectors $\vec{v}^o$, the co-variance matrix Cov_i^o for each cell i can be computed as follows:

$$Cov_i^o = \frac{1}{|V_i^o| - 1} \sum_{j=1}^{|V_i^o|} \begin{pmatrix} x_j^2 & x_j y_j & x_j z_j \\ x_j y_j & y_j^2 & y_j z_j \\ x_j z_j & y_j z_j & z_j^2 \end{pmatrix}.$$

The three eigenvectors $\vec{e}_i^{\,j}$ (j = 1, 2, 3) of the matrix Cov_i^o correspond to the vectors spanning the minimum-bounding ellipsoid of the voxel set V_i^o. The eigenvalues λ_i^j represent the scaling factors for the eigenvectors (cf. Figure 8). Both eigenvalues and eigenvectors are determined by the following equation:

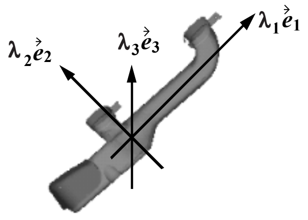
$$Cov_i^o \cdot \vec{e}_i^{\,j} = \lambda_i^j \cdot \vec{e}_i^{\,j}.$$

The interesting values that are inserted in the bins of the histogram are the eigenvalues which describe the scattering along the principal axis of the voxel set. These three values can be computed using the characteristic polynomial:

$$\det(Cov_i^o - \lambda_i^j\, Id) = 0 \text{ for } j = 1, 2, 3.$$

Using this equation we obtain three eigenvalues which are sorted in descending order in the vector $\vec{\lambda}_i$. The highest value represents the variance along the first

Figure 8. The eigenvalue model with the principal axis of a sample object



principal axis, the second value represents the variance along the second principal axis, and the third value represents the variance along the third principal axis.

For each cell *i* of the partitioning the vector $\vec{\lambda}_i$ of the three eigenvalues is computed as just described. It is registered in the according bins of the histogram:

$$f_o^{(i)} = \vec{\lambda}_i = \begin{pmatrix} \lambda_i^1 \\ \lambda_i^2 \\ \lambda_i^3 \end{pmatrix}.$$

Note that for p^3 cells we obtain a feature vector of $3p^3$ dimensions.

Data Partitioning Similarity Models

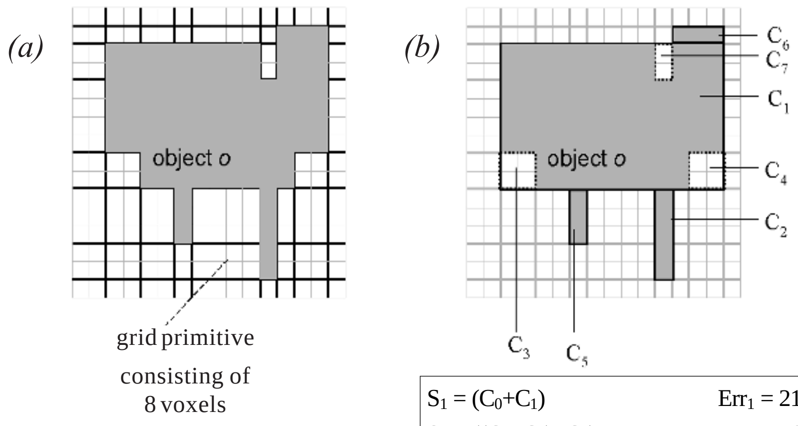
In contrast to the last section where we discussed space partitioning similarity models, we turn our attention to data partitioning similarity models in this section. We introduce two different models, namely the *cover sequence model* (Jagadish, 1991) and the *vector set model* (Kriegel, Brecheisen, Kröger, Pfeifle, & Schubert, 2003). The cover sequence model still uses feature vectors for representing objects, whereas the vector set model uses sets of feature vectors for modelling a three-dimensional voxelized CAD object.

The Cover Sequence Model

The three models described in the foregoing sections are based on a complete partitioning of the data space into disjointed cells. In this section, a model by Jagadish (1991) is adapted to three-dimensional voxelized data which is not restricted to this rigid space partitioning but rather uses a more flexible object-oriented partitioning approach. This model is in the following referred to as *cover sequence model*.

General Idea. As depicted in Figure 9a, each edge of an object can be extended infinitely in either direction, to obtain a grid of lines. Each rectangle in this grid is called grid primitive, and is located either entirely inside the object,

Figure 9. The cover sequence model: (a) grid primitives and (b) cover sequence



$S_1 = (C_0+C_1)$	$Err_1 = 21$
$S_2 = ((C_0+C_1)+C_2)$	$Err_2 = 16$
$S_3 = (((C_0+C_1)+C_2)\ \ C_3)$	$Err_3 = 12$
...	

or entirely out-side of the object. Furthermore, any pair of adjacent grid primitives must also form a hyperrectangle in the data space. The basic idea of this model is to find large clusters of grid primitives, called *covers*, which approximate the object in a best possible way. These covers are organized in a *cover sequence* which provides a sequential description of the object.

Let o be the object being approximated. The quality of a cover sequence S_k of some length $k \in IN$ is measured by the symmetric volume difference Err_k between the object o and the sequence S_k (cf. Figure 9b). Formally, let the covers be drawn from the set C of all possible rectangular covers. Then each unit i of the cover sequence comprises a pair ($C_i \in C, \sigma_i \in \{+, -\}$), where "+" represents set union and "–" represents set difference.

The sequence after k units is:

$$S_k = (((C_0\ \sigma_1\ C_1)\ \sigma_2\ C_2\ )\ ...\ \sigma_k\ C_k),$$

where C_0 is an initial empty cover at the zero point. The symmetric volume difference after k units is:

$$Err_k = |\ o \text{ XOR } S_k|.$$

Note that there exists some natural number N such that $S_k = o$ and $Err_k = 0$ for all k ≥ N. At this point an exact description of the object o has been obtained.

If an object o can be described by a sequence S_j with $j < k$ covers and $Err_j = 0$, we assign $((S_j\, \sigma_{j+1}\, C_0) \ldots \sigma_k\, C_0)$ to S_k. These dummy covers C_0 do not distort our similarity notion, but guarantee that all feature vectors are of the same dimensionality. Thus common spatial index structures (Berchtold, Böhm, Jagadish, Kriegel, & Sander, 2000; Berchtold, Keim, & Kriegel, 1996; Lin, Jagadish, & Faloutsos, 1994) can be used in order to accelerate similarity queries.

Jagadish and Bruckstein (1992) suggest two algorithms for the retrieval of a cover sequence S_k: a *branch and bound* algorithm with exponential run-time complexity, and a *greedy* algorithm with polynomial run-time complexity which tries to minimize Err_i in each step $i \leq k$.

Feature Extraction. Jagadish (1991) describes how a cover sequence

$$S_k = (((C_0\ \sigma_1\ C_1)\ \sigma_2\ C_2\ )\ \ldots\ \sigma_k\ C_k)$$

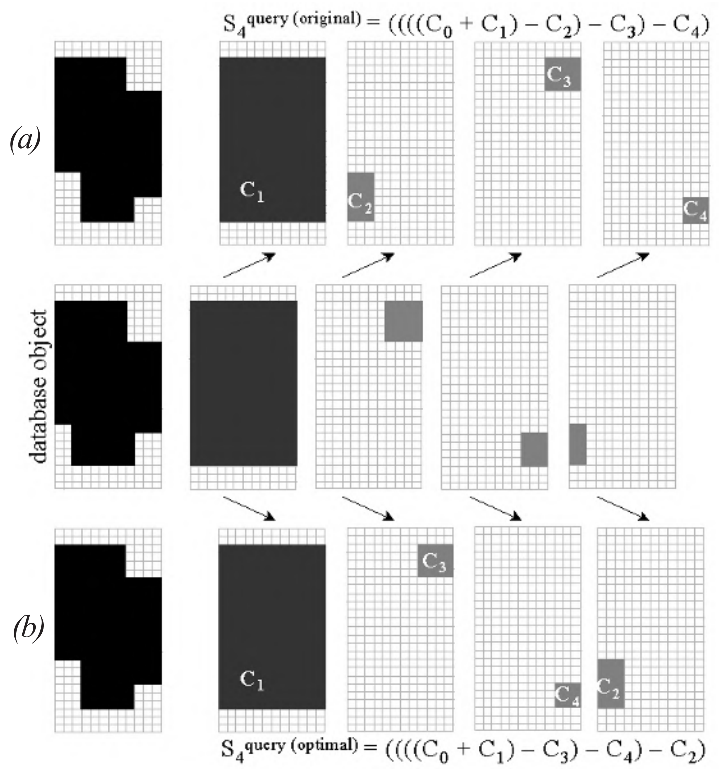
of an object o, can be transformed into a $6k$-dimensional feature vector. Thereby, each cover C_{i+1}, $0 \leq i \leq k$-1, is mapped onto 6 values in the feature vector f_o in the following way:

$f_o^{6i+1} = x$-position of C_{i+1}
$f_o^{6i+2} = y$-position of C_{i+1}
$f_o^{6i+3} = z$-position of C_{i+1}
$f_o^{6i+4} = x$-extension of C_{i+1}
$f_o^{6i+5} = y$-extension of C_{i+1}
$f_o^{6i+6} = z$-extension of C_{i+1}.

The Vector Set Model

As described in the foregoing section, a data object is represented as a feature vector which consists of values obtained from a cover sequence approximation. For similarity queries this method yields a major problem. Always comparing the two covers having the same ranking according to the symmetric volume

Figure 10. Advantages of free permutations: (a) original query object and (b) permuted query object



$S_4^{\text{query (original)}} = ((((C_0 + C_1) - C_2) - C_3) - C_4)$
(a)
C_1
C_2
C_3
C_4
database object
(b)
C_1
C_3
C_4
C_2
$S_4^{\text{query (optimal)}} = ((((C_0 + C_1) - C_3) - C_4) - C_2)$

difference does not make sense in all cases. Two objects can be considered very different, because of the order of their covers, although they are very similar by intuition. The reason for this effect is that the order of the covers does not guarantee that the most similar covers due to size and position will be stored in the same dimensions. Especially for objects generating two or more covers having almost the same volume, the intuitive notion of similarity can be seriously disturbed. Thus, the possibility to match the covers of two compared objects with more degrees of freedom might offer a better similarity measure. Figure 10 displays a two-dimensional example of a comparison between a query object and a very similar database object. The first sequence (cf. Figure 10a) represents the covers of the query object in the order given by the symmetric volume difference. Let us note that the covers C_2, C_3, and C_4 are not very similar to the corresponding covers of the database object and therefore, the calculated similarity is relatively weak. By rearranging the order of these covers, the total distance between the query object and the database object is considerably decreasing, which is displayed in Figure 10b. Thus, the new order preserves the similarity between the objects much better.

To guarantee that the permutation with the minimal distance is used, Kriegel, Brecheisen, et al. (2003) proposed to represent a CAD object by a set of feature vectors $X \subset IR^d$ where $|X| \leq k$.

The Minimal Matching Distance. A distance measure on vector sets that demonstrates to be suitable for defining similarity in our application is based on the *minimal weight perfect matching* of sets. This well-known graph problem can be applied here. Let us first introduce some notations.

Definition 6 (Weighted Complete Bipartite Graph). A *graph* $G = (V, E)$ consists of a (finite) set of vertices V and a set of edges $E \subseteq V \times V$. A *weighted graph* is a graph $G = (V, E)$ together with a weight function w: $E \rightarrow IR$. A *bipartite graph* is a graph $G = (X \cup Y, E)$ where $\mathrm{X} \cap \mathrm{Y} = \emptyset$ and $E \subseteq X \times Y$. A *bipartite graph* $G = (X \cup Y, E)$ is called *complete* if $E = X \times Y$.

Definition 7 (Perfect Matching). Given a bipartite graph $G = (X \cup Y, E)$ a *matching* of X to Y is a set of edges $M \subseteq E$ such that no two edges in M share an endpoint, in other words

$$\forall\ (x_1, y_1), (x_2, y_2) \in M: x_1 = x_2 \Leftrightarrow y_1 = y_2.$$

A matching M of X to Y is *maximal* if there is no matching M' of X to Y such that $|M| < |M'|$. A maximal matching M of X to Y is called a *complete matching* if $|M| = \min\{|X|, |Y|\}$. In the case $|X| = |Y|$ a complete matching is also called a *perfect matching*.

Definition 8 (Minimum Weight Perfect Matching). Given a weighted bipartite graph $G = (X \cup Y, E)$ together with a weight function w: $E \rightarrow IR$. A perfect matching M is called a *minimum weight perfect matching*, iff for any other perfect matching M', the following inequality holds:

$$\sum_{(x,y)\in M} w(x,y) \le \sum_{(x,y)\in M'} w(x,y)$$

In our application, we build a complete bipartite graph $G = (X \cup Y, E)$ between two vector sets $X, Y \subset IR^d$ with $|X|, |Y| \le k$. We set $X' = X \times \{1\}$ and $Y' = Y \times \{2\}$ in order to fulfill the property $X' \cap Y' = \emptyset$. The weight of each edge $((\bar{x},1),(\bar{y},2)) \in X' \times Y'$ in this graph G is defined by the distance $dist(\bar{x},\bar{y})$ between the vectors $\bar{x}$ and $\bar{y}$. For example, the Euclidean distance can be used here. A perfect matching is a subset $M \subseteq X' \times Y'$ that connects each $\bar{x} \in X'$ to exactly one $\bar{y} \in Y'$ and vice versa. A minimal weight perfect matching is a matching with maximum cardinality and a minimum sum of weights of its edges. Since a perfect matching can only be found for sets of equal cardinality, it is necessary to introduce weights for unmatched nodes when defining a distance measure.

Definition 9 (Enumeration of a Set). Let S be any finite set of arbitrary elements. Then π is a mapping that assigns $s \in S$ a unique number $i \in \{1,..., |S|\}$. This is written as $\pi(S) = (s_1,..., s_{|s|})$. The set of all possible enumerations of S is named $\Pi(S)$.

Definition 10 (Minimal Matching Distance). Let $V \subset IR^d$ and let $IR^d \times IR^d \to IR$ be a distance function between two d-dimensional feature vectors. Let $X = \{\bar{x}_1,...,\bar{x}_{|X|}\}, Y = \{\bar{y}_1,...,\bar{y}_{|Y|}\} \in 2^V$ be two vector sets. We assume w.l.o.g. $|X| \le |Y| \le k$. Furthermore, let $w: V \to IR$ be a weight function for the unmatched elements. Then the *minimal matching distance* $d_{mm}^{dist,w} : 2^V \times 2^V \to IR$ is defined as follows:

$$d_{mm}^{dist,w}(X,Y) = \min_{\pi \in \Pi(Y)} \left(\sum_{i=1}^{|X|} dist(\bar{x}_i, \bar{y}_{\pi(i)}) + \sum_{i=|X|+1}^{|Y|} w(\bar{y}_{\pi(i)}) \right).$$

The weight function $w: V \to IR$ provides the penalty given to every unassigned element of the set having larger cardinality. Let us note that the *minimal matching distance* is a specialization of the *netflow distance* which was introduced by Ramon and Bruynooghe (2000). The netflow distance was shown to be a metric and to be computable in polynomial time. Therefore, we derive the following lemma without further proof.

Lemma 1. Let $V \subset IR^d$. The minimal matching distance $d_{mm}^{dist,w}: 2^V \times 2^V \to IR$ is a metric if the underlying distance function $dist: IR^d \times IR^d \to IR$ is a metric and the weight function $w: V \to IR$ meets the following conditions for all $\vec{x}, \vec{y} \in V$:

- $w(\vec{x}) > 0$
- $w(\vec{x}) + w(\vec{y}) \geq dist(\vec{x}, \vec{y})$

Definition 11 (Dummy Vectors). Let $V \subset IR^d$ be a set of d-dimensional vectors. Let $\|\vec{x} - \vec{y}\|_2$ be the Euclidean distance between $\vec{x}, \vec{y} \in IR^d$. Furthermore, let $\vec{w} \in IR^d - V$ be a "dummy" vector. Then $w_{\vec{w}}: V \to IR$, $w_{\vec{w}}(\vec{x}) = \|\vec{x} - \vec{w}\|_2$, denotes a set of weight functions based on dummy vectors.

A good choice of $\vec{w}$ for our application is $\vec{0}$, since it has the shortest average distance within the position and has no volume. Since there are no covers having no volume in any data object, the conditions for the metric character of the minimum matching distance are satisfied (cf. Lemma 1).

By using the method proposed by Kuhn (1955) and Munkres (1957), the minimum matching distance can be computed in polynomial time.

The Industrial Prototype BOSS

In this section, we describe an industrial prototype, called *BOSS* (*B*rowsing *O*PTICS-Plots for *S*imilarity *S*earch) (Brecheisen, Kriegel, Kröger, & Pfeifle, 2004). BOSS is based on the introduced similarity models and on the hierarchical clustering algorithm OPTICS (Ankerst, Kastenmüller, Kriegel, & Seidl , 1999). BOSS is an interactive data browsing tool which depicts the reachability plot computed by OPTICS in a user-friendly way together with appropriate representatives of the clusters. This clear illustration supports the user in his time-consuming task to find similar parts. BOSS helps to reduce the cost of developing and producing new parts by maximizing the reuse of existing parts because it allows the user to browse through the hierarchical structure of the clusters in a top-down way. Thus the engineers get an overview of already existing parts and are able to navigate their way through the diversity of existing variants of products, such as cars. BOSS was designed mainly for two different reasons:

- The evaluation of similarity models (for the scientists), and
- Similarity search (for the industrial user).

Evaluation of Similarity Models

In general, similarity models can be evaluated by computing *k*-nearest neighbor (*k*-nn) queries. A drawback of this evaluation approach is that the quality measure of the similarity model depends on the results of few similarity queries and, therefore, on the choice of the query objects. A model may perfectly reflect the intuitive similarity according to the chosen query objects and would be evaluated as "good" although it produces disastrous results for other query objects. As a consequence, the evaluation of similarity models with sample *k*-nn queries is subjective and error-prone.

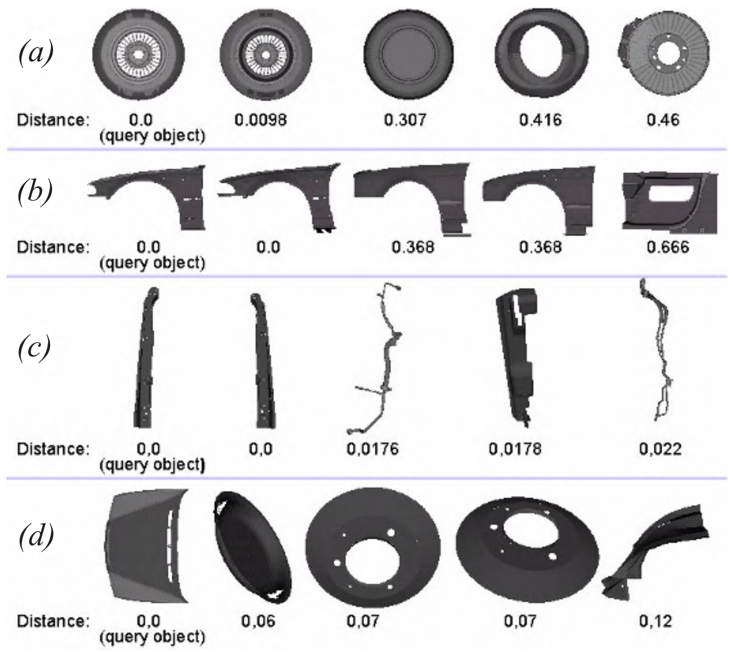
A better way to evaluate and compare several similarity models is to apply a clustering algorithm. Clustering groups a set of objects into classes where objects within one class are similar and objects of different classes are dissimilar to each other. The result can be used to evaluate which model is best suited for which kind of objects.

k-nearest neighbor Queries

A *k*-nn query retrieves the *k* most similar database objects for a given query object. In Figure 11, the results of a 5-nn query for two different similarity models *A* and *B* are presented, whereby model *A* is an inept model and model *B* a suitable similarity model for voxelized CAD data. We achieved satisfying results for each model depending on the query object. For a tire, for example, model *A* performs very well, yielding objects that are intuitively very similar to the query object (cf. Figure 11a). Comparably good results are also produced by model *B* for a part of the fender (cf. Figure 11b).

Although both models deliver rather accurate results for the chosen query objects, we also see in Figure 11 that these results are delusive. Figure 11c shows a nearest neighbor query for an object where there exist several similar parts to this object within our database. Model *A* does not recognize this. Furthermore, there might be objects for which no similarity model can yield any intuitively similar parts (cf. Figure 11d). Obviously, we should not discard a similarity model if the chosen query object belongs to noise. This confirms the

Figure 11. Results of 5-nn queries for a "good" and "bad" similarity model: (a) "good" query object-"bad" model, (b) "good" query object-"good" model, (c) "good" query object-"bad" model, and (d) "bad" query object-"good" model


(a)
Distance: 0.0 (query object) 0.0098 0.307 0.416 0.46
(b)
Distance: 0.0 (query object) 0.0 0.368 0.368 0.666
(c)
Distance: 0,0 (query object) 0,0 0,0176 0,0178 0,022
(d)
Distance: 0,0 (query object) 0,06 0,07 0,07 0,12

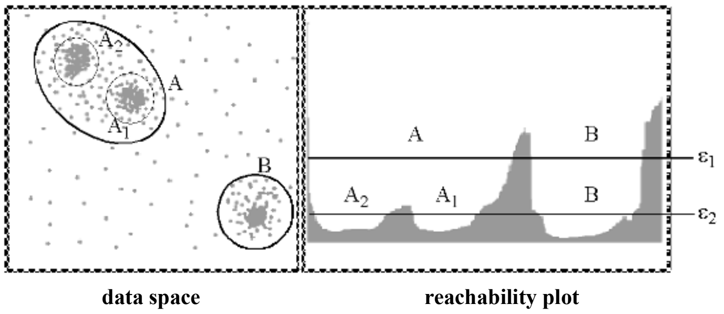
assumption that the method of evaluating similarity models using several *k*-nn queries is subjective and error-prone, due to its dependency on the choice of the query objects.

By means of BOSS, which is based on the density-based hierarchical clustering algorithm OPTICS, we can overcome the earlier-described difficulties. A description of the hierarchical clustering algorithm OPTICS is given in the following section.

OPTICS: A Density-Based Hierarchical Clustering Algorithm

A more objective way to evaluate and compare several similarity models is to apply a clustering algorithm. Clustering groups a set of objects into classes where objects within one class are similar and objects of different classes are

Figure 12. Reachability plot (right) computed by OPTICS for a 2D data set (left)



dissimilar to each other. The result can be used to evaluate which model is best suited for which kind of objects. Furthermore, by using clustering, the evaluation of the models is based on the whole data set and not only on a few sample objects.

Reachability Plots. The output of OPTICS is a linear ordering of the database objects minimizing a binary relation called *reachability* which is, in most cases, equal to the minimum distance of each database object to one of its predecessors in the ordering. This idea is similar to the Single-Link method, but instead of a dendrogram, the resulting reachability-plot is much easier to analyze. The reachability values can be plotted for each object of the cluster-ordering computed by OPTICS. Valleys in this plot indicate clusters; objects having a small reachability value are more similar to their predecessor objects than objects having a higher reachability value.

The reachability plot generated by OPTICS can be cut at any level e parallel to the abscissa. It represents the density-based clusters according to the density threshold ε: A consecutive subsequence of objects having a smaller reachability value than ε belong to the same cluster. An example is presented in Figure 12. For a cut at the level ε_1, we retrieve two clusters denoted as A and B. Compared to this clustering, a cut at level ε_2 would yield three clusters. The cluster A is split into two smaller clusters denoted as A_1 and A_2 and cluster B has decreased its size.

Note that the visualization of the cluster-order is independent from the dimension of the data set. For example, if the objects of a high-dimensional data set are distributed similar to the distribution of the two-dimensional data set depicted in Figure 12, that is, there are three "Gaussian bumps" in the data set, the reachability plot would look very similar to the one presented in Figure 12. Furthermore, density-based clustering is not only restricted to feature spaces, but can be applied to all kinds of metric spaces, for example, to data spaces where objects are represented by vector sets.

Experimental Evaluation of the Voxelized Similarity Models

In this section, we briefly summarize the main results of our experimental evaluation. Generally spoken, the data partitioning similarity models, that is, the *cover sequence model* and the *vector set model,* reflect the intuitive similarity between CAD objects better than the space partitioning similarity models, that is, the *volume model*, the *solid-angle model* and the *eigenvalue model*.

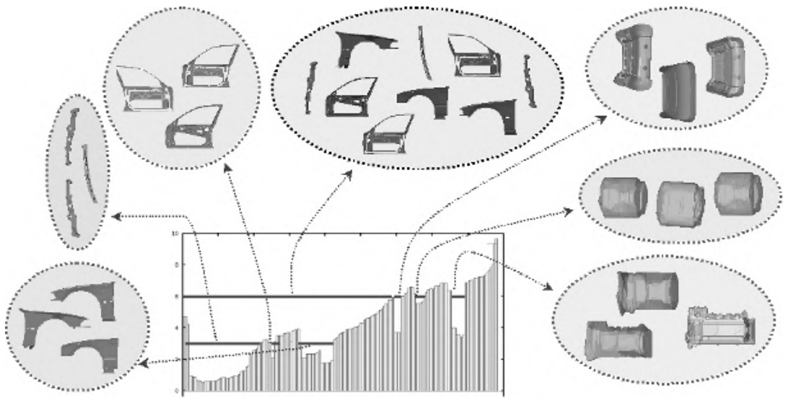
The eigenvalue model produces the most meaningful results among the three introduced space partitioning similarity models. Using the eigenvalue model as basis for the OPTICS run results in clusters containing intuitively similar objects. Nevertheless, the eigenvalue model suffers from the same shortcomings as the data partitioning cover sequence model. Although both models produce rather meaningful clusters, they fail to detect important cluster hierarchies.

Only the vector set model detects these hierarchies, which are important for navigating through massive data sets. To sum up, our new evaluation method based on clustering showed that the combination of the data partitioning cover sequence model and the new paradigm of using sets of feature vectors for representing objects is a very powerful approach for the detection of similar CAD parts.

Similarity Search

BOSS is also an interactive data mining tool which depicts the reachability plot computed by OPTICS in a user-friendly way together with appropriate representatives of the clusters. This clear illustration supports the user in his time-consuming task to find similar parts. From the industrial user's point of view, BOSS meets the following two requirements:

Figure 13. Browsing through reachability plots with different thresholds ε_{cut}



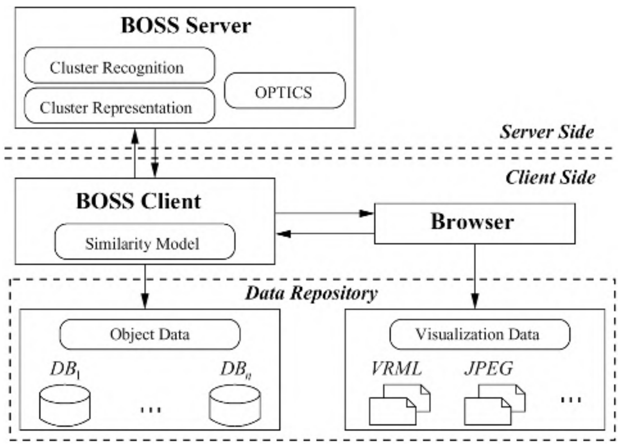
- The hierarchical clustering structure of the data set is revealed at a glance. The reachability plot is an intuitive visualization of the clustering hierarchy which helps to assign each object to its corresponding cluster or to noise. Furthermore, the hierarchical representation of the clusters using the reachability plot helps the user to get a quick overview over all clusters and their relation to each other. As each entry in the reachability plot is assigned to one object, we can easily illustrate some representatives of the clusters belonging to the current density threshold ε_{cut} (cf. Figure 13).
- The user is not only interested in the shape and the number of the clusters, but also in the specific parts building up a cluster. As for large clusters, it is rather difficult to depict all objects; representatives of each cluster should be displayed. We can browse through the hierarchy of the representatives in the same way as through the OPTICS plots.

This way, the cost of developing and producing new parts could be reduced by maximizing the reuse of existing parts, because the user can browse through the hierarchical structure of the clusters in a top-down way. Thus the engineers get an overview of already existing parts and are able to navigate their way through the diversity of existing variants of products, such as cars.

System Architecture

The development of the industrial prototype BOSS is an important step towards developing a comprehensive, scalable, and distributed computing

Figure 14. BOSS distributed architecture

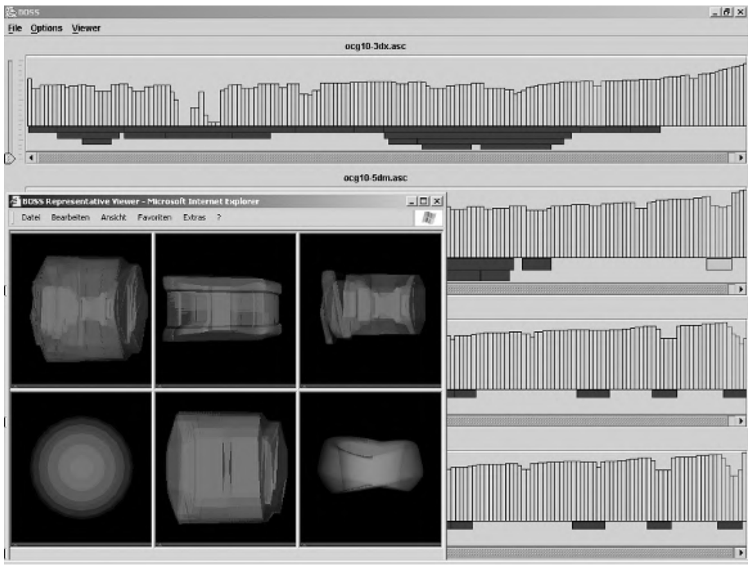


solution designed to make the effectiveness of OPTICS and the proposed cluster recognition and representation algorithms available to a broader audience. BOSS is a client/server system allowing users to provide their own data locally, along with an appropriate similarity model (cf. Figure 14).

The data provided by the user will be comprised of the objects to be clustered, as well as a data set to visualize these objects, for example, VRML files for CAD data (cf. Figure 15) or JPEG images for multimedia data. Since this data resides on the user's local computer and is not transmitted to the server, heavy network traffic can be avoided. In order for BOSS to be able to interpret this data, the user must supply his own similarity model with which the reachability data can be calculated.

The independence of the data processing and the data specification enables maximum flexibility. Further flexibility is introduced through the support of external visual representation. As long as the user is capable of displaying the visualization data in a browser, for example, by means of a suitable plug-in, the browser will then load Web pages generated by BOSS, displaying the appropriate data. Thus, multimedia data such as images or VRML files can easily be displayed (cf. Figure 15). By externalizing the visualization procedure, we can resort to approved software components, which have been specifically developed for displaying objects which are of the same type as the objects within our clusters.

Figure 15. BOSS screenshot


BOSS
File Options Viewer
ocg10-3dx.asc
ocg10-5dm.asc
BOSS Representative Viewer - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?

Conclusion

In this chapter, we introduced three space partitioning similarity models and two data partitioning similarity models for three-dimensional voxelized CAD data. Furthermore, we presented hierarchical clustering as an effective way to analyse and compare similarity models. We showed that hierarchical clustering is more suitable for the evaluation of similarity models than the commonly used *k*-nn queries. Based on this evaluation method and on the introduced similarity models, we described a prototype, called BOSS, which is suitable for industrial use. BOSS helps the user to cope with rapidly growing amounts of data, and helps thereby to reduce the cost of developing and producing new parts.

An interesting direction for future research is to combine the advantages of the eigenwert model and the vector set model. For instance, we could describe a voxelized object not by a set of rectangular covers, but by a set of Gaussian distribution functions. Each of these distribution functions can be represented by their eigenvalues and eigenvectors. Finding a minimal matching between these distribution functions might lead to an improved similarity model for voxelized CAD objects.

References

Agrawal, R., Faloutsos, C., & Swami, A. (1993). Efficient similarity search in sequence databases. In *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms, Lecture Notes in Computer Science, Vol. 730* (pp. 69-84). Chicago: Springer.

Agrawal, R., Lin, K.-I., Sawhney, H., & Shim, K. (1995). Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *Proceedings of the 21st International Conference on Very Large Databases (VLDB)* (pp. 490-501). Zurich, Switzerland: Morgan Kaufman.

Ankerst, M., Breunig, M., Kriegel, H.-P., & Sander, J. (1999). OPTICS: Ordering points to identify the clustering structure. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data* (pp. 49-60). Philadelphia: ACM Press.

Ankerst, M., Kastenmüller, G., Kriegel, H.-P., & Seidl, T. (1999). Nearest neighbor classification in 3D protein databases. In *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology (ISMB'99)* (pp. 34-43). Heidelberg, Germany: AAAI Press.

Badel, A., Mornon, J. P., & Hazout, S. (1992). Searching for geometric molecular shape complementarity using bidimensional surface profiles. *Journal of Molecular Graphics, 10*, 205-211.

Berchtold, S., Böhm, C., Jagadish, H. V., Kriegel, H.-P., & Sander, J. (2000). Independent quantization: An index compression technique for high-dimensional data spaces. In *Proceedings of the International Conference on Data Engineering (ICDE)* (pp. 577-588). San Diego, CA: IEEE Computer Society.

Berchtold, S., Keim, D. A., & Kriegel, H.-P. (1996). The X-tree: An index structure for high-dimensional data. In *Proceedings of the 22nd International Conference on Very Large Databases (VLDB)* (pp. 28-39). Mumbai, India: Morgan Kaufman.

Berchtold, S., Keim, D. A., & Kriegel, H.-P. (1997). Using extended feature objects for partial similarity retrieval. *VLDB Journal, 6*(4), 333-348.

Berchtold, S., & Kriegel, H.-P. (1997). S3: Similarity search in CAD database systems. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data* (pp. 564-567). Tucson, AZ: ACM Press.

Brecheisen, S., Kriegel, H.-P., Kröger, P., & Pfeifle, M. (2004). Visually mining through cluster hierarchies. In *Proceedings of the SIAM International Conference on Data Mining (SDM)* (pp. 400-412). Lake Buena Vista, FL: SIAM.

Connolly, M. (1986). Shape complementarity at the hemoglobin a1b1 subunit inter-face. *Biopolymers, 25*, 1229-1247.

Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data* (pp. 419-429). Minnesapolis, MN: ACM Press.

Hafner, J., Sawhney, H. S., Equitz, W., Flickner, M., & Niblack, W. (1995). Efficient color histogram indexing for quadratic form distance functions. *IEEE Trans. on Pattern Analysis and Machine Intelligence, 17*(7), 729-736.

Higgins, W. E. (1990). *Automatic analysis of 3D and 4D radiological images.* Grant application to the Department of Health and Human Services.

Huang, J., Yagel, R., Filippov, V., & Kurzion, Y. (1998). An accurate method for voxelizing polygon meshes. In *Proceedings of the IEEE Symposium on Volume Visualization* (pp. 119-126). Research Triangle Park, NC: IEEE Computer Society.

Jagadish, H. V. (1991). A retrieval technique for similar shapes. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data* (pp. 208-217). Denver, CO: ACM Press.

Jagadish, H. V., & Bruckstein, A. M. (1992). On sequential shape descriptions. *Pattern Recognition, 25*(2), 165-172.

Kaufman, A. (1987). An algorithm for 3D scan-conversion of polygons. In *Proceedings of Euro-graphics* (pp. 197-208). Amsterdam, The Netherlands: Elsevier.

Keim, D. (1999). Efficient geometry-based similarity search of 3D spatial data-bases. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data* (pp. 419-430). Philadelphia: ACM Press.

Korn, F., Sidiropoulos, N., Faloutsos, C., Siegel, E., & Protopapas, Z. (1996). Fast nearest neighbor search in medical image databases. In

Proceedings of the 22nd International Conference on Very Large Databases (VLDB) (pp. 215-226). Mumbai, India: Morgan Kaufman.

Kriegel, H.-P., Brecheisen, S., Kröger, P., Pfeifle, M., & Schubert, M. (2003). Using sets of feature vectors for similarity search on voxelized CAD objects. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data* (pp. 587-598). San Diego, CA: ACM Press.

Kriegel, H.-P., Kröger, P., Mashael, Z., Pfeifle, M., Pötke, M., & Seidl, T. (2003). Effective similarity search on voxelized CAD objects. In *Proceedings of the 8th International Conference on Database Systems for Advanced Applications (DASFAA)* (pp. 27-36). Kyoto, Japan: IEEE Computer Society.

Kriegel, H.-P., Schmidt, T., & Seidl, T. (1997). 3D similarity search by shape approximation. In *Proceedings of the 5th International Symposium on Large Spatial Databases (SSD), Lecture Notes in Computer Science, Vol. 1262* (pp. 11-28). Berlin, Germany: Springer.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly, 2*, 83-97.

Lin, K., Jagadish, H. V., & Faloutsos, C. (1994). The TV-tree: An index structure for high-dimensional data. *International Journal on Very Large Data Bases (VLDB Journal), 3*(4), 517-542.

McNeely, W. A., Puterbaugh, K. D., & Troy, J. J. (1999). Six degree-of-freedom haptic rendering using voxel sampling. In *Proceedings of the ACM SIGGRAPH International Conference on Computer Graphics and Interactive Techniques* (pp. 401-408). Los Angeles, CA: ACM Press.

Mehrotra, R., & Gary, J. E. (1993). Feature-based retrieval of similar shapes. In *Proceedings of the 9th International Conference on Data Engineering (ICDE)* (pp. 108-115). Vienna, Austria: IEEE Computer Society.

Möller, T., & Haines, E. (1999). *Real-time rendering*. Natick, MA: A. K. Peters.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the SIAM, 6*, 32-38.

Niblack, W., Barber, R., Equitz, W., Flickner, M., Glasmann, E., Petkovic, D., Yanker, P., Faloutsos, C., & Taubin, G. (1993). The QBIC project:

Querying images by content using color, texture, and shape. In *SPIE 1908 International Symposium on Electronic Imaging: Science and Technology Conference, Storage and Retrieval for Image and Video Databases* (pp. 173-187). San Jose, CA: SPIE.

Ramon, J., & Bruynooghe, M. (2001). *A polynomial time computable metric between point sets* (Tech. Rep. No. CW 301). Katholieke Universiteit Leuven, Belgium.

Schneider, R., Kriegel, H.-P., Seeger, B., & Heep, S. (1989). Geometry-based similarity retrieval of rotational parts. In *Proceedings of the International Conference on Data and Knowledge Systems for Manufacturing and Engineering* (pp. 150 160). Gaithersburg, MD: IEEE Computer Society.

Taubin, G., & Cooper, D. B. (1991). Recognition and positioning of rigid objects using algebraic moment invariants. In *Proceedings of the 36th SPIE Conference on Geometric Methods in Computer Vision, Annual International Symposium on Optical and Optoelectronic Applied Science and Engineering* (pp. 175-186). San Diego, CA: SPIE.

Vincent, L. (1991). New trends in morphological algorithms. In *SPIE Proceedings on Non-Linear Image Processing II.* San Jose, CA: SPIE.

Chapter V

STEP-NC to Complete Product Development Chain

Xun W. Xu, University of Auckland, New Zealand

Abstract

This chapter addresses the issue of product development chain from the perspective of data modeling and streamlining. The focus is on an emerging ISO standard, informally known as STEP-NC, and how it may close the gap between design and manufacturing for a complete, integrated product development environment. This new standard defines a new generation of NC programming language and is fully compliant with STEP. There is a whole suite of implementation methods one may utilize for development purposes. STEP-NC brings richer information to the numerically-controlled machine tools; hence, intelligent machining and control are made possible. Its Web-enabled feature gives an additional dimension in that e-manufacturing can be readily supported. A case study toward the end demonstrates a STEP compliant, Web-enabled manufacturing system.

Introduction

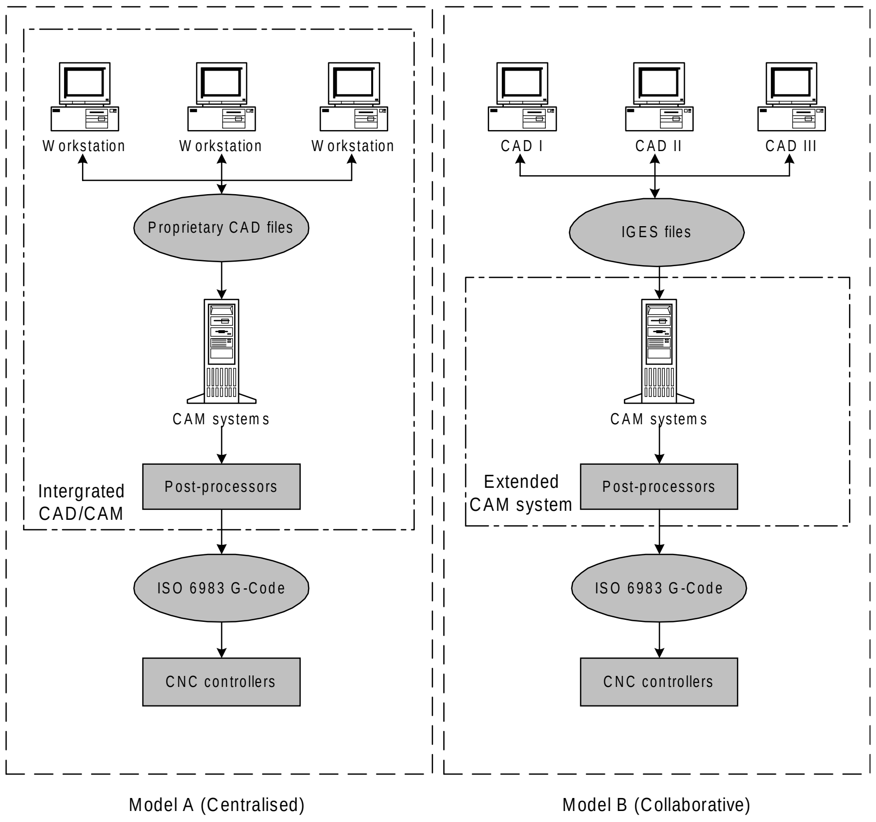
In the manufacturing domain, there are two types of traditional product development models, *centralized model* and *collaborative model*. In a centralized model, manufacturing activities occur within a single manufacturer or a few manufacturers that have similar information infrastructures. In this model, proprietary data formats are commonly used. In a collaborative model, a middle tier is added using a neutral data exchange format. As such, collaborative activities in the manufacturing environment become easier. Figure 1 illustrates the data flows in these two models.

In Model A, both CAD and CAM systems use the same proprietary data format. Over the years, CAD/CAM system vendors have developed different data formats to support their systems throughout the design and manufacturing processes. The benefits of this model are obvious. CAD and CAM systems are unified by the same data format so that data incompatibilities between CAD and CAM systems are eliminated. Furthermore, since there is no data-transferring barrier, system vendors have more freedom to model more information. In addition to pure geometry, integrated CAD/CAM systems can cater for all the activities ranging from design to NC programming. Some of such systems include Pro/ENGINEER (with Pro/NC), CATIA and UGS. However, these systems are not without problems. They assume that data exchange during a product life cycle only occurs within one manufacturer or among a few manufacturers that implement the same CAD/CAM system. When more manufacturers are involved in the product life cycle, it is hard, if not impossible, to unify those manufacturers with a specific proprietary data format. Therefore, the structure represented by Model A is deemed unfit for collaborative manufacturing due to data incompatibility.

Model B aims to solve this problem by using exchangeable neutral data formats such as IGES (Initial Graphics Exchange Specification). Neutral data formats provide a middle tier to connect CAD and CAM systems. With the help of neutral data formats, Model B creates a collaborative manufacturing environment and makes design data exchange possible for large projects at the international level. Yet, some problems still exist:

- IGES was designed to exchange geometrical information only, so additional design or manufacturing information (such as feature information) within a proprietary model is ignored.

Figure 1. Two traditional product development models



- Some information may get lost during data transfer; geometry stitching or model repair is often needed.
- IGES is not an international standard.

There are also problems common to both models. Different data formats (e.g., IGES and ISO 6983 G-Codes (ISO 6983-1, 1982)) are used in the design to manufacturing chain. Data loss occurs in the transaction from design to manufacturing because only low-level, step-by-step sequential machining commands are passed onto the CNC (Computer Numerically Controlled) controllers, leaving the complete product model behind.

Of particular significance is the endeavor made by the International Organization for Standardization (ISO) to introduce the Standard for the Exchange of Product model data (i.e., ISO 10303 STEP [ISO 10303-1, 1994]). STEP is a family of standards defining a robust and time-tested methodology for describing product data throughout the lifecycle of the product. It is now widely used in CAD and Product Data Management (PDM) systems. Major aerospace and automotive companies have proven the value of STEP through production implementations resulting in savings of $150M per year in the U.S. (Gallaher, O'Connor, & Phelps, 2002; PDES, Inc., 2005). Moreover, STEP has recently been extended to cater for manufacturing data modeling and execution with an aim to fill the information gap between CAD/CAPP/CAM and CNC. The standard is informally known as STEP-compliant Numerical Control, or otherwise STEP-NC for short. It was given an ISO name of "ISO 14649: Data Model for Computerized Numerical Controllers (ISO 14649-1, 2003)". Like other STEP models, the information is mostly specified in EXPRESS (ISO 10303-11, 1994), a formal language for the definition of entity-attribute data models.

The objective of this chapter is to take an in-depth look into (1) the STEP-NC standards, (2) the data modeling tools and methods for STEP-NC, and (3) how it can fill the gap between CAD/CAPP/CAM and CNC.

Data Exchange Using STEP and STEP-NC

STEP is regarded as a unified standard for describing all aspects of a product during its life cycle. It does so by establishing various Application Protocols (APs) targeted at different application domains, be it design, manufacturing, or maintenance.

Data Exchange between CAD Systems Using STEP

The STEP standard was initially designed to offer a neutral data exchange method in replacement of IGES. The two APs that have been established to mainly support design data exchanging and sharing are the AP for configuration-controlled 3D designs of mechanical parts and assemblies (AP 203) (ISO 10303-203, 1994), and the AP for core data for automotive mechanical design processes (AP 214) (ISO 10303-214, 1994). Currently, most of the commercial CAD systems can output STEP AP-203 and/or STEP AP-214 files via STEP translators. According to the report from Research Triangle Institute

(1999), when STEP is used as a neutral format to exchange wire frame and surface data between commonly-used commercial CAD systems, it fares better than IGES. This indicates that STEP is ready to replace IGES. However, the STEP standard is much more than a neutral data format that translates geometrical data between CAD systems. The ultimate goal of STEP is to provide a complete computer-interpretable product data format, so that users can integrate business and technical data to support the whole product life cycle: design, analysis, manufacturing, sales and customer services.

Data Flow between CAD, CAM, and CNC Systems

By implementing STEP AP-203 and STEP AP-214 within CAD systems, data exchange barriers are removed in the heterogeneous design environment. Yet, data exchange problems between CAD, CAM, and CNC systems remain unsolved. CAD systems are designed to describe the geometry of a part precisely. CAM systems, on the other hand, focus on using computer systems to generate plans and control the manufacturing operations according to the geometrical information present in a CAD model and the existing resources on the shop-floor. The final result from a CAM system is a set of CNC programs that can be executed on a CNC machine. The neutral data formats such as STEP AP-203 and STEP AP-214 only unify the input data for a CAM system. On the output side of a CAM system, a 50-year-old international standard, ISO 6983, still dominates the control systems of most CNC machines. Outdated yet still widely used, ISO 6983 has become an impediment for the contemporary collaborative manufacturing environment. Some of the technical limits and problems found with ISO 6983 are summarized as follows (Xu & He, 2004):

1. The language focuses on programming the path of the cutter center location (CL) with respect to the machine axes, rather than the machining tasks with respect to the part.
2. The standard defines the syntax of a program statement, but in most cases leaves the semantics ambiguous.
3. Vendors usually supplement the language with extensions that are not covered in the limited scope of ISO 6983; hence, the CNC programs are not exchangeable.
4. It only supports one-way information flow from design to manufacturing. The changes made at the shop-floor cannot be directly fed back to the

designer. Hence, invaluable experiences on the shop-floor cannot be preserved and reused.

5. There is limited control over program execution, and it is difficult to change the program in the workshop.
6. CAD data are not utilized at a machine tool. Instead, they have to be processed by a machine-specific post-processor, only to obtain a set of low-level, incomplete data that makes verifications and simulation difficult, if not impossible.
7. ISO 6983 does not support the spline data, which makes it incapable of controlling five or more axis millings.

These problems collectively make CNC machining a bottleneck in the concurrent manufacturing environment. It also means that a desired CAD/CAPP/CAM/CNC train is broken at the link between CAM and CNC.

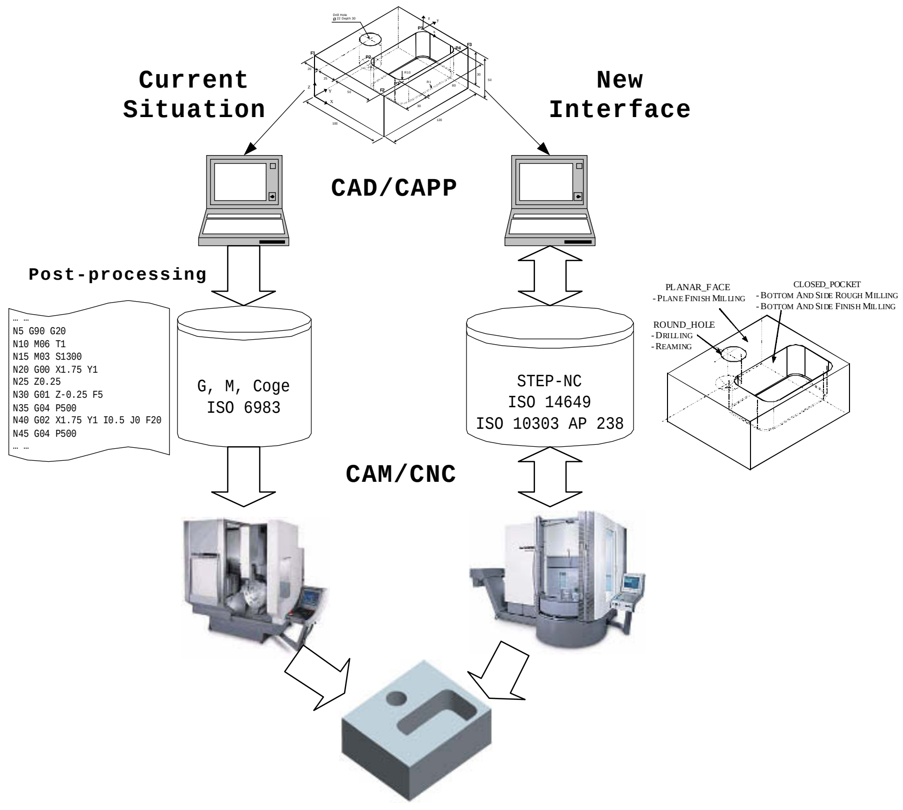
STEP-NC: A Replacement for G-Code

As STEP-NC is an extension of STEP to handling NC processes, it strictly follows the STEP standard. The overriding concept of STEP-NC is that of "Workingsteps". A Workingstep is a manufacturing task that describes information such as a single manufacturing operation, a tool, or a strategy. Other high-level information modeled by STEP-NC includes various NC functions; machine functions; machining strategies; auxiliary commands; technological description such as tool data (dimensions, tool type, conditions, and usage of the tool); and workpiece definitions (surfaces, regions, and features of the finished part).

In essence, STEP-NC describes "what to do", while G-code describes "how to do". STEP-NC describes tasks (pre-drilling, drilling, roughing, finishing) that are based on the machining features (Figure 2), so that the part program supplies the shop-floor with higher-level information, that is, the information about machining tasks and technological data on top of pure geometrical and topological information. As a result, modifications at the shop-floor can be saved and transferred back to the planning department that enables a better exchange and preservation of experience and knowledge.

Some of the benefits with using STEP-NC are as follows (Xu & He, 2004):

Figure 2. Comparison of G-code and STEP-NC



```
… …
N5 G90 G20
N10 M06 T1
N15 M03 S1300
N20 G00 X1.75 Y1
N25 Z0.25
N30 G01 Z-0.25 F5
N35 G04 P500
N40 G02 X1.75 Y1 I0.5 J0 F20
N45 G04 P500
… …
```

- STEP-NC provides a complete and structured data model, linked with geometrical and technological information, so that no information is lost between the different stages of the product development process.
- Its data elements are adequate enough to describe task-oriented NC data.
- The data model is extendable to further technologies and scalable (with Conformance Classes) to match the abilities of a specific CAM, SFP (Shop Floor Programming), or NC system.
- Machining time for small- to medium-sized job lots can be reduced because intelligent optimization can be built into the STEP-NC controllers.
- Post-processor mechanism will be eliminated, as the interface does not require machine-specific information.
- Machine tools are safer and more adaptable because STEP-NC is independent from machine tool vendors.

- Modification at the shop-floor can be saved and fed back to the design department; hence, bi-directional information flow from CAD/CAM to CNC machines can be achieved.
- XML files can be used as an information carrier, hence enable Web-based distributed manufacturing.

Currently two versions of STEP-NC are being developed. The first is the Application Reference Model (ARM), that is, ISO 14649 itself (ISO 14649-1, 2003; ISO 14649-10, 2003; ISO 14649-11, 2003; ISO 14649-111, 2003; ISO 14649-12, 2003; ISO 14649-121, 2003) and the other Application Interpreted Model (AIM), that is, ISO 10303 AP238 (ISO/DIS 10303-238, 2005). Being the ARM model, ISO 14649 provides a detailed analysis of the requirements of CNC applications. The specific objects are accurately defined; so are the relationships among them. On the other hand, the AIM model of STEP-NC, that is, STEP AP-238, is a way to map the application requirement data stipulated in ARM (ISO 14649) using a fixed set of STEP concepts called "generic resources", into an integrated ISO 10303 Application Protocol. More discussions on these two types of STEP-NC models can be found in a later section of this chapter.

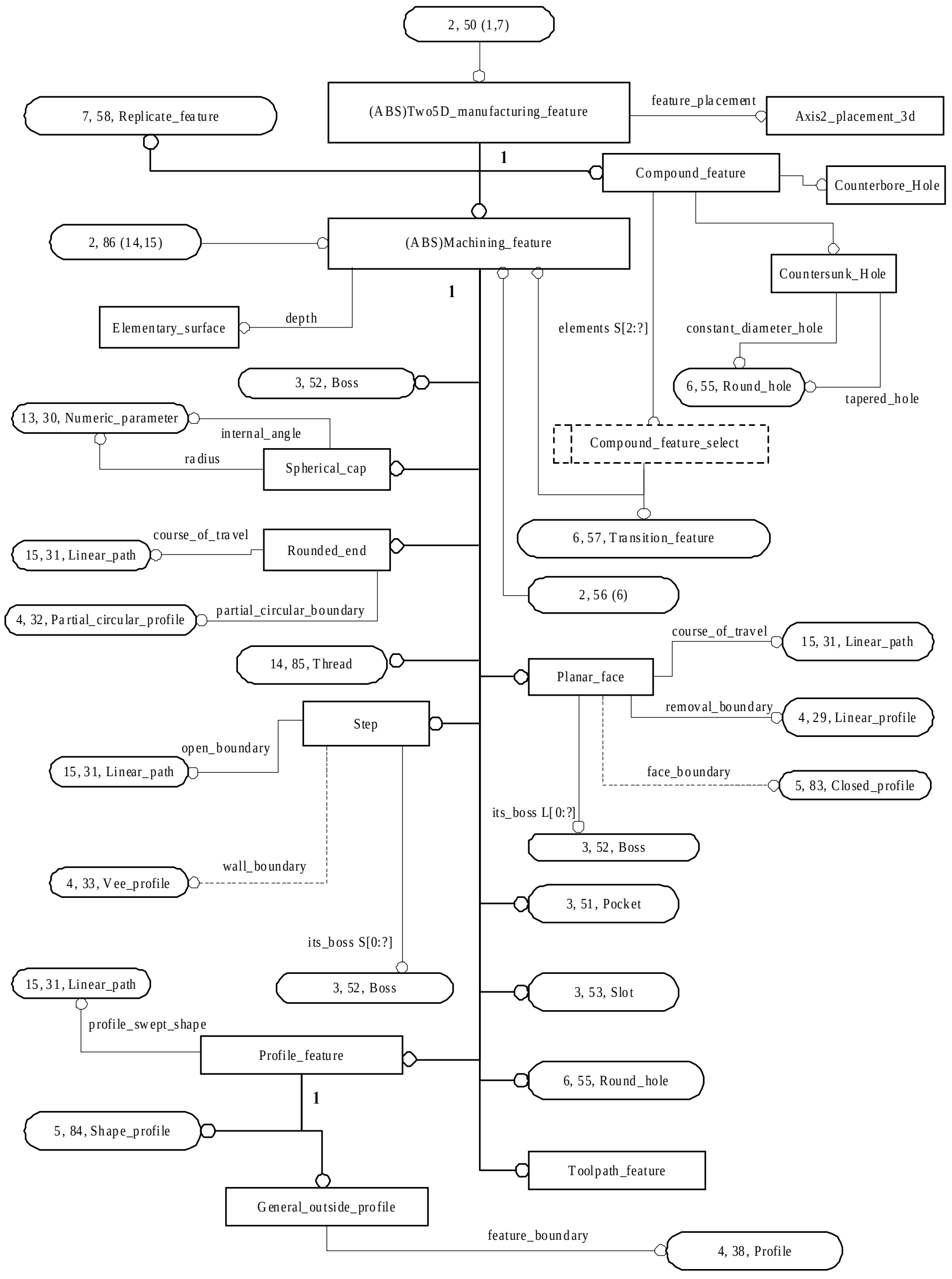
Global Research Endeavor with STEP-NC

The global research in the areas of STEP-NC has remained highly visible with a number of major projects coordinated and conducted across different countries in the same region as well as on a truly international scale. There are three types of projects, those carried out (1) on the international scale, (2) across a few countries in the same region, and (3) within a country. On the international scale, the IMS (Intelligent Manufacturing System) STEP-NC project (IMS STEP-NC Consortium, 2003; Maeder, Nguyen, Richard, & Stark, 2002), endorsed in November, 2001, entails a true international package of actions with research partners from four different regions: European Union, Korea, Switzerland, and the USA. They covered the manufacturers of all systems related to the data interface (CAM systems, controls, and machine tools), the users, and academic institutions. The regional coordinators are Siemens (EU), CADCAMMation (Switzerland), STEP Tools (USA), and ERC-ACI (Korea). Siemens is also the inter-regional coordinator. Formation of the IMS STEP-NC project is seen as the culmination of a number of regional projects carried out by different project groups/consortiums in the past ten years or so, in particular, the European ESPRIT STEP-NC Project, the Super Model Project, the STEP Manufacturing Suite (SMS) Project, and the Rapid Acquisition of Manufactured Parts (RAMP) Project (LSC Group, & R. P. Management (2002). Because of the Workingstep-based feature in STEP-NC, many projects have incorporated prototype/commercial CAPP (Computer-Aided Process Planning) systems or CAM systems with process planning functionalities, that is, STEPturn from ISW Stuttgart, Germany (Storr & Heusinger, 2002; Storr, Pritschow, Heusinger, & Azotov, 2002), ST-Machine from STEP Tools Inc., USA (ST-Machine, 2005; Anonymous, 2003), SFP system from NRL-SNT (National Research Laboratory for STEP-NC Technology), Korean (Suh & Cheon, 2002a; Suh, Cho, & Hong, 2002b; Suh, Cho, Lee, Chung, Cheon, & Hong, 2002c; Suh, Lee, Chung, & Cheon, 2003), the AB-CAM system from Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University, UK (Allen, Newman, Harding, & Rosso, 2003; Newman, Allen, & Rosso, 2003; Roberto, Rosso, Allen, & Newman, 2002), NIST STEP-NC Interpreters, National Institute of Standards and Technology, USA and STEPcNC Converter from the University of Auckland, New Zealand (Xu, 2004; Xu & Mao, 2004).

STEP-NC Data Model

Like all the other parts of the STEP standard, the STEP-NC data model is constructed based on the EXPRESS language (ISO 10303-11, 1994). The EXPRESS language is a formal language for the definition of entity-attribute data models. It is a completely generic modeling language and can therefore be used to model data objects of any type. The STEP-NC EXPRESS information model is organized into schemas. These schemas contain model definitions and serve as a scooping mechanism for subdivision of STEP-NC models. EXPRESS also gives STEP-NC an object-oriented flavor. Its inheritance can be illustrated by the definition of manufacturing features defined in STEP-NC (ISO 14649-10, 2003) (Figure 3). For every 2½D manufacturing feature, there has to be a feature placement value. Therefore, it is defined at the top level (in the Two5D_manufacturing_feature entity). This attribute is inherited by all the "child" entities, that is, machining, replicate, and compound features. Similarly, each sub-type of machining features will have an

Figure 3. EXPRESS-G illustration of STEP-NC manufacturing features



elementary surface to define its depth, and it is defined once for all at the machining feature level.

STEP-NC ARM and AIM Models

The STEP-NC ARM structure written in EXPRESS is effectively a description method. On the other hand, AIM is a way to map the application requirement data (NC data in this case) stipulated in the ARM using a fixed set of STEP "generic resources", into an integrated ISO 10303 Application Protocol, hence AP 238. Because AP-238 is built on the same foundation as the other STEP APs, it can share data seamlessly.

However, often at issue is whether it is best, from an implementer's viewpoint, to implement the STEP-NC ARM (ISO 14649) models directly (a.k.a. ARM implementation), or to implement the STEP-NC AIM (ISO 10303 AP 238) models (a.k.a. AIM implementation). The main difference between these two models is the degree to which they use the STEP representation methods and technical architecture (Feeney, Kramer, Proctor, Hardwick, & Loffredo, 2003; Wolf, 2003). Table 1 compares these two models.

The ISO 14649 standard is more likely to be used in an environment in which CAM systems have exact information from the shop-floor, whereas STEP AP-238, as a part of the STEP standard, is more suitable for complete design and manufacturing integration. The ISO 14649 standard has no mechanism to incorporate other types of STEP data, hence making bi-directional data flow between design and manufacturing more difficult. Unlike ISO 14649, STEP AP-238 encompasses all the information from STEP AP-203 and AP-224 (ISO 10303-224, 2001) plus an interpreted model mapped from ISO 14649. Hence, bi-directional data exchange is enabled.

Table 1. Comparisons between an ARM and AIM model

Comparison criteria	ISO 14649 (ARM) model	ISO 10303-238 (AIM) model
Storage needed	~10 times less than AIM	~10 times more than ARM
Programming	Easy	More complex
Human readable	Difficult	Almost impossible
Compatibilities with STEP	Partly compliant	Fully compliant
Data consistency	Original design information is abandoned	Original design information is preserved

However, STEP AP-238 is not without problems. One problem is that the STEP Integrated Resources used in AP 238 are not adapted to application areas; hence, the data in its files are fragmented and distributed. It only provides an information view of the data, whereas the ARM provides a functional view of the data. An AP 238 file can also become much larger than its equivalent of ISO 14649. STEP AP-238 files are not as easy to decipher as ISO 14649 files. The structure of AP-238 files is more complex and may cost more storage space. In order to work with AIM files, tools and/or libraries such STIX developed by STEP Tools Inc., can alleviate data handling chores for developers (STIX, 2005).

STEP-NC Implementation Methods

EXPRESS language does not define any implementation methods. Therefore, additional implementation methods are defined to describe STEP-NC instances for building product exchange models, for example, ISO 14649 models and ISO 10303 AP 238 models. There are several implementation technologies available:

1. A product model-specific file format called Part 21 physical file (ISO 10303-21, 1994);
2. A variety of programming-language bindings that allow an application programmer to open a data set and access values in its entity instances. Bindings have been developed for C, C++ and Java (ISO 10303-22 1998; ISO 10303-23, 2000; ISO 10303-24, 2001; ISO 10303-27, 2000);
3. The three methods for mapping EXPRESS defined data into XML described by Part 28 Edition 1 (ISO/CD TS 10303-28 [Edition 1], 2002); and
4. The XML schema-governed representation of EXPRESS described by Part 28 Edition 2 (ISO/TS 10303-28 [Edition 2], 2004).

Part 21 Physical File Implementation Method

STEP Part 21 is the first implementation method, which defines the basic rules of storing EXPRESS/STEP data in a character-based physical file. Its aim is to provide a method so that it is possible to write EXPRESS/STEP entities and transmit those entities using normal networking and communication protocols (i.e., FTP [File Transfer Protocol], e-mail, and HTTP [Hyper Text Transfer Protocol]).

A Part 21 file does not have any EXPRESS schemas included. It only defines the relationships between entities that are defined by external EXPRESS schemas. The Part 21 file format uses the minimalist style that was popular before the advent of XML. In this style, the same information is never written twice so that there is no possibility of any contradictions in the data. The style assumes that normally the data will only be processed by software, that people will only look at the data to create test examples or find bugs, and that making the data more easily readable by these people is less important than eliminating redundancies. The Part 21 format is simple and elegant. Each entity instance in a Part 21 file begins with a unique Entity ID and terminates with a semicolon ";". The Entity ID is a hash symbol "#" followed by an integer and has to be unique within the data exchange file. The Entity ID is followed by an equal symbol ("=") and the name of the entity that defines the instance. The names are always capitalized because EXPRESS is case-insensitive. The name of the instance is then followed by the values of the attributes listed between parentheses and separated by commas. The following is the excerpt of a STEP-NC ARM file.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION((‘A STEP-NC testing file’),’1’);
FILE_NAME(‘sample_part1.stp’,$,(‘AUMS’),(‘’),’Prototype Mill’,’’,’’);
FILE_SCHEMA((‘STEP-NC milling schema’));
ENDSEC;
DATA;
// Project and Workplan
#1=PROJECT(‘Contour’,#2,(#3));
#2=WORKPLAN(‘Work plan’,(#4),$,#5);
#3=WORKPIECE(‘Workpiece’,#6,0.01,$,$,#8,());
// Workingsteps
#4=MACHINING_WORKINGSTEP(‘Rough Contour’,#13,#16,#17);
#5=SETUP(‘main_setup’,#44,#48,(#51));
#6=MATERIAL(‘ST-50’,’Steel’,(#7));
#7=PROPERTY_PARAMETER(‘E=200000 N/mm^2’);
```

```
#8=BLOCK(‘Block’,#9,260.000,210.000,110.000);
// Geometric data
#9=AXIS2_PLACEMENT_3D(‘BLOCK’,#10,#11,#12);
…………
// Manufacturing features
#16=GENERAL_OUTSIDE_PROFILE(‘Profile’,#3,(#17),#18,#22,$,$,$,$,#23,$,$);
// Operation data
#17=SIDE_ROUGH_MILLING($,$,’Contour
profile’,#38,10.000,#39,#40,#43,$,$,$,20.000,5.000,0.000);
#18=AXIS2_PLACEMENT_3D(‘Position of contour’,#19,#20,#21);
#19=CARTESIAN_POINT(‘Position of contour’,(40.000,90.000,100.000));
#20=DIRECTION(‘’,(0.0,0.0,1.0));
#21=DIRECTION(‘’,(1.0,0.0,0.0));
#22=TOLERANCED_LENGTH_MEASURE(20.000,$,$,$);
#23=COMPOSITE_CURVE(‘Contour Profile’,(#24,#25,#56),.F.);
…………
// Tool data
#40=CUTTING_TOOL(‘Endmill 10mm’,#41,(),(50.000),50.000);
#41=TAPERED_ENDMILL(#42,3,.RIGHT.,.F.,$,$);
#42=TOOL_DIMENSION(10.000,$,$,$,$,$,$);
// Machining technology
#43=MILLING_TECHNOLOGY($,.TCP.,$,3.3333,$,0.10,.T.,.F.,.F.);
#44=AXIS2_PLACEMENT_3D(‘Reference point to Machine zero’,#45,#46,#47);
#45=CARTESIAN_POINT(‘’,(20.000,30.000,10.000));
…………
#56=COMPOSITE_CURVE_SEGMENT(.CONTINUOUS.,.T.,#57);
#57=POLYLINE(‘Second cut of the contour’,(#29,#30,#31,#32,#33,#27));
ENDSEC;
END-ISO-10303-21;
```

Data Access Implementation Methods

STEP Data Access Interface (SDAI) reduces the costs of managing integrated product data by making complex engineering applications portable across data implementations. Currently, four international standards have been established for SDAI:

- Standard data access interface (ISO 10303-22, 1998);
- C++ language binding to the standard data access interface (ISO 10303-23, 2000);
- C language binding of standard data access interface (ISO 10303-24, 2001); and
- Java Programming language binding to the standard data access interface with Internet/intranet extensions (ISO 10303-27, 2000).

Each standard defines a specific way of binding the EXPRESS data with a particular computer programming language. Binding is a terminology given to an algorithm for mapping constructs from the source language to the counter-parts of another. Generally speaking, the binding defined in SDAI can be classified into early and late binding. The difference between them is whether the EXRESS data dictionary is available to the software applications. There is no data dictionary in an early binding, whereas in a late binding, the EXPRESS schema definition is needed by late binding applications at run-time. For example, the SDAI for C++ language binding is a typical early binding approach; while the SDAI for C language binding is a late binding approach.

The early binding approach generates specific data structure according to the EXPRESS schemas and the programming language definitions. The entities defined in EXPRESS schemas are converted to C++ or Java classes. The inheritance properties in the EXPRESS schemas are also preserved in those classes. The advantage of an early binding is that the compiler of the programming language can perform additional type checking. But because of the complexities of EXPRESS schemas (for example, the average definitions in a STEP-NC AIM model is up to 200, and each definition in the early binding approach needs to have a corresponding class in the programming language), the initial preparation, compiling, and link of an early binding approach can be time-consuming.

The late binding approach, on the other hand, does not map EXPRESS entities into classes. It uses EXPRESS entity dictionaries for accessing data. Data values are found by querying those EXPRESS entity dictionaries. Only a few simple functions need to be defined in the late binding approach to get or set values. A late binding approach is suitable for a programming language that does not have strong type checking such as C language or an environment that may have multiple EXPRESS schemas (when EXPRESS schema changes, a late binding application can use a new dictionary without changing the application itself). A late binding is simpler than an early binding approach because there is no need to generate the corresponding classes. However, the lack of type checking destines that the late binding approach is not suitable for large systems.

A mixed binding approach may provide the advantages of an early binding (compile-time type checking and semantics as functions in a class) and late binding (simplicity). For example, a mixed binding takes advantage of the observation that applications rarely use all of the structures defined by an AP AIM (e.g., AP 238). The subset of structures that are used, called the working set, can be early-bound, while the rest of the AP is late-bound. All data is still available, but the application development process is simplified. The number of classes and files that are needed are reduced dramatically, resulting in quicker compilations, simpler source control and more rapid development.

XML Implementation Method (*Part 28, Edition 1*)

XML consists of different rules for defining semantic tags that breaks a document into parts and identifies the different parts of the document. Furthermore, it is a meta-markup language that defines a syntax in which other field-specific markup languages can be written (Harold, 2002). Essentially, XML defines a character-based document format. The following is a simple XML document defining a milling cutter:

```
<?xml version=”1.0"?>
   <MILLING_TOOL>
      MILL 18MM
   </MILLING_TOOL>
```

The first line is the XML declaration. It is usually made up of an attribute named “version” and its value “1.0”. Lines two to four define a “MILLING_TOOL” element with “<MILLING_TOOL>” as the start tag and “</MILLING_TOOL>” the end tag. “MILL 18MM” is the content, or in another word, the value of the element. XML is flexible because there is no restriction to those tag names. Hence, it is possible to assign more human-understandable tag names in an XML document, while computers just interpret an XML document according to a pre-defined formula. It is obvious that the use of meaningful tags can make an XML document human-understandable as well as computer-interpretable.

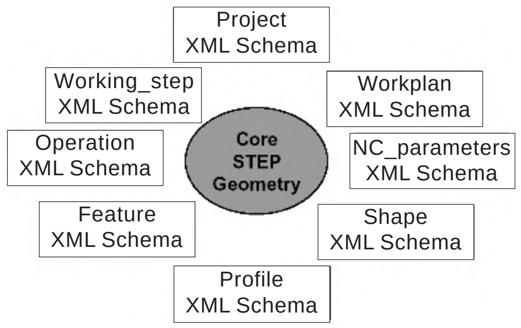
When representing EXPRESS schemas, Part 28, Edition 1 (ISO/CD TS 10303-28 [Edition 1], 2002) specifies an XML markup declaration set based on the syntax of the EXPRESS language. EXPRESS text representation of schemas is also supported. The markup declaration sets are intended as formal specifications for the appearance of markup in conforming XML documents. These declarations may appear as part of Document Type Definitions (DTDs) for such documents.

Like the method used in SDAI, STEP Part 28, Edition 1 (ISO/CD TS 10303-28 [Edition 1], 2002) defined two broad approaches for representation of data corresponding to an EXPRESS schema. One approach is to specify a single markup declaration set that is independent of the EXPRESS schema and can represent data of any schema. This approach is called XML late binding. The second approach is to specify the results of the generation of a markup declaration set that is dependent on the EXPRESS schema. This approach is called XML early binding. STEP Part 28, Edition 1, defines one late-binding approach and two early-binding approaches.

XML Implementation Method (*Part 28, Edition 2*)

It has soon become evident that the use of DTD syntax to specify mappings of EXPRESS to XML as prescribed in Part 28, Edition 1, results in a sub-optimal solution. Recognizing the limitations of the first edition such as those discussed in the previous section, ISO has begun to work on the second edition of Part 28 employing W3C XML schema. The Part 28, Edition 2, EXPRESS-to-XML-schema mapping and configuration language (ISO/TS 10303-28 [Edition 2], 2004) is still under development. The main theme of the new implementation method is its two-level method. At the lower level, CAD authoring systems can continue to read and write STEP data sets. The only difference on this level is that these data sets can now have an XML format to make them more compatible with the higher level. At the upper level, the data sets are modularized by inserting information from the mapping tables into the XML

Figure 4. Definitions generated by the new method



data to explain the meaning of each entity sequence. The new method can open up the definition of an Application Protocol into a series of interconnected XML schemas. As shown in Figure 4, each XML schema defines the representation required for one of the STEP-NC ARM objects.

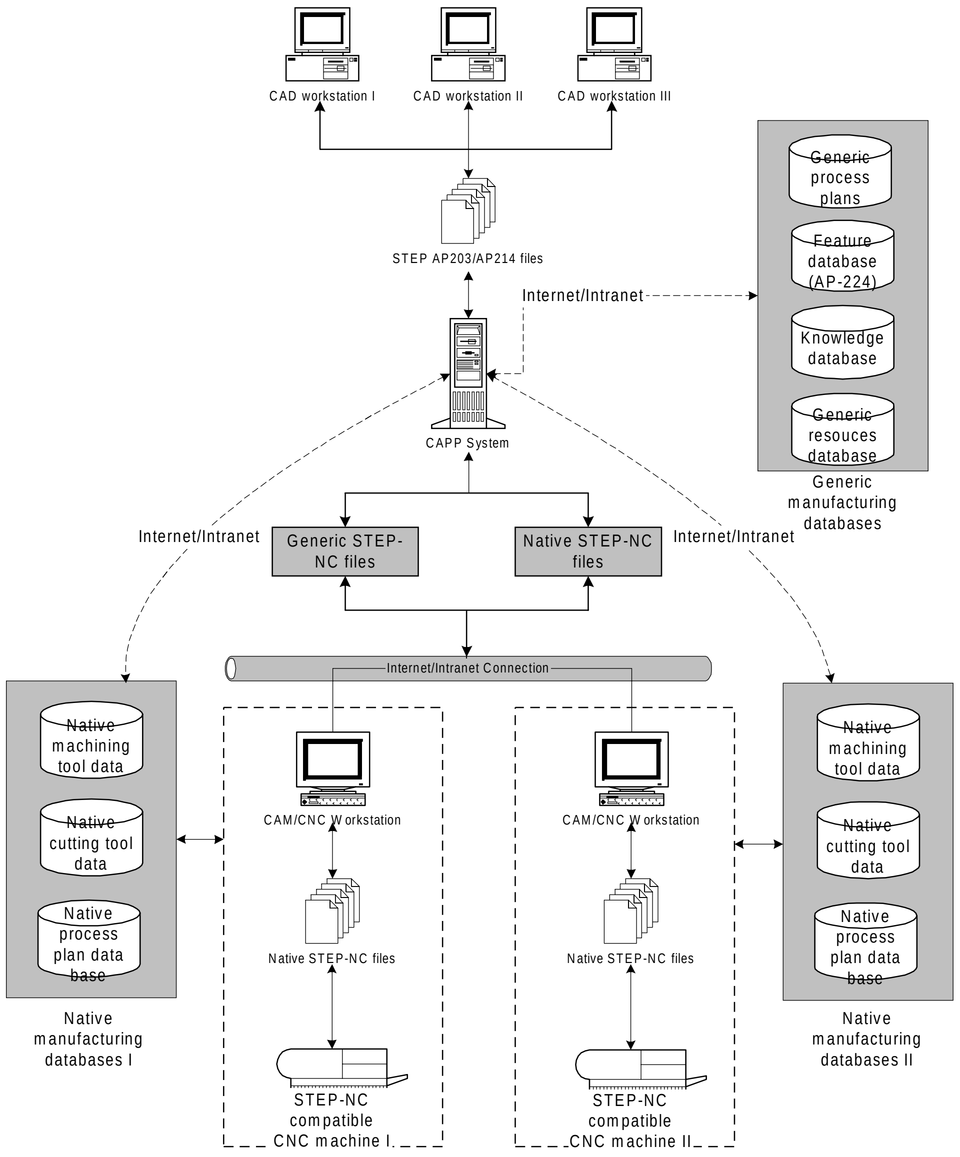
This method is implemented using two languages, a configuration language for describing how to map EXPRESS information into an XML defined form, and the existing STEP mapping table language converted into an XML form.

Case Study

STEP and EXPRESS languages have only been around for a short span of time. XML and other Web tools and languages are also relatively young. The first STEP-NC standard was published in 2003. The research work carried out in the Manufacturing Systems Laboratory of the University of Auckland aims to achieve a total STEP-compliant product development environment (Figure 5) (Xu et al., 2005). The system used STEP (Part 21 and Part 28) and STEP-NC (AP 238) standards to construct a universal data model.

In the design phase, STEP AP-203 or AP-214 is used as the neutral data format to exchange design data between different CAD systems or between CAD and CAPP systems. Two different manufacturing information databases (generic and native) co-exist to support CAPP data exchange. The generic manufacturing databases contain abstract information about machine tools and cutting tools of any kind. Hence, process plans generated using generic manufacturing resources cannot be executed directly at the shop-floor. This is because a STEP-NC-based process plan at this stage has only information

Figure 5. STEP-compliant collaborative manufacturing model


CAD workstation I
CAD workstation II
CAD workstation III
STEP AP203/AP214 files
Internet/Intranet
Generic process plans
Feature database (AP-224)
Knowledge database
Generic resouces database
Generic manufacturing databases
CAPP System
Internet/Intranet
Generic STEP-NC files
Native STEP-NC files
Internet/Intranet
Internet/Intranet Connection
Native machining tool data
Native cutting tool data
Native process plan data base
Native manufacturing databases I
CAM/CNC Workstation
Native STEP-NC files
STEP-NC compatible CNC machine I
CAM/CNC Workstation
Native STEP-NC files
STEP-NC compatible CNC machine II
Native machining tool data
Native cutting tool data
Native process plan data base
Native manufacturing databases II

about "what to do", that is, the tasks. Examples of what-to-do information include machining features and the description of requirements of machine tool(s) and cutting tool(s). At this stage, no information about selection and determination of specific machine tool(s) and cutting tool(s) is present in the STEP-NC program. Generic process plans are therefore machine-independent. The native manufacturing databases reflect the actual conditions of a shop-floor including existing machine tools and cutting tools that can be used for populating and optimizing a generic process plan and generating native process plans for final execution. To this end, a native manufacturing database can be considered as a "DNA" bank for all the available manufacturing facilities.

The basic element of a STEP-NC file is Workingstep instead of the cutting tool's positions. Workingsteps are built based on machining features. The system uses inputs from CAD systems, feature recognizers, CAPP algorithms, and manufacturing resources to generate a STEP-NC file. As the CAPP system is designed based on the three-tiered architecture, it has the ability to switch between different manufacturing databases to generate generic or native STEP-NC files to provide a maximal flexibility to support collaborative manufacturing. As shown in dashed lines in Figure 5, when the CAPP system is connected to a generic database, the output STEP-NC files will be universal and machine-tool independent. Under this condition, the CAM system can later populate and optimize a generic STEP-NC file based on the native manufacturing database on the shop-floor to obtain a suitable STEP-NC file for a specific CNC machine. When the CAPP system is directly connected to a native manufacturing database, it will be able to optimize the machining sequence, select machine tools and cutting tools at the process planning stage, and generate a STEP-NC file which can be directly used by a targeted CNC machine. Figure 6 shows the detailed information flow in the proposed system.

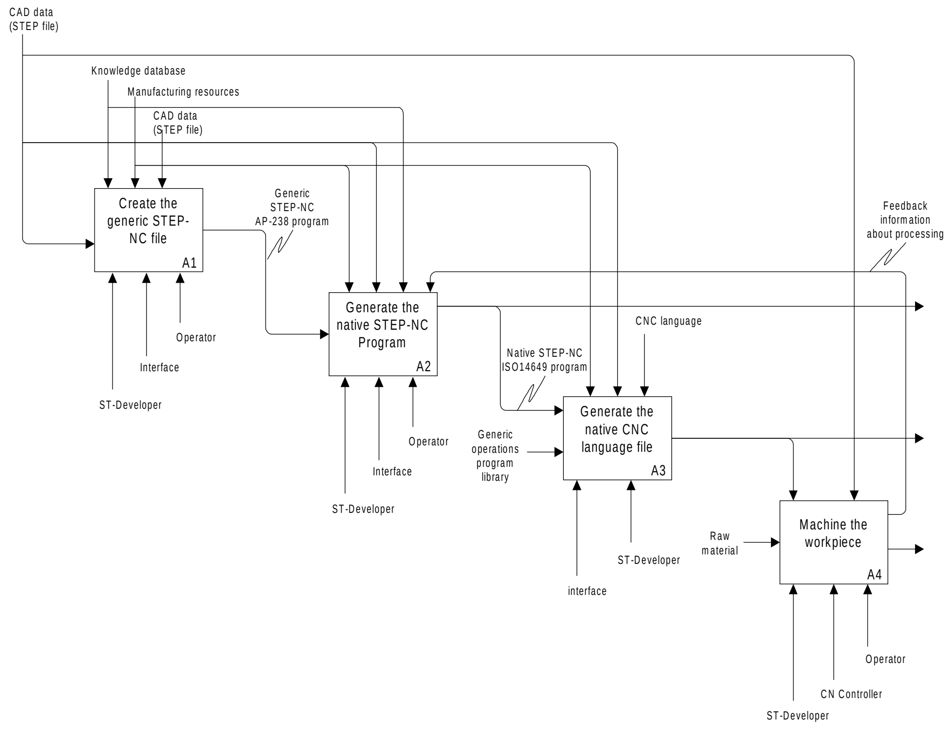
In this scenario, CAM systems are more likely to be integrated with STEP-NC-enabled CNC machines, or rather their controllers. The main functions of a CAM system are therefore to optimize the generic STEP-NC information and offer data connections to a CAPP system instead of calculating tool trajectories and generating CNC programs, which will be handled by the built-in functions of the STEP-NC controller.

System Model

The abstract model of the proposed STEP-compliant manufacturing system is illustrated in Figure 7. In order to support the collaborative manufacturing environment, the system is of a three-tiered network hierarchy.

The client tier is effectively a GUI, consisting of a set of applications and a Web browser to enable interactions between users and the system. The main functions of the client tier are to analyze the necessary interactions between users and the entire system, as well as to provide an effective way to realize the interactions using existing technologies.

Figure 6. IDEF0 diagram of the STEP-compliant collaborative manufacturing model



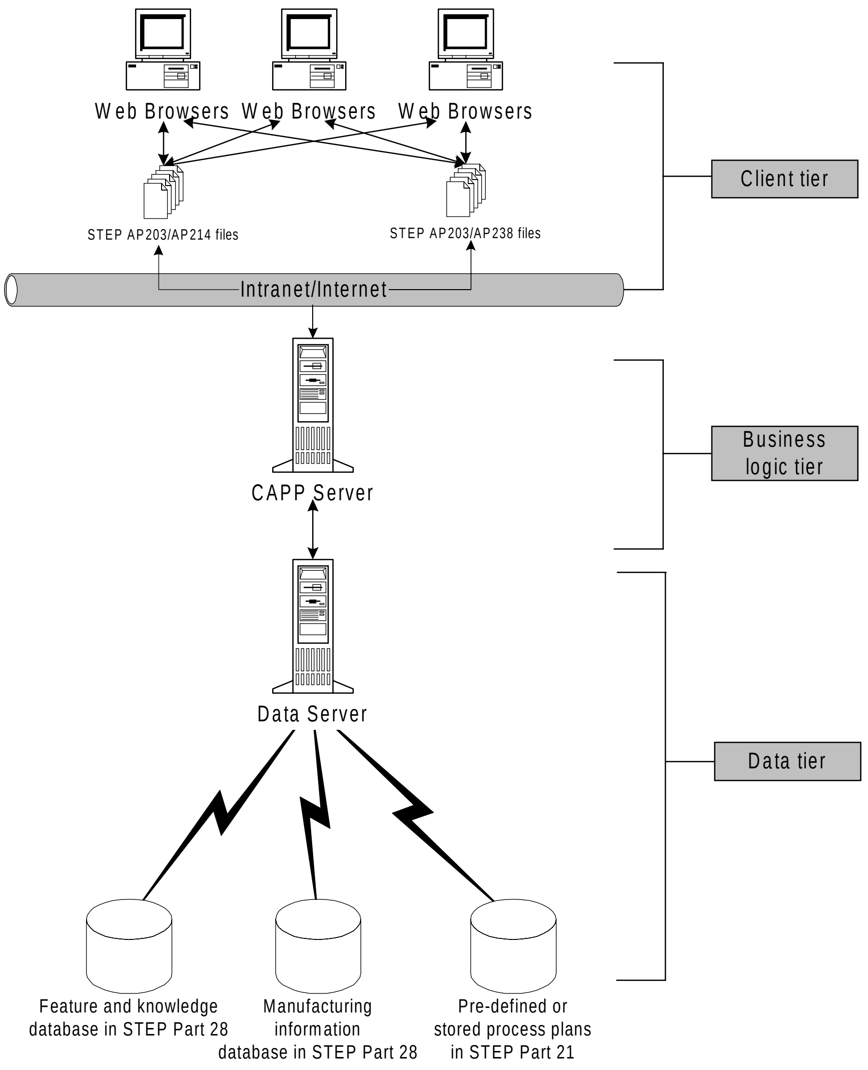
Business logic tier is the core of the proposed system; it acts as a CAPP server. The main functions in this tier are similar to those of a traditional CAPP system. Common CAPP tasks such as feature recognition, designation of machine tool/ cutting tool, and operation optimizations are carried out on the CAPP server.

Data tier supports the CAPP server. It represents generic or local shop-floor information pertaining to process planning. By switching between different data sources at the data tier, the CAPP system can better support collaborative manufacturing.

Client Tier: User Interface

The client tier directly interacts with users. As a STEP-compliant system, there are a number of different modules that are needed to provide the required functions in the client tier. They are:

Figure 7. Abstract system model of the STEP compliant manufacturing system



- A user interface that can view process plans through a specific query statement;
- A STEP physical file interpreter that interprets the STEP and STEP-NC structures;
- A GUI that displays 3-D models based on the geometrical information in a STEP file;

- A module that presents manufacturing plans in STEP-NC terms, that is, Workplans and Workingsteps;
- A module that presents, and allows the user to modify, manufacturing information such as features, machine tools, cutting tools, and tolerances in a Workingstep;
- A module that allows users to alter the sequence of Workingsteps and/or Workplans; and
- An XML interpreter that can interpret both the generic manufacturing information from the CAPP server and native manufacturing information from a database in XML format.

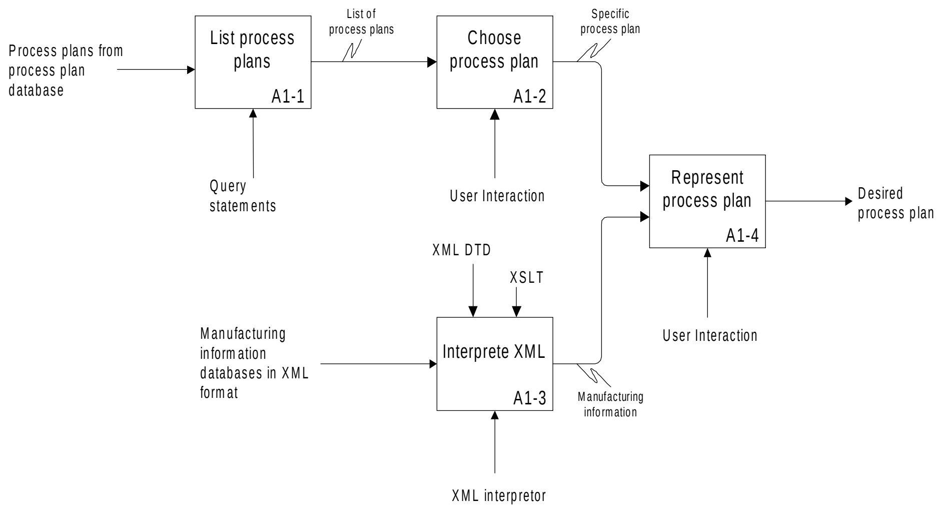
Figure 8 illustrates the information flow among different modules. The client tier starts with listing the existing process plans in a process plan database through a query statement. When a specific process plan is chosen, Workplans and the solid models in the plan can be presented to the user via GUI. At this stage, the XML interpreter provides the interpreted XML manufacturing information to the clients. XML DTD, XSLT, and XML schema and/or the controls for keeping the XML data are retrieved from the manufacturing databases in a desired manner. In doing so, the manufacturing information within the XML data can be easily used to modify the current process plan. In response to the requirements from the client tier, the most suitable framework in which those modules can be implemented is a Web browser.

Business Logic Tier: CAPP Server

Two different types of business logic tiers are represented as shown in Figures 9 and 10. The difference between them is the way in which the Workingstep optimizer works with the process planner. In Model I, the CAPP server is able to access different native manufacturing resources to generate different native process plans, hence an “integrated” scenario. Model II, on the other hand, generates the generic and native process plans in tandem. The latter is likely to be generated at the shop-floor. Therefore, it supports an “interfacing” scenario.

In both models, feature recognition is the first task. The inputs to this module are data files conforming to ISO 10303 AP-203 or AP-214. The controls include ISO 10303 AP-203, AP-214, AP-224 and a feature database compatible with AP-224. AP-203 and AP-214 are used to describe the pure

Figure 8. Information flow in the client tier



geometrical information of a part, whereas AP-224 is used to describe machining features in a process plan. The goal of using AP-224 as a control here is to provide a universal and STEP-compliant machining feature library in place of different proprietary feature libraries from different system vendors. AP 224 is also used to define STEP-NC machining features. The feature recognition module may have two different modes, automatic and manual.

The main function of the process planning module in Model I is to assign manufacturing resources to the features generated by the feature recognizer. The controls of the process-planning module include ISO 10303 AP-238 (STEP-NC AIM), Workingstep optimizing algorithms, and native manufacturing resources databases. The native manufacturing resources databases conform to ISO 14649, Part 111 (tools for milling) and Part 121 (tools for turning). As the manufacturing information is stored in STEP Part 28, Edition 1, XML format, the connection between the CAPP server and the native resources databases is via the Internet. If the native shop-floor manufacturing resources are connected, the process planning module can directly assign specific manufacturing resources such as machine tools and cutting tools to each feature for creation of Workingsteps. Workingsteps are optimized and properly sequenced to generate a process plan which can be executed immediately at the shop-floor.

In Model II, the Workingstep optimization mechanism is separated from process planning and forms a new module. This may be due to the fact that the native manufacturing resources are still pending. In this case, the outputs of the

Figure 9. CAPP server Model I (Integrated model)

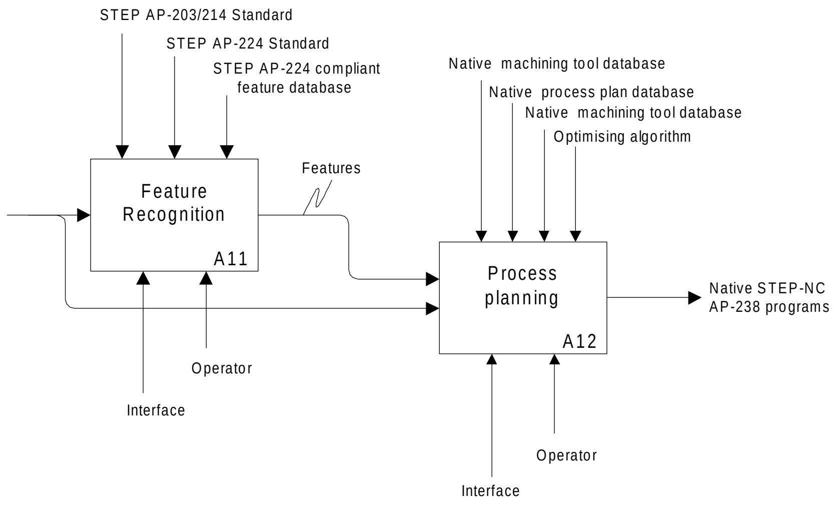
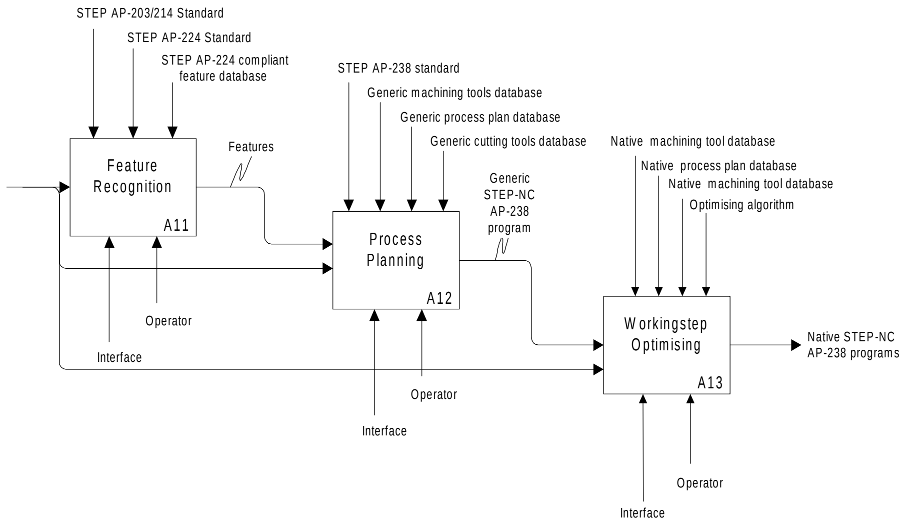

STEP AP-203/214 Standard
STEP AP-224 Standard
STEP AP-224 compliant feature database
Native machining tool database
Native process plan database
Native machining tool database
Optimising algorithm
Features
Feature Recognition
A11
Process planning
A12
Native STEP-NC AP-238 programs
Operator
Interface
Operator
Interface

Figure 10. CAPP server Model II (Interfacing model)


STEP AP-203/214 Standard
STEP AP-224 Standard
STEP AP-224 compliant feature database
STEP AP-238 standard
Generic machining tools database
Generic process plan database
Generic cutting tools database
Native machining tool database
Native process plan database
Native machining tool database
Optimising algorithm
Features
Feature Recognition
A11
Generic STEP-NC AP-238 program
Process Planning
A12
Workingstep Optimising
A13
Native STEP-NC AP-238 programs
Operator
Interface
Operator
Interface
Operator
Interface

process-planning module are generic process plans. They will be "populated" in the Workingstep optimization module with the information from an identified native manufacturing resource to give a native process plan. Essentially, generic process planning is a process of "enriching" the machining features, represented as the AP-224 format in this case, with the necessary syntax information to form entities defined by STEP-NC, for example, Workplans and Workingsteps.

Some preliminary decisions such as Workingstep ordering and set-up planning will be mainly based on the feature information and the information from a generic manufacturing database. Note that whatever decision is reached at this stage, changes can be easily made once the native manufacturing information becomes available. Many information slots in an STEP-NC file will remain empty or carry default values at this stage. This is intended by the standard, that is, STEP-NC has the ability to just model what-to-do information.

To recap, both Models I and II seem to perform similar functions. There is, however, a fundamental difference between them. In Model I, the output from the CAPP server is a specific, "how-to-do" process plan, which can be directly used by a specific manufacturing facility. This "how-to-do" information may not be used by other manufacturing facilities. In Model II, there is an intermediate result from the process-planning module, that is, a generic process plan. This generic process plan contains "what-to-do" instead of "how-to-do" information. It is, therefore, machine tool independent. The "what-to-do" information maintains its generic nature until the last moment when the CAM system of the chosen machine tool populates it with the native manufacturing information so as to generate a specific (how-to-do) process plan. Therefore, Model II possesses the required flexibility and portability to support collaborative manufacturing.

Data Tier: Data Model

The databases in the data tier are constructed by applying the Part 28, Edition 1, rules to the EXPRESS schemas. For example, the feature database is constructed by applying the Part 28 rules to ISO 10303 AP-224 schemas, and the cutting tool database is constructed by applying the Part 28 rules to ISO 14649, Part 111 and Part 121 schemas. The following XML codes from the cutting tool database define a center drill. Figure 11 shows such information in a Web browser.

```
<STEP-XML xmlns:ceb="urn:iso10303-28:ceb">
<cutting_tool ceb:id="66" ceb:copies="4">
  <id>CENTER_DRILL_5MM</id>
  <its_tool_body>
```

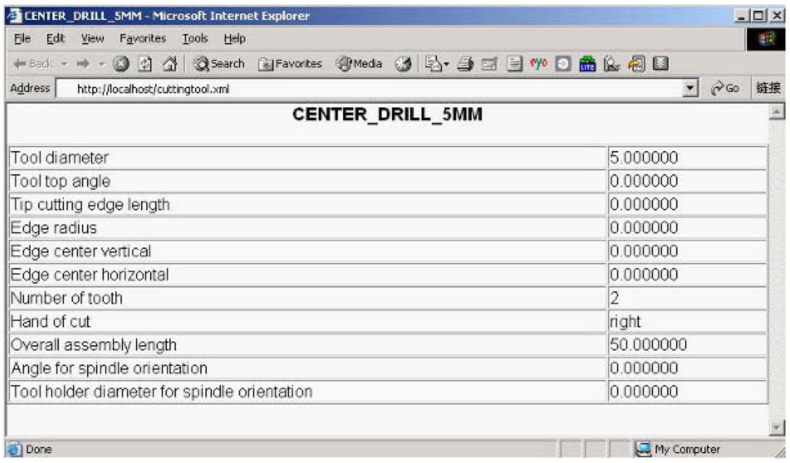
```
    <center_drill ceb:id=”65" ceb:copies=”4">
        <dimension>
            <tool_dimension ceb:id=”59" ceb:copies=”4">
                <diameter>5.000000</diameter>
                <tool_top_angle>0.000000</tool_top_angle>
                <tipcutting_edge_length>0.000000</tipcutting_edge_length>
                <edge_radius>0.000000</edge_radius>
                <edge_center_vertical>0.000000</edge_center_vertical>
                <edge_center_horizontal>0.000000</edge_center_horizontal>
            </tool_dimension>
        </dimension>
        <number_of_teeth>2</number_of_teeth>
        <hand_of_cut>
            <hand>right</hand>
        </hand_of_cut>
    </center_drill>
  </its_tool_body>
  <overall_assembly_length>50.000000</overall_assembly_length>
  <angle_for_spindle_orientation>0.000000</angle_for_spindle_orientation>
    <tool_holder_diameter_for_spindle_orientation>0.000000</
tool_holder_diameter_for_spindle_orientation>
</cutting_tool>
</STEP-XML>
```

The RDBMS (Relational DataBase Management System) is used in the data tier. In order to keep the original structure within an XML document, XML documents are stored as a whole in the RDBMS or as an external file outside the RDBMS. Once such a database is constructed, the information required by the CAPP server can be carried by the XML documents and transferred via the Internet. The XML documents are readily viewable in Web browsers and/or interpreted by a STEP-XML interpreter in the CAPP server to obtain specific manufacturing information.

Framework Development

In the interest of space, only two types of development work are discussed. The objectives are to (1) enable a process planner to view and manually edit the

Figure 11. Cutting tool information in XML format shown in a Web browser


CENTER_DRILL_5MM - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/cuttingtool.xml
Done
My Computer

CENTER_DRILL_5MM

Tool diameter	5.000000
Tool top angle	0.000000
Tip cutting edge length	0.000000
Edge radius	0.000000
Edge center vertical	0.000000
Edge center horizontal	0.000000
Number of tooth	2
Hand of cut	right
Overall assembly length	50.000000
Angle for spindle orientation	0.000000
Tool holder diameter for spindle orientation	0.000000

existing process plans/STEP-NC programs in STEP AP-238, Part 21 file format using the manufacturing resources provided in STEP Part 28 XML format and (2) enable access to, and modification of, manufacturing databases across the Internet.

Client Tier Implementation

The prototype of the client tier has been developed and implemented under the Microsoft Windows® environment. All client applications are unified within Microsoft Internet Explorer® 6.0. A set of development tools and technologies are used:

- Microsoft Visual C++® 6.0 and Microsoft Foundation Classes (MFC®);
- ST-Developer® (ST-Developer, 2005) and STIX® (STIX, 2005);
- OpenGL® (Open Graphics Library); and
- ActiveX® technology.

ST-Developer® is a software development package for developing and working with STEP applications. It offers libraries for reading, writing, processing, and checking STEP data of Part 21 formats. It also provides EXPRESS early binding with C++ classes. These features help to develop additional STEP-compatible applications. Applications written in Visual C++® can offer functions to read and write STEP Part 21 files, as well as create, delete, traverse, and change any EXPRESS-defined data sets compiled as objects in C++ style. STIX® is a STEP

IndeX library for STEP AP-238 from the same company. It contains a C++ library which provides useful functions to process manufacturing data in STEP AP-238 format. Therefore, STIX® simplifies implementation and processing of STEP AP-238 information in programs written in Microsoft Visual C++®.

Figure 12 shows the client user interface. A STEP AP-238 file in STEP Part 21 format is represented in an ActiveX control. The left panel is a tree structure listing all the Workplans and Workingsteps in the STEP-NC file, whereas the right panel represents the geometrical model of the finished part. Client users can modify the information such as cutting tools, tolerance and manufacturing technologies.

Business Logic Tier Implementation

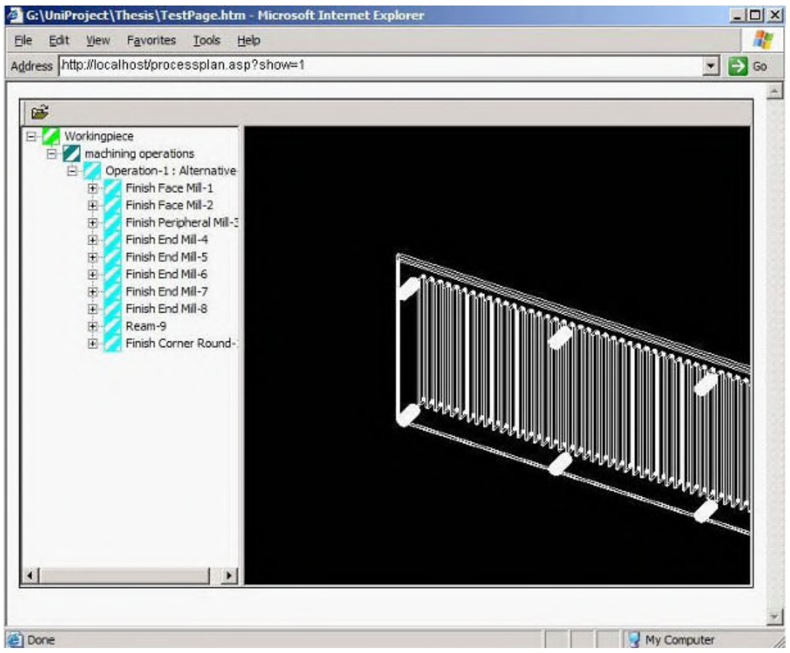
In the business logical tier, a Web server has been constructed. It utilizes Internet Information Server (IIS®), Visual Basic® IIS application, and Active Sever Pages (ASP®) to generate dynamic Web pages for different client users. The Web server separates users into two groups: the process planners and the database administrators.

A process planner can access a list of existing process plans generated by the Web server. The process planner can then choose the desired process plan to modify. Each hyperlink leads to one existing process plan represented in an AP-238 Part 21 physical file and stored in the data tier. Once a process plan is chosen, the process plan file will be downloaded into the ActiveX control and represented in a Web browser for modifications. New process plans can be uploaded back to the data server after modifications.

For database administrators, the tier functions differently. The STEP Part 28 XML documents that contain manufacturing information are parsed by the Web server before transmission. Hence, database administrators can focus on the manufacturing information in the XML file. The generated dynamic Web page for database administration is illustrated in Figure 13.

The left frame in the interface is a tree menu structure presenting the structure of an XML document. The upper-right frame represents the extracted manufacturing information. The lower-right frame is the editing area, in which, the "Name" section represents tags in an XML document, and the "Value" section refers to the detailed manufacturing information in the XML tags. Once the modifications are submitted, the Web server generates a new XML document based on the new values. The corresponding values in the manufacturing databases are also updated to reflect the changes.

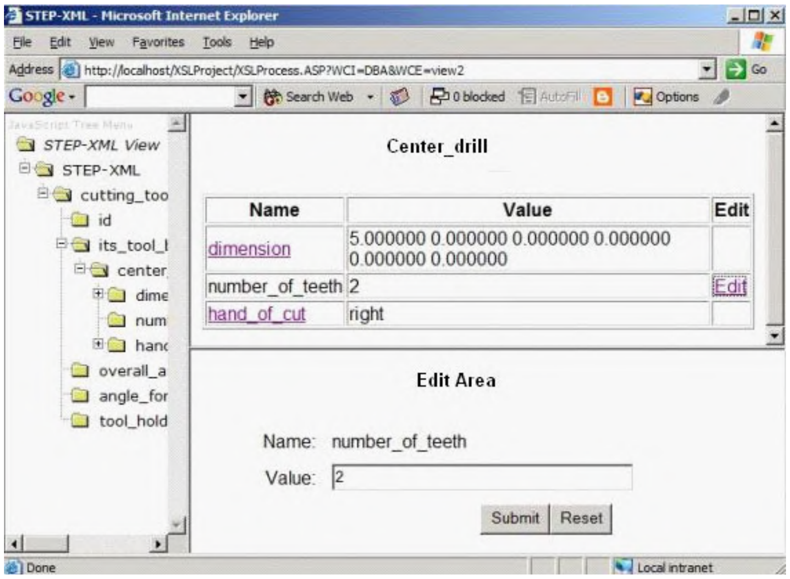
Figure 12. Prototype of client tier interface



Data Tier Implementation

The data tier is currently implemented using Microsoft Access 2000®. The manufacturing information is stored as XML files in the operating system on which the data tier is implemented. The complete file paths of those XML files are stored. Some important attributes such as tool diameter, tool length, and tool name by which the manufacturing information can be identified are also extracted and stored in the tables to enable more specific query statements. The main benefit of such an implementation method is that the database is easy to construct, and both the original XML structure and the flexibility provided by SQL can still be preserved. For example, a simple query statement, "*Select filepath from drilltools where tooldiameter = 5.0 and overallasslength = 50.0*", will return the file path for the existing drilling tools with a diameter 5.0 and overall assembly length 50.0. The CAPP server can then extract the detailed manufacturing information from the XML documents according to the file paths. Figure 14 illustrates an implemented table which describes tools for drilling.

Figure 13. Interface for database administration



Center_drill

Name	Value	Edit
dimension	5.000000 0.000000 0.000000 0.000000 0.000000 0.000000	
number_of_teeth	2	Edit
hand_of_cut	right	

Edit Area

Name: number_of_teeth

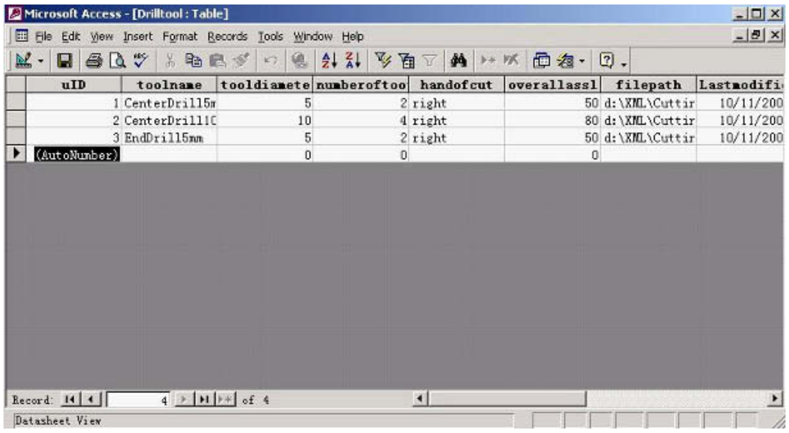
Value: 2

Conclusion

First published in 1994, the initial scope of STEP covered geometry, product structure, and technical drawings. Since then, the standard has been extensively implemented on major commercial projects in the aerospace, automotive, and other industries. Typical implementations are the use of STEP to combine the information on shape and other characteristics of individual parts with assembly structures to form a single integrated representation of a complex assembly or product. This information is gathered from a range of application systems and consolidated into a STEP file which can be transferred to other companies and unloaded into their corresponding systems. The advantage from combining this data is that it guarantees consistency for information deliveries, and avoids the administrative cost of ensuring that data is consistent between multiple systems.

Now STEP is on the verge of a new success with the release of a specification for defining the data input to CNC controllers — STEP-NC. Currently, these controllers are driven by vector codes developed in the 1960s. STEP-NC provides a unique NC program format for CAPP, CAM, and NC; avoids post processing; and entails a truly exchangeable format. Operators can now be supported at the machine tool level by complete information containing

Figure 14. An implemented table that describes tools for drilling


Microsoft Access - [Drilltool : Table]
File Edit View Insert Format Records Tools Window Help

uID	toolname	tooldiamete	numberoftoo	handofcut	overallassl	filepath	Lastmodifi
1	CenterDrill5m	5	2	right	50	d:\XML\Cuttir	10/11/200
2	CenterDrill1C	10	4	right	80	d:\XML\Cuttir	10/11/200
3	EndDrill5mm	5	2	right	50	d:\XML\Cuttir	10/11/200
(AutoNumber)		0	0		0		

Record: 4 of 4
Datasheet View

understandable geometry (machining features); task oriented operations (Workingsteps and Workplans); and strategies and tool definitions. CNC machines implementing STEP-NC can have a more open and adaptable architecture, making it easier to integrate with other manufacturing facilities such as workpiece handling devices. Relatively high magnitude of funding strength over a number of STEP-NC-related projects during a short span of time sufficiently demonstrated the importance of the STEP-NC-related development work. Participation of, and collaboration among, a wide variety of organizations such as end users, academic and research institutions, and manufacturers of CAM systems, controls and machine tools, echoes the significance and relevance of this work, in particular from the industry perspective. STEP-NC comes in two "forms", Application Requirements Model (ISO 14649) and Application Interpreted Model (ISO 1030 AP-238). AP-238 provides an information view of the data, whereas ISO 14649 provides a functional view of the data.

STEP-NC can also support distributed and collaborative manufacturing scenarios. This is demonstrated by the case study. The framework in the case study supports a bi-directional information flow throughout the design and manufacturing chain. Design information in its entirety is available in the manufacturing model. Manufacturing information is feature-based and task-oriented. In a collaborative manufacturing environment, designers and manufacturers are often geographically dispersed. Therefore, the framework adopts a three-tiered, Web-based network architecture to provide an open structure for the system. This architecture provides convenient ways in exchanging design and manufacturing data in STEP Part 21 and/or Part 28 file format through the Internet. The client user interface is implemented within a Web browser so that the implementation and maintenance costs can be reduced. Manufacturing information databases implemented in STEP Part 28 XML format enables the CAPP server to switch between geographically-dispersed shop-floor resources through Internet connections to realize collaborative manufacturing.

References

Allen, R. D., Newman, S. T., Harding, J. A., & Rosso, R. S. U., Jr. (2003, June 9-11). The design of a STEP-NC compliant agent based CAD/CAM system. In *Proceedings of the 13th International Conference on Flexible Automation and Intelligent Manufacturing Conference (FAIM2003)*, Tampa, FL (pp. 530-540). Kinco's Inc.

Anonymous. (2003). STEP-NC demonstrated at JPL. *Manufacturing Engineering, 130*(3), 34-35.

Feeney, A. B., Kramer, T., Proctor, F., Hardwick, M., & Loffredo, D. (2003, March). STEP-NC implementation — ARM or AIM? (White Paper). In *ISO T2 STEP-Manufacturing Meeting*, San Diego. Geneva, Switzerland: ISO TC184/SC4.

Gallaher, M. P., O'Connor, A. C., & Phelps, T. (2002). Economic impact assessment of international Standard for the Exchange of Product model data (STEP) in transportation equipment industries. *NIST Planning Report 02-5*. Retrieved October 14, 2005, from http://www.mel.nist.gov/msid/sima/step_economic_impact.pdf.

Hardwick, M. (2004). *A modular XML implementation method for STEP*. Troy, NY: STEP Tools, Inc.

Harold, E. R. (2002). *XML bible, Gold edition*. Hungry Minds, Inc. Wiley.

IMS STEP-NC Consortium. (2003). *STEP-compliant data interface for numerical controls (STEP-NC)* (Tech. Rep. No. 3 of IMS Project 97006). Erlangen, Germany: MIS Germany.

ISO 10303-1. (1994). Industrial automation systems — Product data representation and exchange. Part 1: Overview and fundamental principles.

ISO 10303-11. (1994). Industrial automation systems and integration — Product data representation and exchange. Part 11: Description methods: The EXPRESS language reference manual.

ISO 10303-203. (1994). Industrial automation systems and integration — Product data representation and exchange. Part 203: Application protocol: Configuration controlled 3D designs of mechanical parts and assemblies.

ISO 10303-21. (1994). Industrial automation systems and integration — Product data representation and exchange. Part 21: Implementation methods: Clear text encoding of the exchange structure.

ISO 10303-214. (1994). Industrial automation systems and integration — Product data representation and exchange. Part 214: Application protocol: Core data for automotive mechanical design processes.

ISO 10303-22. (1998). Industrial automation systems and integration — Product data representation and exchange. Part 22: Implementation methods: Standard data access interface.

ISO 10303-23. (2000). Industrial automation systems and integration — Product data representation and exchange. Part 23: C++ language binding to the standard data access interface.

ISO 10303-27. (2000). Industrial automation systems and integration — Product data representation and exchange. Part 27: JavaÔ programming language binding to the standard data access interface with Internet/Intranet extensions.

ISO 10303-224. (2001). Industrial automation systems and integration — Product data representation and exchange. Part 224: Application protocol: Mechanical product definition for process plans using machining features.

ISO 10303-24. (2001). Industrial automation systems and integration — Product data representation and exchange. Part 24: C language binding of standard data access interface.

ISO 6983-1. (1982). Numerical control of machines — Program format and definition of address words. Part 1: Data format for positioning, line motion and contouring control systems.

ISO/CD TS 10303-28 (Edition 1). (2002). Product data representation and exchange: Implementation methods: EXPRESS to XML binding, Draft Technical Specification, ISO TC184/SC4/WG11 N169, 2002-02-14.

ISO/DIS 10303-238. (2003). Industrial automation systems and integration — Product data representation and exchange. Part 238: Application protocols: Application interpreted model for computerized numerical controllers.

ISO/TS 10303-28 (Edition 2). (2004). ISO ISO/WD 10303-28. Product data representation and exchange: Implementation methods: XML schema-governed representation of EXPRESS schema governed data, TC184/SC4/WG11 N223, 2004-02-17.

ISO 14649-1. (2003). Data model for computerized numerical controllers. Part 1: Overview and fundamental principles.

ISO 14649-10. (2003). Data model for computerized numerical controllers. Part 10: General process data.

ISO 14649-11. (2003). Data model for computerized numerical controllers. Part 11: Process data for milling.

ISO 14649-111. (2003). Data model for computerized numerical controllers. Part 111: Tools for milling.

ISO 14649-12. (2003). Data model for computerized numerical controllers. Part 12: Process data for turning.

ISO 14649-121. (2003). Data model for computerized numerical controllers. Part 121: Tools for turning.

LSC Group, & R. P. Management. (2002). *Driving down the cost of spares provisioning* (RAMP White Paper). LSC Group, Warship Support Agency.

Maeder, W., Nguyen, V. K., Richard, J., & Stark, J. (2002, September). Standardization of the manufacturing process: The IMS STEP-NC project. In *Proceedings of the IPLnet (National Network of Competence on Integrated Production and Logistics) Workshop 5.5/1-3*, Saas Fee, Switzerland.

Newman, S. T., Allen, R. D., & Rosso, R. S. U., Jr. (2003). CAD/CAM solutions for STEP-compliant CNC manufacture. *International Journal of Computer Integrated Manufacturing, 16*(7-8), 590-597.

PDES, Inc. (2005). *STEP success stories*. Retrieved October 2005, from http://pdesinc.aticorp.org/success_stories.html

Research Triangle Institute. (1999). Interoperability cost analysis of the U.S automotive supply chain: Final report. Research Triangle Park, NC.

Roberto, S. U., Rosso, J., Allen, R. D., & Newman, S. T. (2002). Future issues for CAD/CAM and intelligent CNC manufacture. In *Proceedings of the 19th International Manufacturing Conference — IMC-19/ 2002*, Belfast, Queen's University.

ST-Developer for CAD/CAE. (2005). Retrieved October 2005, from http:// www.steptools.com/products/stdev/

STIX. (2005). Retrieved October 2005, http://www.steptools.com/stix/

ST-Machine for CAM/CNC. (2005). Retrieved October 2005, http:// www.steptools.com/products/stmachine/

Storr, A., & Heusinger, S. (2002). STEP-NC — Grundlage Einer CAD/NC-Prozesskette Das STEP-NC-Prozessmodell für die Drehbearbeitung.

Storr, A., Pritschow, G., Heusinger, S., & Azotov, A. (2002). Arbeitsschrittplanung Beim Drehen mit STEP-NC Planungsmethoden zur Anwenderunterst¸tzung. ZWF Jahrg. 97 (2002) 7-8. Carl Hanser Verlag, München, Germany.

Suh, S. H., & Cheon, S. U. (2002a). A framework for an intelligent CNC and data model. *Advanced Manufacturing Technology, 19*, 727-735.

Suh, S. H., Cho, J. H., & Hong, H. -D. (2002b). On the architecture of intelligent STEP-compliant CNC. *Computer Integrated Manufacturing, 15*(2), 168-177.

Suh, S. H., Cho, J. H., Lee, B. E., Chung, D. H., Cheon, S. U., & Hong, H. D. (2002c). Developing an integrated STEP-compliant CNC prototype. *Journal of Manufacturing Systems, 21*(5), 350-362

Suh, S. H., Lee, B. E., Chung, D. H., & Cheon, S. U. (2003). Architecture and implementation of a shop-floor programming system for STEP-compliant CNC. *Computer-Aided Design, 35*, 1069-1083.

Wolf, J. (2003, March). Requirements in NC machining and use cases for STEP-NC — Analysis of ISO 14649 (ARM) and AP 238 (AIM) (White Paper). *ISO T24 STEP-Manufacturing Meeting*, San Diego, CA.

Xu, X. (2004, November 13-19). Development of a G-code free, STEP-compliant CNC lathe. In *Proceedings of the 2004 International Mechanical Engineering Congress and Exposition (IMECE), 2004 ASME Winter Conference, IMECE2004-60346, CIE-2* (pp. 1-5), Anaheim, CA.

Xu, X., & He, Q. (2004). Striving for a total integration of CAD, CAPP, CAM and CNC. *Robotics and Computer Integrated Manufacturing, 20*, 101-109.

Xu, X., & Mao, J. (2004, March 25-27). A STEP-compliant collaborative product development system. In *Proceedings of the 33rd International Conference on Computers and Industrial Engineering, CIE598*, Jeju, Korea.

Xu, X., Wang, H., Mao, J., Newman, S. T., Kramer, T. R., Proctor, F. M. et al. (2005, Sept). Step-compliant NC research: The search for intelligent CAD/CAPP/CAM/CNC integration. *International Journal of Production Research, 43*(1), 3703-3743.

Chapter VI

Semantic-Based Dynamic Enterprise Information Integration

Jun Yuan, The Boeing Company, USA

Abstract

Decision support systems or mission control systems for network-centric operations typically need input from multiple heterogeneous information sources. While the number and size of information sources increase, information integration and its semantic interoperability are becoming a growing concern and a major challenge to information management. In this chapter, we will share our experience of enabling semantic-based dynamic information integration across multiple heterogeneous information sources. While data is physically stored in existing/legacy data systems across the networks, the information is integrated based upon its semantic meanings. Informally, it sounds like a virtual data warehousing technique without any physical data conversion required in advance. Ontology is used to describe the semantics of global information content, and semantic enhancement is achieved by mapping the local metadata onto the ontology. For better system reliability, a unique mechanism is introduced to perform appropriate adjustments upon detecting environmental changes.

Introduction

Information Integration has been a high priority in business for many years. For building applications such as enterprise-wide decision support systems or mission control systems in a network-centric environment, many companies have actually been struggling against integration of legacy/existing data systems for some time. With the recent explosion of Internet and Web-based resources, information integration and its semantic interoperability are becoming an even greater concern.

In this chapter, we propose a framework that enables dynamic information integration across multiple heterogeneous information sources, with special attention to semantic interoperability and adaptations to a dynamic environment. While data is physically stored in existing/legacy data systems across the networks, the information is integrated based upon its semantic equivalence. Ontology is used to explicitly describe the semantics of global information content. Such ontology is independent of any particular information sources, but only based upon domain experts' knowledge. Individual systems may have their own intended models, and there are likely various types of semantic heterogeneity in between those models. Semantic heterogeneity resolution methods are provided by mapping those local models onto the ontology with a unique mapping mechanism associated with the proposed framework. Our approach is leveraging with the state-of-the-art Semantic Web standards. For instance, the domain ontology as well as the mapping knowledge can be exported into RDF or OWL documents.

In our framework, each participating information source assumes full local autonomy. All existing software applications that were built on top of those information resources will remain to be functioning the same way as they did previously. There is no need to re-develop any of these pre-existing applications. The integration itself is thus a purely incremental enhancement.

A single information access point is provided. Users are able to access information as if it were a centralized information repository. More interestingly, it is a semantic information access point, which allows users to query against ontology, for example, concepts and relationships, directly. Users are thus able to retrieve information from multiple heterogeneous data systems without having to understand lower-level details such as physical distribution, semantic heterogeneity, and logical conflicts among information sources. In addition, query answers are also presented by an instantiation of the ontology, and the semantics can be easily captured and understood by both human beings and computer systems.

Background

In the past decade, many researchers, from both academia and industry, have conducted extensive research on information or data integration. The information sources involved in information integration efforts are inherently heterogeneous on many aspects. Different sources may use different data models to describe the data contents, which might be a relational model, a hierarchical model, a network model, an XML schema/DTD, a piece of ontology, or even a free-text layout. A specific data model usually has its own implied capabilities and limitations, in terms of power of expressiveness, query languages, processing capacities, information qualities, and so forth. Such capabilities or limitations may differ greatly from one to another. Even for systems with the same data model, the detailed logical data structures for information with same or similar semantics may differ significantly from one to another, mainly because of the factors such as: preferences on how data should be physically organized; levels of data generalization/aggregation; and different naming strategies.

Data warehousing technology (Jarke, Lenzerini, & Vassiliou, 1999) is one of the solutions to integrate multiple data systems together. Overviews of data warehousing as well as OLAP technology can be found in some articles (Chaudhuri & Dayal, 1997; Widom, 1995). With the support of commercial data warehousing products, there have already been many successful stories of building enterprise-wide data warehouses to perform various information management tasks (Bontempo & Zagelow, 1998; Gardner, 1998; Scott, 1998; Sutter, 1998). In general, it is to create a separate, physical data repository that contains data from multiple sources. It has been used to resolve problems caused by heterogeneous systems. Environmental variation can be substantially reduced through the pre-conversion of data from its source into a homogeneous data warehouse representation. A single hardware platform and DBMS are chosen, generally using a common set of standards. The data representation and format issues are similarly resolved, although often with more difficulty. Resolution of representational disparities is typically based on varying levels of schema normalization or dimensional modeling, depending on the scope and intent of the data warehouse design goals. The format of the data is resolved through transformation of data into consistent data types, units of measure, and translation of code sets into a consistent form during the load processes. Resolution of semantic heterogeneities is currently addressed only in an *ad hoc* fashion, usually at access time. In conclusion, the data warehouse approach does not eliminate the need for resolving heterogeneity; it merely splits resolution into a pre-conversion during the loading of data copy, and then provides conversion as needed during data access. In many cases, such conversions are embedded in code and are not persistently recorded as a maintainable piece of metadata.

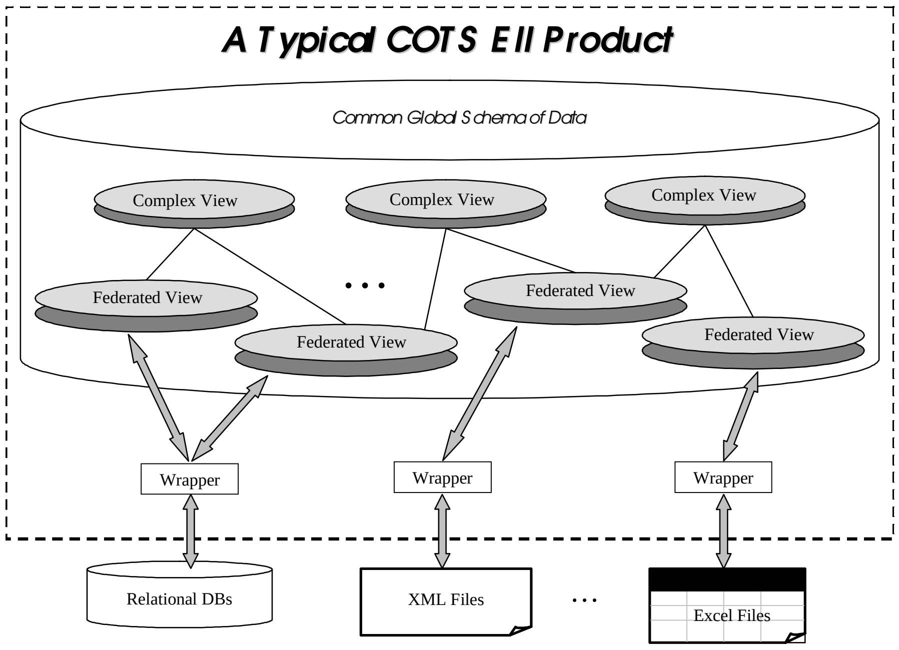
Resource requirements for creation and maintenance of a data warehouse can be substantial. A data warehouse maintains copies of the data, that is, duplicated sets of data, from multiple systems. Storage requirements easily double those of the participating data systems, and the ultimate capacity of the warehouse provides a fixed limit on the scope of the data to be warehoused. A new team of personnel is required to maintain the new environment. Data integration is a prerequisite. Restructuring and reformatting of data already residing in the data warehouse must be performed at any time when an extension to the design occurs, as does restructuring of existing loading procedures. While the number of sources feeding the data warehouse grows, scheduling of updates becomes increasingly difficult. To ensure integrity of the data, proper cross-correlations in the data warehouse must be synchronized, forcing synchronization of the loading processes. If there are more than four or five systems supplying information, resolution of these dependencies can be arduous, time-consuming, and thus expensive.

An alternative approach that emerged recently is the so-called Enterprise Information Integration (EII). EII aims at providing a unified access point of data from disparate systems. The data should be accessible in "real time," meaning that the information consumer should be accessing the system directly as opposed to accessing static data from a previously-captured snapshot. Unlike the data warehousing approach, EII does not have any physical instantiations of its member system's data, so hardware and software resources are comparatively minimal. Additional maintenance of the member system data is not required since there is no copy, and the latest information is always available directly from the member data systems. A few EII products have become commercially available in the past couple of years. A review on a number of today's Commercial Off-the-Shelf (COTS) EII products is available in (Macvittie, 2004). For any EII product, distributed and heterogeneous data access capability is one of the most fundamental problems that needs to be addressed. There are two major issues: One is to answer queries in a distributed environment, and the other is to resolve various types of heterogeneity, particularly the query capability heterogeneity, among participating data systems. The bottom line is: how to generate a query execution plan that answers the query as quickly as possible with help of utilizing, if any, member systems' querying capability.

As shown in Figure 1, a typical COTS EII product offers different types of wrappers, each for a specific type of data repository like RDBMS, XML documents, Excel files, packaged applications, and so forth. A wrapper is a software module that provides an alternative interface for accessing a given software resource, and thus a wrapper herein can provide a common interface for heterogeneously-structured data sources. A so-called "federated view" will be created for each schematic element in the member data system. A federated view is essentially a virtual entity, and users can perform any supported data manipulating operations over it, for example, defining a selection, a projection, and a join. Complex views can be further created on top of one or many federated views. Both federated views and complex views constitute the Common Global Schema of Data. In general, a user would think the whole integrated system as if it were a single virtual data warehouse. Upon receiving a query request, the internal distributed query engine in EII product will take the whole responsibility of decomposing the original query into sub-queries, dispatching sub-queries to corresponding member systems, and merge the intermediate query answers together.

The COTS EII products have provided a very interesting capability, that is, answering queries against a variety of information sources across the networks

Figure 1. COTS EII product architecture



with a single unified query interface. This offers tremendous help in resolving some lower-level heterogeneity, including: platform (e.g., Operating Systems) heterogeneity, network heterogeneity, data model heterogeneity, and more importantly, querying capability heterogeneity. However, two major issues are missing in almost every COTS EII product that we have seen so far: semantic interoperability and dynamic adjustments upon changes.

For these COTS products, if any of the data elements in a query become not accessible due to any possible reasons, the query execution would stop. Users will end up with an error message indicating that an error has arisen. An extreme case could be: The failure of one member data system may cause the whole integrated system to not function well and therefore poor system reliability.

The common global schema is represented by using some lower-level data models, for example, relational model or XML schema/DTD files. The semantics of the integrated information are usually not explicitly expressed. Instead, they are implicitly embedded in the schemata. Data is still retrieved by using low-level query language like SQL or XQuery, with which users must have enough database knowledge in understanding both the schema and the query language. It is not user-friendly for the so-called average users, who usually do not have enough knowledge on databases, but are very familiar with the domain.

EII products do not provide users with assistance in understanding the meaning of the data, nor do they help to resolve semantic inconsistencies. They only provide data consistency. Users are expected to "learn" EII terminology (attribute/view names, for instance) in order to access data from the EII server. Heterogeneity between data structures may be resolved, but heterogeneity between the physical data structure and semantics of the user domain is not.

For the last several years, research has been focused on semantic interoperability of information (Doan, Noy, & Halevy, 2004; Ouksel & Sheth, 1999; Sheth, 1998). Just to name a few: The InfoSleuth project (Fowler, Perry, Nodine, & Bargmeyer, 1999) used an agent-based architecture for semantic integration of information sources; the OntoBroker project (Decker, Erdmann, Fensel, & Studer, 1999) employed the Resource Description FrameWork (RDF) to represent semantic level metadata; and the DAML project (Hendler & McGuinness, 2000) has extended that representation and applied it to semantic integration of services. While the focus of the research community has been on developing comprehensive solutions to semantic integration issues, our focus has been on applying these developments to specific business problems. We have reported our previous work on semantic information integration in

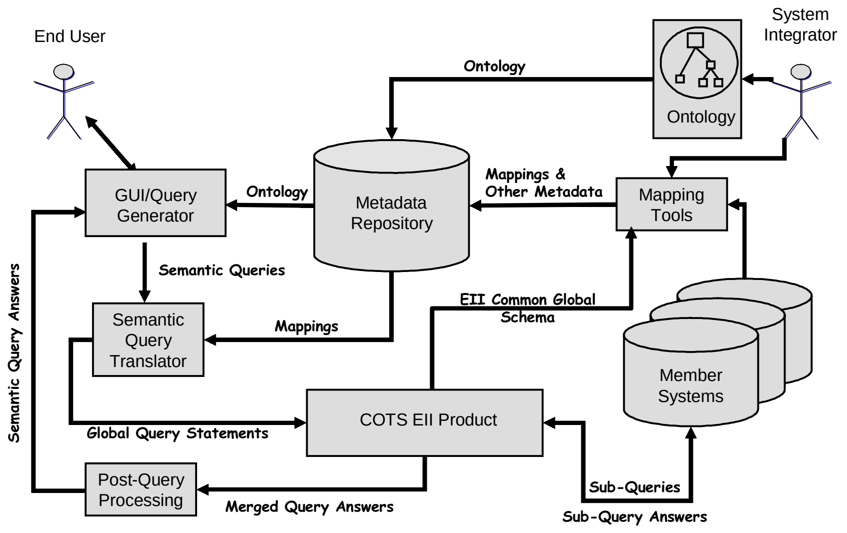
(Barrett, Jones, Yuan, Sawaya, Uschold, Adams, et al., 2002). At that time, since most of the COTS EII products were not available, an agent-based infrastructure was proposed to apply Semantic Web technology to the integration of corporate information. The infrastructure consists of two major components: resource brokers and domain brokers. Every information resource that we wish to make available has an associated resource broker. The role of resource broker is to handle the communication to and from the underlying resource, and it is actually a semantic wrapper for each information resource. The role of domain broker is to perform the semantic integration for particular domain ontology upon receiving information from multiple resource brokers. Multiple domain brokers are allowed to co-exist for multiple domains, and a resource broker can work with multiple domain brokers simultaneously as well. This infrastructure has provided tremendous flexibility in terms of semantic interoperability. However, it also requires huge efforts on query optimization, especially because many mature relational-based distributed query optimization techniques (Kossmann, 2000) cannot be applied directly.

With the emergence of many COTS EII products, we are proposing another framework, benefiting from using a better-optimized EII query engine. The basic idea is to utilize the EII products' available capability on data integrating in maximum, and provide additional features that EII products do not have. In subsequent sections, our discussion will start with the overall architecture of our framework, following with some detailed discussions on several major issues including: integrated metadata management, querying against semantics, and dynamic adjustments upon changes.

Semantic-Based Information Integration Architecture

An overall architecture of our proposed semantic-based dynamic enterprise information integration framework is presented in Figure 2. The semantics of information contents in all participating member systems are represented by ontology. The ontology is persistently stored in the metadata repository. A COTS EII product is used to perform low-level data integration. Different types of utility tools that come with the selected COTS EII product can be used to hook the EII product to each participating system. An EII common global schema will be established as an output of this hookup. With help of our home-

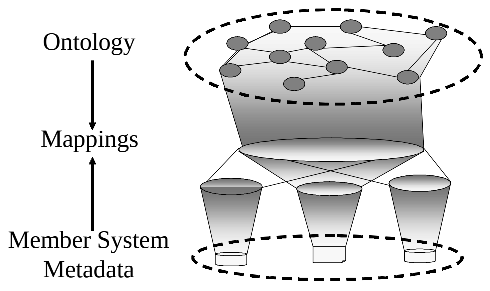
Figure 2. Semantic-based dynamic enterprise information integration framework



grown mapping tools, a system integrator can create mappings between the ontology and local metadata, and thus the EII common global schema. Such mappings are also stored in the metadata repository persistently, and will be utilized to conduct query translation from a user-defined semantic query to one or multiple global query statements that are acceptable by the EII query engine. The query generator, usually a GUI, allows end users to define semantic queries against the ontology. Upon receiving a semantic query, the semantic query translator will translate the original query into one or more query statements according to the mapping information. Such global query statements are submitted to the underlying COTS EII query engine, and the query results of COTS EII query engine will be sent to the post-query processing module, mainly to transform the query answers into an instantiation of the ontology. In the end, the semantic query answers will be presented to end users via GUI.

In this framework, different tasks in regards to semantic information integration and interoperability are performed at two different layers. The task of resolving lower-level heterogeneity is actually pushed down into the COTS EII product, while the various types of semantic heterogeneities are resolved by both domain ontology and the semantic mappings between domain ontology and EII common global schema. This separation allows us to better utilize many distributed query optimization strategies that have been deployed in the COTS EII products.

Figure 3. Major metadata elements



Integrated Metadata Management

Metadata has always been one of the most essential components in every data intensive system, and it is the same case here. In fact, metadata management is undoubtedly playing a very important role in regards to semantic enhancement. In this section, we will discuss issues relating to metadata management in a semantic-based information integration environment from different aspects, such as: What is the metadata content from information integration's viewpoint; where and how to get/maintain the metadata; what kind of correlations between different pieces of metadata need to be captured and managed from different types of data repositories; and with what kind of unique capabilities the metadata management is going provide with end users, query engines, system administrators, or whoever else to fulfill information management requests.

There are three major types of metadata elements captured in our proposed metadata architecture. As shown in Figure 3, they are semantic data representation as represented in a piece of ontology; physical/logical layer data descriptions (member system metadata) for each member system; and mappings between the ontological layer and the physical/logical layers of each participating system.

Ontology

An ontology is essentially an information model of the domain of interest. The most commonly quoted definition of ontology is "a formal, explicit specification of a shared conceptualization" (Gruber, 1993). A detailed introduction to ontologies as well as the current state of the art for applying ontologies to achieve semantic connectivity is presented in Uschold and Gruninger (2004). In our approach, ontology is used to subsume the semantics of the data across multiple data systems, and it is independent of any member system's data representation. The ontology is accessed by both users and client software components to capture and clarify domain semantics. Use of ontology to abstract and encapsulate multiple data sources serves to obscure the complexity of the individual data elements and indeed provides an opportunity for a more "user-centric" approach to maturation application development. This is because how data is organized physically is usually of far less interest than how the available information is semantically described. With ontology, from end users' or application developers' perspectives, the integrated system then looks like a single information resource, containing all of the available information within a specific domain. They would be relieved of the need to know where data is located — responsibility for location and identification of correct and relevant data elements would belong to the mediation. Ultimately, users should not need to know the details of the data sources — that is the responsibility of an integrated system's query formulation process based on knowledge about physical system particulars.

There may potentially be many choices of representation languages, for instance, first-order logic- (FOL) based logical languages or description logic-based ontological specifications such as RDF, DAML, and OWL. While each of them may have its own advantages as well as disadvantages, our focus is only on the basic elements of ontology, that is, concepts and relationships between concepts, which are available in almost every ontological representation language. The metadata architecture here is, therefore, technically applicable to almost every ontological specification available today.

Member System Metadata

In our framework, with full local autonomy supported, the data remains to be stored and maintained in each member system. In order to achieve better performance, user-defined query requests will be answered by dynamically executing sub-queries over each member system. To construct those sub-queries correctly, one has to fully understand the access requirements and other details about how data is described and organized in these member systems, no matter they are well-formed database schemata or any semi-structured data representation. Considerable effort must be put into understanding the physical and logical design of the system, and a keen understanding of the individual data elements must be attained. It is prudent to take advantage of metadata that already exists in member systems so as to minimize redevelopment costs and ensure consistency. The metadata most likely to be available are physical models (usually available), logical models (less likely), and data dictionaries (even less likely, although less formal glossaries are sometimes available or can be gleaned from design documentation).

The physical model identifies the layout of the data, a prerequisite to find and access data. Although one can draw significant knowledge from the physical model, accurate understanding of the data elements themselves must also be achieved. The names used to label information tend to be cryptic and inconsistently applied (if at all), relationships in the data structure are difficult to recognize directly, and no descriptive information is generally available for reference and precise interpretation. A physical model provides the means to traverse those relationships supported by the physical design and may supply some measure of descriptive information on the design components (DBMS, XML structure, tables, fields, etc.).

The logical model identifies conceptual relationships within the data. A logical model is used to explicitly identify those relationships without regard to the platform-specific optimizations that are provided in the physical models. It also tends to apply a naming structure that is more useful in understanding the data relationships within a particular data system. These relationships, whether explicit or implicit, are a source of metadata needed to provide semantic understanding of the data. Within our approach, the logical model is used as an essential source of interpretation for logical data structure and determining part of semantic interpretation of the data content and therefore, is ultimately one of the key elements in bridging the gap between the ontology and the physical data.

A data dictionary provides a textual description of the data elements defined within a model. It is the primary source for explicit semantic interpretation of the data within a data system. The data dictionary is not a semantic model per se, but it does provide crucial understanding of the data content from a semantic perspective. In conjunction with the supporting model, it is used heavily to ensure proper interpretation of the data content and correlation of the mappings to the ontology during the mapping process. Dictionaries are typically not provided, and in those cases must be derived from whatever sources are available: glossaries, other design documentation, system experts (either administrators or users), and so forth.

Mappings

Creation of appropriate mappings are the most critical and challenging element of our approach. Resolution of heterogeneity has always been a major challenge for any information integration effort. Our information integration approach aims at resolving all types of heterogeneities dynamically when processing queries, and does not require a pre-conversion of data. However, the dynamic mappings are generally more difficult to create than performing static pre-conversion.

There are three key aspects to create mappings that support dynamic heterogeneity resolution: (1) the different types of mappings that are required, (2) how they are represented, and (3) utilization issues. Each of these will be discussed in further detail in this section.

In our definition, mappings are not only correlations between any two (or more) models, but also in general bridge some portion of the conceptual gap between the physical model and the ontology. Since the ontology is not a model supplied by any participating data system, the mappings to the ontology must be developed. Unfortunately, there is little in the way of formal methodologies for identifying, developing and managing mappings, yet it is critical to the operation of a mediated information integration system. What follows provides a high level description of the approach developed in our effort.

Types of Mappings

While formal methodologies for creating mappings may be scant, there are many classifications of mappings presented in the literature. Our approach attacks the issue from how user-defined ontological queries can be answered. Mappings in this mediated architecture are divided into two major categories, that is, horizontal mappings and vertical mappings.

Horizontal mapping is a kind of association between elements at the same level, such as mapping among concepts from multiple ontological models, or mapping among different ontological representation standards such as mapping ontology from RDF to FOL. Horizontal mappings are important because they provide relations and constraints between ontological metadata elements, and consequently result in solutions for better interoperability at the ontological layer. Sometimes, the terms that users utilize and those employed by the ontology may have no commonality even when they denote the “same” concept. This is significant, as the objects of communication (*terms*) no longer denote the objects of interest (*concepts*) unambiguously. An example of this semantic mismatch that may be familiar to readers in aircraft maintenance domain is the terminology used to describe avionics "black boxes" that are replaced as part of a flight line maintenance action. Depending on the organization performing the maintenance, the replaceable unit may be referred to as a Line Replaceable Unit (LRU) or a Weapons Replaceable Assembly (WRA). This simple example illustrates the semantic mismatch issue that proliferates throughout all maintenance data systems. Again, what a product engineer might call a "system fault" may be known as an "inherent failure" in one maintenance data system and be referenced by a cryptic code in another. Semantic issues of this nature can be perplexing for humans but can make creation of automated analysis tools an extremely arduous task.

Vertical mapping, on the other hand, refers to the association between metadata elements at different levels of abstraction, such as: mappings between ontology and lower-level schemata (e.g., a logical or physical schema); mappings between ontological queries and lower-level queries; and mappings between local security policies and global security policies.

In our approach, the vertical mappings provide direction to the query processor, which uses such information to evaluate and optimize ontological queries against member systems efficiently and effectively. The integrated system may consist of a number of member data systems, each of which has its own logical/physical schema. These schemata from multiple member systems may or may not share common naming standards across their logical/physical schema; however, they do share common semantics as expressed in the ontology. Mappings between the ontology and lower-level schemata connect each lower-level schematic element with one or more corresponding concepts in the ontology. They are constructed on the basis of semantic equivalence. The semantic equivalence does not mean "absolute equity" only, but can also represent "approximate equivalence," in which case a conversion method will be attached to the mappings for data transformation.

Recalling that the goal of our approach is to access data from multiple heterogeneous data systems using a single ontological view of the data, the global query engine must have the ability to translate queries and transform query results from one format to another between ontological and physical layers. To achieve this, the query engine must be able to reconcile the differences in query capability between member systems, because different member systems are likely to deploy different data models and use different query languages. Some types of data sources, for example, unstructured data sources such as plain text files, online documents, and email systems, do not have a query facility at all, or have a very limited query mechanism, such as a keyword-based search engine. Even for a standard query language such as SQL, dissimilarities can be found among various commercial products, especially for advanced operations such as outer joins. Thus, the mappings between an ontological query and a lower-level data system query must provide the information required by the query engine by characterizing and mapping member system query capabilities so that the engine will be able to apply appropriate translation rules when needed.

Mapping Representations

After a suitable mapping methodology has been created, the next question is: Where and in what format should they be stored? In our architecture, the metadata repository is the primary place where most of the mapping information resides. It is a stand-alone information repository. Rule-based logic can be used to extract knowledge from the repository for various purposes such as ontological-to-physical query translation and physical-to-ontological result reconciliation. Different types of mappings are represented in different formats in the metadata repository. For example, some horizontal mappings (relations) between ontological elements are embedded in the ontology as part of the ontology alignment; some vertical mappings between ontological elements and physical schematic elements are described by using a proprietary set of mapping predicates that can be developed by system experts using an interface that manipulates a set of primitives and macros; and some other vertical mappings (such as query capability mappings) are either implied or hard-coded in different types of algorithms implemented by using either FOL rules or, in a few cases, programming code written specifically for individual member systems.

Each software component has its own requirements for information that must be supplied during query development processing, and in general they usually require more sophisticated mapping knowledge than the basic mapping predicates themselves. This implies that some derivations must be made before they can be used. The derivations might be something as simple as transitive closure calculation or as complex as evaluating an advanced logic program. This imposes a requirement for an inference capability on the ontology and mappings, which makes the choice of a deductive database sound more beneficial as the persistent storage medium. With a deductive database, such derivations can be pre-defined in terms of rules; the built-in inference engine will then be able to execute these stored rules to expand the original ontological query to an executable series of queries using the pre-defined series of basic mapping facts.

Using Mapping for Semantic Wrapper

Recent efforts to enhance interoperability among heterogeneous data sources focuses on structured data sources such as databases. Examples include Relational and Life Sciences Data Connect at IBM, OLEDB at Microsoft, Virtual Table Interface at Informix, and Adaptive Component Architecture at Sybase. Though the detailed techniques differ from one to another, they have a common approach to achieve data access interoperability, that is, to wrap the individual data source by providing a single relational access point. The aforementioned wrapping technology has provided a very interesting capability, that is, a homogeneous SQL-based querying interface for various types of databases. However, they mainly concentrate on interoperability among structured database systems rather than semi-structured or unstructured data sources. More importantly, the data is still represented by a relational model, where semantics are not explicitly presented. Semantic links between tables are implied by integrity constraints or joins, and query results are presented in a simple tabular format. In fact, it has brought little, if any, help on improving the semantic interoperability.

Instead of wrapping the information resource with a relational model, a more preferable approach, in our opinion, is to use an ontology to wrap the underlying data source, so that the data content is exposed and can be retrieved as an instantiation of the ontology. In general, a semantic wrapper offers a unified semantic view over the wrapped information resource, and performs on-demand semantic translations. It also wraps multiple heterogeneous querying interfaces with a unified and mutually-compatible semantic querying interface. Information consumers use this to query information by semantics, without having to understand lower level details such as logical/physical data structures. For those information resources with less powerful querying capability, the semantic wrapper is not only a semantic enabler, but also a querying capability provider.

Upon receiving a semantic information retrieval request, a semantic wrapper will generate an information access and transformation plan based on the mappings between the ontology and local metadata. Depending on the power of the querying capability of the wrapped information resource, one or more sub-queries are generated and then pushed down into the wrapped information resource. The query answers of these sub-queries are then merged and translated into an instantiation of the ontology and returned to the software component that has posted the query.

In building a semantic wrapper, three pieces of information are important: local metadata for wrapped information resource, ontology, and mappings in between the two. Actually, all of these pieces of information are captured in our metadata repository. Next, we are going to provide an example of how ontological elements, for example, concepts and relationships, are used to map across multiple physical schemata for the purpose of answering the query. For the sake of readability, we assume that all participant data systems are relational databases, and we use pseudo-SQL to represent some of the mapping logic.

An Example

We have applied the metadata management methodology presented in this section in building a data collection service for airplane health management (Wilmering, Yuan, & VanRossum, 2003). Such system requires advanced data analysis against many types of data, including airplane design data, airplane performance data, airplane maintenance data, and logistics data. One of the primary issues that we were facing in trying to collect data sufficient to construct a useful analytical model is that: It exists in multiple data systems that are designed and maintained by multiple organizations, for instance, both airplane manufactures and airline companies. These systems are heterogeneous in nature — no thought has ever been given to the relationship of one system to another, let alone the meaning of the data elements contained therein. A key issue when accessing heterogeneous information sources is how to resolve the difference that may exist among one and another. Approaches may vary considerably, depending on the type of information sources and the degree of reasoning/inference required. The example that follows briefs our approach in bridging the difference with the proposed mapping representation.

A simplified three-entity ontology is presented in Figure 4. It contains only three concepts, among which AIRCRAFT is a sub-concept of PART, and therefore AIRCRAFT inherits all attributes and relationships from PART. Two binary relationships are defined in this sample ontology. isPartOf recursively relates the hierarchical structure between instances of PART, and hasRecords indi-

Figure 4. A piece of simplified ontology

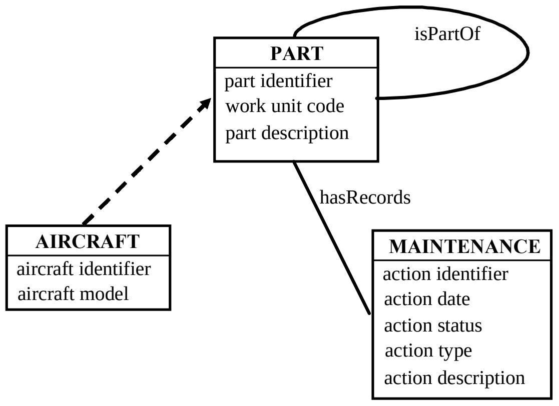
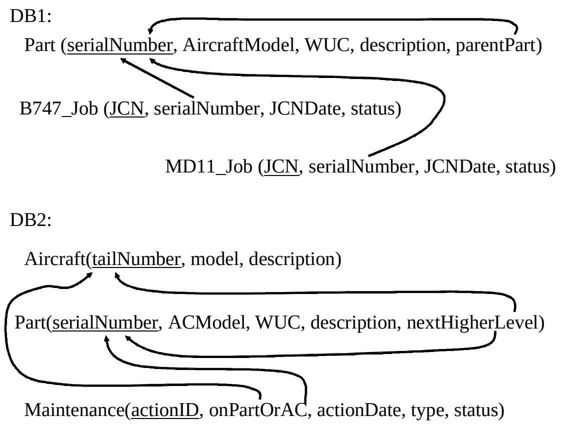


Figure 5. Relational schema


DB1:
Part (serialNumber, AircraftModel, WUC, description, parentPart)
B747_Job (JCN, serialNumber, JCNDate, status)
MD11_Job (JCN, serialNumber, JCNDate, status)
DB2:
Aircraft(tailNumber, model, description)
Part(serialNumber, ACModel, WUC, description, nextHigherLevel)
Maintenance(actionID, onPartOrAC, actionDate, type, status)

cates the instances of MAINTENANCE that have been performed with respect to each instance of PART (including AIRCRAFT).

In a simplified scenario (Figure 5), we assume that there are two relational databases, DB1 and DB2, and each of them contains three tables as shown in Figure 5. The primary key attributes are underlined, and arrow lines denote relationships between tables.

Mappings between the sample ontology and schematic elements in DB1 and DB2 are presented in Figures 6 and 7, respectively, shown in boxes with italic text. We omit the detailed mapping for each attribute in the ontology for the sake of economy.

Figure 6. Mapping ontology with DB1

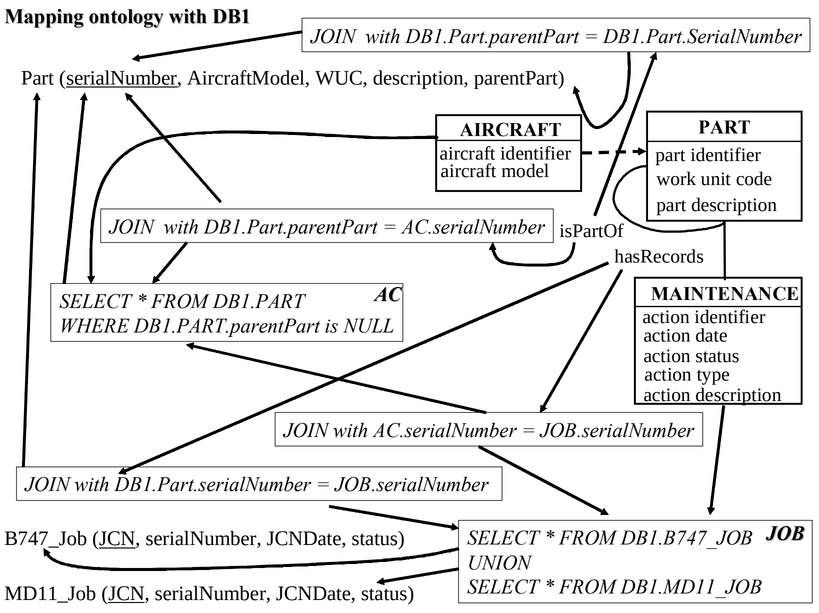
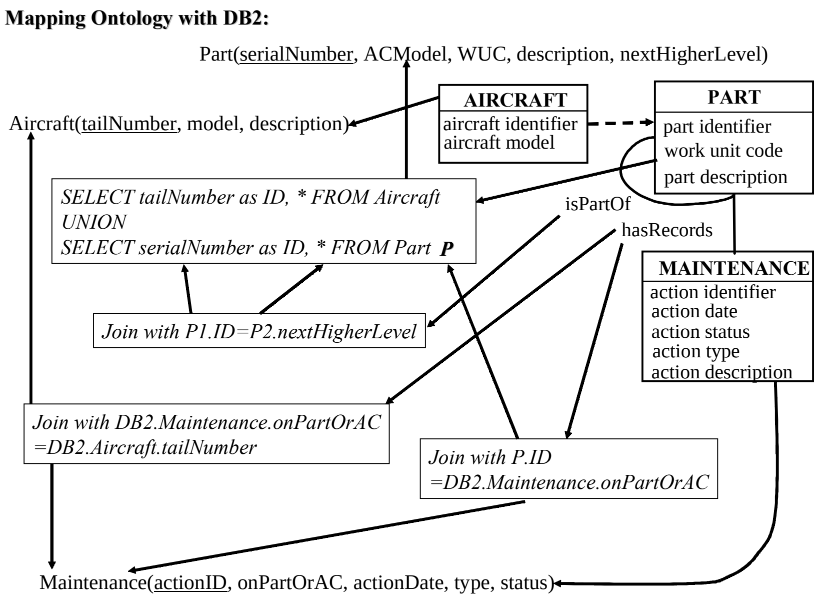

Mapping ontology with DB1
JOIN with DB1.Part.parentPart = DB1.Part.SerialNumber
Part (serialNumber, AircraftModel, WUC, description, parentPart)
AIRCRAFT
aircraft identifier
aircraft model
PART
part identifier
work unit code
part description
JOIN with DB1.Part.parentPart = AC.serialNumber
isPartOf
hasRecords
SELECT * FROM DB1.PART WHERE DB1.PART.parentPart is NULL
AC
MAINTENANCE
action identifier
action date
action status
action type
action description
JOIN with AC.serialNumber = JOB.serialNumber
JOIN with DB1.Part.serialNumber = JOB.serialNumber
B747_Job (JCN, serialNumber, JCNDate, status)
SELECT * FROM DB1.B747_JOB UNION SELECT * FROM DB1.MD11_JOB
JOB
MD11_Job (JCN, serialNumber, JCNDate, status)

Figure 7. Mapping ontology with DB2


Mapping Ontology with DB2:
Part(serialNumber, ACModel, WUC, description, nextHigherLevel)
AIRCRAFT
aircraft identifier
aircraft model
PART
part identifier
work unit code
part description
Aircraft(tailNumber, model, description)
SELECT tailNumber as ID, * FROM Aircraft UNION SELECT serialNumber as ID, * FROM Part
P
isPartOf
hasRecords
MAINTENANCE
action identifier
action date
action status
action type
action description
Join with P1.ID=P2.nextHigherLevel
Join with DB2.Maintenance.onPartOrAC =DB2.Aircraft.tailNumber
Join with P.ID =DB2.Maintenance.onPartOrAC
Maintenance(actionID, onPartOrAC, actionDate, type, status)

Please note that there is no individual table in DB1 that stores information about aircraft. In fact, an aircraft is treated as a special type of PART (i.e., the topmost element in the PART hierarchy). Therefore, the concept AIRCRAFT is mapped to table PART with an additional condition, that is, "parentPart is NULL." The maintenance actions in DB-1 are recorded in two tables with identical structure according to the aircraft models (the tables Boeing747-Job and MD11-Job). So, the ontological concept MAINTENANCE must be mapped to a union of those two tables, that is, the relationships in the ontology are mapped with semantics implied by JOIN operations in relational algebra. Because of the inheritance between the concept AIRCRAFT and the concept PART, there are multiple mappings associated with every relationship coming out of concept PART.

DB2, however, does have a specific table for aircraft. Therefore, the concept AIRCRAFT can be mapped to that table directly. This also means that the PART table in DB2 no longer contains aircraft information. In order to maintain the semantic consistency of concept PART, that is, being a super-concept of AIRCRAFT, concept PART is mapped to a union of the AIRCRAFT and PART tables in DB-2. Since there is no horizontal partition on maintenance information in DB2, concept MAINTENANCE is mapped to a single relational table. Similar to DB1, relationships are mapped to JOIN operations between relational tables.

Query Against Semantics

In database systems, queries are questions to be answered or solved by database engines. A particular database system usually offers a specific querying interface, for example, a database query language, which allows users to formulate their querying requests in a specific format. A query language usually has a tight connection with the underlying data model, and therefore it may vary significantly from one to another on both syntax and the power of expressiveness. Normally, an underlying database schema not only details the structure of how data should be organized, but it also provides guidelines about how data should be queried. While dealing with different types of information sources across the networks, users are likely required to switch query languages from one to another in order to integrate related pieces of information together. A unified query interface that is able to operate over different types of information sources becomes greatly preferable in order to enhance the interoperability of heterogeneous data systems.

The semantic correlation that exists among different pieces of information in a heterogeneous environment is the very reason why information integration shall happen. Despite the fact that data is presented quite differently on many aspects, it has one commonality, that is, the shared semantics. Unfortunately, such semantics cannot be explicitly represented by many of today's data models. Instead, they are implicitly represented, if not missing, by some internal mechanisms, for example, database integrity constraints. Similarly, traditional query languages that are built on top of these data models do not have the ability to express explicit semantics either. Taking relational query languages, for example, it is very common that one or more join operations need to be defined to inquire connections between data items in multiple tables. In order to have those join operations exactly reflect the semantics, users have to have substantial knowledge on both relational database theory and query language itself.

The advent of the Internet has brought millions of end users into the information cyberspace. All these so-called average users usually have little, if any, training in both traditional query languages and database systems. As a result, they have some particular concerns about the easy-ability of query interface. Their preference might be something other than the current database languages such as SQL or XQuery. They wish to have a higher-level query language, which enable them to retrieve information only by their semantic understanding on the domain, or at least, to retrieve the information without having to completely understand things like the physical data distribution or the lower-level data structures.

Information-sharing capability has been playing a more and more important role in today's information technology. Query answers have to be shared among many information consumers. The information consumers are not only limited to human beings, but also computers or any software components in the loop. The implication of computer systems being information consumers is: The semantics of query answers should be both human- and machine- understandable.

Semantic information access methodology refers to a mechanism that enables the information consumer to retrieve information via semantics only. The major distinction between a semantic information access methodology and traditional information access methodology, such as database query languages, is the semantic understandability. Machine-understandable semantics need to be provided explicitly for all involved parties during the query processing. Such semantics include the semantics of the domain, the semantics of information content in each information source, the semantics of a user-defined query itself, and the semantics of returned query answers.

Comparing with some conventional query paradigms, a semantic information retrieval mechanism should have some unique features:

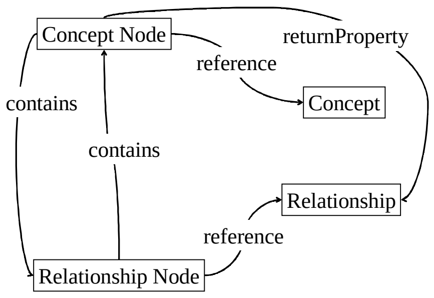
1. It is neutral to any particular semantic representation format such as EER model, RDF/DAML-OIL, or Datalog. Instead, it is built upon basic semantic elements, that is, concepts and relationships, in any semantic model.
2. It requires little prior training and is relatively intuitive to be utilized by average users.
3. It is powerful enough for users to express their query criteria.
4. It hides complex logical data structures and data distribution from users, and therefore, they do not have to know a lot about underlying data sources.
5. It enables users to formulate their queries in a semantic way based on their knowledge of the application domain.
6. Query answers are delivered as an instantiation of the referenced ontology, and therefore are semantic understandable.

Query Representation

We defined a query structure by ontology, and specified the individual queries as instantiations of that ontology. The internal formalism of our query structure is designed neutral to any particular ontological representation language or standard. It is proposed based on two fundamental ontological elements, that is, concept and relationship, the counterparts of which are available in almost every ontological language specification. Therefore, as shown in Figure 8, there are two fundamental constructs, that is, concept node and relationship node in the query structure.

A concept node in the query references a concept in the domain ontology, and it may contain one or more relationship nodes. Likewise, a relationship node references a relationship defined in the domain ontology, and it basically tells a navigating operation from one concept to another via the referenced relationship. A relationship node may also contain one or more concept nodes, and

Figure 8. Query representation structure



these concept nodes refer to the ending points of an ontological navigation. A concept node may also have links to zero or many relationships in the domain ontology (via returnProperty), and these links tell us what are the requested query targets with respect to the concept node.

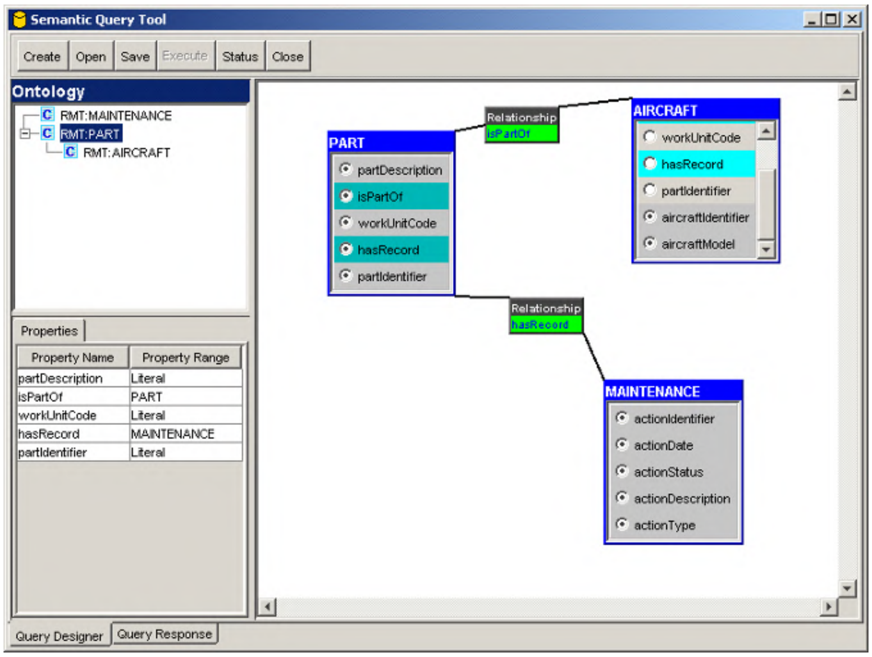
You may treat concept nodes as variables in a query, and they are placeholders for instances of concepts. When a property value is requested (via returnProperty), it will always be in the context of a specific instance of a class. This provides a benefit when merging query answers from multiple sub-queries such that we only include the bindings for concept nodes because the requested property values can be easily calculated from the answers of sub-queries. This makes the post-query processing more efficient.

Our query structure also supports query constraints and aggregation requests, which can be attached to either concept node or relationship node. A query constraint may be either a simple constraint with comparison functions (EQ, NE, GT, ...) or a more complex logical expression in which multiple simple constraints are connected by logical operators (AND, OR, NOT).

As mentioned early, another important criterion for semantic information access methodology is the semantic understandability of query answers. In our query structure, the query results are returned as instantiations of referenced ontology. The data elements in query answers are represented as instances of concepts and relationships defined in the ontology, and therefore the semantics of query answers can be easily captured.

Based on the internal query representation, we have also implemented a graphical query designer that allows users to conveniently compose semantic queries. With such a query designer, when defining a query, a user typically picks a concept as a starting point, simply drag-and-drop the concept into the

Figure 9. A sample query with query designer



Property Name	Property Range
partDescription	Literal
isPartOf	PART
workUnitCode	Literal
hasRecord	MAINTENANCE
partIdentifier	Literal

designing panel, and the query definition thereafter is more like navigating the whole ontology through selected relationships.

Different from many conventional relational query designers provided by major database vendors, our query designer enables users to define queries by semantics only. Therefore, there is no concern, from end users' perspective, about issues such as union compatibility and joinable columns, which usually requires fairly amount of database knowledge in understanding the database schemata.

Some generic ontology tools such as Protégé[1] and OilEd[2], may allow users to create queries by importing the query representation ontology. However, they are not user-friendly enough largely because it is extremely difficult, if not impossible, for end users to fully understand the abstract and profound query representation ontology. In contrast, our query designer does not require users to understand query representation ontology at all. Instead, semantic queries can be defined against domain ontology directly, in a much easier navigational manner.

A very simple semantic query that complies with our query structure is presented in Figure 9. There are three concept nodes (PART, AIRCRAFT, and MAINTENANCE) and two relationship nodes (isPartOf and hasRecord) defined in the query. Relationship node isPartOf refers to a navigation from PART to AIRCRAFT, and hasRecord refers to a semantic navigation from PART to MAINTENANCE.

Dynamic Adjustments upon Changes

In addition to all kinds of heterogeneity that needs to be taken care of at the first place, changes of different types are happening over the time. For example, the accessibility and capability of each individual information source may change; new data systems are added or existing data systems are removed; the underlying data model of particular information sources may be modified; and network disconnectivity and congestion may occur occasionally. All these changes sometimes prevent users from getting information correctly, and they usually have big impacts on related operations. Therefore, a mechanism needs to be established, which not only can actively monitor changes in real time, but should also be able to take appropriate reactions upon detecting those changes. Such reactions can be regarded as to perform a real-time system re-configuration and information re-integration.

With today's COTS EII products, for example, if any of the data elements in a query are not accessible due to any possible reasons, the query execution would stop. Users will end up with no information being retrieved but only an error message indicating that something was wrong during query evaluation.

In order to fulfill such dynamic information re-integration, the query processor needs to be able to understand what the semantic implications are for each type of the changes. For instance, the fact that a particular data element becomes inaccessible may imply different things for different groups of users in different circumstances. Such semantic implications are actually analytical results on artifacts including: who is the user; what is the semantics of data; what is the semantics of user-defined queries, which finally lead to appropriate adjustments on query evaluation.

Looking at this from another perspective, the information integration here is no longer the traditional information integration strategy, where information is integrated in a purely static manner. The proposed dynamic information integration strategy has its unique feature in dynamic query processing. Generally, given any query request, with a static information integration strategy, there is only one query execution plan be generated to retrieve the query result in any circumstance. If anything goes wrong during the query execution, it fails. However, a dynamic information integration strategy here will be able to generate different query execution plans based on related metadata information, and guarantee users to get what they are supposed to get, and of course, according to their tolerance to approximation in query results.

We have developed a query rewriting technique to perform various types of dynamic adjustments based on the accessibility of the data elements that are mapped to either concepts or relationships in the ontology. The basic idea of this query rewriting technique is to exclude the inaccessible data elements from the translated COTS global query statements such that the underlying COTS EII products can execute those query statements successfully.

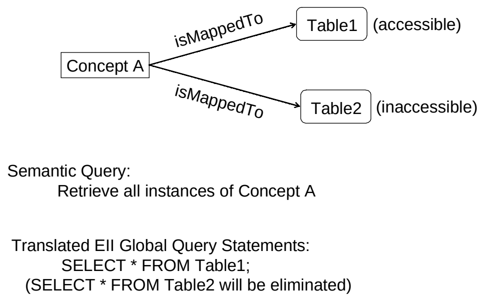
As stated in the previous section, concept node and relationship node are two fundamental constructs in a semantic query. Remember that a concept node will reference one and only one concept in the ontology. Thus, when encountering a concept node in the semantic query, our semantic query translator will utilize the corresponding mapping information for the referenced concept to process the query. Taking relational-based EII products, for example, we have mentioned in previous sections that a concept in ontology is usually mapped to one or many relational tables. When receiving a user-defined semantic query on that concept, the query translator will then generate, according to the number of mapped relational tables, one or many SQL statements. If a particular mapped table is inaccessible at the time when user-defined query is received, the SQL statement against that table will be eliminated, and therefore not submitted to the underlying EII query engine for execution.

As shown in Figure 10, suppose that Concept A in ontology is mapped to both Table1 and Table2. For the semantic query "Retrieve all instances of Concept A," the query translator usually will generate two SQL statements, for Table1 and Table2, respectively. However, because Table2 is inaccessible, the query generator will make the adjustment by removing "SELECT * from Table2" so that the underlying EII query engine will retrieve query answers only from accessible tables.

Relationship node is another basic construct in the semantic query representation. Similar to a concept node, a relationship node references a unique relationship in ontology. For relational-based COTS EII products, a relationship between two concepts is typically mapped to a join operation between two tables.

As shown in Figure 11, Concept A is mapped to Table1 and Table2, and Concept B is mapped to Table3 and Table4. The Relationship (R-AB) is

Figure 10. Mapping concept

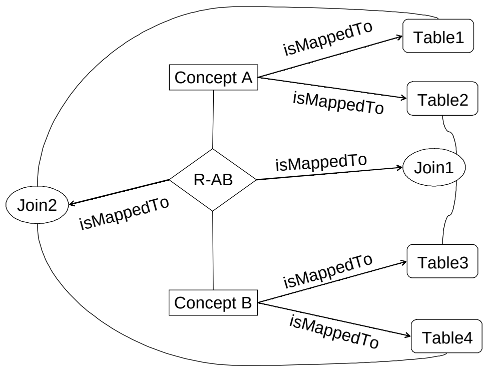


Semantic Query:
Retrieve all instances of Concept A

Translated EII Global Query Statements:

```
SELECT * FROM Table1;
(SELECT * FROM Table2 will be eliminated)
```

Figure 11. Mapping relationship



mapped to two join operations, that is, Join1 and Join2, while Join1 is between Table2 and Table3 and Join2 is between Table 1 and Table4. We omit the detailed join information, such as join criteria, due to space limitation.

There are different types of join operations defined in relational world, for example, inner join and outer join. The mapping mechanism in our approach does not specify the type of join. Instead, an appropriate join type will be chosen based on users' preference on semantic navigations. We allow users to define two types of navigations: optional navigation and mandatory navigation. An optional navigation from Concept A to Concept B via R-AB indicates users' intention of obtaining instances of Concept A even if there is no related instance of Concept B under R-AB; whereas a mandatory navigation means that users are not interested in getting instances of Concept A at all if there is no related instance of Concept B under R-AB. It is easy to find out that the meaning of an optional navigation is very similar to that of a relational outer join

Figure 12. Different scenarios of query translation

Navigation		Table1	Table2	Table3	Table4	EII Global Query Statements (in pseudo SQL)
Optional Navigation	A	YES	YES	YES	YES	select * from Table2 left outer join Table3 on … select * from Table1 left outer join Table4 on …
	B	YES	YES	YES	NOT	select * from Table2 left outer join Table3 on ... select * from Table1
	C	YES	NOT	YES	YES	select * from Table1 left outer join Table4 on …
	D	YES	NOT	YES	NOT	select * from Table1
	E	YES	YES	NOT	NOT	select * from Table2 select * from Table1
	F	NOT	NOT	YES	YES	(none Query statement generated)
Mandatory Navigation	A	YES	YES	YES	YES	select * from Table2 inner join Table3 on … select * from Table1 inner join Table4 on …
	B	YES	YES	YES	NOT	select * from Table2 inner join Table3 on …
	C	YES	NOT	YES	YES	select * from Table1 inner join Table4 on …
	D	YES	NOT	YES	NOT	(none Query statement generated)
	E	YES	YES	NOT	NOT	(none Query statement generated)
	F	NOT	NOT	YES	YES	(none Query statement generated)

operation, and the meaning of a mandatory navigation is similar to that of an inner join operation.

In addition to selections of different join types, when the accessibility of mapped table changes, our query rewriting technique will also make different adjustments for optional and mandatory navigations. The table in Figure 12 presents several typical scenarios (labeled with "A," "B," "C," "D," "E," and "F"), with varying combination of accessibility in regards to the mapped tables for the previous mapping example. For each table, "YES" means accessible and "NOT" means in accessible. For your convenience, pseudo SQL statements are provided as examples of translated EII global query statements.

For scenario A, while all four tables are accessible, two query statements are generated. However, left outer joins are used for optional navigations, but inner joins are used for mandatory navigations.

For scenario B, there are still two query statements generated for optional navigations, but the join between Table1 and Table4 is not presented in the second query statements. This is because Table4 is inaccessible, and query statement will only return instances of concept A. For mandatory navigations, due to the inaccessibility of Table4, there is only one query statement generated. The second query statement is eliminated mainly because the semantics enforced in mandatory navigation.

For scenario C, there is only one query statement generated for either optional navigations or mandatory navigations. This is because the inaccessibility of Table2 tells us that one of the mapped tables for concept A is not available, and concept A is the starting point of the navigation.

Scenario D is actually a combination of scenario B and scenario C. As a result, there is only one query statement generated for optional navigation, but the join between Table1 and Table4 is omitted because of the inaccessibility of Table4. For mandatory navigations, there is no query generated.

Scenario E is very similar to scenario B. For optional navigations, two query statements are generated but neither of them has join operations presented, and there is no query generated for mandatory navigations.

Scenario F is very similar to scenario C; neither optional navigation nor mandatory navigation has query statement generated. This is because both of the mapped tables for concept A, which is the starting point of the navigation, are not accessible in this case.

Future Trends

Ontology plays an important to expose the implicit semantics of the information content that has been stored in multiple heterogeneous data systems. In our approach, we say that the semantics are described explicitly by a single ontology. However, people may argue that it is not viable since universal agreement on semantics can never be reached among different groups, particularly for a relatively larger domain. We acknowledge this, and we have in fact experienced such difficulty when applying this technology to different business units. In our opinion, a single ontology does not necessarily imply that such ontology has to be created within a single step. On the contrary, there are different ways to develop such ontology. Different groups have choices of developing their own pieces of ontology based on their own semantic understanding, and these pieces of ontology are to be merged together through the process of ontology alignment. Ontology alignment and management has been discussed in the research community (Doan, Madhavan, Dhamankar, Domingos, & Halevy, 2003; Maedche, Motik, & Stojanovic, 2003). However, a globally-recognized approach has not been reached yet. Ontology alignment is actually one of the objectives of our horizontal mappings as mentioned in early sections. In addition to ontology alignment, ontology evolving is another issue that needs to be addressed more carefully. As ontology migrates from one version to another, an appropriate mechanism needs to be established to track the semantic difference between versions. Such difference usually has impacts, more or less, on the underlying mappings between ontology and local schemata. A good version control mechanism on ontology may provide great potentials to increase the degree of automation on maintenance of mappings.

As said in previous sections, mapping between models is one of the key factors toward semantic-based information integration. Though mappings between the ontology and local metadata can always be identified in a pure manual manner, an improved degree of automation in capturing the mapping information is highly desired to significantly reduce the cost of building an integrated system. In Rahm and Bernstein (2001), the authors have presented a survey on recent work about automatic schema matching, in which a few methods were mentioned ranging from structural matching algorithms on schematic elements to linguistic-based matching algorithms on data instances (He & Chang, 2003; Kang & Naughton, 2004; Yan, Miller, Haas, & Fagin, 2001). However, a completely automatic schema matching solution seems not yet viable at this time. Our colleagues at Boeing (Coen, 2002) also proposed database lexicography, a metadata analysis discipline that applies lexical graph theory to database design. Database lexicography can be further applied to analyze existing database models, and thus provide potentials on automatic schema matching with additional help from lexical tools, such as WordNet (Fellbaum, 1998).

Conclusion

We have presented a framework that supports semantic-based dynamic enterprise information integration. With COTS EII query engine, we offer a mediated, or "virtual warehouse" approach, in contrast to an explicit data warehouse for reasons of economy, reliability, maintainability, and scalability. The resulting synthesis of information management and ontological principles provides a strong basis for effective computational support for advanced application development. We have discussed a number of key issues related to the semantic integration of corporate resources. Along with giving examples from our implementation, we identified the necessary technology components covering a number of aspects, including integrated metadata management, semantic querying and wrapping mechanism, and dynamic adjustments upon changes.

A COTS EII product is used in our approach to conduct low-level distributed query processing, by which our approach assumes that many types of low-level heterogeneity are able to be resolved. Two pieces of major improvement are made on top of the COTS EII products, for the purpose of increasing the semantic understandability of information, as well as system reliability and robustness.

With a unique mapping mechanism introduced between ontology and local metadata, users are able to query the integrated system by semantics only, without having to understand how the data is actually organized physically. Moreover, the retrieved information is directly delivered as an instantiation of the ontology, with a common set of both concepts and relationships between them. This greatly improves the semantic understandability of retrieved information, and may eliminate the need to further classify instances as it may be required by many advanced applications. As a result of this effort, we have found that even a relatively sparse semantic representation can give significant improvement in the time it takes to find information.

A mechanism is introduced to adjust the query dynamically. Upon detecting environmental changes, for example, the accessibility of some specific data elements, the query processor will make appropriate adjustments on the underlying execution plan to ensure that semantically- relevant information can be retrieved from available data sources, instead of throwing an exception as most of the COTS EII products do. In practice, failure to make such appropriate and dynamic adjustments, in response to changes, may result in performance shortfalls in all operational missions.

References

Barrett, T., Jones, D., Yuan, J., Sawaya, J., Uschold, M., Adams, T., et al. (2002, May). RDF representation of metadata for semantic integration of corporate information resources. In *Proceedings of the International Workshop on Real World RDF and Semantic Web Applications*. Retrieved May 7, 2002, from http://www.cs.rutgers.edu/~shklar/www11/final_submissions/paper3.pdf

Bontempo, C., & Zagelow, G. (1998). The IBM data warehouse architecture. *Communications of ACM, 41*(9), 38-48.

Chaudhuri, S., & Dayal, U. (1997). An overview of data warehousing and OLAP technology. *ACM SIGMOD Record, 26*(1) 65-74.

Coen, G. (2002). Database lexicography. *Data and Knowledge Engineering, 42*(3), 293-314.

Decker, S., Erdmann, M., Fensel, D., & Studer, R. (1999). Ontobroker: Ontology based access to distributed and semi-structured information. In R. Meersman, Z. Tari, S. Stevens, & R. Meserman (Eds.), *Semantic issues in multimedia systems*. Boston: Kluwer Academic Publisher.

Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., & Halevy, A. (2003). Learning to match ontologies on Semantic Web. *The VLDB Journal, 12*(4), 303-319.

Doan, A., Noy, N. F., & Halevy, A. (2004). Introduction to the special issue on semantic integration. *ACM SIGMOD Record, 33*(4), 11-13.

Fellbaum, C. D. (Ed.). (1998). *WordNet: An electronic lexical database*. Cambridge, MA: The MIT Press.

Fowler, J., Perry, B., Nodine, M., & Bargmeyer, B. (1999). Agent-based semantic interoperability in InfoSleuth. *SIGMOD Record, 28*(1), 60-67.

Gardner, S. R. (1998). Building the data warehouse. *Communications of ACM, 41*(9), 52-60.

Gruber, T. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition, 5*, 199-220.

He, B., & Chang, K. (2003). Statistical schema matching across Web query interfaces. In *Proceedings of the ACM SIGMOD International Conference on Management of Data,* San Diego, CA (pp. 217-228). ACM.

Hendler, J., & McGuinness, D. L. (2000). The DARPA agent markup language. *IEEE Intelligent Systems, 15*(6), 67-73.

Jarke, M., Lenzerini, M., & Vassiliou, Y. (1999). *Fundamentals of data warehousing*. Berlin-Heidelberg, Germany: Springer-Verlag.

Kang, J., & Naughton, J. (2003). On schema matching with opaque column names and data values. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 205-216). San Diego, CA: ACM.

Kossmann, D., (2000). The state of the art in distributed query processing. *ACM Computing Surveys, 32*(4), 422-469.

Maedche, A., Motik, B., & Stojanovic, L. (2003). Managing multiple and distributed ontologies on the semantic Web. *The VLDB Journal, 12*(4), 286-302.

Macvittie, L. (2004, September 16). Enterprise information integration suites — Don't fear the data. *Network Computing Magazine*. Retrieved from http://www.nwc.com/showitem.jhtml?docid=1518f2

Ouksel, A. M., & Sheth, A. (1999). Semantic interoperability in global information systems: A brief introduction to the research area and special section. *ACM SIGMOD Record, 28*(1), 5-12.

Rahm, E., & Bernstein, P. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal, 10*(4), 334–350.

Scott, J. (1998). Warehousing over the Web. *Communications of ACM, 41*(9), 64-65.

Sheth, A. (1998). *Changing focus on interoperability in information systems: From system, syntax, structure to semantics. Interoperating Geographic Information Systems*. Boston: Kluwer Publishers.

Sutter, J. R. (1998). Project-based warehouses. *Communications of ACM, 41*(9), 49-51.

Uschold, M, & Gruninger, M. (2004). Ontologies and semantics for seamless connectivity. *ACM SIGMOD Record, 33*(4), 58-64.

Widom, J. (1995). Research problems in data warehousing. In *Proceedings of the 1995 International Conference on Information and Knowledge Management (CIKM)* (pp. 25-30). Baltimore: ACM.

Wilmering, T., Yuan, J., & VanRossum, D. (2003). A metadata architecture for mediated integration of product usage data. In *Proceedings of AUTOTESTCON 2003, IEEE Systems Readiness Technology Conference* (pp. 564-575). Anaheim, CA: IEEE.

Yan, L., Miller, R., Haas, L., & Fagin, R. (2001). Data driven understanding and refinement of schema mappings. In *Proceedings of ACM SIGMOD International Conference on Management of Data* (pp. 485-496). Santa Barbara, CA: ACM.

Endnotes

1 Protégé is an ontology editor developed at Stanford University (http://protege.stanford.edu).

2 OilEd is an ontology editor developed at University of Manchester (http://oiled.man.ac.uk).

Chapter VII

Web Service Integration and Management Strategies for Large-Scale Datasets

Yannis Panagis, University of Patras and
Research Academic Computer Technology Institute, Greece

Evangelos Sakkopoulos, University of Patras and
Research Academic Computer Technology Institute, Greece

Spyros Sioutas, University of Patras, Greece

Athanasios Tsakalidis, University of Patras and
Research Academic Computer Technology Institute, Greece

Abstract

This chapter presents the Web Service architecture and proposes Web Service integration and management strategies for large-scale datasets. The main part of this chapter presents the elements of Web Service architecture, the challenges in implementing Web Services whenever large-scale data are involved and the design decisions and business *process re-engineering steps to integrate Web Services in an enterprise information system. The latter are presented in the context of a case study involving the largest private-sector telephony provider in Greece, where the provider's billing system datasets are utilized. Moreover, scientific work on Web Service discovery is presented along with experiments on implementing an elaborate discovery strategy over real-world, large-scale data. Thereby, this chapter aims to illustrate the necessary actions to implement Web Services in a corporate environment, stress the associated benefits, to present the necessary business process re-engineering procedures and to highlight the need for more efficient Web Service discovery.*

Introduction

Web Services (WS) is one of the few architectures that were unanimously adopted by the information technology (IT) industry. From the first drafts of WS specifications, the WS architecture has been established as the dominating distributed software development paradigm.

In a nutshell, *Web Services* are collections of operations — parts of larger applications — that are *remotely available* through common Web protocols, without posing any restrictions on the platforms at both communication ends.

The Web Services framework consists of essentially three basic components: The *Web Service Description Language* (WSDL), a language to allow formal functional characterization of the provided functionalities, the *Simple Object Access Protocol* (SOAP), a protocol that defines the format of the information interchange, and the *Universal Description, Discovery, and Integration* (UDDI), which is a catalog of Web Service descriptions.

All three of the components just mentioned are specified using extensions to the common XML language. Every WS transaction is taking place over established Web protocols such as HTTP and FTP. The adoption of cross-platform implemented protocols is what has facilitated the wide acceptance of Web Services as a platform for implementing a wide gamut of applications. These range from major services such as business interaction and customer relationship management, to much more limited services such as checking the price of stock quotes and checking in for a flight.

Despite the wide acclaim of the WS architecture, some very important issues arise when implementing Web Services in the context of enterprise application development, especially with large-scale datasets involved. Large dataset processing entails long response times even before the first results are returned to the interested end user. This is because Web Service execution duration varies depending on the current status of the infrastructure supporting it, and according to the available resources executing the Web Service. Consequently, the customer may have to wait extremely long for the Web Service to respond and thus experience frustration. In the latter case, he may even decide to request results from another source with, presumably, higher availability. Web Service availability is also an important issue. More concretely, a mere advertisement of a Web Service does not provide any guarantees about its availability or absence from its expected point of presence. These are only two examples of some shortcomings of the current Web Service framework, which highlight the difficulties of implementing tasks with the use of Web Services.

This chapter aims at presenting the main elements of the Web Service framework as well as to propose a methodology for implementing Web Services in corporate environments that utilize very large, time-sensitive, and resource-intensive data. Firstly, the motivation that led to the development of WS architecture along with a short presentation of the WS framework will be presented. Consequently, the reasons that dictate the use of WS in business environments will be discussed, together with related experience in implementing business applications and important steps-forward in the area of WS discovery. Separate sections are devoted to discussion of the required architectural design for the incorporation of Web Services to a business environment as well as the required business process re-engineering steps that have to be taken. Furthermore, a real-world example is presented, where a Web Service framework was designed to support the requirements of the billing system of a large private-sector telephony corporation. Measurements are also presented that highlight the behavior of Web Service-driven architecture, when carrying out tasks in the presence of large amounts of processed data. Finally, the chapter is concluded with summary of the approach and directions for further research.

Web Services Background

In this section we will provide some introductory information on the Web Services and what led to their development, along with key elements and related standards. An important concept in modern software development is that of *Service Oriented Architecture* or SOA (World Wide Web Consortium [W3C], 2004a). A SOA is an architecture where the implemented distributed system has the following traits: (1) It provides operation transparency; (2) It operates on message-driven communication; (3) Its services are accompanied by service descriptions; and (4) It is platform independent.

Quite a few well-known architectures besides WS also deal with some elements of distributed application development. The list of WS precursors includes, UNIX RPC, Microsoft's COM/DCOM, CORBA and Java RMI. Each has failed to succeed due to several reasons including: very complex implementation and infrastructure requirements; complex and/or proprietary communication protocols; lack of support of large software companies; and platform restrictions.

The Web Service Framework

The WS framework has been developed as an answer to most of the issues mentioned previously. First of all, a *Web Service* is a set of software functionalities — stand-alone or part of a larger software module — that allow the interaction between distant processes over a networked environment, by means of a flexible and cross-platform, XML-based infrastructure.

The Web Service alternative to distributed software development has been adopted by the community due to its flexibility, ease of development, and platform independency, characteristics of the core protocols of WSs, namely XML, HTTP, and related WWW protocols.

As discussed, the protocols that comprise Web Service architecture are HTTP, XML, SOAP, WSDL, and UDDI. HTTP and XML 1.0 (W3C, 2000) are de facto accepted standards for the versatile and platform-neutral exchange of information. HTTP and similar information transfer protocols are just the mediators for the data transfer among distant Web Services (TCP/IP has the same role, for example, for HTTP data transfer). XML, due to its extensibility, provides a wrapper to the required functionalities of the Web Service architecture. The rest of the protocols base their specifications on top of XML and HTTP.

Simple Object Access Protocol, SOAP for short, (W3C, 2003), is a protocol that defines the format of message interchange. This interchange takes place when discovering, binding, or consuming a Web Service.

Web Service Definition Language, WSDL for short (W3C, 2004c), is a language to describe the functionalities of a Web Service and to provide additional details about the ways it can be accessed, the points it can be reached, and so forth.

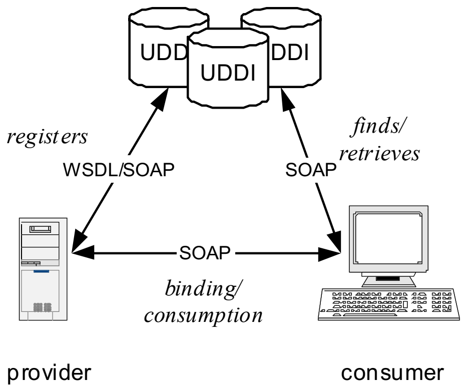
Universal Data, Discovery, and Integration, UDDI for short (OASIS, 2004), defines a separate entity (a registry) that mediates in the development process by hosting descriptions of Web Services.

The next sub-sections refer to the organization of the Web Service architecture with more detail.

The Web Services Functional Model

The typical model under which Web Services operate is the so-called "publish-find-bind" model (Ran, 2003, p. 1). This model consists of three entities: *the provider*, *the consumer*, and *the registry*. The provider registers its services to the UDDI registry using SOAP/WSDL and waits for service requests. A consumer process, queries the registry for a Web Service, using SOAP messages. Note that the queries provided by the standardized UDDI querying APIs address only functional requirements. Efforts to add more intelligence are discussed in a separate section. After retrieving the results, the consumer process determines the binding procedure and from this point on it requests, with SOAP messages, service provision from the provider. The provider ideally responds, executes and returns results in SOAP messages. The process is illustrated in Figure 1.

Figure 1. The Web Services model



Using Web Services in a Business Environment

As argued, Web Services have gained wide use for distributed application development. Nevertheless, it might still not be that clear whether the Web Services architecture is suitable for application development in business and industrial environments. This section provides arguments for the use of Web Services and points out some of their drawbacks.

Contemporary enterprise information systems are becoming increasingly complex and hard to maintain. A way for large corporation networks to reduce the amount of the overall complexity is to distribute computation effort among physically-separated computational nodes. Consequently, the need to execute remotely-located computational processes occurs. Quite a few examples of such practices exist, from Internet banking and e-commerce solutions to phone billing procedures or performing statistical analyses at a company's subsidiary from the central branch and vice versa.

Another ubiquitous need is to coordinate procedure execution, often among different entities. A typical example (Koshutanski & Massacci, 2003) is: A buyer process initiates a request on a seller service with necessary credits provided; the agent on the seller side needs to validate the buyer's credits by initiating a call to a third-party procedure. This illustrates a case where many parties are involved. Coordination of processes may also take place within the same organization. For instance, suppose that a local airline operator needs to check a client's eligibility for a frequent-flyer bonus program prior to the reservation. The operator then needs to call a remote procedure to collect data for the specific client before invoking the appropriate call to the reservation procedure.

A further important factor is flexibility. The term "flexibility" can have multiple interpretations including the ease of incorporation of new business processes, introduction of new external collaborators, or even independence of individual machine platforms at the communication ends.

The Web Services framework, in sharp contrast to the previously appeared, similar distributed protocols, provides satisfying answers to all the earlier requirements. Web Services provide a flexible, conceptually easy, and platform-independent way for process sharing and remote procedure execution. Web Services are easier to consume and do not require a long list of a priori agreements on the communication methods. Furthermore, integration of new processes or partners and extensions to standards are still possible since they are all built on the versatility of XML-flavored specifications. This very attribute is what enables platform independence between endpoints, allowing for extra degrees of freedom. As for the coordination of remote processes, this is natively supported in the recent specifications of Web Service architecture in what is called "Web Service Choreography" (W3C, 2004b, ¶2.3.2.3).

Albeit the Web Service framework, in its initial specifications (W3C, 2004a), exhibits some serious drawbacks. First of all, even if the model is distributed, the discovery part is mostly centralized. UDDI catalogues receive all the requests for WS consumption, which may lead to single point failures and bottlenecks. Furthermore, the entities stored in the registry provide only technically-oriented descriptions of WSs. Therefore, a consumer process may have to "trial and error" while locating which server process to bind. Another drawback is that Quality of Service (QoS) characteristics are not by any means included in the WSDL descriptions. This can create problematic instances in many situations since there is nothing to indicate that, for instance, a link to a tModel is not obsolete or simply a service is not running anymore. Orth (2002) also mentions "accountability" (p. 115) as a loophole of the initial specification, in the sense that enforcing an accounting scheme to the use of a WS is not an easy task. He also argues that WSs lack what would be valuable in enterprise environment, namely "asynchrony" (p. 116).

Security and authentication issues, although of paramount importance in business application development, are not explicitly treated in the Web Service specification. Nevertheless, recent proposals seem to circumvent the lack of security specifications. A short treatment is given by Salz (2003) and a more comprehensive analysis by Siddiqui (2003a, 2003b, 2003c, 2003d). Siddiqui (2003a) presents the motivation and typical deployment scenarios for security in Web Services. Siddiqui (2003b) argues that the main underlying concept for secure WS provision is to provide XML-enabled firewalls, that is, firewalls that can authenticate, certify, and decrypt SOAP messages, which contain extra information in their headers. The introduction of this concept is facilitated by the existence of standards such as XML Digital Signature (XMLDS) (W3C, 2002a), a standard to cater for the integration of signatures to XML documents, and XML Encryption (W3C, 2002b). In this vein, XMLDS and XML Encryption can also be included in SOAP messages to provide message integrity, authentication, and content encryption. Actually, this is proposed in the WS-Security Specification by IBM (IBM, 2002). Security Assertion Markup Language (SAML) (OASIS, n.d.) can also be used to allow authentication sharing between interacting applications.

Related Work

The WS standards are rapidly maturing due to the wide acceptance of the WS framework as a standard for remote execution of processes and inter-business collaboration. The current standards that comprise the WS architecture on the XML part are: XML 1.0 (W3C, 2000), XML Namespaces (W3C, 1999), and XML schema (W3C, 2001a; W3C, 2001b). As for the layers comprising the WS framework, the SOAP format is currently in version 1.2 (W3C, 2003), the new version of WSDL, version 2.0, was very recently published (W3C, 2004c), and UDDI has reached version 3.0.2 (OASIS, 2004).

The need for: (1) expressing business workflows in terms of Web Services, and (2) interoperability among different corporate workflows, was recognized relatively early. In this respect, vendor- specific implementations exist, such as Microsoft's XLANG (Microsoft, 2001), Java Enterprise Java Beans, and .NET C#, whereas languages to control business processes have also been developed, including *Business Process Execution Language for WS* (BPEL4WS) (Curbera, Goland, Klein, Leymann, Roller, Thatte, et al., 2002), *WS Flow Language* (WSFL) (IBM, 2001), and *WS Business Process Execution Language Version 2.0* (WS-BPEL) (OASIS, 2004). A very important recent development on the choreography field is the announcement of the *Web Services Choreography Description Language Version 1.0* (W3C, 2004b). This initiative aims to serve as an additional layer on top of the corporate workflows in order to orchestrate their interactions.

The WS Discovery procedure has also received much attention from major software platforms. Windows 2003 has a UDDI server preinstalled with the OS, whereas many J2EE vendors build UDDI instances into their application servers (Lacey, 2004). Sun Microsystems has also included its Java API for XML Registries (JAXR) as a single, general purpose API for interoperating with multiple registry types. JAXR allows its clients to access the Web Services provided by a Web Services implementer, by exposing Web Services built upon an implementation of the JAXR specification. The UDDI specifications do not directly define a Java-based API for accessing a UDDI registry. The Programmer's API specification only defines a series of SOAP messages that a UDDI registry can accept. Thus, a Java developer who wishes to access a UDDI registry can utilize a number of ways:

1. Through a Java-based SOAP API,
2. Through a custom Java-based UDDI client API, or
3. Through JAXR.

Microsoft had originally developed its own standard for WS discovery, with DISCO, in the form of a discovery (DISCO) file. A published DISCO file is an XML document with links to other resources describing the Web Service. Since the widespread adoption of UDDI, however, Microsoft has been supporting it instead of DISCO, in order to maximize interoperability.

Handling Large Datasets: Motivation and Architecture

This section discusses the challenges of large and "online" data management, especially in the sector of telecommunication service providers. Before presenting details of the proposed approach, a couple of paradigms of traditional large data sets management will be presented to illustrate that existing solutions can be inadequate. In particular, the case of large data set management is usually handled using performance tuning and orchestration at the back-end enterprise DBMSs. This solution might be effective in configurations where departments of a single enterprise are participating in the process of handling the large data sets. This usually involves a unique type of Enterprise Resource Planning (ERP) system.

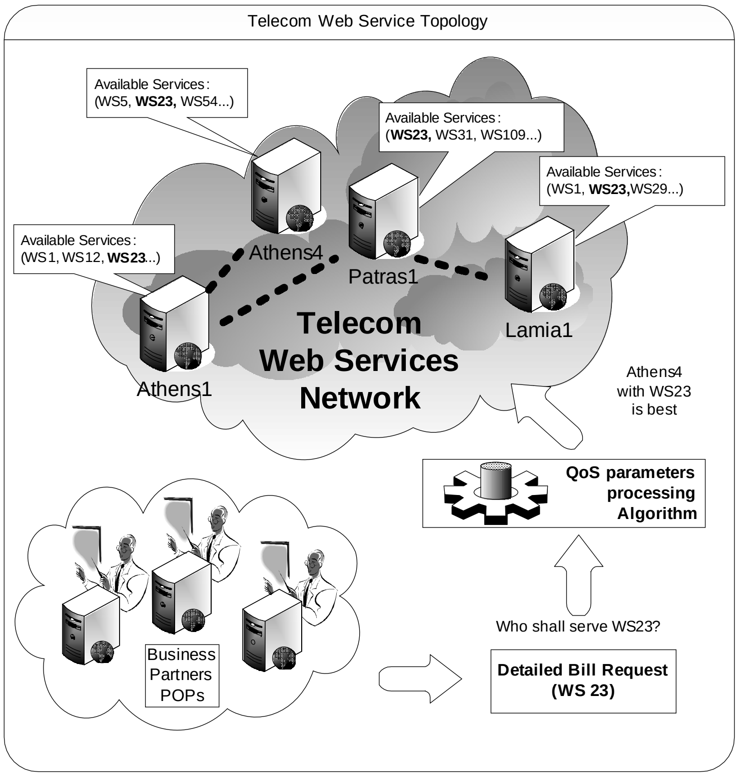
Actually, this is the way that bank branches communicate and interchange data with other branches of the same bank, since they share a common semantic representation in business and exact database schema of their data. In this case, the notion of "online" data availability exists. It is also commonplace for the customer to demand full analysis of his bank-account transactions without having to wait at all. The exchange of large data sets becomes quite complicated when different banks are involved. Additionally, it turns out to be even harder to interchange data between banks and enterprises of other commercial industries/service providers. In that event, WS technology provides a non-transparent solution for data sets' communication and interchange. The "online" notion loses its actual meaning in these cases and a customer may even have to wait for a whole day to receive the analysis of his account (e.g., whenever his transactions were performed in foreign country banks or even at branches of different banks).

However, Web Services penetration in enterprises has reached several more industrial and business environments. In the latter, time-sensitive and data-intensive transactions using Web Services are required between the customers and the provided services/products. Web Service technology does not include a standardized mechanism to deal with such "online" cases. Web Services have been designed to serve a loosely-coupled interconnection of enterprises without dealing with quality of service (QoS) parameters such as execution time and response duration.

This inadequacy becomes an obstacle in businesses such as telecommunication service providers. A common example is the delivery and management of customers' detailed bill records of telephone calls. There are carriers that have to handle millions of bills and billions of transactions per hour. In the DBMS world, these numbers of records can possibly be handled efficiently and even "online" with advanced, well-known strategies and mechanisms such as replication, clustering, and network sharing. Nevertheless, Web Services are not DBMSs but only an interconnection technology, which unfortunately lacks QoS. In this example, a Web Service would have to patiently search for all the unbilled calls of a whole day's transactions to return detailed analysis of customer calls of the day. Such a Web Service would have to wait even more in order to evaluate and respond to "the call-billing" service. Delay overheads occur, because the back-end DBMS has to deal with very large data sets of gigabytes per hour. The scenario is getting even more complicated when QoS parameters have to be incorporated such as availability of the Web Service and Internet-working performance between the consumer of the Web Service and the WS provider. As a result, QoS specifications have to be taken into consideration when consuming a Web Service. In this work, the efficient and effective QoS-enabling techniques proposed in Makris, Panagis, Sakkopoulos, and Tsakalidis (2004) have been applied. A high-level architectural view of this approach is displayed in Figure 2. The most crucial component in the introduced architecture lies in the implementation of the QoS-enabling selection algorithm. A detailed analysis of the QoS-aware algorithm of Makris, Panagis, Sakkopoulos, and Tsakalidis (2004) goes beyond the scope of this chapter. In the paragraphs to follow, we highlight the basic underlying concepts.

Two main QoS factors are taken into account: Network delay between requester and provider, and the number of distinct functions implemented at a potential WS provider. First of all, the set of candidate WS provider nodes is

Figure 2. General architecture of the proposed solution for the Telecom case


Telecom Web Service Topology
Available Services: (WS5, **WS23,** WS54...)
Available Services: (**WS23,** WS31, WS109...)
Available Services: (WS1, **WS23,**WS29...)
Available Services: (WS1, WS12, **WS23**...)
Athens4
Patras1
Lamia1
Athens1
Telecom Web Services Network
Athens4 with WS23 is best
QoS parameters processing Algorithm
Who shall serve WS23?
Detailed Bill Request (WS 23)
Business Partners POPs

identified. This set is pruned by ruling out nodes that are sub-optimal according to the two QoS criteria, an indication that there is at least one node with better QoS performance than them. The pruning step is carried out via a simple geometric observation. In the sequel, the authors select the node with the smallest "projected" execution time; the actual execution time cannot be foreseen, but instead each node keeps an execution history, which the authors leverage to calculate projected execution times. The calculation process requires querying the remaining nodes for their current state. A heuristic is also presented, in order to avoid too many network traffic overhead, during the calculations.

Apart from efficient selection, a synthesis of all applications that the enterprise needs has to be designed, using Web Service management and performance tuning techniques. In this way, the resulting system will boost productivity and response time, and therefore, increase customer satisfaction while reducing administrative costs. Key features of the solution proposed are: (1) the object-oriented method followed for smooth and swift incorporation of the re-engineered processes with the already existing infrastructure; (2) the adaptive Web-based environment utilized in both intranet and extranet users; and (3) the Web-unification of all the enterprise applications under the common umbrella of a single architecture using Web Services.

Functional and Operational Desiderata

The functional and operational requirements of the proposed solution of the case study are discussed in this section. In the sequel, we present the list of prime key requirements included in the design of the technological environment. After a series of interviews with telecommunication carrier top IT executives and the distribution of a two-page-long questionnaire to the telecommunication IT department (12 computer and telecom engineers), the following functional specifications were outlined:

- Handling of Web Services for data interchange with a wide enterprise collaborators' national network of 13,000 points of presence (PoP).
- Implementation and orchestration of Web Services that manage and deliver sensitive data directly into the infrastructure of the provided telecom services.
- Online customer document management, nationwide, for any PoP of the enterprise.
- Enabling communication and business with four different hardware suppliers to facilitate single-day request, delivery, and installation of end-user VoIP gateways and DSL routers.
- Establishing a technical knowledge base for Web Service support.

From an operational perspective, the previous specifications imply several important details that affect Web Services providing the corresponding services-operations:

- *Management of Services.* In the operations concerning the management of the provided customer services and the creation of new ones, we have designed a framework that covers a number of different configuration options. Each process covers completely the creation, activation, interconnection and de-activation or deletion of the corresponding service. Management of services influences directly the interconnection with the identification code of each customer.

  The reader should take into consideration that the majority of both switched and VoIP telephony hardware supports mainly text-based database interfaces and scarcely partial database interfaces. Consequently, database communication and processing performance is severely degraded and the corresponding Web Services have to wait for several minutes to receive a result.
- *Administration of the Detailed, Bill-Issuing Procedure.* Producing detailed bill records for a telecommunication carrier is a quite complicated task. In high level it includes the issuing and the distribution of bills. Detailed bills are issued and distributed either using a pre-defined time schedule or in an ad-hoc sense, whenever a customer requests a detailed bill snapshot (for call transactions that are not included in previous bill issues).
- *Document Management.* The document management system includes tens of millions of scanned documents. These documents include customer applications, previous bill issues, and other sensitive documents, such as ID, customer passport, and so forth. They have to be delivered through Web Services supporting high security standards for authorization and authentication. It is further required to use a limited number of central points, as well as efficient load-balancing techniques.

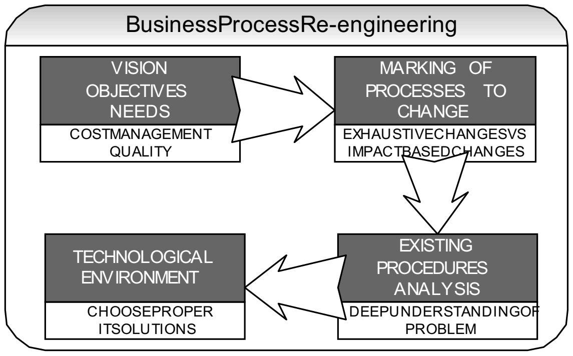
Business Rules Re-Engineering and Integration

This section describes the business data management processes that have to be followed for the organization and execution of legacy corporate tasks with the use of Web Services. Teng, Grover, and Fielder (1994) described Business Process Re-engineering (BPR) as the process during which thorough analysis and complete redesign of current business processes is performed in order to reach major improvements in performance. In this case, the introduction of Service Oriented Architecture in the Information Technology systems has followed all the discrete features of BPR as described in Davenport (1993). All involved processes have been radically changed having as a starting point a clean state. Overall, the design and implementation time required for the processes transformation lasted less than a semester. However, the processes that involved all 13,000 national business partners had been later revisited for further enhancements. The scope of changes was broad and cross-functional. Especially, in the case of re-engineering the "management of services" process, this task had practically had its effects across all productive sectors of the enterprise. As a natural consequence, BPR was coupled with structural changes in the enterprise. Before the integration of Web Services, only the positions of managing director and managers existed. The most important structural "side-effect" has been the appointment of a new General Manager (GM) in the enterprise. In order to ensure effective implementation of the IT-based processes, the enterprise chose a new GM with long experience in the implementation of IT-oriented services from the Department of Information Technology and Service Implementation.

To successfully deliver Web Services, the following three-layer logic is assumed, following the business process re-engineering roadmap of Figure 3.

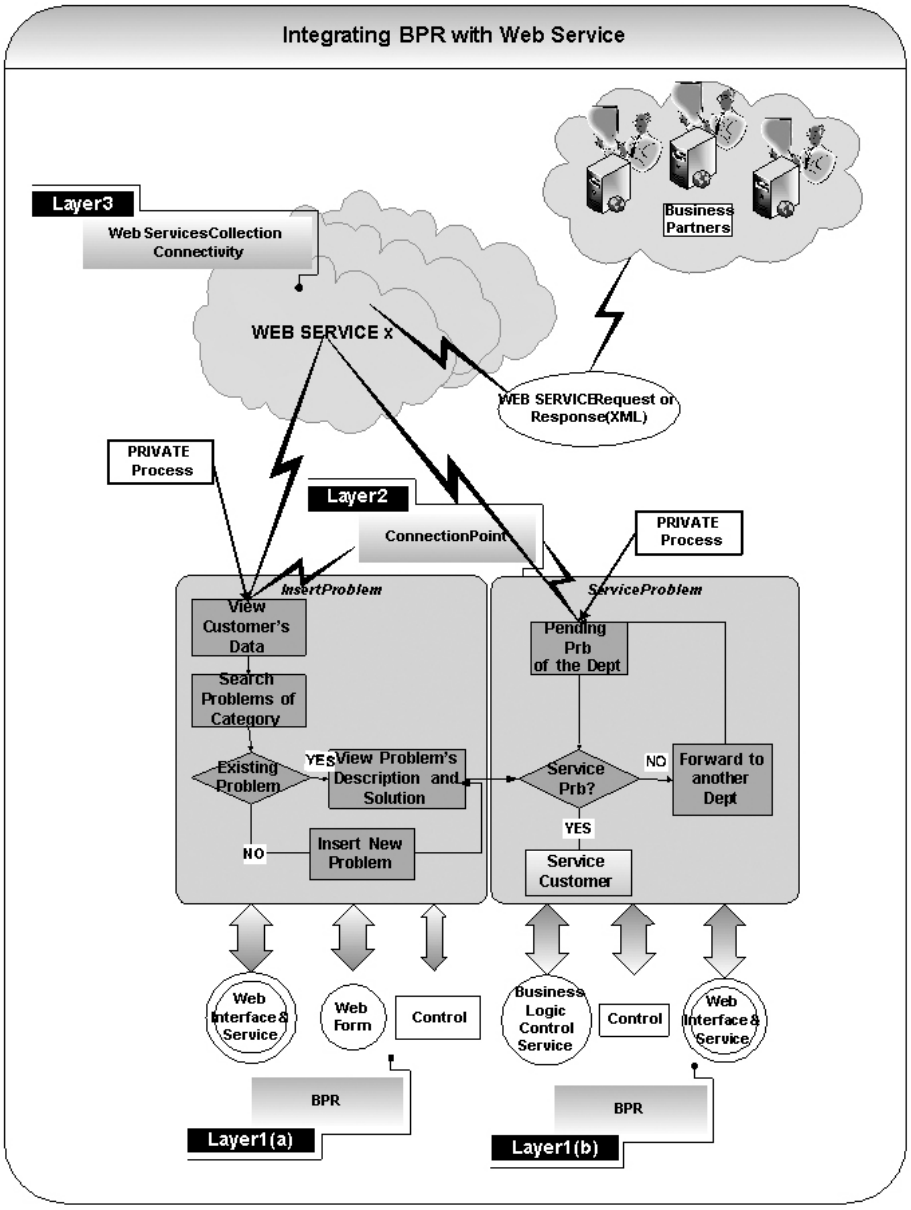
Initially, integration of the re-engineered business processes is achieved. Next, the connection points between Web Services and business action points are identified and interconnected. Finally, publishing, testing, and adjustment of those Web Services is performed to put them in business. The three different layers are depicted in Figure 4. A piece of a simplified business process

Figure 3. Business process re-engineering roadmap


BusinessProcessRe-engineering
VISION OBJECTIVES NEEDS
COSTMANAGEMENT QUALITY
MARKING OF PROCESSES TO CHANGE
EXHAUSTIVECHANGESVS IMPACTBASEDCHANGES
EXISTING PROCEDURES ANALYSIS
DEEPUNDERSTANDINGOF PROBLEM
TECHNOLOGICAL ENVIRONMENT
CHOOSEPROPER ITSOLUTIONS

flowchart is presented with a corresponding outline of the final implementation. Furthermore, the internal connection points are located as the potential access points for a user coming from the external hyperspace (indicated on the same figure as "layer 2"). Ultimately, the infrastructure presented as "layer 3",

Figure 4. Integration procedure using BPR for problem-solving Web Services



supports the Web Service that would enable all customers and associates to take advantage of the company's business services.

Selecting and Consuming Web Services

Since an important subtask preceding the consumption or invocation of Web Services is their selection, this section will deal with the selection components of the Web Service architecture. The UDDI approach will be presented, and the main problems arising by its use will be pointed out. More elaborate and efficient in terms of search quality approaches will also be presented. For a more comprehensive treatment see Garofalakis, Panagis, Sakkopoulos, and Tsakalidis (2004).

Architectural Issues: UDDI, Its Flavors, and Alternatives

The standard UDDI discovery approach requires the potential user to make use of the available UDDI query API or query interface to provide description keywords or part of the function name to the registry. Consequently, the user is faced with a list of results, related or unrelated, which can be browsed to check their availability and functional requirements. Several shortcomings occur from this simplistic viewpoint mainly due to several kinds of heterogeneities that we are faced with: (1) technological heterogeneities (different platforms or different data formats), (2) ontological heterogeneities (domain-specific terms and concepts within services that can differ from one another, especially when developed by different vendors), and (3) pragmatic heterogeneities (different development of domain-specific processes and different conceptions regarding the support of domain-specific tasks).

More importantly perhaps, the current registry-discovery approach does not take any quality of service parameters into account. A very common example has its analogue in the case of a broken link in the returned results of a Web search engine; nothing can guarantee that an advertised WS truly exists.

In Lacey (2004) an approach to include quality information (e.g., expected meantime between failures, maximum response time, maximum data throughput, etc.) to white and yellow UDDI pages is proposed. Overhage and Thomas (2003) proposed a similar kind of changes applied to green pages. Some work has also been conducted in the area of dynamic binding with the introduction of the Active UDDI (Jeckle & Zengler, 2003). Moreau, Avila-Rosas, Dialani, Miles, and Liu (2002) give an alternative approach to the discovery of Web Services in the context of grid-based agent system; the UDDI is extended to allow agent descriptions to fit in and to support ontology-based pattern matching.

Several papers deal with decentralizing or distributing the UDDI registry. These efforts fall into two categories: construction of UDDI federations, and Peer-to-Peer (P2P) solutions. The concept of federation is to have several UDDI nodes connected together, forming a service that, while appearing to be virtually a single component, is composed of an arbitrary number of operator nodes (Rompothong & Senivongse, 2003). An operator node is responsible for the data published at a particular node; in UDDI terms, it is the "custodian" of that part of the data.

The P2P approach has gained more attention. The majority of related work is based on a well-known, dynamic, load-balanced P2P protocol, the Chord (Stoica, Morris, Karger, Kaashoek, & Balakrishnan, 2003). The key concept in P2P solutions is to allow all the participants to enter the discovery game by hosting some part of the service descriptions.

Retrieval Models for Web Services

This section reviews proposed methodologies for modeling Web Service description in order to enhance the retrieval process. Two approaches are discussed: this of information retrieval approach and that of Semantic Web.

Information Retrieval Based Representations

The simplest data model is the *Keyword-Based*. This model is followed by the legacy UDDI standard and the discovery mechanism it supports. The provider supplies textual descriptions recorded in the business entities section. Moreover, tModels are used as texts representing Web Services. The query keywords are then matched against the stored descriptions. This approach is the Web Services equivalent of the classic Boolean information retrieval model; see, for example Baeza-Yates and Ribeiro-Neto (1999, chap. 2).

A direct extension is to exploit different IR models such as the vector-space model; see again Baeza-Yates and Ribeiro-Neto (1999, chap. 2). That was exactly the approach of Sajjanhar, Hou, and Zhang (2004). They represent WS textual descriptions as document column-vectors. A description text, thus, corresponds to a vector $\vec{d}$ in the vector space spanned by all the terms used in all Service description texts. These vectors are collectively represented in a term-document matrix. *Latent Semantic Indexing* (Berry, Dumais, & O'Brien, 1995) is used in the sequel, to reduce the dimensionality of the vector space and map query vectors more closely to WS vectors.

A Web Service modelled as a d-dimensional vector, can also be thought of as a point in d-dimensions. In this respect, a geometric data structure for indexing multi-dimensional data can be deployed in indexing and querying Web Services. Multi-dimensional data structures are rather hard to implement in a distributed or a P2P environment. Therefore, Schmidt and Parashar (2004) use a transformation, which injectively maps points in higher dimensions, to numbers. This transformation is called *space-filling curve*. In the system of Schmidt and Parashar (2004), a unique ID is generated for each Web Service, through the Hilbert curve mapping. The IDs are then stored in a Chord (Stoica et al., 2003) of Web Service peers. An interesting merit of the given approach is that it can efficiently support partial match queries, due to Hilbert mapping. Other work on P2P Web Service discovery efforts includes that of Li, Zou, Wu, and Ma (2004).

Semantic Web Approaches

The semantics community has devoted great effort in enriching WS description standards with semantic descriptions so that ontology-based matching can be performed. This approach has resulted in specialized, ontology-description languages for Web Services, such as DAML-S (The DAML-S Coalition, 2002) and its successor OWL-S (The OWL Services Coalition, 2003). In this vein, Paolucci, Kawamura, Payne, and Sycara (2002) present a framework to allow WSDL and UDDI to perform semantic matching. Web Services are modelled as ontologies, ontologies are described by other non-functional attributes, and a specialized ontology, the DAML-S Matchmaker, undertakes the binding procedure. The matchmaker does not extend any of the UDDI page categories; it is treated as an add-on, which undertakes semantic matching and the mapping of ontologies to UDDI descriptions.

A recent development was the introduction of a new language, namely OWL-S, to combine semantic annotation of Web Services with their discovery and invocation with WSDL and SOAP (Sycara, 2004). This approach has led to the OWL-S Matchmaker module. Implementation frameworks for semantic WS annotations are provided by Hu (2003) and Overhage and Thomas (2003). Hu describes a semantic implementation of WS in an industrial environment, where data are modelled by domain ontologies and operates on them with operation ontologies. Ontology representation, relationship encoding, and binding are coded in XML, which allows incorporation to UDDI entities. Overhage and Thomas (2003), provide an extension to UDDI, the E-UDDI, by introducing "blue pages," pages that contain semantic descriptions in DAML-S.

Case Study Implementations and Evaluation

The architectural approach outlined in Figure 2 has been based on the needs of the largest private-sector telecom carrier in Greece. It supports Web Services for processing large-scale datasets with domain-specific enhancements in the discovery strategies according to Makris, Panagis, Sakkopoulos, and Tsakalidis (2004). The technologies utilized for the implementation of the mechanisms, the Web Services, and the evaluation procedures include the MS .NET framework version 1.1 (Microsoft "Microsoft.NET framework", n.d.) and the C# language (Microsoft, "Microsoft C#", n.d.). The .NET framework has been chosen by the end-user enterprise as a strategic development and solution platform. Without losing its generality, the proposed approach can be implemented utilizing any Web Service development platform.

Evaluation results are examined to determine the quality of the proposed architecture. Two main phases are followed: verification and validation of the outcomes. Verification refers to building the environment correctly that is substantiating that a system correctly implements its specifications. Validation refers to building the right system that is substantiating that the system performs with an acceptable level of accuracy. During verification phase, software architects tested the proposed environment in the laboratory. For this purpose, test cases were performed. As the goal of these tests was to achieve a representative cross-sectional test of the system, test scenarios were designed to cover a broad range of potential user inputs (including common as well as rare-use cases). Among the validation methodologies available, we opted for

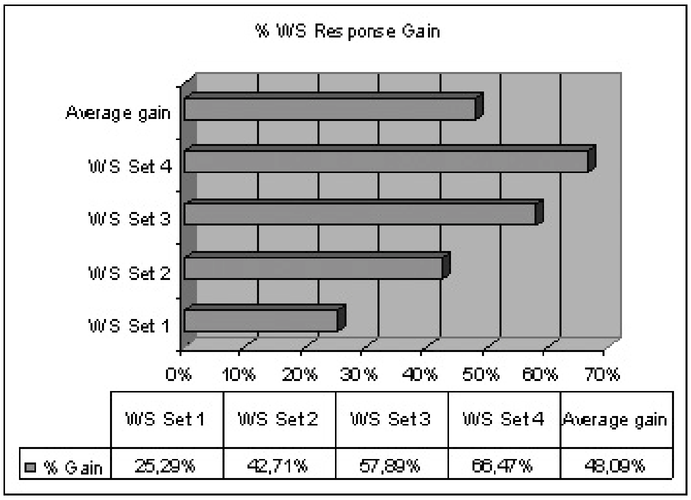
Table 1. Experimental sets utilized

	Execution Time (sec)	Network Time (sec)
WS Group 1	10 up to 60	1 up to 15
WS Group 2	30 up to 320	1 up to 30
WS Group 3	60 up to 600	1 up to 30
WS Group 4	180 up to 3600	10 up to 60

quantitative validation. Four sets of experiments were designed in the laboratory in order to measure the efficiency of the proposed solution. The experiments treated the involved Web Services in groups with similar characteristics to facilitate comparison of the resulting measurements. The different groups of WS formed are depicted in Table 1, together with their characteristics. In particular, the sets were differentiated in terms of the data volume involved (expressed indirectly in terms of execution time to process the data) and the network latency to deliver the data to the requester (cf. Table 1).

In the corporate environment, the mean network latency is far less than the mean execution time. This is true due to the fact that Web Services dealing with telecommunication call-log data sets need to perform several series of cross-checks and operations. Despite that, the number of call transactions is limited only to some hundreds of calls per phone number and, therefore, it takes relatively shorter time for the service to be delivered. Experiments include also a scenario where execution duration is close to network latency. The presented comparison depicts the execution time of the WS selected by the proposed algorithm in Makris, Panagis, Sakkopoulos, and Tsakalidis (2004) vs. the WS's execution time based on common and typical Web Service technologies, such as UDDI-based selection. These measurements only highlight the default behavior when carrying out tasks in the presence of large amount of processed data, in contrast to the proposed Web Service-driven architecture. When adopting the proposed architecture, we can observe an average of 48.09% gain in response time, as illustrated in Figure 5. However, in the operational environment, even higher gain is expected as the Web Services used can be mainly classified to groups 3 and 4 of Table 1, which indicate the presence of very large data sets.

Figure 5. Response time gain with intelligent selection and execution of Web Services: The larger the data set handled, the greater the percentage gain is



	WS Set 1	WS Set 2	WS Set 3	WS Set 4	Average gain
▨ % Gain	25,29%	42,71%	57,89%	66,47%	48,09%

Future Steps

Future work includes incorporation of quality of Web Service metrics into processes that involve Web Service and database interconnection. This would enable IT solution providers to perform tuning of Web Service-oriented environments just like they do in the case of DBMS. This can radically change the performance and allow further penetration of the Web Services standardization framework. Additionally, dealing with performance tuning of Web Service workflows and non-composite Web Services is also an open issue. There is a long roadmap towards standardizing and interconnecting series of Web Services that depend on each other's results. At the end it will allow the enterprises to take best advantage of virtual service providers and efficiently outsource parts of their operational processes. Outsourcing will allow them to cut down on software and hardware expenses. In fact, such outsourcing would upgrade service response time, especially for large, database-dependant processes. Processes of this sort require very expensive hardware and software that only a virtual service provider can offer in cost-effective solutions.

Conclusion

Throughout this chapter, Web Services, an emerging architecture for building distributed corporate applications, was presented. Web Services offer the chance to build robust, loosely-coupled and platform-independent applications, boosting thus the adoption of the Web Services framework by the software engineering community. Web Service design and deployment in real-life scenarios was also presented, providing both the design decisions that allowed incorporation of legacy corporate processes and the outline of the adopted architectural design. In the case presented, Web Services provided the needed cross-platform solution. Web Services provided the necessary framework for inter-operation and non-transparent data exchange with a number of suppliers (procurement) and collaborating service providers. Additionally, the need for compatibility and interconnection with different ERP systems and other billing systems of thousands of business partners has been fulfilled using the XML-based Web Service Architecture.

Moreover, the presence of data-intensive tasks, as those carried out in a telecom business environment, highlights several weaknesses of the standardized WS framework; most notably the lack of QoS guarantees as well as the lack of true WS retrieval rather than a mere matching of service specifications. Some of the approaches to tackle the retrieval problem were presented. We have also presented experimental results on large datasets that merely indicate the margin for improvement in the enforcement of a QoS policy upon the WS framework.

The WS framework has helped to untangle the complex process of developing distributed, large-scale, inter-corporate applications. However, there is still need for the standardization to follow the surge for QoS provisioning and improved retrieval capabilities.

Acknowledgments

Part of this research has been supported by the Karatheodoris Research Program (B391) of the University of Patras. We would like to thank the anonymous reviewers for their valuable suggestions to improve the presentation of this chapter.

References

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. New York: Addison-Wesley.

Berry, M. W., Dumais, S. T., & O'Brien, G. W. (1995). Using linear algebra for intelligent information retrieval. *SIAM Review, 37*(4), 573-595.

Curbera, F., Goland, Y., Klein, J., Leymann, F., Roller, D., Thatte, S., et al. (Eds.). (2002). *Business process execution language for Web Services (BPEL4WS), Version 1.0*. BEA, IBM, Microsoft. Retrieved January 3, 2005, from http://www-106.ibm.com/developerworks/webservices/library/wsbpel/

Davenport, T. H. (1993). *Process innovation*. Boston, MA: Harvard Business School Press.

Garofalakis, J., Panagis, Y., Sakkopoulos, E., & Tsakalidis, A. (2004). Web Service discovery mechanisms: Looking for a needle in a haystack? (Electronic Version). In *International Workshop on Web Engineering: Hypermedia Development & Web Engineering Principles and Techniques: Put Them in Use, in conjunction with ACM Hypertext 2004*, Santa Cruz, CA. Retrieved January 5, 2005, from http://www.ht04.org/workshops/WebEngineering/ HT04WE_Garofalakis.pdf

Hu, Z. (2003). Using ontology to bind Web Services to the data model of automation systems. In A. B. Chaudhri, M. Jeckle, E. Rahm, & U. Unland (Eds.), *Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems, Lecture Notes in Computer Science* (vol. 2593, pp. 154-168). Berlin; Heidelberg: Springer Verlag.

IBM. (2001). *Web Services Flow Language (WSFL) Version 1.0*. Retrieved January 3, 2005, from http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

IBM. (2002). *Web Services Security*. Retrieved January 3, 2005, from http://www-106.ibm.com/developerworks/webservices/library/ws-secure/

Jeckle, M., & Zengler, B. (2003). Active UDDI — An extension to UDDI for dynamic and fault-tolerant service invocation. In A. B. Chaudhri, M. Jeckle, E. Rahm, & U. Unland (Eds.), *Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems, Lecture Notes in Computer Science* (vol. 2593, pp. 154-168). Berlin; Heidelberg: Springer Verlag.

Koshutanksi, H., & Massacci, F. (2003). An access control framework for business processes for Web Services. In *Proceedings of the ACM Workshop on XML Security* (pp. 15-24). Fairfax, VA: ACM.

Lacey, P. (2004, February). UDDI & dynamic Web Service discovery. *Doctor Dobb's Journal*. Retrieved May 3, 2004, from http://www.ddj.com/articles/2004/0402

Li, Y., Zou, F., Wu, Z., & Ma, F. (2004). PWSD: A scalable Web Service discovery architecture based on peer-to-peer overlay network. In Y. J. Xu, X. Lin, X. Lu, & Y. Zhang (Eds.), *Proceedings of the Asian-Pacific Web Conference 2004*, *Lecture Notes in Computer Science, Vol. 3007* (pp. 291-300). Springer Verlag.

Makris, C., Panagis, Y., Sakkopoulos, E., & Tsakalidis, A. (2004). *Efficient search algorithm for large scale Web Service data* (Tech. Rep. No. CTI TR 2004/09/01). Patras, Greece: Research Academic Computer Technology Institute (RACTI). Retrieved January 3, 2005, from http://www.ru5.cti.gr/reports/CTI-TR-2004-9-1.pdf

Microsoft. (2001). *XLANG Web Services for business process design*. Retrieved January 5, 2005, from http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm

Microsoft. (n.d.). *Microsoft .NET Framework Developer Center*. Retrieved, April 29, 2005, from http://msdn.microsoft.com/netframework/

Microsoft. (n.d.). *Microsoft Visual C# Developer Center*. Retrieved, April 29, 2005, from http://msdn.microsoft.com/vcsharp/

Moreau, L., Avila-Rosas, A., Dialani, V., Miles, S., & Liu, X. (2002). Agents for the grid: A comparison with Web Services (Part II: Service discovery). In *Proceedings of Workshop on Challenges in Open Agent Systems* (pp. 52-56). Retrieved January 3, 2005, from http://www.agentcities.org/Challenge02/Proc/Papers/ch02_51_avila-rosas.pdf

OASIS. (2004). *UDDI Version 3.0.2*. Retrieved January 3, 2005, from http://uddi.org/pubs/uddi-v3.0.2-20041019.htm

OASIS. (2004). *Web Services Business Process Execution Language Version 2.0*. Retrieved January 3, 2005, from http://www.oasis-open.org/committees/download.php/10347/wsbpel-specification-draft-120204.htm

OASIS. (n.d.). *OASIS Security Services (SAML) Technical Committee, Security Assertion Markup Language (SAML) v2.0 Information*.

Retrieved April 10, 2005, from http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security#samlv20

Orth, T. (2002). *The Web Services framework: A survey of WSDL, SOAP and UDDI*. Master's thesis, TU Wien, Wien, Austria.

Overhage, S., & Thomas, P. (2003). On specifying Web Services using UDDI improvements. In A. B. Chaudhri, M. Jeckle, E. Rahm, & U. Unland (Eds.), *Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems, Erfurt, Germany, Lecture Notes in Computer Science* (vol. 2593, pp. 100-119). Berlin; Heidelberg: Springer Verlag.

Paolucci, M., Kawamura, T., Payne, T. R., & Sycara, K. (2002). Semantic matching of Web Services capabilities. In I. Horrocks, & J. A. Hendler (Eds.), *Proceedings of the 1st International Semantic Web Conference, Lecture Notes in Computer Science* (vol. 2342, pp. 333-347). Berlin; Heidelberg: Springer Verlag.

Ran, S. (2003). A model for Web Services discovery with QoS. *ACM SIGecom Exchanges, 1*(4), 1-10.

Rompothong, P., & Senivongse, T. (2003). A query federation of UDDI registries. In *Proceedings of the 1st International Symposium on Information and Communication Technologies* (pp. 561-566). Dublin, Ireland: ACM International Conference Proceeding Series.

Sajjanhar, A., Hou, J., & Zhang, Y. (2004). Algorithm for Web Services matching. In Y. J. Xu, X. Lin, X. Lu, & Y. Zhang (Eds.), *Proceedings of the Asian-Pacific Web Conference 2004, Lecture Notes in Computer Science* (vol. 3007, pp. 665-670). Berlin; Heidelberg: Springer Verlag.

Salz, R. (2003). Securing Web Services. *O'Reilly Web Services.XML*. Retrieved April 10, 2005, from http://www.xml.com/pub/a/ws/2003/01/15/ends.html

Schmidt, C., & Parashar, M. A. (2004). A peer-to-peer approach to Web Service discovery. *World Wide Web: Internet and Web Information Systems, 7*, 211-229.

Siddiqui, B. (2003a, March 4). Web Services security, Part 1. *O' Reilly Web Services.XML*. Retrieved April 10, 2005, from http://www.xml.com/pub/a/ws/2003/03/04/security.html

Siddiqui, B. (2003b, April 1). Web Services security, Part 2. *O' Reilly Web Services.XML*. Retrieved April 10, 2005, from http://www.xml.com/pub/a/ws/2003/04/01/security.html

Siddiqui, B. (2003c, May 13). Web Services security, Part 3. *O'Reilly Web Services.XML*. Retrieved April 10, 2005, from http://www.xml.com/pub/a/ws/2003/05/13/security.html

Siddiqui, B. (2003d, July 22). Web Services security, Part 4. *O'Reilly Web Services.XML*. Retrieved April 10, 2005, from http://www.xml.com/pub/a/ws/2003/07/22/security.html

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., & Balakrishnan, H. (2003). Chord: A scalable peer-to-peer lookup service for Internet applications. *IEEE/ACM Transactions on Networking, 1*(11), 17-32.

Sycara, K. (2004). Dynamic discovery, invocation, and composition of semantic Web Services. In A. Vouros, & T. Panayiotopoulos (Eds.), *Proceedings of Methods and Applications of Artificial Intelligence, Third Helenic Conference on AI, SETN 2004*, *Lecture Notes in Computer Science* (vol. 3025, pp. 3-12). Berlin; Heidelberg: Springer Verlag.

Teng, J. T. C., Grover, V., & Fielder, K. D. (1994). Business process reengineering: Charting a strategic path for the information age. *California Management Review, 36*(3), 9-31.

The DAML-S Coalition. (2002). DAML-S: Web Service description for the semantic web. In I. Horrocks, & J. A. Hendler (Eds.), *Proceedings of the 1st International Semantic Web Conference, Lecture Notes in Computer Science* (vol. 2342, pp. 348-363). Berlin; Heidelberg: Springer Verlag.

The OWL Services Coalition. (2003). *OWL-S: Semantic Markup for Web Services.* Retrieved January 3, 2005, from http://www.daml.org/services/owl-s/1.0/owl-s.html

W3C. (1999). *Namespaces in XML* (W3C). Retrieved January 3, 2005, from http://www.w3.org/TR/REC-xml-names

W3C. (2000). *Extensible markup language (XML) 1.0* (W3C Recommendation, 2nd ed.). Retrieved January 3, 2005, from http://www.w3.org/TR/2000/REC-xml-20001006

W3C. (2001a). *XML schema part 1: Structures* (W3C). Retrieved January 3, 2005, from http://www.w3.org/TR/xmlschema-1/

W3C. (2001b). *XML schema part 2: Datatypes* (W3C). Retrieved January 3, 2005, from http://www.w3.org/TR/xmlschema-2/

W3C. (2002a). *XML-signature syntax and processing* (W3C). Retrieved January 3, 2005, from http://www.w3.org/TR/xmldsig-core/

W3C. (2002b). *XML encryption syntax and processing* (W3C). Retrieved January 3, 2005, from http://www.w3.org/TR/xmlenc-core/

W3C. (2003). *SOAP Version 1.2 part 1: Messaging framework* (W3C). Retrieved January 3, 2005, from http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

W3C. (2004a). *Web Services architecture* (W3C). Retrieved January 3, 2005, from http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

W3C. (2004b). *Web Services choreography description language Version 1.0* (W3C). Retrieved January 3, 2005, from http://www.w3.org/TR/ws-cdl-10/

W3C. (2004c). *Web Services description language (WSDL) Version 2.0 part 1: Core language* (W3C). Retrieved January 3, 2005, from http://www.w3.org/TR/wsdl20

Chapter VIII

Business Data Warehouse: The Case of Wal-Mart

Indranil Bose, The University of Hong Kong, Hong Kong
Lam Albert Kar Chun, The University of Hong Kong, Hong Kong
Leung Vivien Wai Yue, The University of Hong Kong, Hong Kong
Li Hoi Wan Ines, The University of Hong Kong, Hong Kong
Wong Oi Ling Helen, The University of Hong Kong, Hong Kong

Abstract

The retailing giant Wal-Mart owes its success to the efficient use of information technology in its operations. One of the noteworthy advances made by Wal-Mart is the development of the data warehouse which gives the company a strategic advantage over its competitors. In this chapter, the planning and implementation of the Wal-Mart data warehouse is described and its integration with the operational systems is discussed. The chapter also highlights some of the problems encountered in the developmental process of the data warehouse. The implications of the recent advances in technologies such as RFID, which is likely to play an important role in the Wal-Mart data warehouse in future, is also detailed in this chapter.

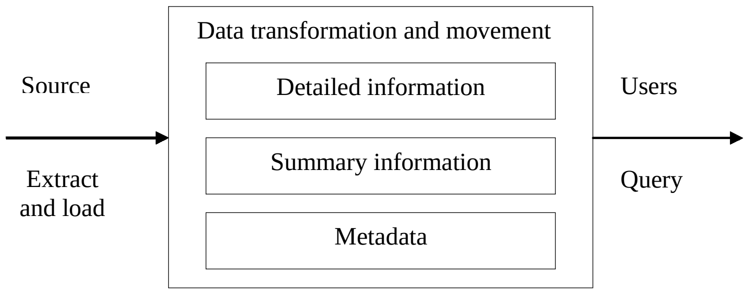
Introduction

Data warehousing has become an important technology to integrate data sources in recent decades which enables knowledge workers (executives, managers, and analysts) to make better and faster decisions (SCN Education, 2001). From a technological perspective, Wal-Mart, as a pioneer in adopting data warehousing technology, has always adopted new technology quickly and successfully. A study of the applications and issues of data warehousing in the retailing industry based on Wal-Mart is launched. By investigating the Wal-Mart data warehouse from various perspectives, we review some of the critical areas which are crucial to the implementation of a data warehouse. In this chapter, the development, implementation, and evaluation of the Wal-Mart data warehouse is described, together with an assessment of the factors responsible for deployment of a successful data warehouse.

Data Warehousing

Data warehouse is a subject-oriented, integrated, time-variant, non-updatable collection of data used in support of management decision-making (Agosta, 2000). According to Anahory and Murray (1997), "a data warehouse is the data (meta/fact/dimension/aggregation) and the process managers (load/warehouse/query) that make information available, enabling people to make informed decisions". Before the use of data warehouse, companies used to store data in separate databases, each of which were meant for different functions. These databases extracted useful information, but no analyses were carried out with the data. Since company databases held large volumes of data, the output of queries often listed out a lot of data, making manual data analyses hard to carry out. To resolve this problem, the technique of data warehousing was invented. The concept of data warehousing is simple. Data from several existing systems is extracted at periodic intervals, translated into the format required by the data warehouse, and loaded into the data warehouse. Data in the warehouse may be of three forms — detailed information (fact tables), summarized information, and metadata (i.e., description of the data). Data is constantly transformed from one form to another in the data warehouse. Dedicated decision support system is connected with the data warehouse, and it can retrieve required data for analysis. Summarized data are presented to managers, helping them to make strategic decisions. For example, graphs showing

Figure 1. Process diagram of a data warehouse (adapted from Anahory and Murray [1997])



sales volumes of different products over a particular period can be generated by the decision support system. Based on those graphs, managers may ask several questions. To answer these questions, it may be necessary to query the data warehouse and obtain supporting detailed information. Based on the summarized and detailed information, the managers can take a decision on altering the production volume of different products to meet expected demands. The major processes that control the data flow and the types of data in the data warehouse are depicted in Figure 1. For a more detailed description of the architecture and functionalities of a data warehouse, the interested reader may refer to Inmon and Inmon (2002) and Kimball and Ross (2002).

Background

Wal-Mart is one of the most effective users of technology (Kalakota & Robinson, 2003). Wal-Mart was always among the front-runners in employing information technology (IT) to manage its supply chain processes (Prashanth, 2004). Wal-Mart started using IT to facilitate cross docking in the 1970s. The company later installed bar codes for inventory tracking, and satellite communication system (SCS) for coordinating the activities of its supply chain. Wal-Mart also set-up electronic data interchange (EDI) and a computer terminal network (CTN), which enabled it to place orders electronically to its suppliers and allowed the company to plan the dispatch of goods to the stores appropriately. Advanced conveyor system was installed in 1978. The point of sale (POS) scanning system made its appearance in 1983, when Wal-Mart's key suppliers placed bar-codes on every item, and Universal Product Code (UPC) scanners were installed in Wal-Mart stores. Later on, the electronic purchase order management system was introduced when associates were equipped with handheld terminals to scan the shelf labels. As a result of the adoption of these technologies, inventory management became much more efficient for Wal-Mart. In the early 1990s, Wal-Mart information was kept in many different databases. As its competitors, such as Kmart, started building integrated databases, which could keep sales information down to the article level, Wal-Mart's IT department felt that a data warehouse was needed to maintain its competitive edge in the retailing industry.

Since the idea of data warehouse was still new to the IT staff, Wal-Mart needed a technology partner. Regarding data warehouse selection, there are three important criteria: compatibility, maintenance, and linear growth. In the early 1990s, Teradata Corporation, now a division of NCR, was the only choice for Wal-Mart, as Teradata was the only merchant database that fulfilled these three important criteria. Data warehouse compatibility ensured that the data warehouse worked with the front-end application, and that data could be transferred from the old systems. The first task for Teradata Corporation was to build a prototype of the data warehouse system. Based on this prototype system, a business case study related to the communication between the IT department and the merchandising organizations was constructed. The case study and the prototype system were used in conjunction to convince Wal-Mart executives to invest in the technology of data warehouse.

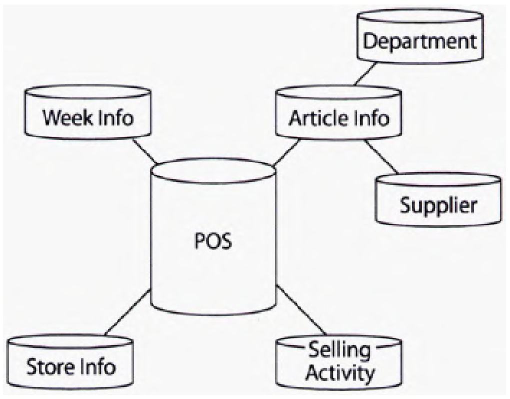
Once approved, the IT department began the task of building the data warehouse. First, information-based analyses were carried out on all of the historical merchandising data. Since the IT department did not understand what needed to be done at first, time was wasted. About a month later, there was a shakedown. The IT department focused on the point-of-sales (POS) data. Four teams were formed: a database team, an application team, a GUI team, and a Teradata team. The Teradata team provided training and overlooked everything. The remaining teams held different responsibilities: the database team designed, created, and maintained the data warehouse, the application team was responsible for loading, maintaining, and extracting the data, and the GUI team concentrated on building the interface for the data warehouse. While working on different parts of the data warehouse, the teams supported the operations of each other.

Hardware was a limitation in the data warehouse implementation at Wal-Mart. Since all data needed to fit in a 600 GB machine, data modeling had to be carried out. To save up storage space, a technique called "compressing on zero" was used (Westerman, 2001). This technique was created by the prototype teams. The technique assumed that the default value in the data warehouse was zero, and when this was the case, there was no need to store this data or allocate physical space on the disk drive for the value of zero. This was quite important since it required equal space to store zero or any large value. This resulted in great disk space savings in the initial stages of the database design. Data modeling was an important step in Wal-Mart data warehouse implementation. Not only did it save up storage but was responsible for efficient maintenance of the data warehouse in the future. Hence, it is stated by Westerman (2001), "If you logically design the database first, the physical implementation will be much easier to maintain in the longer term."

After the first implementation, Wal-Mart data warehouse consisted of the POS structure (Figure 2). The structure was formed with a large fact-base table (POS) surrounded by a number of support tables.

The initial schema was a star schema with the central fact table (POS) being linked to the other six support tables. However, the star schema was soon modified to a snowflake schema where the large fact-table (POS) was surrounded by several smaller support tables (like store, article, date, etc.) which in turn were also surrounded by yet smaller support tables (like region, district, supplier, week, etc.). An important element of the POS table was the activity sequence number which acted as a foreign key to the selling activity table. The selling activity table led to performance problems after two years, and Wal-Mart decided to merge this table with the POS table. The next major change that took place several years later was the addition of the selling time

Figure 2. Star schema for Wal-Mart data warehouse (Source: Westerman, 2001)



attribute to the POS table. The detailed description of the summary and fact tables can be obtained from Westerman (2001).

Main Thrust

Approximately one year after implementation of the data warehouse in Wal-Mart, a return on investment (ROI) analysis was conducted. In Wal-Mart, the executives viewed investment in the advanced data warehousing technology as a strategic advantage over their competitors, and this resulted in a favorable ROI analysis. However, the implementation of the data warehouse was marked by several problems.

Problems in Using the Buyer Decision Support Systems (BDSS)

The first graphical user interface (GUI) application based on the Wal-Mart data warehouse was called the BDSS. This was a Windows-based application created to allow buyers to run queries based on stores, articles, and specific weeks. The queries were run and results were generated in a spreadsheet format. It allowed users to conduct store profitability analysis for a specific article by running queries. A major problem associated with the BDSS was that the queries run using this would not always execute properly. The success rate of query execution was quite low at the beginning (i.e., 60%). BDSS was rewritten several times and was in a process of continual improvement. Initially, the system could only access POS data, but in a short period of time, access was also provided to data related to warehouse shipments, purchase orders, and store receipts. BDSS proved to be a phenomenal success for Wal-Mart, and it gave the buyers tremendous power in their negotiations with the suppliers, since they could check the inventory in the stores very easily and order accordingly.

Problems in Tracking Users with Query Statistics

Query Statistics was a useful application for Wal-Mart which defined critical factors in the query execution process and built a system to track the queries.

Tracking under this query statistics application revealed some problems with the warehouse. All users were using the same user-ID and password to log on and run queries, and there was no way to track who was actually running the specified query. Wal-Mart did manage to fix the problem by launching different user-IDs with the same password “walmart”. But this in turn led to security problems as Wal-Mart’s buyers, merchandisers, logistics, and forecasting associates, as well as 3,500 of Wal-Mart’s vendor partners, were able to access the same data in the data warehouse. However, this problem was later solved in the second year of operation of the data warehouse by requiring all users to change their passwords.

Performance Problems of Queries

Users had to stay connected to Wal-Mart’s bouncing network and database, throughout its entire 4,000-plus store chain and this was cost-ineffective and time-consuming when running queries. The users reported a high failure rate when the users stayed connected to the network for the duration of the query run time. The solution to this problem was deferred queries, which were added to enable a more stable environment for users. The deferred queries application ran the query and saved the results in the database in an off-line mode. The users were allowed to see the status of the query and could retrieve the results after completion of the query. With the introduction of the deferred queries, the performance problems were solved with satisfactory performance, and user confidence was restored as well. However, the users were given the choice to defer the queries. If they did not face any network-related problems they could still run the queries online, while remaining connected to Wal-Mart’s database.

Problems in Supporting Wal-Mart’s Suppliers

Wal-Mart’s suppliers often remained dissatisfied because they did not have access to the Wal-Mart data warehouse. Wal-Mart barred its suppliers from viewing its data warehouse since they did not want suppliers to look into the Wal-Mart inventory warehouse. The executives feared, if given access to the inventory warehouse, suppliers would lower the price of goods as much as they could, and this in turn would force Wal-Mart to purchase at a low price, resulting in overstocked inventory. Later on, Wal-Mart realized that since the goals of the supplier and the buyer are the same (i.e., to sell more merchandise), it is not beneficial to keep this information away from the suppliers. In fact, the information should be shared so that the suppliers could come prepared. In order to sustain its bargaining power over its suppliers and yet satisfy them, Wal-Mart built Retail Link, a decision support system that served as a bridge between Wal-Mart and its suppliers. It was essentially the same data warehouse application like the BDSS but without the competitors' product cost information. With this the suppliers were able to view almost everything in the data warehouse, could perform the same analyses, and exchange ideas for improving their business. Previously, the suppliers used to feel quite disheartened when the buyers surprised them with their up-to-date analyses using the BDSS. The suppliers often complained that they could not see what the buyers were seeing. With the availability of the Retail Link, the suppliers also began to feel that Wal-Mart cared about their partners, and this improved the relationship between the suppliers and the buyers.

Once the initial problems were overcome, emphasis was placed on integration of the data warehouse with several of the existing operational applications.

Integration of the Data Warehouse with Operational Applications

When it comes to integration, the main driving force for Wal-Mart was the ability to get the information into the hands of decision makers. Therefore, many of the applications were integrated into the data warehouse (Whiting, 2004). As a result, the systems were able to feed data into the data warehouse seamlessly. There were also technical reasons for driving integration. It was easier to get data out of the integrated data warehouse, thus making it a transportation vehicle for data into the different computers throughout the company. This was especially important because this allowed each store to pull new information from the data warehouse through their replenishment system. It was also very effective since the warehouse was designed to run in parallel, thus allowing hundreds of stores to pull data at the same time. The following is a brief description of Wal-Mart's applications and how they were integrated into the enterprise data warehouse.

Replenishment System

The process of automatic replenishment was critically important for Wal-Mart since it was able to deliver the biggest ROI after the implementation of the data warehouse. Since the replenishment application was already established, the system was quite mature for integration. The replenishment system was responsible for online transaction processing (OLTP) and online analytical processing (OLAP). It reviewed articles for orders. The system then determined whether an order was needed and suggested an order record for the article. Next these order records were loaded into the data warehouse and transmitted from the home office to the store. The store manager then reviewed the suggested orders, changed prices, counted inventory, and so on. Before the order was placed, the store managers also reviewed the flow of goods by inquiring about article sales trends, order trends, article profiles, corporate information, and so on. These were examples of OLAP activities. This meant that the order was not automatically placed for any item. Only after the store manager had a chance to review the order and perform some analyses using the data warehouse was it decided whether the order was going to be placed or not. The order could either be placed if the order could be filled from one of the Wal-Mart warehouses, or the order could be directed to the supplier via electronic data interchange (EDI). In either of the two cases, the order would be placed in the order systems and into the data warehouse.

Distribution via Traits

The traiting concept was developed as an essential element of the replenishment system. The main idea was to determine the distribution of an article to the stores. Traits were used to classify stores into manageable units and could include any characteristics, as long as it was somewhat permanent. Furthermore, these traits could only have two values: TRUE and FALSE. Table 1 is an example of what a store trait table might look like.

Traits could also be applied to articles in a store where a different table could be created for it. These different trait tables were used as part of the replenishment system. The most powerful aspect of this traiting concept was the use of a replenishment formula based on these traits. The formula was a Boolean formula where the outcome consisted of one of two values. If the result

Table 1. An example store trait table

Store ID	Pharmacy	Fresh Deli	Bakery	Beach	Retirement	University	<60K Sqft	>120K Sqft	Kmart Comp	Target Comp	Real Comp	etc.
2105	N	N	N	N	Y	N	N	Y	Y	N	N	…
2106	Y	Y	Y	N	N	Y	N	Y	N	Y	N	…

was true, the store would receive an article and vice versa. This concept was very important for a large centrally-managed retail company like Wal-Mart, since the right distribution of goods to the right stores affected sales and hence the image of the company. A distribution formula might look like this:

*Store distribution for Article X = (pharmacy * fresh deli * bakery * ¬ < 60K sq. ft.).*

This formula indicated that a store which had a pharmacy, a fresh deli, a bakery, and had a size of more than 60,000 sq. ft., should receive the order. From Table 1, we can see that store 2106 satisfies all these conditions and hence should receive the article X. In this manner, each article had its own unique formula, helping Wal-Mart distribute its articles most effectively amongst its stores.

All this information was very valuable for determining the allocation of merchandise to stores. A data warehouse would provide a good estimate for a product based on another, similar product that had the same distribution. A new product would be distributed to a test market using the traiting concept, and then the entire performance tracking would be done by the data warehouse. Depending on the success or failure of the initial trial run, the traits would be adjusted based on performance tracking in the data warehouse, and this would be continued until the distribution formula was perfected. These traiting methods were replicated throughout Wal-Mart using the data warehouse, helping Wal-Mart institute a comprehensive distribution technique.

Perpetual Inventory (PI) System

The PI system was used for maintenance of inventory of all articles, not just the articles appearing in the automatic replenishment. Like the replenishment system, it was also an example of an OLAP and OLTP system. It could help managers see the entire flow of goods for all articles, including replenishment articles. This data was available in the store and at the home office. Thus, with the use of the replenishment and PI systems, managers could maintain all information related to the inventory in their store electronically. With all this information in the data warehouse, there were numerous information analyses that could be conducted. These included:

- The analysis of the sequence of events related to the movement of an article;
- Determination of operational cost; and
- Creation of "plan-o-grams" for each store for making planning more precise. This could allow buyers and suppliers to measure the best selling locations without physically going to the store.

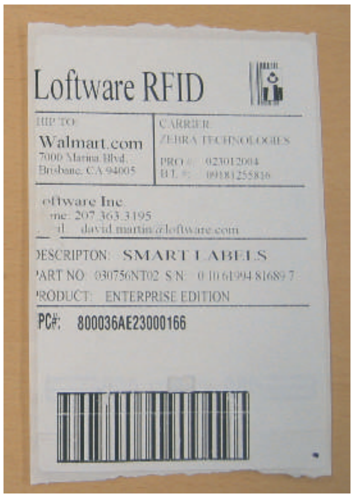
The PI system using the enterprise data warehouse could also provide benefits to the customer service department. Managers could help customers locate certain products with certain characteristics. The system could allocate the product in the store, or identify if there were any in storage, or if the product was in transit and when it could arrive or even if the product was available in any nearby stores. This could be feasible due to the data provided by the PI system and the information generated by the data warehouse.

Future Trends

Today, Wal-Mart continues to employ the most advanced IT in all its supply chain functions. One current technology adoption in Wal-Mart is very tightly linked with Wal-Mart's data warehouse, that is, the implementation of Radio Frequency Identification (RFID). In its efforts to implement new technologies to reduce costs and enhance the efficiency of supply chain, in July 2003, Wal-Mart asked all its suppliers to place RFID tags on the goods, packed in pallets and crates shipped to Wal-Mart (Prashanth, 2004). Wal-Mart announced that its top 100 suppliers must be equipped with RFID tags on their pallets and crates by January, 2005. The deadline is now 2006 and the list now includes all suppliers, not just the top 100 (Hardfield, 2004). Even though it is expensive and impractical (Greenburg, 2004), the suppliers have no choice but to adopt this technology.

The RFID technology consists of RFID tags and readers. In logistical planning and operation of supply chain processes, RFID tags, each consisting of a microchip and an antenna, would be attached on the products. Throughout the distribution centers, RFID readers would be placed at different dock doors. As a product passed a reader at a particular location, a signal would be triggered and the computer system would update the location status of the associated

Figure 3. RFID label for Wal-Mart (Source: E-Technology Institution (ETI) of the University of Hong Kong [HKU])



Loftware RFID

HIP TO
Walmart.com
7000 Marina Blvd.
Brisbane, CA 94005

CARRIER
ZEBRA TECHNOLOGIES
PRO #: 023012004
B.L #: 09181255816

oftware Inc.
me: 207.363.3195
il: david.martin@loftware.com

ESCRIPTON: SMART LABELS
ART NO: 030756NT02 S/N: 0 10 61994 81689 7
RODUCT: ENTERPRISE EDITION

PC#: 800036AE23000166

product. According to Peak Technologies (http://www.peaktech.com), Wal-Mart is applying SAMSys MP9320 UHF portal readers with Moore Wallance RFID labels using Alien Class 1 passive tags. Each tag would store an Electronic Product Code (EPC) which was a bar code successor that would be used to track products as they entered Wal-Mart's distribution centers and shipped to individual stores (Williams, 2004). Figure 3 is an example of the label. The data stored in the RFID chip and a bar code are printed on the label, so we know what is stored in the chip and also the bar code could be scanned when it became impossible to read the RFID tag. According to Sullivan (2004, 2005), RFID is already installed in 104 Wal-Mart stores, 36 Sam's Clubs, and three distribution centers, and Wal-Mart plans to have RFID in 600 stores and 12 distribution centers by the end of 2005.

The implementation of RFID at Wal-Mart is highly related to Wal-Mart's data warehouse, as the volume of data available will increase sufficiently. The industry has been surprised by estimates of greater than 7 terabytes of item-level data per day at Wal-Mart stores (Alvarez, 2004). The large amount of data can severely reduce the long-term success of a company's RFID initiative. Hence, there is an increasing need to integrate the available RFID data with the Wal-Mart data warehouse. Fortunately, Wal-Mart's data warehouse team is aware of the situation and they are standing by to enhance the data warehouse if required.

Conclusion

In this chapter we have outlined the historical development of a business data warehouse by the retailing giant Wal-Mart. As a leader of adopting cutting edge IT, Wal-Mart demonstrated great strategic vision by investing in the design, development and implementation of a business data warehouse. Since this was an extremely challenging project, it encountered numerous problems from the beginning. These problems arose due to the inexperience of the development team, instability of networks, and also inability of Wal-Mart management to forecast possible uses and limitations of systems. However, Wal-Mart was able to address all these problems successfully and was able to create a data warehouse system that gave them phenomenal strategic advantage compared to their competitors. They created the BDSS and the Retail Link which allowed easy exchange of information between the buyers and the suppliers and was able to involve both parties to improve sales of items. Another key achievement of the Wal-Mart data warehouse was the Replenishment system and the Perpetual Inventory system, which acted as efficient decision support systems and helped store managers throughout the world to reduce inventory, order items appropriately, and also to perform ad-hoc queries about the status of orders. Using novel concepts such as traiting, Wal-Mart was able to develop a successful strategy for efficient distribution of products to stores. As can be expected, Wal-Mart is also a first mover in the adoption of the RFID technology which is likely to change the retailing industry in the next few years. The use of this technology will lead to the generation of enormous amounts of data for tracking of items in the Wal-Mart system. It remains to be seen how Wal-Mart effectively integrates the RFID technology with its state-of-the-art business data warehouse to its own advantage.

References

Agosta, L. (2000). *The essential guide to data warehousing*. Upper Saddle River, NJ: Prentice Hall.

Alvarez, G. (2004). What's missing from RFID tests. *Information Week*. Retrieved November 20, 2004, from http://www.informationweek.com/story/showArticle.jhtml?articleID=52500193

Anahory, S., & Murray, D. (1997). *Data warehousing in the real world: A practical guide for building decision support systems*. Harlow, UK: Addison-Wesley.

Greenburg, E. F. (2004). Who turns on the RFID faucet, and does it matter? *Packaging Digest*, 22. Retrieved January 24, 2005, from http://www.packagingdigest.com/articles/200408/22.php

Hardfield, R. (2004). The RFID power play. *Supply Chain Resource Consortium*. Retrieved October 23, 2004, from http://scrc.ncsu.edu/public/APICS/APICSjan04.html

Inmon, W. H., & Inmon, W. H. (2002). *Building the data warehouse* (3rd ed.). New York: John Wiley & Sons.

Kalakota, R., & Robinson, M. (2003). *From e-business to services: Why and why now?* Addison-Wesley. Retrieved January 24, 2005, from http://www.awprofessional.com/articles/article.asp?p=99978&seqNum=5

Kimball, R., & Ross, M. (2002). *The data warehouse toolkit: The complete guide to dimensional modeling* (2nd ed.). New York: John Wiley & Sons.

Prashanth, K. (2004). *Wal-Mart's supply chain management practices (B): Using IT/Internet to manage the supply chain*. Hyderabad, India: ICFAI Center for Management Research.

SCN Education B. V. (2001). *Data warehousing — The ultimate guide to building corporate business intelligence* (1st ed.). Vieweg & Sohn Verlagsgesellschaft mBH.

Sullivan, L. (2004). Wal-Mart's way. *Information Week*. Retrieved March 31, 2005, from http://www.informationweek.com/story/showArticle.jhtml?articleID=47902662&pgno=3

Sullivan, L. (2005). Wal-Mart assesses new uses for RFID. *Information Week*. Retrieved March 31, 2005, from http://www.informationweek.com/showArticle.jhtml?articleID=159906172

Westerman, P. (2001). *Data warehousing: Using the Wal-Mart model*. San Francisco: Academic Press.

Whiting, R. (2004). Vertical thinking. *Information Week*. Retrieved March 31, 2005, from http://www.informationweek.com/showArticle.jhtml?articleID=18201987

Williams, D. (2004). The strategic implications of Wal-Mart's RFID mandate. *Directions Magazine*. Retrieved October 23, 2004, from http://www.directionsmag.com/article.php?article_id=629

Chapter IX

A Content-Based Approach to Medical Image Database Retrieval

Chia-Hung Wei, University of Warwick, UK
Chang-Tsun Li, University of Warwick, UK
Roland Wilson, University of Warwick, UK

Abstract

Content-based image retrieval (CBIR) makes use of image features, such as color and texture, to index images with minimal human intervention. Content-based image retrieval can be used to locate medical images in large databases. This chapter introduces a content-based approach to medical image retrieval. Fundamentals of the key components of content-based image retrieval systems are introduced first to give an overview of this area. A case study, which describes the methodology of a CBIR system for retrieving digital mammogram database, is then presented. This chapter is intended to disseminate the knowledge of the CBIR approach to the applications of medical image management and to attract greater interest from various research communities to rapidly advance research in this field.

Introduction

John Doe, a radiologist in a university hospital, takes X-rays and MRI scans for patients producing hundreds of digital images each day. In order to facilitate easy access in the future, he registers each image in a medical image database based on the modality, region, and orientation of the image. One day Alice Smith, a surgeon, comes to discuss a case with John Doe as she suspects there is a tumor on the patient's brain according to the brain MRI. However, she cannot easily judge if it is a benign or malign tumor from the MRI scan, and would like to compare with previous cases to decide if this patient requires a dangerous operation. Understanding Alice's needs, John helps Alice find similar-looking tumors from the previous MRI images. He uses the query-by-example mode of the medical image database, delineates the tumor area in the MRI image, and then requests the database to return the brain MRI images most similar to this one. Alice finds eleven similar images and their accompanying reports after reviewing the search results. Alice compares those cases and verifies the pattern of the tumor. Later on, she tells her patient that it is a benign tumor and the operation is unnecessary unless the tumor grows.

This scenario briefly describes the creation of medical images, categorization of medical images, and a content-based access approach. Although a mature content-based access technology has not appeared yet, this field is developing actively. In the last decade, a large number of digital medical images have been produced in hospitals. Large-scale image databases collect various images, including X-ray, computed tomography (CT), magnetic resonance imaging (MRI), ultrasound (US), nuclear medical imaging, endoscopy, microscopy, and scanning laser ophtalmoscopy (SLO). The most important aspect of image database management is how to effectively retrieve the desired images using a description of image content. This approach of searching images is known as content-based image retrieval (CBIR), which refers to the retrieval of images from a database using information directly derived from the content of images themselves, rather than from accompanying text or annotation (El-Naqa, Yang, Galatsanos, Nishikawa, & Wernick , 2004; Wei & Li, in press).

The main purpose of this chapter is to disseminate the knowledge of the CBIR approach to the applications of medical image retrieval and to attract greater interest from various research communities to rapidly advance research in this field. The rest of the chapter is organized as follows: The second section addresses the problems and challenges of medical image retrieval and describes potential applications of medical CBIR. The third section reviews the existing medical CBIR systems. The fourth section provides greater details on the key components of content-based image retrieval systems for medical imaging applications. The fifth section presents a case study, which describes the methodology of CBIR systems for digital mammograms. The sixth section discusses potential research issues in the future research agenda. The last section concludes this chapter.

Medical Image Database Retrieval

This section will discuss the problems of image retrieval using the conventional text-based method and addresses the challenges of the CBIR approach. Potential applications of the CBIR approach will also be discussed.

Challenges in Medical Image Retrieval

Before the emergence of content-based retrieval, medical images were annotated with text, allowing the images to be accessed by text-based searching (Feng, Siu, & Zhang, 2003). Through textual description, medical images can be managed based on the classification of imaging modalities, regions, and orientation. This hierarchical structure allows users to easily navigate and browse the database. Searching is mainly carried out through standard Boolean queries.

However, with the emergence of massive image databases, the traditional text-based search suffers from the following limitations (Shah et al., 2004; Wei & Li, 2005):

- Manual annotations require too much time and are expensive to implement. As the number of images in a database grows, the difficulty in finding desired images increases. Muller, Michous, Bandon, and Geissbuhler (2004a) reported that the University Hospital of Geneva produced approximately 12,000 medical images per day. It is not feasible to manually annotate all attributes of the image content for this number of images.
- Manual annotations fail to deal with the discrepancy of subjective perception. The phrase, "an image says more than a thousand words," implies

that the textual description is not sufficient for depicting subjective perception. Typically, a medical image usually contains several objects, which convey specific information. Nevertheless, different interpretations for a pathological area can be made by different radiologists. To capture all knowledge, concepts, thoughts, and feelings for the content of any images is almost impossible.

- The contents of medical images are difficult to be concretely described in words. For example, irregular organic shapes cannot easily be expressed in textual form, but people may expect to search for images with similar contents based on the examples they provide.

These problems limit the feasibility of text-based search for medical image retrieval. In an attempt to overcome these difficulties, content-based retrieval has been proposed to automatically access images with minimal human intervention (Eakins, 2002; Feng et al., 2003). However, due to the nature of medical images, content-based retrieval for medical images is still faced with challenges:

- Low resolution and strong noise are two common characteristics in most medical images (Glatard, Montagnat, & Magnin, 2004). With these characteristics, medical images cannot be precisely segmented and extracted for the visual content of their features. In addition, medical images obtained from different scanning devices may display different features, though some approaches to image correction and normalization have been proposed (Buhler, Just, Will, Kotzerke, & van den Hoff, 2004);
- Medical images are digitally represented in a multitude of formats based on their modality and the scanning device used (Wong & Hoo, 2002). Another characteristic of medical images is that many images are represented in gray level rather than color. Even with the change of intensity, monochrome may fail to clearly display the actual circumstance of lesion area.

Medical Applications of Content-Based Image Retrieval

Content-based image retrieval has frequently been proposed for various applications. This section will discuss three potential applications of medical CBIR.

PACS/Health Database Management

Content-based image retrieval has been proposed by the medical community for inclusion into picture archiving and communication systems (PACS) (Lehmann, Wein, & Greenspan, 2003). The idea of PACS is to integrate imaging modalities and interfaces with hospital and departmental information systems in order to manage the storage and distribution of images to radiologists, physicians, specialists, clinics, and imaging centers (Huang, 2003). A crucial point in PACS is to provide an efficient search function to access desired images. Image search in the digital imaging and communication in medicine (DICOM) protocol is currently carried out according to the alphanumerical order of textual attributes of images. However, the information which users are interested in is the visual content of medical images rather than that residing in alphanumerical format (Lehmann et al., 2003). The content of images is a powerful and direct query which can be used to search for other images containing similar content. Hence, content-based access approaches are expected to have a great impact on PACS and health database management. In addition to PACS, medical imaging databases that are unconnected to the PACS can also obtain benefits from CBIR technology.

Computer-Aided Diagnosis

Computer-aided diagnosis has been proposed to support clinical decision making. One clinical decision-making technique is case-based reasoning, which searches for previous, already- solved problems similar to the current one and tries to apply those solutions to the current problem (Hsu & Ho, 2004; Schmidt, Montani, Bellazzi, Portinale, & Gierl, 2001). This technique has a strong need to search for previous medical images with similar pathological areas, scrutinize the histories of these cases which are valuable for supporting certain diagnoses, and then reason the current case (Chang et al., 2004).

Medical Research, Education, and Training

CBIR technology can benefit any work that requires the finding of images or collections of images with similar contents. In medical research, researchers can use CBIR to find images with similar pathological areas and investigate their association. In medical education, lecturers can easily find images with particular pathological attributes, as those attributes can imply particular diseases. In addition, CBIR can be used to collect images for medical books, reports, papers, and CD-ROMs based on the educational atlas of medical cells, where typical specimens are collected according to the similarity of their features, and the most typical ones are selected from each group to compose a set of practical calibrators.

Existing Medical CBIR Systems

Although content-based image retrieval has frequently been proposed for use in medical image management, only a few content-based retrieval systems have been developed specifically for medical images. These research-oriented systems are usually constructed in research institutes and continue to be improved, developed, and evaluated over time. This section will introduce several major medical content-based retrieval systems.

ASSERT (Automatic Search and Selection Engine with Retrieval Tools)

Developers: Purdue University, Indiana University, and University of Wisconsin Hospital, USA.

Image Database: High-Resolution Computed Tomography (HRCT) of lung.

Selected References: Shyu, Brodley, Kak, Kosaka, Aisen, and Broderick (1999), and Brodley, Kak, Dy, Shyu, Aisen, and Broderick, (1999).

Web site: http://rvl2.ecn.purdue.edu/~cbirdev/WWW/CBIRmain.html

Main Characteristics:

- The ASSERT system uses a physician-in-the-loop approach to retrieving images of HRCT of the lung. This approach requires users to delineate the pathology-bearing regions and identify certain anatomical landmarks for each image;
- This system extracts 255 features of texture, shape, edges, and gray-scale properties in pathology-bearing regions;
- A multi-dimensional hash table is constructed to index the HRCT images.

CasImage

Developer: University Hospital of Geneva, Switzerland.

Image Database: A variety of images from CT, MRI, and radiographs, to color photos.

Selected References: Muller, Rosset, Vallee, and Geissbuhler (2004b), and Rosset, Ratib, Geissbuhler, and Vallee (2002).

Web site: http://www.casimage.com/

Main Characteristics:

- The CasImage system, which has been integrated into a PACS environment, contains a teaching and reference database, and the medGIFT retrieval system, which is adapted from the open-source GIFT (GNU Image Finding Tool) (Squire, Muller, Muller, Marchand-Maillet, & Pun, 2001);
- The medGIFT retrieval system extracts global and regional color and texture features, including 166 colors in the HSV color space, and Gabor filter responses in four directions each at three different scales;
- Combinations of textual labels and visual features are used for medical image retrieval.

IRMA (Image Retrieval in Medical Applications)

Developer: Aachen University of Technology, Germany.

Image Database: Various imaging modalities.

Selected References: Lehmann, Guld, Keysers, Deselaers, Schubert, Wein, and Spitzer (2004a), and Lehmann, Guld, Thies, Plodowski, Keysers, Ott, and Schubert (2004b).

Web site: http://libra.imib.rwth-aachen.de/irma/

Main Characteristics:

- The IRMA system is implemented as a platform for content-based image retrieval in medical applications;

- This system splits the image retrieval process into seven consecutive steps, including categorization, registration, feature extraction, feature selection, indexing, identification, and retrieval.

NHANES II (The Second National Health And Nutrition Examination Survey)

Developer: National Library of Medicine, USA.

Image Database: 17,000 cervical and lumbar spine X-ray images.

Selected References: Antani, Lee, Long, and Thoma (2004a), and Antani, Xu, Long, and Thoma (2004b).

Web site: http://archive.nlm.nih.gov/proj/dxpnet/nhanes/nhanes.php

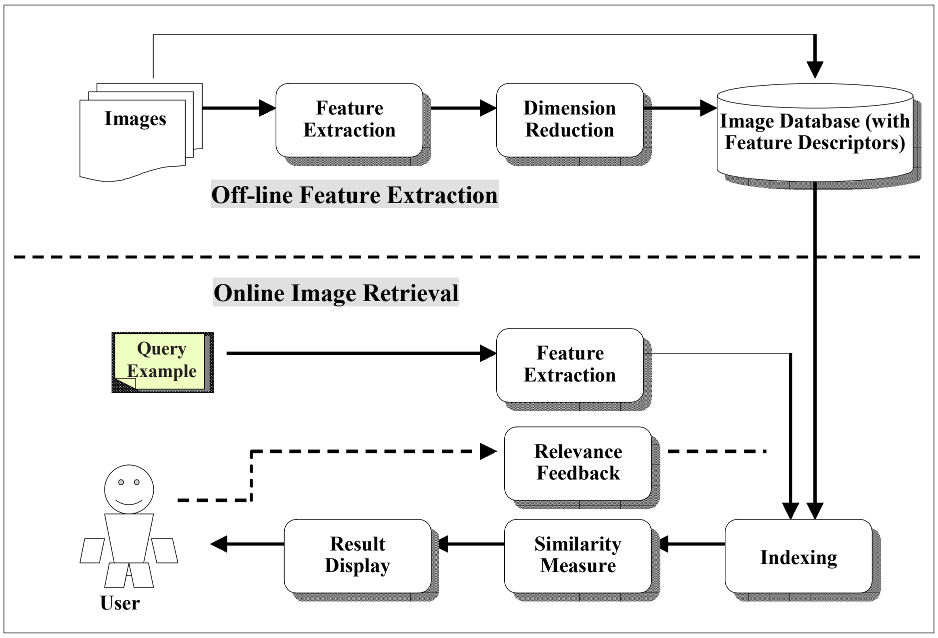
Main Characteristics:

- This system contains the Active Contour Segmentation (ACS) tool, which allows the users to create a template by marking points around the vertebra. If the segmentation of a template is accepted, the ACS tool will estimate the location of the next vertebra, place the template on the image, and then segment it;
- In data representation, a polygon approximation process is applied for eliminating insignificant shape features and reducing the number of data points. The data obtained in the polygon approximation process represent the shape of vertebra. Then, the approximated curve of vertebra is converted to tangent space for similarity measurement.

Content-Based Retrieval Systems

Content-based retrieval uses the contents of images to represent and access the images (Wei & Li, in press). A typical content-based retrieval system is divided into *off-line feature extraction* and *online image retrieval*. A conceptual framework for content-based image retrieval is illustrated in Figure 1. In off-line feature extraction, the contents of the images in the database are extracted and described with a multi-dimensional feature vector, also called descriptor.

Figure 1. A conceptual framework for content-based image retrieval



The feature vectors of the image constitute a feature dataset stored in the database. In online image retrieval, the user can submit a query example to the retrieval system in search of desired images. The system represents this example with a feature vector. The distances (i.e., similarities) between the feature vectors of the query example and those of the media in the feature dataset are then computed and ranked. Retrieval is conducted by applying an indexing scheme to provide an efficient way of searching the image database. Finally, the system ranks the search results and then returns the results that are most similar to the query examples. If the user is not satisfied with the search results, the user can provide relevance feedback to the retrieval system, which contains a mechanism to learn the user's information needs. The following sections will clearly introduce each component in the system.

Feature Extraction

Representation of images needs to consider which features are most useful for representing the contents of images and which approaches can effectively code the attributes of the images. Feature extraction of the image in the database is typically conducted off-line so computation complexity is not a significant issue. This section will introduce two features — texture and color — which are used most often to extract the features of an image.

Color

Color is a powerful descriptor that simplifies object identification (Gonzalez & Woods, 2002) and is one of the most frequently used visual features for content-based image retrieval. To extract the color features from the content of an image, a proper color space and an effective color descriptor have to be determined.

The purpose of a color space is to facilitate the specification of colors. Each color in the color space is a single point represented in a coordinate system. Several color spaces, such as *RGB*, *HSV*, *CIE L*a*b, and CIE L*u*v,* have been developed for different purposes. Although there is no agreement on which color space is the best for CBIR, an appropriate color system is required to ensure perceptual uniformity. Therefore, the *RGB* color space, a widely used system for representing color images, is not suitable for CBIR because it is a perceptually non-uniform and device-dependent system (Gevers, 2001).

The most frequently used technique is to convert color representations from the *RGB* color space to the *HSV*, CIE *L*u*v*, or CIE *L*a*b* color spaces with perceptual uniformity (Li & Yuen, 2000). The *HSV* color space is an intuitive system, which describes a specific color by its hue, saturation and brightness value. This color system is very useful in interactive color selection and manipulation; The CIE *L*u*v* and CIE *L*a*b* color spaces are both perceptually uniform systems, which provide easy use of similar metrics for comparing color (Haeghen, Naeyaert, Lemahieu, & Philips, 2000).

After selecting a color space, an effective color descriptor should be developed in order to represent the color of the global or regional areas. Several color descriptors have been developed from various representation schemes, such as color histograms (Quyang & Tan, 2002), color moments (Yu et al., 2002), color edge (Gevers & Stokman, 2003), color texture (Guan & Wada, 2002), and color correlograms (Moghaddam, Khajoie, & Rouhi, 2003). For example, color histogram, which represents the distribution of the number of pixels for each quantized color bin, is an effective representation of the color content of an image. The color histogram can not only easily characterize the global and regional distribution of colors in an image, but also be invariant to rotation about the view axis.

For the retrieval of medical images, color allows images to reveal many pathological characteristics (Tamai, 1999). Color also plays an important role in morphological diagnosis (Nishibori, Tsumura, & Miyake, 2004). Color medical images are usually produced in different departments and by various devices. For example, color endoscopic images are taken by a camera that is put into the hollow organs of the body such as stomachs and lungs. A common characteristic in such kind of images is that most colors are made of various stains, though fine variations of natural colors are crucial for diagnosis. Nishibori (2000) pointed out that problems in color medical images include inaccurate color reproduction, rough gradations of color, and insufficient density of pixels. Therefore, effective use of the various color information in images includes absolute color values, ratios of each tristimulus color, differences in colors against adjacent areas, and estimated illumination data. In addition, many medical images are represented in gray level. For this kind of gray level images, CBIR can only regard color as secondary features because gray levels provide limited information about the content of an image. For specific purposes, some gray level images have pseudo-color added to enhance specific areas instead of gray level presentation. Such processing increases difficulties in retrieval.

Texture

Texture in CBIR can be used for at least two purposes (Sebe & Lew, 2002). First, an image can be considered to be a mosaic that consists of different texture regions. These regions can be used as examples to search and retrieve similar areas. Second, texture can be employed for automatically annotating the content of an image. For example, the texture of an infected skin region can be used for annotating regions with the same infection.

Textural representation approaches can be classified into statistical approaches and structural approaches (Li, 1998). Statistical approaches analyze textural characteristics according to the statistical distribution of image intensity. Approaches in this category include gray level co-occurrence matrix, fractal model, Tamura feature, Wold decomposition, and so on (Feng et al., 2003). Structural approaches characterize texture by identifying a set of structural primitives and certain placement rules.

If medical images are represented in gray level, texture becomes a crucial feature, which provides indications about scenic depth, the spatial distribution of tonal variations, and surface orientation (Tourassi, 1999). For example, abnormal symptoms on female breasts include calcification, architectural distortion, asymmetry, masses, and so forth. All of those reveal specific textural patterns on the mammograms. However, selection of texture features for specifying textural structure should take account of the influence from the modulation transfer function on texture (Veenland, Grashuis, Weinans, Ding, & Vrooman, 2002). As the intensifying screens are used to enhance the radiographs, the blurring effect also changes texture features, that is, spatial resolution, contrast, and sharpness are all reduced in the output. Low resolution and contrast result in difficulties in measuring the pattern of tissue and structure of organs (Majumdar, Kothari, Augat, Newitt, Link, Lin, & Lang, 1998).

Dimension Reduction

In an attempt to capture useful contents of an image and to facilitate effective querying of an image database, a CBIR system may extract a large number of features from the content of an image. Feature set of high dimensionality causes the “curse of dimension” problem in which the complexity and computational cost of the query increase exponentially with the number of dimensions (Egecioglu, Ferhatosmanoglu, & Ogras, 2004).

To reduce the dimensionality of a large feature set, the most widely-used technique in image retrieval is principal component analysis (PCA). The goal of principal component analysis is to specify as much variance as possible with the smallest number of variables (Partridge & Calvo, 1998). Principal component analysis involves transforming the original data into a new coordinate system with low dimension, thus creating a new set of data. The new coordinate system removes the redundant data, and the new set of data may better represent the essential information. However, there is a trade-off between the efficiency obtained through dimension reduction and the completeness of the information extracted. As data is represented lower dimensions, the speed of retrieval is increased, but some important information may be lost in the process of data transformation. In the research of medical CBIR, Sinha and Kangarloo (2002) demonstrated the PCA application to the image classification of 100 axial brain images.

Similarity Measure

Selection of similarity metrics has a direct impact on the performance of content-based image retrieval. The kind of feature vectors selected determines the kind of measurement that will be used to compare their similarity (Smeulders, Worring, Santini, Gupta, & Jain, 2000). If the features extracted from the images are presented as multi-dimensional points, the distances between corresponding multi-dimensional points can be calculated. Euclidean distance is the most common metric used to measure the distance between two points in multi-dimensional space (Qian, Sural, Gu, & Pramanik, 2004). For other kinds of features such as color histogram, Euclidean distance may not be an ideal similarity metric or may not be compatible with the human-perceived similarity. Histogram intersection was proposed by Swain and Ballard (1991) to find known objects within images using color histograms. A number of other metrics, such as Mahalanobis Distance, Minkowski-Form Distance, Earth Mover's Distance, and Proportional Transportation Distance, have been proposed for specific purposes. Antani, Long, Thoma, and Lee (2003) used several approaches to code the shape features for different classes of spine X-rays. Each class used a specific similarity metric to compare the distance between two feature vectors.

Multi-Dimensional Indexing

Retrieval of an image is usually based not only on the value of certain features, but also on the location of a feature vector in the multi-dimensional space (Fonseca & Jorge, 2003). A retrieval query on a database of multimedia with multi-dimensional feature vectors usually requires fast execution of search operations. To support such search operations, an appropriate multi-dimensional access method has to be used for indexing the reduced but still high dimensional feature set. Popular multi-dimensional indexing methods include the R-tree (Guttman, 1984) and the R*-tree (Beckmann, Kriegel, Schneider, & Seeger, 1990).

The R-tree, which is a tree-like data structure, is mainly used for indexing multi-dimensional data. Each node of an R-tree has a variable number of entries. Each entry within a non-leaf node can have two pieces of data. The goal of the R-tree is to organize the spatial data in such a way that a search will visit as few spatial objects as possible. The decision on which nodes to visit is made based on the evaluation of spatial predicates. Hence, the R-tree must be able to hold some sort of spatial data on all nodes. The R*-tree, an improved version of the R-tree, applies more complex criteria for the distribution of minimal- bounding rectangles through the nodes of the tree, such as: The overlap between the minimal- bounding rectangles of the inner nodes should be minimized; the perimeter of a directory rectangles should be minimized; and storage utilization should be maximized. Both techniques perform well in low-dimensional features-space with a limit of up to 20 dimensions. For high-dimensional features-space, it is necessary to reduce the dimensionality using statistic multi-variate analysis techniques such as the aforementioned principal component analysis.

With regard to medical CBIR research, Shyu et al. (1999) successfully applied multi-dimensional indexing in the ASSERT system. In this system, lobular feature sets (LFS) on HRCT images are translated into an index for archiving and retrieval. A multi-dimensional hash table for the LFS classes is constructed for the system. A decision tree algorithm is used to construct a minimum-entropy partition of the feature space where the LFS classes reside. After translating a decision tree to a hash table, the system prunes the set of retrieved LFS classes and candidate images.

Relevance Feedback

Relevance feedback was originally developed for improving the effectiveness of information retrieval systems. The main idea of relevance feedback is for the retrieval system to understand the user's information needs. For a given query, the retrieval system returns initial results based on pre-defined similarity metrics. Then, the user is required to identify the positive examples by labeling those that are relevant to the query. The system subsequently analyzes the user's feedback using a learning algorithm and returns refined results.

A typical relevance feedback mechanism contains a learning component and a dispensing component. The learning component uses the feedback data to estimate the target of the user. The approach taken to learn feedback data is key to the relevance feedback mechanism. In addition to Rocchio's (1971) and Rui and Huang's (2002) learning algorithms, recent work has reported that support vector machine (SVM) is a useful learning approach in relevance feedback (Ferecatu, Crucianu, & Boujemaa, 2004a; Hoi, Chan, Huang, Lyu, & King, 2004; Tao & Tang, 2004). The dispensing component should provide the most appropriate images after obtaining feedback from the user. However, the dispensing component has two conflicting goals during each feedback round. On the one hand, the dispensing component has to provide as many relevant images as possible. On the other hand, the dispensing component, based on the information needs of the user, has to investigate the images of unknown relevance to the target (Ferecatu, Crucianu, & Boujemaa, 2004b). As the dispensing component returns more relevant images to the user, it has fewer images to mine the needs of the user at each round, and vice versa. A sensible strategy also plays an important role in relevance feedback. Hence, approaches to learning user feedbacks and dispensing strategies for returning the results both determine the performance of relevance feedback mechanisms.

Medical images have a unique characteristic in that their contents always reflect pathological attributes or symptoms for specific diseases. Image classification is often used to group the similar features based on their contents. With this characteristic, relevance feedback is expected to assist in mining the common features of relevant images and finding a specific class where the query example should reside. Those images grouped in the same class have the same semantics and are likely to be target images. El-Naqa, Yang, Galatsanos, and Wernick (2003) proposed a relevance feedback approach based on incremental learning for mammogram retrieval. They adapted support vector machines (SVM) to develop an online learning procedure for similarity learning. The approach they proposed was implemented using clustered micro-calcifications images. They reported that the approach significantly improves the retrieval effectiveness. In addition, El-Naqa et al. (2004) also demonstrated a hierarchical two-stage learning network, which consists of a cascade of a binary classifier and a regression module. Relevance feedback is incorporated into this framework to effectively improve precision based on online interaction with users.

Case Study

This section will propose a general CBIR framework and its application to mammogram retrieval, and demonstrate its method. Breast cancer continues to be a serious disease across the world. Mammography is a reliable method for detection of breast cancer. There are an enormous number of mammograms generated in hospitals. How to effectively retrieve a desired image from mammogram databases is a challenging problem. This study concentrates on textural analysis based on gray level co-occurrence matrices for the content-based retrieval of mammograms. The objectives of this study are as follows:

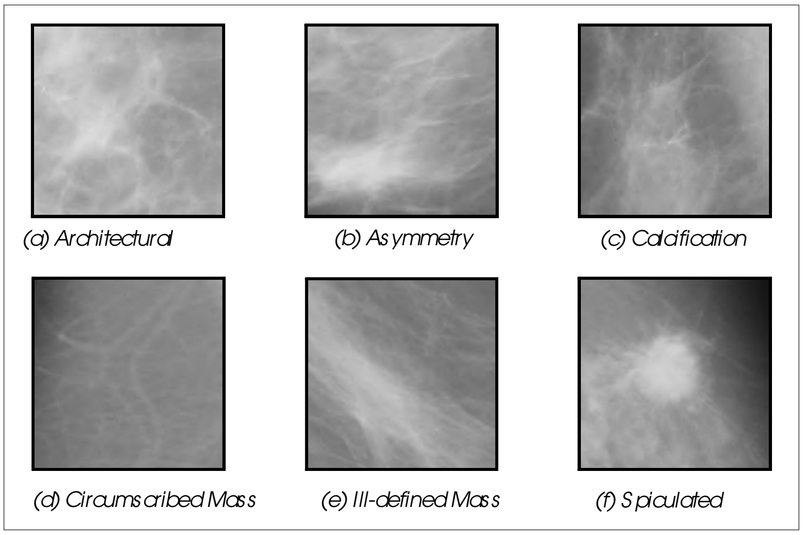
1. To analyze and examine the textural features present in the ROI (Region of Interest) of abnormal breast tissue as compared to the same information presented in normal tissue;
2. To develop the optimal mammographic descriptor generated from gray level co-occurrence matrices; and
3. To evaluate the effectiveness of the CBIR system using descriptors with different unit pixel distances.

The method in this work contains two major stages — *image analysis* and *image retrieval*. The objective of the image analysis stage is to examine the textural features of mammograms, and then test the statistical significance of the differences between normal and abnormal mammograms. These discriminating features are selected to construct a textural descriptor of mammograms. The descriptor constructed in the image analysis stage is embedded into the CBIR system. The feature descriptor is extracted from the query image in order to retrieve the mammograms relevant to the query image. The performance of the CBIR system is then evaluated. The detailed steps and components of the experiment are described in the following sections.

Mammogram Dataset

Mammograms were obtained from the database of the Mammographic Images Analysis Society (MIAS) (Suckling, Parker, Dance, Astley, Hutt, Boggis, Ricketts, Stamatakis, Cerneaz, Kok, Taylor, Betal, & Savage, 1994). The size of each image was 1024×1024 pixels. All of the images have been annotated for class, severity and location of abnormality, character of background tissue, and radius of circle enclosing the abnormality. Abnormalities are classified into calcifications architectural distortions, asymmetries, circumscribed masses, speculated masses, and ill-defined masses. Sub-images of size 200×200 pixels were cropped as ROIs from each mammogram. One hundred and twenty-two sample ROIs (including 29 images in calcification class, 19 in architectural distortion class, 15 in asymmetry class, 25 in circumscribed masses class, 19 in speculated masses class, and 15 in other or ill-defined

Figure 2. Abnormal mammograms are classified into calcification, architectural distortion, asymmetry, circumscribed masses, speculated masses, and ill-defined masses



masses class) were selected deliberately from abnormal tissues. Another 207 ROIs were obtained arbitrarily from normal tissues. These 329 ROIs were used to analyze their textural features based on gray level co-occurrence matrices.

Feature Analysis

The presence of a breast lesion may cause a disturbance in the homogeneity of tissues, and result in architectural distortions in the surrounding parenchyma (Cheng & Cui, 2004). Therefore, the textures of digital images contain a lot of valuable information for further research and application. This study applies gray level co-occurrence matrices, a statistical textural method, to analyze the textural features of mammograms and develop descriptors for content-based image retrieval. Gray level co-occurrence matrices will be introduced in the following section.

Gray Level Co-Occurrence Matrices

Gray level co-occurrence matrix (GLCM) is a statistical method for computing the co-occurrence probability of textural features (Haralick, 1979). Given an image $f(x, y)$ of size $L_r \times L_c$ with a set of N_g gray levels, define the matrix $p(i, j, d, \theta)$ as:

$$\begin{aligned} P(i, j, d, \theta) = card\{((x_1, y_1), (x_2, y_2)) \in (L_r \times L_c) \;\; (L_r \times L_c) \mid \\ (x_2, y_2) = (x_1, y_1) + (d\cos\theta, d\sin\theta), \\ f(x_1, y_1) = i, f(x_2, y_2) = j, 0 \le i, j < N_g\} \end{aligned} \qquad (1)$$

where d denotes the distance between pixels (x_1, y_1) and (x_2, y_2) in the image, θ denotes the orientation aligning (x_1, y_1) and (x_2, y_2), and card $\{\cdot\}$ denotes the number of elements in the set. Texture features that can be extracted from gray level co-occurrence matrices (Haralick, Shanmugan, & Dinstein, 1973) are:

$$\text{Angular Second Moment (ASM)} = \sum_i \sum_j \{p(i, j)\}^2 \qquad (2)$$

$$\text{Contrast} = \sum_{n=0}^{Ng-1} n^2 \left\{ \sum_{i=1}^{Ng} \sum_{\substack{j=1 \\ |i-j|=n}}^{Ng} p(i, j) \right\} \qquad (3)$$

$$\text{Correlation} = \frac{\sum_i \sum_j (ij) p(i, j) - \mu_x \mu_y}{\sigma_x \sigma_y} \qquad (4)$$

$$\text{Variance} = \sum_i \sum_j (i - j)^2 p(i, j) \qquad (5)$$

$$\text{Inverse Difference Moment (ID\_Mom)} = \sum_i \sum_j \frac{1}{1 + (i - j)^2} p(i, j) \qquad (6)$$

$$\text{Sum Average (Sum\_Aver)} = \sum_{i=2}^{2Ng} i p_{x+y}(i) \qquad (7)$$

$$\text{Sum Variance (Sum\_Var)} = \sum_{i=2}^{2Ng} (i - Sum\_Entro)^2 p_{x+y}(i) \tag{8}$$

$$\text{Sum Entropy (Sum\_Entro)} = -\sum_{i=2}^{2Ng} p_{x+y}(i) \log\{p_{x+y}(i)\} \tag{9}$$

$$\text{Entropy} = -\sum_i \sum_j p(i,j) \log p(i,j) \tag{10}$$

$$\text{Different Variance (Diff\_Vari)} = \text{variance of } p_{x-y} \tag{11}$$

$$\text{Different Entropy (Diff\_Entro)} = -\sum_{i=0}^{Ng-1} p_{x-y}(i) \log\{p_{x-y}(i)\} \tag{12}$$

Mammogram Analysis Using GLCM

In order to develop the tailored descriptors described in the next section, it is necessary to analyze the features of the mammogram. In this study, 12 GLCMs are constructed in order to compute each ROI in the 0°, 45°, 90°, and 135° directions, each with unit pixel distances of 1, 3, and 5, respectively. The 11 features described earlier are computed for the 12 GLCMs, thus resulting in a total of 132 texture features for each ROI.

Feature Selection for Image Retrieval

At the stage of feature analysis, 132 texture features are generated for each ROI. We obtain 5,148 texture feature sample images from 20 normal and 19 abnormal images. To select the most discriminant features, a statistical multivariate *t*-test is used to assess the significance of the difference between the means of two sample set *A* and *B*, which are independent of each other in the obvious sense, that is, the individual measures in set *A* are in no way related to any of the individual measures in set *B*. The value of the *t*-test is obtained as follows (Serdobolskii, 2000):

$$D_a = \sum (A_i - \mu_a)^2 \tag{13}$$

$$D_b = \sum (B_i - \mu_b)^2 \tag{14}$$

$$V = \frac{D_a + D_b}{(n_a - 1) + (n_b - 1)} \tag{15}$$

$$\sigma = \sqrt{\frac{V}{n_a} + \frac{V}{n_b}} \tag{16}$$

$$t = \frac{\mu_a - \mu_b}{\sigma} \tag{17}$$

where A_i and B_i in equations (13) and (14) are the *i*th element of the set *A* and *B*, while μ_a and μ_b are the means of the set *A* and *B*, respectively. D_a and D_b in the equations (13) and (14) are the sum of squared deviates of *the* set A and B. *V* in equation (15) is the estimated variance of the source population. σ in equation (16) is the standard deviation of the sampling distribution of sample-mean differences. *t* in equation (17) is the value of the *t*-test. The degree of freedom (d.f.) is $(n_a - 1) + (n_b - 1)$.

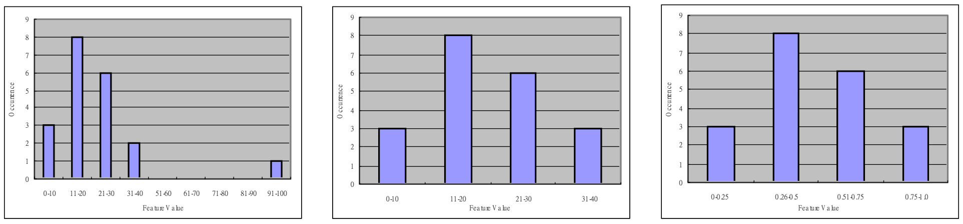
In our case study with 20 normal images (set *A*) and 19 abnormal images (set *B*), the degree of freedom (d.f.) is 37. According to the Table of Critical Values of *t* (Rencher, 1998), the *t* value for 37 degrees of freedom (d.f.) is 1.305. When the value *t* obtained in this experiment is greater than 1.305, it means that there is a significant mean difference between normal and abnormal mammograms with regard to the given feature.

The descriptor is composed of features with significant differences in the *t* statistic. Individual descriptors were developed for three distances (*d* = 1, 3, and 5) in the gray level co-occurrence matrices.

Data Normalization

The purpose of normalization in this experiment is to assign a weight to all features in order to measure their similarity on the same basis. The technique

Figure 3. Process of normalization



(a) The original distribution *(b) The trimmed distribution* *(c) The normalized distribution*

used was to project each feature onto a unit sphere. However, a potential problem exists — if a few elements in a feature space are extremely large, other elements may be dominated by these large ones after normalization.

To solve this problem, the value located at the point of the top 95% of the distribution is taken as the nominal maximum. All features greater than the nominal minimum in the feature space were clipped to the nominal maximal value, that is, the top 5% of distribution are trimmed. Then all values are divided by the maximal values. An example is given to illustrate the use of this approach to normalization. Figure 3a shows the original distribution of feature values in a feature vector. The results of trimming the top 5% and normalization are illustrated in Figures 3b and 3c.

Similarity Measure

The similarity measure of two images I_a and I_b is the distance between their descriptors f_a and f_a. In this work, L_2 norm was adopted to measure the similarity between the query image and each ROI. L_2 is defined as follows:

$$\| d_{ab} \|_2 = | f_a - f_b |^2 = \sqrt{\sum_{i=1}^{n} | f_{a,i} - f_{b,i} |^2}, \qquad (18)$$

where d_{ab} is the similarity distance between descriptors f_a and f_b, $f_{a,i}$ and $f_{b,i}$ are the ith element of f_a and f_a, respectively, and n is the number of elements of the descriptors. The smaller the distance is, the more similar the two images are. After calculating the distance, our CBIR system ranks similarity in descending order and then returns the top five images that are most similar to the query image.

Performance Evaluation

Relevance judgment is a vital part of performance evaluation. The relevance criteria described in Table 1 were developed and used in this work. For example, suppose the query image belongs to the calcification class, the retrieved image would score 0.5 if it belongs to any of the following abnormal classes: ill-defined masses, circumscribed masses, speculated masses, architectural distortion, and asymmetry.

Precision and recall are basic measures used in evaluating the effectiveness of an information retrieval system. Precision is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved (Baeza-Yates, & Ribeiro-Neto, 1999). It indicates the subject score assigned to each of the top five images in this experiment. The formula is expressed as follows:

$$p = \frac{\sum_{i=1}^{n} S_i}{N} \tag{19}$$

where S_i is the score assigned to the ith hit, N is the number of top hits retrieved.

Recall is the ratio of the number of relevant records retrieved to the total number of relevant records in the database (Baeza-Yates & Ribeiro-Neto, 1999). It is defined as follows:

Table 1. Criteria for measurement of performance evaluation of CBIR

Score	Criteria
1.0	The retrieved image belongs to the class of query image.
0.5	The retrieved image belongs to one of the abnormal classes, but not the class of query image.
0	The retrieved image does not belong to any abnormal class.

$$R = \frac{R_n}{T_n} \tag{20}$$

where R_n is the number of retrieved relevant hits, and T_n is the total number of relevant images in the database.

Results of *t*-Test

Table 2 presents the results of the t statistic for $d = 1$ of gray level co-occurrence matrices. From the results, it can be seen that the differences between the mean values of ASM, Correlation, Sum_Var (sum variance), and Diff_Entro (difference entropy) of the normal and abnormal ROIs are significant ($t > 1.305$). As a result, these four features are selected to construct the descriptor. Table 3 shows that ASM is the only feature with significant discriminating power for two groups of ROIs when $d = 3$. The descriptor with $d = 3$ contains only ASM.

Table 4 shows that Sum_Var (sum-variance) is the only feature with significant discriminating power for two groups of ROIs when $d = 5$. The descriptor with $d = 5$ contains only Sum_Var.

Table 2. Comparison of mean values obtained by co-occurrence matrices with the distance of 1

	Normal		**Abnormal**		
Feature	μ_a	D_a	μ_b	D_b	t (37 .d.f.)
ASM	1.7721	0.6127	1.3890	0.5542	1.6042
Contrast	5.8209	2.2742	5.6187	3.7038	0.3755
Correlation	-0.3392	0.0742	-0.6135	0.5345	1.6071
Variance	0.2413	0.0768	1.5446	1.9193	-4.2257
ID_Mom	2.9460	0.0671	3.0654	0.1442	-1.1812
Sum_Aver	2.1943	0.3391	2.6984	1.2137	-1.8444
Sum_Var	3.2636	2.2970	0.7699	1.3261	5.9097
Sum_Entro	-6.7253	2.7305	-0.7922	0.1327	-15.6895
Entropy	-4.0729	0.1476	-3.8611	0.3262	-1.3995
Diff_Vari	5.2007	2.3020	6.7532	1.4977	-3.5953
Diff_Entro	-1.7865	0.0332	-1.9603	0.0141	3.5986

Table 3. Comparison of mean values obtained by co-occurrence matrices with the distance of 3

	Normal		Abnormal		
Feature	μ_a	D_a	μ_b	D_b	t (37 .d.f.)
ASM	1.2846	0.3212	1.0075	0.2905	1.6046
Contrast	1.3822	0.2398	1.6247	0.4604	-1.3191
Correlation	0.8311	0.0402	3.6871	15.2704	-3.3534
Variance	1.5943	3.4598	1.1012	0.9300	1.0584
ID_Mom	1.465	0.0518	1.5281	0.1695	-0.6123
Sum_Aver	1.6135	0.1835	1.9774	0.6539	-1.8161
Sum_Entro	-4.9365	1.4720	-0.5789	0.0708	-15.7084
Entropy	-2.6801	0.1052	-2.4949	0.4599	-1.1262
Diff_Vari	4.1858	1.2969	5.3510	0.8446	-3.5984
Diff_Entro	-1.6118	0.0184	1.7424	0.0088	-91.7379

Table 4. Comparison of mean values obtained by co-occurrence matrices with the distance of 5

	Normal		Abnormal		
Feature	μ_a	D_a	μ_a	D_b	t (37 .d.f.)
ASM	1.2466	0.3019	0.9784	0.2730	0.7924
Contrast	2.1675	0.8227	2.9725	2.0665	-1.5708
Correlation	0.8822	0.0387	4.6453	14.6955	-7.1316
Variance	1.4357	2.8461	1.0654	0.8467	1.0316
ID_Mom	1.2008	0.0497	1.227	0.1625	-0.0742
Sum_Aver	1.5855	0.1775	1.9376	0.6293	-0.8292
Sum_Var	2.3006	2.6245	0.5359	0.6438	4.5870
Sum_Entro	-4.8432	1.4197	-0.5657	0.0677	-8.0337
Entropy	-2.5128	0.1179	-2.3037	0.2179	-0.4203
Diff_Vari	4.1858	1.2969	5.3510	0.8446	-1.6684
Diff_Entro	-1.6118	0.0184	-1.7424	0.0088	0.3150

Results of Performance Evaluation

Precision can be used to describe the accuracy of the proposed CBIR system in finding only relevant images on a search for query images. Table 5 shows that precision rates for the three descriptors (*d* = 1, 3, and 5) are 47%, 50%, and 51%, respectively. The descriptor with *d* = 5 obtained the highest value in precision and the smallest values in standard deviation. This indicates that the performance of the descriptor (*d* = 5) is more stable. On the whole, about half of the results retrieved by these three descriptors are relevant to the query images.

Recall measures how well the CBIR system finds all relevant images in a search for a query image. Table 6 indicates that the descriptor with *d* = 5 outperforms the other two. However, the recall values are very close. The largest difference is only 1.5%. The three descriptors can retrieve, on average, about 18% of relevant images in the database. In theory, as precision goes up, recall goes down. The relationship explains why the three recall values are low.

The experimental results also show that the descriptor with the largest distance (*d* = 5) has the best performance in both precision and recall. The descriptor with *d* = 3 outperforms the descriptor with *d* = 1 in both measures. Although the larger distance has better performance in this experiment, it is still too early to make any conclusions.

Table 5. Precision for the 3 descriptors

	CALC	CIRC	SPIC	MISC	ARCH	ASYM	Mean	Std
***d*=1**	42%	44%	54%	44%	46%	54%	47%	5.32%
***d*=3**	57%	43%	54%	42%	54%	51%	50%	6.24%
***d*=5**	48%	53%	54%	50%	50%	48%	51%	2.51%

Table 6. Recall for the 3 descriptors

	CALC	CIRC	SPIC	MISC	ARCH	ASYM	Mean	Std
***d*=1**	10.69%	12.80%	20.53%	18.67%	16.84%	26.00%	17.59%	5.51%
***d*=3**	13.45%	11.60%	20.00%	20.00%	18.42%	25.33%	18.13%	4.97%
***d*=5**	12.07%	15.20%	21.05%	24.00%	20.00%	22.00%	19.05%	4.51%

(Notes: CALC = calcification; CIRC = circumscribed masses, SPIC = speculated masses; ARCH = architectural distortion; ASYM = asymmetry; MISC = other or ill-defined masses.)

The main contribution of this work is to present a sound CBIR methodology for mammograms. The methodology is divided into image analysis and image retrieval stages. The purpose of the image analysis is to collect samples from the database, obtain the image signature, and then apply it for the feature extraction in the image retrieval stage. A complete CBIR system based on gray level co-occurrence matrices was implemented. A technique was also proposed to improve the effectiveness of normalization. Three descriptors were evaluated by query images to retrieve the ROIs for the mammogram dataset consisting of 122 images of six sub-classes from abnormal class, and 207 images from normal class. The best precision rate of 51% and recall rate of 19% were achieved with the descriptor using gray level co-occurrence matrices with the pixel distance of 5.

Research Issues

Content-based retrieval for medical images is still in its infancy. There are many challenging research issues. This section identifies and addresses some issues in the future research agenda.

Bridging the Semantic Gap

An ideal medical CBIR system from a user perspective would involve semantic retrieval, in which the user submits a query like “find MRIs of brain with tumor”. This kind of open-ended query is very difficult for the current CBIR systems to distinguish brain MRI’s from spine MRIs even though the two types of images are visually different. Current medical CBIR systems mainly rely on low-level features like texture, color, and shape.

Systems Integration

Most medical retrieval systems are designed for one particular type of medical image, such as mammogram or MRIs of spine. Specific techniques and modalities are developed based on the characteristics of these highly homogeneous images in their databases. However, medical image databases across different medical institutions have been expected to connect through PACS.

With PACS, a user may make a request to search for medical images among different databases, where images display different characteristics such as degree of resolution, degree of noise, use of color, shape of object, and texture of background. In other words, PACS can be seen as a single retrieval system with distributed image databases, which collect medical images with various modalities. Therefore, systems capable of finding images across heterogeneous image databases are desirable.

Human-Computer Interaction and Usability

Current research on medical CBIR concentrates on the effectiveness of the system, rarely evolving the relationship between CBIR and user interface design. However, innovative retrieval systems alone may not obtain user acceptance as users of medical CBIR systems may include radiologists, surgeons, nurses, or other users without specific knowledge of these systems. The user's experience, or how the user experiences the system, is the key to acceptance. Good user interface design is usually required for the end user to easily learn and use the system. Also, empirical usability testing permits naïve users to provide information about the usability of individual system functions and components.

Performance Evaluation

The National Institute of Standards and Technology (NIST) has developed TREC (Text REtrieval Conference) as the standard test-bed and evaluation paradigm for the information retrieval community (Smeaton, 2003). The image retrieval community still awaits the construction and implementation of a scientifically-valid evaluation framework and standard test bed. To construct a test bed for medical CBIR, imaging modalities, regions, and orientations of images should be taken into account. Due to the complexity of medical images, how to construct a common test bed for medical CBIR is a research issue.

Conclusion

The goal of medical image databases is to provide an effective means for organizing, searching, and indexing large collections of medical images. This requires intelligent systems that have the ability to recognize, capture, and understand the complex content of medical images. Content-based retrieval is a promising approach to achieve these tasks and has developed a number of techniques used in medical images. Despite recent developments, medical content-based image retrieval still has a long way to go and more efforts are expected to be devoted to this area. Ultimately, a well-organized image database, accompanied by an intelligent retrieval mechanism, can support clinical treatment, and provide a basis for better medical research and education.

References

Antani, S., Lee, D. J., Long, L. R., & Thoma, G. R. (2004a). Evaluation of shape similarity measurement methods for spine X-ray images. *Journal of Visual Communication and Image Representation, 15*(3), 285-302.

Antani, S., Long, L. R., Thoma, G. R., & Lee, D. J. (2003). Evaluation of shape indexing methods for content-based retrieval of X-ray images. In *Proceedings of IS&T/SPIE 15th Annual Symposium on Electronic Imaging, Storage, and Retrieval for Media Databases* (pp. 405-416). Santa Clara, CA: SPIE.

Antani, S., Xu, X., Long, L. R., & Thoma, G. R. (2004b). Partial shape matching for CBIR of spine X-ray images. In *Proceedings of IS&T/SPIE Electronic Imaging — Storage and Retrieval Methods and Applications for Multimedia 2004* (pp. 1-8). San Jose, CA: SPIE.

Baeza-Yates, R., & Ribeiro-Neto, B. (Eds.). (1999). *Modern information retrieval*. Boston: Addison-Wesley.

Beckmann, N., Kriegel, H.-P., Schneider, R., & Seeger, B. (1990). The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of ACM SIGMOD International Conference on Management of Data* (pp. 322-331). Atlantic City, NJ: ACM Press.

Brodley, C. E., Kak, A, Dy, J., Shyu, C. R., Aisen, A., & Broderick, L. (1999). Content-based retrieval from medical image database: A synergy of human interaction, machine learning, and computer vision. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence* (pp. 760-767). Orlando, FL: AAAI Press / MIT Press.

Buhler, P., Just, U., Will, E., Kotzerke, J., & van den Hoff, J. (2004). An accurate method for correction of head movement in PET. *IEEE Transactions on Medical Imaging, 23*(9), 1176-1185.

Chang, C.-L., Cheng, B.-W., & Su, J.-L. (2004). Using case-based reasoning to establish a continuing care information system of discharge planning. *Expert Systems with Applications, 26*(4), 601-613.

Cheng, H. D., & Cui, M. (2004). Mass lesion detection with a fuzzy neural network. *Pattern Recognition, 37*(6), 1189-1200.

Eakins, J. P. (2002). Towards in intelligent image retrieval. *Pattern Recognition, 35*(1), 3-14.

Egecioglu, O., Ferhatosmanoglu, H., & Ogras, U. (2004). Dimensionality reduction and similarity computation by inner-product approximations. *IEEE Transactions on Knowledge and Data Engineering, 16*(6), 714-726.

El-Naqa, I., Yang, Y., Galatsanos, N. P., & Wernick, M. N. (2003). Relevance feedback based on incremental learning for mammogram retrieval. In *Proceedings of the International Conference of Image Processing 2003* (pp. 729-732). Barcelona, Spain: IEEE Press.

El-Naqa, I., Yang, Y., Galatsanos, N. P., Nishikawa, R. M., & Wernick, M. N. (2004). A similarity learning approach to content-based image retrieval: Application to digital mammography. *IEEE Transactions on Medical Imaging, 23*(10), 1233-1244.

Feng, D., Siu, W. C., & Zhang, H. J. (Eds.). (2003). *Multimedia information retrieval and management: Technological fundamentals and applications*. Berlin: Springer.

Ferecatu, M., Crucianu, M., & Boujemaa, N. (2004a). Retrieval of difficult image classes using SVM-based relevance feedback. In *Proceedings of the 6th ACM SIGMM International Workshop on Multimedia Information Retrieval* (pp. 23-30). New York: ACM Press.

Ferecatu, M., Crucianu, M., & Boujemaa, N. (2004b). Sample selection strategies for relevance feedback in region-based image retrieval. In *Pacific-Rim Conference on Multimedia 2004* (pp. 497-504). Tokyo, Japan: IEEE Press.

Fonseca, M. J., & Jorge, J. A. (2003). Indexing high-dimensional data for content-based retrieval in large database. In *Proceedings of the Eighth International Conference on Database Systems for Advanced Applications* (pp. 267-274). Kyoto, Japan: IEEE Press.

Gevers, T., (2001). Color-based retrieval. In M. S. Lew (Ed.), *Principles of visual information retrieval* (pp. 11-49). London: Springer.

Gevers, T., & Stokman, H. (2003). Classifying color edges in video into shadow-geometry, highlight, or material transitions. *IEEE Transactions on Multimedia, 5*(2), 237-243.

Glatard, T., Montagnat, J., & Magnin, I. E. (2004). Texture based medical image indexing and retrieval: Application to cardiac imaging. In *Proceedings of the ACM SIGMM International Workshop on Multimedia Information Retrieval* (pp. 135-142). New York: ACM Press.

Gonzalez, R. C., & Woods, R. E. (2002). *Digital image processing*. Upper Saddle River, NJ: Prentice Hall.

Guan, H. & Wada, S. (2002). Flexible color texture retrieval method using multi-resolution mosaic for image classification. In *Proceedings of the 6th International Conference on Signal Processing: Vol. 1* (pp. 612-615). Beijing, China: IEEE Press.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *Proceedings of ACM SIGMOD International Conference on Management of Data* (pp. 47-54). New York: ACM Press.

Haeghen, Y. V., Naeyaert, J. M. A. D., Lemahieu, I., & Philips, W. (2000). An imaging system with calibrated color image acquisition for use in dermatology. *IEEE Transaction on Medical Imaging, 19*(7), 722-730.

Haralick, R. M. (1979). Statistical and structural approaches to texture. In *Proceedings of the IEEE, 67*(5), 786-804.

Haralick, R. M., Shanmugan, K., & Dinstein, I. (1973). Textural features for image classification. *IEEE Transactions on Systems, Man and Cybernetics, 3*(6), 610-621.

Hoi, C.-H., Chan, C.-H., Huang, K., Lyu, M. R., & King, I. (2004). Biased support vector machine for relevance feedback in image retrieval. In *Proceedings of International Joint Conference on Neural Networks* (pp. 3189-3194). Budapest, Hungary: IEEE Press.

Hsu, C.-C., & Ho, C.-S. (2004). A new hybrid case-based architecture for medical diagnosis. *Information Sciences, 166*(1-4), 231-247.

Huang, H. K. (2003). PACS, image management, and imaging informatics. In D. Feng, W. C. Siu, & H. J. Zhang (Eds.), *Multimedia information retrieval and management: Technological fundamentals and applications* (pp.347-365). New York: Springer.

Lehmann, T. M., Guld, M. O., Keysers, D, Deselaers, T., Schubert, H., Wein B. B., & Spitzer, K. (2004a). Similarity of medical images computed from global feature vectors for content-based retrieval. *Lecture Notes in Artificial Intelligence* (pp. 989-995).

Lehmann, T. M., Guld, M. O., Thies, C., Plodowski, B., Keysers, D., Ott, B., & Schubert, H. (2004b). IRMA — Content-based image retrieval in medical applications. In *Proceedings of the 14th World Congress on Medical Informatics* (pp. 842-848).

Lehmann, T. M., Wein, B. B., & Greenspan, H. (2003). Integration of content-based image retrieval to picture archiving and communication systems. In *Proceedings of Medical Informatics Europe*. Amsterdam, The Netherlands: IOS Press.

Li, C. H., & Yuen, P. C. (2000). Regularized color clustering in medical image database. *IEEE Transaction on Medical Imaging, 19*(11), 1150-1155.

Li, C.-T. (1998). *Unsupervised texture segmentation using multi-resolution Markov random fields*. Doctoral dissertation, University of Warwick, Coventry, UK.

Majumdar, S., Kothari, M., Augat, P., Newitt, D. C., Link, T. M., Lin, J. C., & Lang, T. (1998). High-resolution magnetic resonance imaging: Three-dimensional trabecular bone architecture and biomechanical properties. *Bone, 22*, 445-454.

Moghaddam, H. A., Khajoie, T. T., & Rouhi, A. H. (2003). A new algorithm for image indexing and retrieval using wavelet correlogram. In *Proceedings of the International Conference on Image Processing 2003: Vol. 3* (pp. 497-500). Barcelona, Catalonia, Spain: IEEE Press.

Muller, H., Michous, N., Bandon, D., & Geissbuhler, A. (2004a). A review of content-based image retrieval systems in medical applications — Clinical benefits and future directions. *International Journal of Medical Informatics, 73*(1), 1-23.

Muller, H., Rosset, A. Vallee, J.-P., & Geissbuhler, A. (2004b). Comparing features sets for content-based image retrieval in a medical-case database. In *Proceedings of IS&T/SPIE Medical Imaging 2004: PACS and Imaging Informatics* (pp. 99-109).

Nishibori, M. (2000). Problems and solutions in medical color imaging. In *Proceedings of the Second International Symposium on Multi-Spectral Imaging and High Accurate Color Reproduction* (pp. 9-17). Chiba, Japan: SPIE.

Nishibori, M., Tsumura, N., & Miyake, Y. (2004). Why multi-spectral imaging in medicine? *Journal of Imaging Science and Technology, 48*(2), 125-129.

Partridge, M., & Calvo, R. A. (1998). Fast dimensionality reduction and simple PCA. *Intelligent Data Analysis, 2*(1-4), 203-214.

Ouyang, A., & Tan, Y. P (2002). A novel multi-scale spatial-color descriptor for content-based image retrieval. In *Proceedings of the 7th International Conference on Control, Automation, Robotics and Vision: Vol. 3* (pp. 1204-1209).

Qian, G.., Sural, S., Gu, Y., & Pramanik, S. (2004). Similarity between Euclidean and cosine angle distance for nearest neighbor queries. In *Proceedings of 2004 ACM Symposium on Applied Computing* (pp. 1232-1237). Nicosia, Cyprus: ACM Press.

Rencher, A. C. (1998). *Multi-variate statistical inference and applications*. New York: John Wiley & Sons.

Rocchio, J. J. (1971). Relevance feedback in information retrieval. In G. Salton (Ed.), *The SMART retrieval system-experiments in automatic document processing* (pp.313-323). Englewood Cliffs, NJ: Prentice Hall.

Rosset, A., Ratib, O., Geissbuhler, A., & Vallee, J. P. (2002). Integration of a multimedia teaching and reference database in a PACS environment. *Radiographics, 22*(6), 1567-1577.

Rui, Y., & Huang, T. (2002). Learning based relevance feedback in image retrieval. In A. C. Bovik, C. W. Chen, & D. Goldfof (Eds.), *Advances in image processing and understanding: A festschrift for Thomas S. Huang* (pp. 163-182). New York: World Scientific Publishing.

Schmidt, R., Montani, S., Bellazzi, R., Portinale, L., & Gierl, L. (2001). Cased-based reasoning for medical knowledge-based systems. *International Journal of Medical Informatics, 64*(2-3), 355-367.

Sebe, N., & Lew, M. S. (2002). Texture features for content-based retrieval. In M. S. Lew (Ed.), *Principles of visual information retrieval* (pp.51-85), London: Springer.

Serdobolskii, V. (2000). *Multivariate statistical analysis: A high-dimensional approach*. London: Kluwer Academic Publishers.

Shah, B., Raghavan, V., & Dhatric, P. (2004). Efficient and effective content-based image retrieval using space transformation. *Proceedings of the 10th International Multimedia Modelling Conferenc* (p. 279). Brisbane, Australia. IEEE Press.

Shyu, C., Brodley, C., Kak, A., Kosaka, A., Aisen, A., & Broderick, L. (1999). ASSERT: A physician-in-the-loop content-based image retrieval system for HRCT image databases. *Computer Vision and Image Understanding, 75*(1/2), 111-132.

Shyu, C.-R., Pavlopoulou, C., Kak, A. C., Brodly, C. E., & Broderick, L. (2002). Using human perceptual categories for content-based retrieval from a medical image database. *Computer Vision and Image Understanding, 88*(3), 119-151.

Smeaton, A. F., & Over, P. (2003). TRECVID: Benchmarking the effectiveness of information retrieval tasks on digital video. In *International Conference on Image and Video Retrieval*, 18-27.

Smeulders, A. W. M., Worring, M., Santini, S., Gupta, A., & Jain, R. (2000). Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(12), 1349-1380.

Sinha, U., & Kangarloo, H. (2002). Principal component analysis for content-based image retrieval. *RadioGraphics, 22*, 1271-1289.

Squire, D. M., Muller, H., Muller, W., Marchand-Maillet, S., & Pun, T. (2001). Design and evaluation of a content-based image retrieval system. In S. M. Rahman (Ed.), *Design and management of multimedia information systems: Opportunities and challenges* (pp. 125-151). Hershey, PA: Idea Group Publishing.

Squire, D. M., Muller, W., Muller, H., & Pun, T. (2000). Content-based query of image databases: inspirations from text retrieval, *Pattern Recognition Letters, 21*(13-14), 1193-1198.

Suckling, J., Parker, J., Dance, D. R., Astley, S., Hutt, I., Boggis, C. R. M., Ricketts, I., Stamatakis, E., Cerneaz, N., Kok, S.-L., Taylor, P., Betal, D., & Savage, J. (1994). The mammographic image analysis society digital mammogram database. In *Proceedings of the 2nd International Workshop on Digital Mammography*, 375-378.

Swain, M. J., & Ballard, D. H. (1991). Color Indexing. *International Journal of Computer Vision, 7*(1), 11-32.

Tamai, S. (1999). The color of digital imaging in pathology and cytology. *Proceedings of the First Symposium of the "Color" of Digital Imaging in Medicine*. Tokyo, Japan: SPIE.

Tao, D., & Tang, X. (2004). Random sampling based SVM for relevance feedback image retrieval. In *The IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2004*, 647-652. Washington, DC: IEEE Press.

Tourassi, G. D. (1999). Journey toward computer-aided diagnosis: Role of image texture analysis. *Radiology*, 317-320.

Veenland, J. F., Grashuis, J. L., Weinans, H., Ding, M., & Vrooman, H. A. (2002). Suitability of texture features to assess changes in trabecular bone architecture. *Pattern Recognition Letters, 23*(4), 395-403.

Veltkamp, R. C., & Hagedoorn, M. (2002). State of the art in shape matching. In M. S. Lew (Ed.), *Principles of visual information retrieval* (pp.87-119), London: Springer.

Wei, C.-H, & Li, C.-T. (2005). Design of content-based multimedia retrieval. In M. Pagani (Ed.), *Encyclopedia of multimedia technology and networking* (pp. 116-122). Hershey, PA: Idea Group Reference.

Wong, S., & Hoo, K. S. (2002). Medical imagery. In V. Castelli, & L. D. Bergman (Eds.), *Image databases: Search and retrieval of digital imagery* (pp. 83-105). New York: John Wiley & Sons.

Wong, S. T. C. (1998). CBIR in medicine: Still a long way to go. In *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries*, 114. Santa Barbara, CA: IEEE Press.

Yu, H., Li, M., Zhang, H.-J., & Feng, J. (2002). Color texture moments for content-based image retrieval. In *Proceedings of the International Conference on Image Processing 2002: Vol. 3* (pp. 929-932).

Section II: Generic Database Modeling

Chapter X

Conceptual Modeling for XML: A Myth or a Reality

Sriram Mohan, Indiana University, USA
Arijit Sengupta, Wright State University, USA

Abstract

The process of conceptual design is independent of the final platform and the medium of implementation, and is usually in a form that is understandable and usable by managers and other personnel who may not be familiar with the low-level implementation details, but have a major influence in the development process. Although a strong design phase is involved in most current application development processes (e.g., Entity Relationship design for relational databases), conceptual design for XML has not been explored significantly in literature or in practice. Most XML design processes start by directly marking up data in XML, and the metadata is typically designed at the time of encoding the documents. In this chapter, the reader is introduced to existing methodologies for modeling XML. A discussion is then presented comparing and contrasting their capabilities and deficiencies, and delineating the future trend in conceptual design for XML applications.

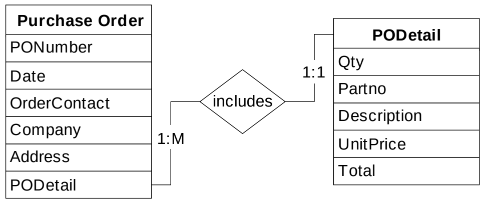
Introduction

With advances in the structural and functional complexity of XML, a standardized method for designing and visually presenting XML structures is becoming necessary.

XML modeling techniques can be generally classified based on the chosen approach into one of the following three major categories: (1) Entity Relationship (ER), (2) Unified Modeling Language (UML), and (3) Structured Hierarchical Model. Literature reveals the existence of several methodologies for modeling XML that are derived from these three categories. Several proprietary commercial tools that can be adapted to design and model XML structures have been introduced in recent years. In this chapter, we present six such academic tools and four commercial methodologies relevant in modeling XML structures and provided an overview of the same is provided by making use of appropriate examples. In order for the survey to be more comparative, a common working example is chosen and equivalent conceptual models are developed to illustrate a model's capabilities. To conclude, a discussion summarizing the capabilities of each of the methods and their suitability as a conceptual model for XML is analysed to help answer the question posed by the chapter: *Is developing a conceptual model for XML a Myth or a Reality?*

Several business situations arise where a conceptual model is necessary. A good conceptual model can help planners by providing a framework for developing architectures, assigning project responsibilities, and selecting technology (Mohr, 2001). For XML in particular, the verbose and syntax-heavy nature of the schema languages makes them unsuitable for providing this type of framework. As an illustration, consider the typical business problem of data interchange between different organizations. These type of applications, often used with the term EDI (Electronic Data Interchange), is already being moved to XML (Kay, 2000; Ogbuji, 1999). The non-proprietary nature of XML and its descriptive markup make it suitable for exchange of information between organizations. Ogbuji (1999) uses a purchase order example to illustrate how the interchange process can be facilitated with XML. However, a quick look at the illustration reveals that XML data and structure syntax, although more generalized and more descriptive than the EDI notation used by the article (ANSI X12 Transaction set), it is not going to be suitable for use in the presentation of the data to its potential users. A conceptual model of this purchase order, shown in Figure 1, reveals the internal structure of the order and items, and is more suited for understanding the conceptual structure of the application, and this is exactly the aim of this chapter.

Figure 1. A conceptual model for the purchase order application



In the rest of this chapter, we intend to demonstrate how conceptual models can in fact handle the complexities of XML, and the advances of such models in current literature as well as commercial applications. Toward that goal, we first further motivate the problem in the second section, and then discuss the interesting problems that arise when creating a conceptual model for XML in the third section. We then discuss six research-based methods in the fourth section, followed by four commercial tools in the fifth section. Finally, we compare these various tools in the sixth section and draw conclusions on the state of the current development in conceptual modeling for XML in the seventh section.

Motivation

Since its introduction in 1996, the use of XML has been steadily increasing, and it can be considered as the "format of choice" for data with mostly textual content. XML is widely used as the data format for a variety of applications, encompassing data from financial business transactions to satellite and scientific information. In addition, XML is also used to represent data communication between disparate applications. The two leading Web application development platforms .NET (Microsoft, 2003) and J2EE (Sun, 2004) both use XML Web Services (a standard mechanism for communication between applications, where the format for the exchange of data, and the specification of the services are modeled using XML).

Literature shows different areas of application of design principles that apply to XML. XML has been around for a while, but only recently has there been an effort towards formalizing and conceptualizing the model behind XML. These modeling techniques are still playing "catch-up" with the XML standard. The World Wide Web Consortium (W3C) has developed a formal model for XML — DOM (Document Object Model), a graph-based formal model for

XML documents (Apparao, Champion, Hors, & Pixley, 1999). For the purpose of querying, W3C has also proposed another data model called XPath data model, recently re-termed as the XQuery 1.0/XPath 2.0 data model (Fernandez, Malhotra, Marsh, Nagy, et al., 2003). However, both of these models are lowlevel models, representing the tree structure of XML, and are not designed to serve conceptual modeling purposes. The Object Management Group (OMG) has developed the XML Metadata Interchange (XMI) Specification[1] which comprises XML vocabulary and permits ASCII-based exchange of metadata between UML modeling tools (OMG, 2003). The XMI specification includes production rules for obtaining a schema (actually a DTD) from an XMI-encoded UML meta-model. SOAP (Simple Object Access Protocol) is another XML-based method that allows representation, serialization, and interchange of objects using XML. Although several of the lessons learned from such protocols provide valuable insight to developing a model for XML objects, the focus of this chapter is on more conceptual models that have a distinct user view and are not completely textual.

Data modeling is not a new topic in the field of databases. The most common is the relational model, which is a formal model, based on set-theoretic properties of tuples (Codd, 1970). The entity-relationship model (Chen, 1976) is a widely accepted conceptual model in relational databases. Similarly, object-oriented databases have the object-oriented model (Nguyen & Hailpern, 1982) at the underlying formal layer, and unified modeling language (UML) (Booch, Rumbaugh, & Jacobson, 1998) at the conceptual layer. Although XML has been in existence for over seven years, it is not based on a formal or conceptual model. XML has a grammar-based model of describing documents, carried over from its predecessor SGML (Standard Generalized Markup Language). Although fairly suitable for describing and validating document structures, grammar is not ideally suited for formal or conceptual description of data. The popularity of XML, however, necessitates a convenient method for modeling that would be useful for understanding and formalizing XML documents.

Conceptual Model

Before getting into the details of potential conceptual models for XML, the question that should be answered is, "What does the term *Conceptual Model* mean?" Batini, Ceri, and Navathe (1992, p. 6) define a conceptual model as:

A conceptual schema is a high-level description of the structure of the database, independent of the particular system that is used to implement the database. A conceptual model is a language that is used to describe the conceptual schema.

Typically a conceptual model should have the following properties:

- A conceptual model should map the requirements, and not the structure. Although it may be possible to generate the logical design of a system from its conceptual model, a conceptual design primarily needs to capture the abstract concepts and their relationships.
- A conceptual model should lend itself to be used for the generation of the logical design, such as the database schema. This allows the conceptual model to be changed late in the design phase, and avoids more expensive changes to the database itself.
- Conceptual models are for developers and non-developers alike. Non-developers should be able to understand the concepts without needing to know the details of the underlying database concepts.
- The conceptual design should not be considered as an intermediate design document and disregarded after the logical and physical design, but should remain as an integral part of the specification of the application itself.

Surprisingly little research has been done on visual modeling of XML documents. Six different directions of XML modeling were identified in literature. Each of the methods surveyed here is scrutinized to determine if they could be considered as a conceptual model, and not just as a visual representation of the structure.

Modeling Issues in XML

Modeling of XML document structures is not a trivial task. Unlike relational structures which are inherently flat, XML structures are hierarchical, and the data in XML documents is presented in a specific order, and this order is often a significant property of the data that needs to be preserved. In addition, there are many considerations one needs to make while designing XML structures.

In fact, the object-oriented model, because of its generalized structural constructs, comes closer to XML than the Relational and Entity Relationship models. Before presenting the main issues with XML modeling, we first discuss the two main approaches for metadata representation with XML.

Metadata Representation with XML

As is typical with XML, the two primary structure representation formats for XML are highly textual. The DTD (Document Type Definition) is a concept inherited in XML from its predecessor SGML. However, the increasing complexity of requirements have led to the creation of the new XML schema construct, which is the W3C-recommended format for representing XML metadata. Some more details of these two languages are given in subsequent paragraphs.

XML DTD

The DTD or the Document Type Definition is a concept that was inherited from SGML (ISO, 1986). The DTD has been the de facto standard for XML schema languages since the introduction of XML. It has limited capabilities compared to the other schema languages and uses elements and attributes as its main building blocks. These hierarchical structures have to be used to model the real world. The basic representation of a DTD resembles a grammatical representation such as BNF (Backus-Naur Form) (Naur, 1963). DTDs do not have support for any data typing, and a DTD in its current textual format lacks clarity and readability; therefore erroneous design and usage are inevitable. Often DTDs are tricky to design because of the limitations of the DTD constructs. Table 1 shows the DTD for a structured paper with citations. (Example adapted from Psaila [2000]).

XML Schema

XML schema (Malhotra & Maloney, 1999) is part of an ongoing effort by the W3C to aid and eventually replace DTD in the XML world. XML schema is more expressive than DTDs and can be used in a wide variety of applications. XML schema has support for data types, structures, and limited inheritance, which can be used to model XML structures appropriately. But like the DTD, XML schema

Table 1. The DTD for structured papers

```
<!ELEMENT PAPER (TITLE, SECTION*, BIBLIOGRAPHY?) >
<!ATTLIST PAPER name ID #REQUIRED>
<!ELEMENT TITLE (#PCDATA) >
<!ELEMENT SECTION (#PCDATA | CITE | PARA| EMPH )*>
<!ATTLIST SECTION title CDATA #REQUIRED >
<!ELEMENT BIBLIOGRAPHY (BIBITEM +) >
<!ELEMENT BIBITEM (#PCDATA)>
<!ATTLIST BIBITEM label ID #IMPLIED>
<!ELEMENT CITE EMPTY >
<!ATTLIST CITE label IDREF #REQUIRED >
<!ELEMENT EMPH (#PCDATA) >
<!ELEMENT PARA (#PCDATA)>
```

suffers from the fact that its textual format lacks clarity and readability. Typically an XML schema, like a DTD, consists of a series of definitions of elements and their contents. The most significant aspect of XML schema is that it uses XML as its underlying language. Because of this, for even simple structures, the corresponding XML schema can be highly verbose. This is demonstrated by the fact that the equivalent schema for the structured paper DTD generated using a DTD to Schema Translator — DTD2Xs translator (Rieger, 2003), shown in Table 2, is over five times larger than the DTD shown in Table 1.

XML Modeling Issues

Several issues arise when attempting to conceptually represent the structure of XML data. Some of these issues are as follows:

1. *Order:* XML objects are inherently ordered — there is a specific ordering between elements and different instances of the same element.
2. *Hierarchy:* XML does not have a direct way to support many-to-many relationships, since the structure is essentially hierarchical.
3. *Heterogeneous types:* XML structures often involve heterogeneous types, a concept by which different instances of an element may have different structures.
4. *Complex content:* Individual element structures can be complex. XML structures allow an element to contain a combination of multiple groups of elements combined using sequence, optional, and required constraints. Sub-elements can also repeat in many different ways. Structure of elements could be directly or indirectly recursive as well.
5. *Mixed content:* An element in XML may have mixed content — with atomic values as well as non-atomic values at the same time.

Table 2. XML schema for structured paper

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="PAPER">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="TITLE" /> <xs:element minOccurs="0"
              maxOccurs="unbounded" ref="SECTION" />
     <xs:element minOccurs="0" ref="BIBLIOGRAPHY" />
    </xs:sequence>
    <xs:attribute name="name" type="xs:ID" use="required" />
   </xs:complexType>
 </xs:element>

 <xs:element name="TITLE" type="xs:string" />

 <xs:element name="SECTION">
  <xs:complexType mixed="true">
   <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element ref="CITE" />
    <xs:element ref="PARA" />
    <xs:element ref="EMPH" />
   </xs:choice>
   <xs:attribute name="title" type="xs:string"
use="required"/>
  </xs:complexType>
 </xs:element>

 <xs:element name="BIBLIOGRAPHY">
  <xs:complexType>
   <xs:sequence>
    <xs:element maxOccurs="unbounded" ref="BIBITEM" />
   </xs:sequence>
  </xs:complexType>
 </xs:element>

 <xs:element name="BIBITEM">
  <xs:complexType>
   <xs:simpleContent>
    <xs:extension base="xs:string">
     <xs:attribute name="label" type="xs:ID" />
    </xs:extension>
   </xs:simpleContent>
  </xs:complexType>
 </xs:element>

 <xs:element name="CITE">
  <xs:complexType>
   <xs:attribute name="label" type="xs:IDREF" use="required"
/>
  </xs:complexType>
 </xs:element>
 <xs:element name="EMPH" type="xs:string" />
 <xs:element name="PARA" type="xs:string" />
</xs:schema>
```

6. *Namespaces:* Last but not the least, XML supports many related concepts such as namespaces that would make a straightforward conceptual design difficult to attain.

The issues with XML need to be resolved for any conceptual model to faithfully incorporate all the structural nuances of XML. Literature reveals a number of highly capable methodologies which could be considered as potentials for a widely accepted model for XML. We present six such methods from academic literature in this chapter. In order to demonstrate the current state of tools for XML structure design, we also include the modeling capabilities of four different commercial tools. To make the comparisons more intuitive, we choose a structure that we consistently use to create the diagrams for all of the surveyed models. We have used the DTD for structured papers from (Psaila, 2000) for this purpose. Table 1 shows the document type definition for structured papers.

Research Systems

Most of the methodologies explored in literature are based on currently existing modeling techniques. Two popular modeling methodologies that are in vogue currently are the Entity Relationship (ER) model and UML (Unified Modeling Language). Both of these have been used for modeling XML. The research on modeling methodologies for XML can be broadly classified in three categories:

1. *ER-based methods:* Methods in this category use the Entity Relationship model as a basis. Typically these methods extend the ER model to incorporate the complexities of XML.
2. *UML-based methods:* The Unified Modeling Language (UML) is a highly powerful model for object-oriented concepts. UML is much more powerful than XML structures, and has to be adapted by toning down some of its complexities for use with XML.
3. *Other modeling methods:* There are other methods such as the semantic network model which have also been used for modeling XML.

The rest of this section provides details on six academic articles which address issues related to these methods, and show how the nuances of XML structures can be appropriately represented in these respective models.

ER-Based Models

Methods based on the Entity-Relationship model face the challenge of extending the ER model to a more complex structure. Two main challenges faced here are:

1. The ER model is designed for flat and unordered relational structures, and the problem is in extending it to an ordered and more complex hierarchical structure.
2. One of the differences between the ER model and the intrinsic XML model is that the ER model is a network-based model, while XML is hierarchical. The problem comes from the fact that a network diagram has cycles, and potential many-to-many relationships, which are not directly representable in XML, although they can be implemented using XML IDs and IDREFs.

Typically, methods based on the Entity Relationship method extend the ER model to handle the two problems. We present two techniques that use the ER model as a basis.

Entity Relationship for XML (ERX)

Psaila (2000) introduces ERX (Entity Relationship for XML) as a conceptual model based on the Entity Relationship model (Chen, 1976). ERX is designed primarily to provide support for the development of complex XML structures. It provides a conceptual model to help better represent the various document classes that are used and their interrelationships. ERX is not nested, but rather exploits a flat representation to explain XML concepts and their relationships.

This model has some of the necessary modifications to cope with the features that are peculiar to XML. The basic building blocks of an ER model such as Entities, Relationships, and Attributes, have been modified or extended to support XML-specific features such as order, complex structures, and document classes. An ERX Entity describes a complex concept in a source document. An entity is typically represented by using a solid rectangle with the name of the entity mentioned inside the rectangle. Entities can have attributes which describe elementary concepts associated with an entity. In ERX, attributes are represented by using small oval circles which are connected to the Entity. Attributes in ERX can be extended by making use of qualifiers such as (R) and (I) — required and implied attributes, respectively.

ERX Relationships denote an association between two entities and is represented by a rhombus connected to the two associated entities. The cardinality constraints are mentioned in a manner similar to that of ER. ERX supports different kinds of relationships and also supports specialization hierarchies to denote various XML-specific concepts such as the "Choice" tag in XML schema. ERX supports the concept of "interface" which can be used to divide two parts of the conceptual model that are derived from two distinct classes of source documents. Hierarchies and generalizations are also supported in ERX. Psaila demonstrates in detail the capabilities of the ERX system and also provides a detailed explanation for the various elements that constitute ERX and their graphical notations. Order is partially supported in ERX by modeling it as the qualifier (O) for the attribute which determines where the specific instance of the entity appears in the document.

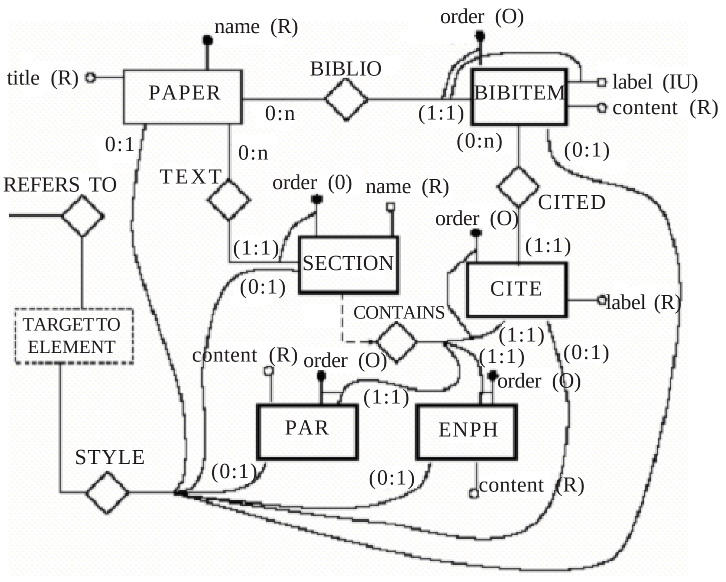
ERX, however, does not support some XML-specific features such as mixed content, and does not describe how complex types with their various nuances can be modeled into the system. ERX does not provide support for ordered and unordered attributes. The qualified attributed "Order" supported in ERX serves to establish the order between the various instances of a complex concept in XML; however, there is no mechanism to determine the order of attributes within an XML concept. ERX is not constrained by the syntactic structure of XML and is specifically focused on the data management issues. ERX, however, establishes the reasoning that a conventional ER model can be used to describe a conceptual model for XML structures and serves as an effective support for the development of complex XML structures for advanced applications. In another related article, Psaila (2003) also describes algorithms to translate XML DTDs into a corresponding ERX model.

Figure 2 shows the ERX for the structured paper DTD from Table 1. The ERX diagram is obtained from Psaila (2000) and includes only the relevant part of the diagram without the style sheet components.

Extensible Entity Relationship Model (XER)

XER (Sengupta, Mohan, & Doshi, 2003) is a conceptual modeling approach that can be used to describe XML document structures in a simple visual form

Figure 2. The ERX model for the structured DTD



reminiscent of the ER model. The XER approach has the capability to automatically generate XML document type definitions and schema from the XER diagrams and vice-versa. XER introduces a canonical view of XML documents called Element Normal Form (ENF) (Layman, 1999), which simplifies some of the modeling issues by removing the notion of attributes from the document. Instead, all XML attributes are converted to simple elements. More details are available in Sengupta et al. (2003).

The XER model includes all the basic constructs of the ER model, and introduces some new constructs of its own. The basic building blocks of the ER model — Entities, Attributes, and Relationships — are preserved with similar semantics in XER. The XER entity is the basic conceptual object in XER. A XER entity is represented using a rectangle with a title area showing the name of the entity and the body showing the attributes. XER attributes are properties of entities that are usually atomic, but can also be optional or multi-valued. Attributes are shown in the model by placing the names of the attributes in the body of the entity. Attributes are ordered by default, and the ordering in the diagram is top-to-bottom. Multi-valued attributes are allowed as mentioned before with the multiplicity shown in parentheses. Depending on the type of the schematic element being modeled, there are subtle changes in the representation of the XER entity. A XER entity can be of the following types: (1) ordered, (2) unordered, and (3) mixed. Each of these types has a unique graphical representation to enable easy design and comprehension. Relationships, which denote a connection between two or more entities, are introduced in XER when a complex entity contains a complex element as one of its sub-elements. Relationships can be one-to-one, one-to-many, or many-to-many. The cardinality of a relationship is equivalent to the minOccurs and maxOccurs tags present in the XML schema.

XER also supports other XML-specific features such as order, as well as advanced ER concepts such as weak entities, ternary relationships, aggregations, and generalizations. XML-specific features such as complex type structures, schematic restrictions, group-order, and choice indicators are supported in a highly presentable and easy-to-understand graphical form.

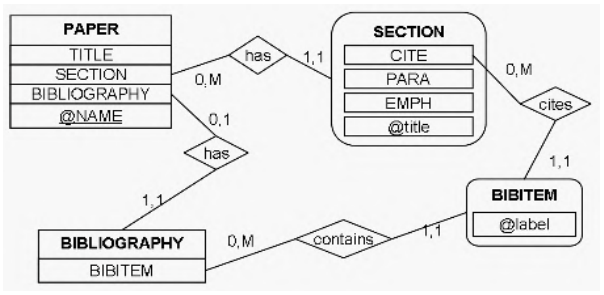
The XER diagram that is constructed bears a lot of resemblance to an ER diagram and supports more or less every facet available in the XML schema. XER does not fully incorporate intricate XML features such as namespaces. XER also fails to handle the "Any" construct, which the authors argue results in bad design. The authors also provide detailed algorithms to convert a XER diagram to a DTD and XML schema and vice versa. A prototype has been implemented using Dia (a GTK+ drawing program) (Dia, 2004) and XSLT that can be used to create XER models and convert them to XML schema and vice-versa.

Figure 3 presents the XER diagram for the structured paper DTD shown in Table 1. The diagram was generated by first converting the DTD to its ENF representation, and then converting the resulting DTD into an equivalent XML schema using DTD2Xs (Rieger, 2003). The resulting schema was imported into the XER Creator which generated the model in Figure 3.

UML-Based Models

The Unified Modeling Language (UML) (Booch et al., 1998) is one of the most popular models for object-oriented design. UML, however, is mismatched with XML because it is designed for fully object-oriented systems and is based on object inheritance, unlike XML. However, UML has support for some of the problems in modeling XML such as complex hierarchies and order, and is hence a good choice for developing a conceptual model for XML documents. To model XML properly with UML, some of the complexities in UML (such as methods, use cases) need to be trimmed to fit the XML domain. Two articles that use UML to model XML are discussed here.

Figure 3. XER model of the structured paper DTD



Unified Modeling Language (UML) and DTD

An approach to model DTDs conceptually using UML (Unified Modeling Language) has been studied in Conrad, Scheffner, and Freytag (2000). This paper incorporates relevant UML constructs such as the static view and the model management view to perform transformations. The static view consists of classes and their relationships such as association, generalization, and various kinds of dependencies, while the model management view describes the organization of the model. UML enables the application of object-oriented concepts in the design of XML and helps improve redesign and also reveal possible structural weaknesses.

Conrad, et al. describe various UML constructs such as classes, aggregation, composition, generalization, and packages, and explains their transformation into appropriate DTD fragments. It also extends UML to take advantage of all facets that DTDs offer. UML classes are used to represent XML element type notations. The element name is represented using the class name, and the element content is described by using the attributes of the class. Since UML classes do not directly support order, the authors introduce an implicit top-bottom order in a manner similar to that seen in XER. DTD constructs for element types which express an element — sub-element relationship are modeled by using aggregations. The authors argue that the multiplicity specification for UML aggregations is semantically as rich as cardinality constraints and use the same to express relationship cardinalities. Generalizations are supported by using UML generalizations.

The conceptual model as proposed by the authors can handle most of the constructs that are commonly used in a DTD. Further, some of the UML constructs such as UML attributes for classes do not have an equivalent XML representation and are suitably modified to adapt UML to represent most of the

XML specific features. This method is fairly successful in the conversion of DTD fragments into corresponding conceptual models. But since the author's work restricts the model to DTDs, the expressive power of the model is limited. The UML-based method also entails the user designing an XML conceptual model to learn the concepts of UML.

The authors project UML as the link between software engineering and document design as it provides a mechanism to design object-oriented software together with the necessary XML structures. The main goal behind conceptual modeling is to separate the designer's intention from the implementation details. The authors use UML to help achieve this by combining object-oriented design with XML document structures.

Figure 4 shows the UML for the structured paper DTD shown in Table 1 using the methodology in Conrad et al. (2000). This diagram was created manually by following the structural modeling examples shown in the Conrad article.

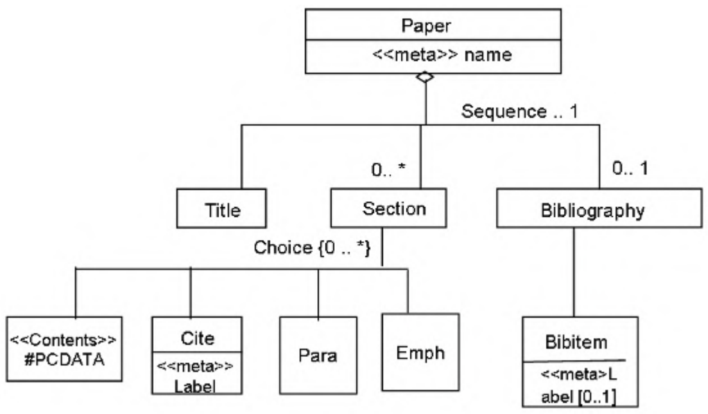
Unified Modeling Language (UML) and XML Schema

Routledge, Bird, and Goodchild (2002) attempt to define a mapping between the UML class diagrams and XML schema using the traditional three-level database design approach, which makes use of the conceptual, logical and physical design levels. The conceptual and logical levels are represented using UML class diagrams and they make use of the XML schema to specify the physical level.

The first step in this methodology is to model the domain using a conceptual level UML class diagram and to use this diagram to describe the various entities. This stage makes use of standard UML notations, and also extends the notation to represent attribute, relationships, and can also represent some conceptual constraints. Some of the common ER concepts are represented by modifying standard UML notations. For example, elements are represented as UML classes by making use of rectangles, and attributes are listed within the associated rectangles. Relationships are represented by lines linking two or more classes. Attributes and Relationships can have multiplicity constraints and are represented using standard UML notations. However, other conceptual constraints like Primary Key cannot be directly represented in UML, and instead some non-standard notations (such as affixing {P} to indicate a primary key attribute) are used.

Once the conceptual model has been validated (which requires a domain expert), the process involves the automatic conversion of the model to a logical level diagram, which describes the XML schema in a graphical and abstract manner.

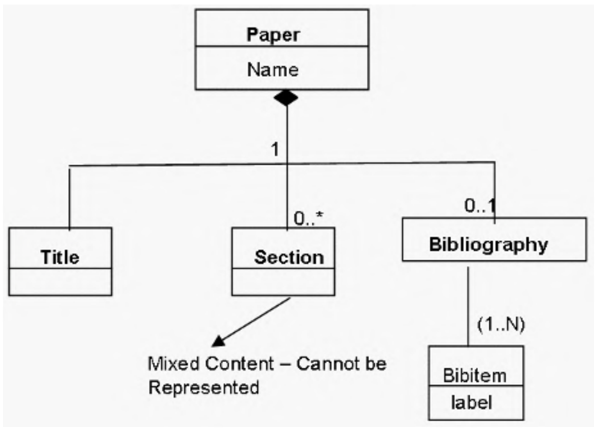
Figure 4. Conrad UML for the structured paper DTD



The logical level model, in most cases, serves as a direct representation of the XML schema data structures. The logical level model uses standard XML stereotypes such as "Simple Element" and "Complex Element" and "Sequence" that are defined in the UML profile for the XML schema. The previous definitions enable a direct representation of the XML schema components in UML. More details of the UML profile can be found in (Routledge et al., 2002). The paper describes in detail the steps needed to obtain a conceptual model and then to convert the conceptual model into a logical model. The third and final stage is the physical level, and involves the representation of the logical level diagram in the implementation language, namely XML schema. The authors have not included algorithms to directly convert the logical model to a schema and vice versa. This model entails the use of the conceptual and the logical view to define the XML schema. Since UML is aimed at software design rather than data modeling, new notations have to be added to fully describe the XML schema. Further mixed content in XML cannot be easily defined in UML, and the syntax to be used is different from the normal XML schema regular expression.

Figure 5 shows the UML for the structured paper DTD shown in Table 1 as obtained using the methodology in Routledge, et al. (2002). The diagram was generated by first converting the DTD into an equivalent XML schema using DTD2Xs (Rieger, 2003). The resulting schema was then converted manually using the mapping rules used by Routledge et al.

Figure 5. Routledge UML representation of structured paper DTD



Other Modeling Methods

Semantic Modeling Networks

XML schema does not concentrate on the semantics that underlie these documents, but instead depicts a logical data model. A conceptual model taking into account the semantics of a document has been proposed by Feng, Chang, and Dillon (2002). The methodology described by Feng, et al. can be broken into two levels: (1) semantic level and (2) schema level.

The first level is based on a semantic network, which provides a semantic model of the XML document through four major components:

1. Set of atomic and complex nodes representing real-world objects;
2. Set of directed edges representing the semantic relationships between objects;
3. Set of labels denoting the different types of semantic relationships such as aggregation, generalization, and so forth; and
4. Set of constraints defined over nodes and edges to constrain these relationships.

A semantic network diagram (see Figure 6) consists of a series of nodes interconnected using direct-labeled edges. It is possible to define constraints over these nodes and edges. Nodes can be either basic or complex, corresponding to simple or complex content respectively. Edges are used to connect nodes, thereby indicating a semantic relationship between them. This binary relationship is used to represent the structural aspect of real-world objects. Using edges, one can represent generalizations, associations, aggregations, and the "of-property". Constraints can be specified over nodes and edges to support various requirements such as domain, cardinality, strong/weak adhesion, order, homogeneity/heterogeneity, exclusion, and so forth. Cycles are possible in semantic diagrams, and to transform these diagrams into XML schema, it is necessary to convert the cyclic structures into an acyclic-directed graph.

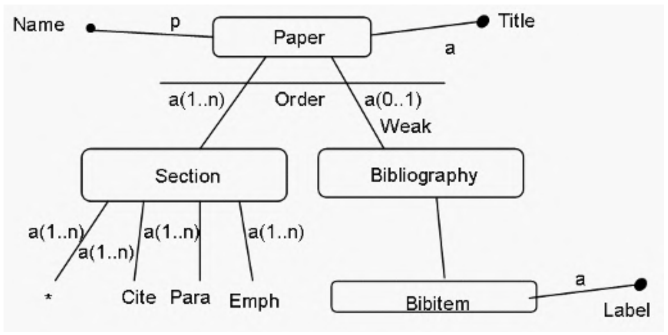
The second level is based on a detailed XML schema design, including element/attribute declarations and simple/complex type definitions. The main idea is that the mapping between these two levels can be used to transform the XML semantic model into a XML schematic model, which can then be used to create, modify, manage, and validate documents.

Figure 6 shows the Semantic Model representation for the structured paper DTD shown in Table 1. The diagram was generated by first converting the DTD into an equivalent XML schema using DTD2Xs (Rieger, 2003). The resulting schema was then converted using the semantic model-mapping rules mentioned by Feng et al.

XGrammar and the EER Model

Semantic data modeling capabilities of XML schemas are under utilized and XGrammar (Mani, Lee, & Muntz, 2001) makes an attempt to understand the

Figure 6. Semantic network model for the structured paper DTD



mapping between features of XML schema and existing models. Mani, et al. use a systematic approach to data description using XML schema and compare it to the ER model. The study formalizes a core set of features found in various XML schema languages into XGrammar — a commonly used notation in formal language theory. XGrammar is an extension of the regular tree grammar definition in (Murata, Lee, & Mani, 2001) which provided a six-tuple notation to describe a formal model for XML schema languages. XGrammar has extended this notation to include the ability to represent attribute definitions and data types.

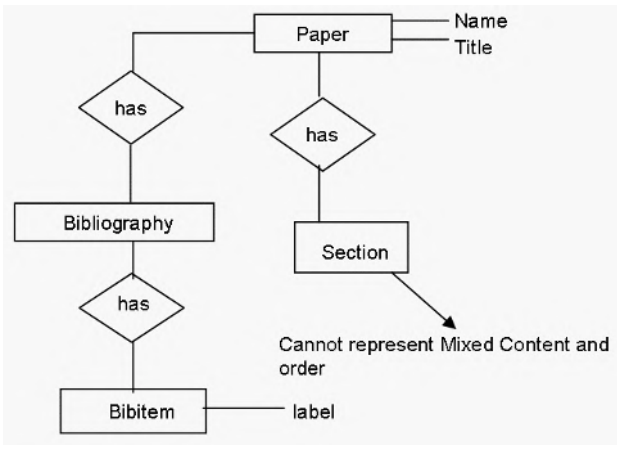
XGrammar has three features, namely: (1) ordered binary relationships, (2) union and Boolean operations, and (3) recursive relationships. Mani, et al. compare them to the standard ER model and, based on the comparison, extend the ER model to better support XML. This extension, called the Extended Entity Relationship Model (EER) has two main differences from the ER model: (1) modification of the ER constructs to better support order, and (2) introduction of a dummy "has" relationship to describe the element — sub-element relationship that is prevalent in XML.

The EER model (a typical EER model is shown in Figure 7) proposed by this study depends implicitly on the power of XGrammar. XGrammar introduces the ability to represent ordered binary relationships, recursive relationships, and also to represent a set of semantically equivalent but structurally different types as one. XGrammar also supports the ability to represent composite attributes, generalization hierarchy, and n-ary relationships. XGrammar, however, suffers in its representation because its grammar is loosely based on several existing schema languages rather than a generalized representation.

The article provides detailed rules necessary to convert XGrammar to EER and vice-versa. However, conversion rules for the more predominantly-used XML schema and DTD are not explored. Although the intermediate conversion to XGrammar is necessary to check for completeness, standard modeling practice can potentially generate the EER model directly without the intermediate step. As in the case of ERX, the EER model presented by (Mani et al., 2001) does not support XML-specific features such as mixed content, group indicators, and complex type entities. EER also lacks support for generalizations, and its main thrust is just on ordered binary relationships and IDREFs.

Figure 7 shows the XGrammar representation for the structured paper DTD shown in Table 1. This diagram was generated manually by creating an equivalent XGrammar first, and consequently mapping it into EER following the mapping methodologies described by Mani et al.

Figure 7. The XGrammar visualization for the structured paper DTD



Commercial Systems

Although XML has been in use as a standard for over five years, the task of designing XML structures has traditionally been a non-standard process. As the previous sections demonstrate, the research support in this area has been less than satisfactory. Tool support for the design of XML structures is also not well defined. Different tools have their own proprietary methods for graphically designing XML Structures. Most of these editors rely on the tree-based nature of XML schema and just provide a method for editing XML schemas. The underlying technology of these tools does not permit the ability to construct schemas or to enforce constraints. Tool support for providing interactive XML structure design is also not adequate as a standard mechanism for conceptual design of XML documents. Many companies have come up with several variants of XML-schema editors that will graphically present the main constructs of a schema to the user, including the ones mentioned in the chapter.

As before, we categorize the commercial tools into three broad categories: (1) ER-like models, (2) UML-based models, and (3) other models. Most commercial XML tools include graphical XML schema editors which resemble hierarchy editors. The visual structures of these editors are essentially the same, and in this section, we will hence refer to the third category as “tree-like models”.

ER-Like Models

Visual Studio .NET

Microsoft's Visual Studio .NET (Microsoft, 2003) includes XML Designer, which is a graphical XML schema editor. .NET uses connected rectangular blocks to present an overview of a schema with most of the structural details being hidden in dialog boxes. XML Designer provides a set of visual tools for working with XML schema, ADO.NET datasets, XML documents, and their supporting DTDs. The XML Designer provides the following three views (or modes) to work on XML files, XML schema, and XML datasets:

1. Schema View,
2. XML View, and
3. Data View.

The schema view provides a visual representation of the elements, attributes, types, and other constructs that make up XML schema and ADO.NET datasets. In the schema view, one can construct schema and datasets by dropping elements on the design surface from either the XML schema tab of the Toolbox or from Server Explorer. Additionally, one can also add elements to the designer by right-clicking the design surface and selecting Add from the shortcut menu. The schema view shows all complex types in table-like structures, and related types are connected through the type that relates them. Unfortunately, when the "ref" structure is used in XML schema, the connection is not shown, resulting in multiple separate disconnected structures (see Figure 8).

The Data view provides a data grid that can be used to modify ".xml" files. Only the actual content in an XML file can be edited in Data view (as opposed to actual tags and structure). There are two separate areas in Data view: Data Tables and Data. The Data Tables area is a list of relations defined in the XML file, in the order of their nesting (from the outermost to the innermost). The Data area is a data-grid that displays data based on the selection in the Data Tables area. The XML view provides an editor for editing raw XML and provides IntelliSense and color coding. Statement completion is also available when working on schema (.xsd) files and on XML files that have an associated schema. Specific details of XML Designer can be obtained from the Visual studio help page.

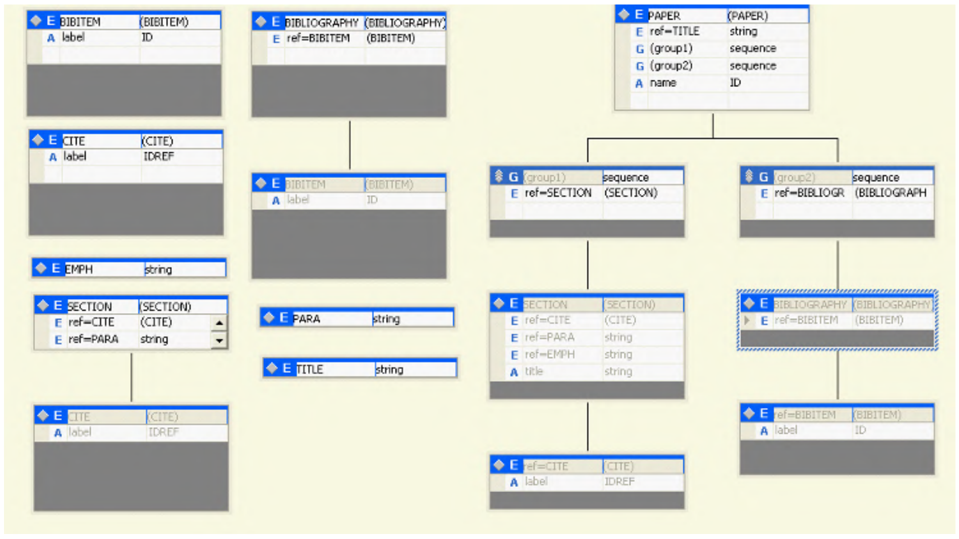
Figure 8 shows the Visual representation of the structured paper DTD shown in Table 1 using Visual Studio .NET. This diagram was generated by converting the DTD into its equivalent XML schema using DTD2Xs (Rieger, 2003) and importing the resulting schema into Visual Studio.

UML-Based Models

HyperModel

HyperModel (Ontogenics, 2003) by Ontogenics is a new approach to modeling XML applications. It introduces an agile design methodology and combines it with a powerful integration tool to aid in the software development process. HyperModel attempts to build a bridge between UML and XML development silos. The powerful model transformation capability of HyperModel allows different software integration technologies to intersect in a common visual language. HyperModel is different from other UML tools and has the ability to seamlessly integrate hundreds of industry standard XML schemas with various UML models. HyperModel supports the following features:

Figure 8. Visual Studio .Net visualization of the structured paper DTD



1. Imports any XML schema into a common UML-based presentation;
2. Facilitates integration between new XML design tools and widely deployed UML modeling tools;
3. Reverse engineers any W3C XML schema and produces an XML document that may be imported into other UML tools;
4. Generates XML schema definitions from any UML model, and is supported by a comprehensive UML extension profile for customization; and
5. Enables object-oriented analysis and design of XML schemas.

Figure 9 shows the HyperModel of the structured paper DTD shown in Table 1 using Ontogenics 1.2. This diagram was generated by converting the DTD into its equivalent XML schema using DTD2Xs (Rieger, 2003) and importing the resulting schema into HyperModel.

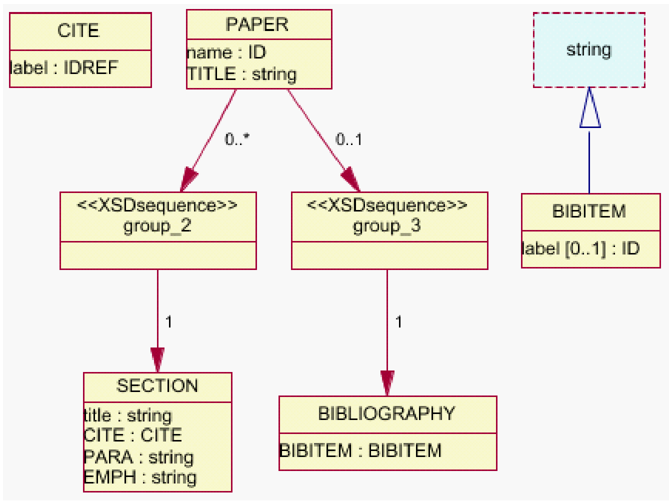
Tree-Like Models

XML Authority

XML Authority (Tibco, 2003) provides a visual representation of a DTD or a XML schema. It supports two different views, a tree representation and a tabular representation listing the various elements and attributes of the schema or a DTD in the Elements Pane. The Elements pane contains the element type declarations within a given schema. The pane is divided into three parts: a graphical view of the content model (the Content Model pane), a pane that lists element types that may contain the currently active elements (the Usage Context pane), and an editable list of element types and content models (the Element List pane).

The Content Model pane is located in the upper left hand area of the Elements pane and provides a graphical display of the Content Model for the currently active element type. Elements are represented as rectangles, and relationships between elements are displayed as lines connecting elements. Occurrence indicators and sequence indicators are also represented graphically. The Usage Context pane, located in the upper right hand area of the Elements pane, displays the possible parent elements of the currently selected element type. Element type declarations are defined and listed in the Element List pane at the bottom of the Elements Pane.

Figure 9. Visualization of the structured paper DTD using HyperModel



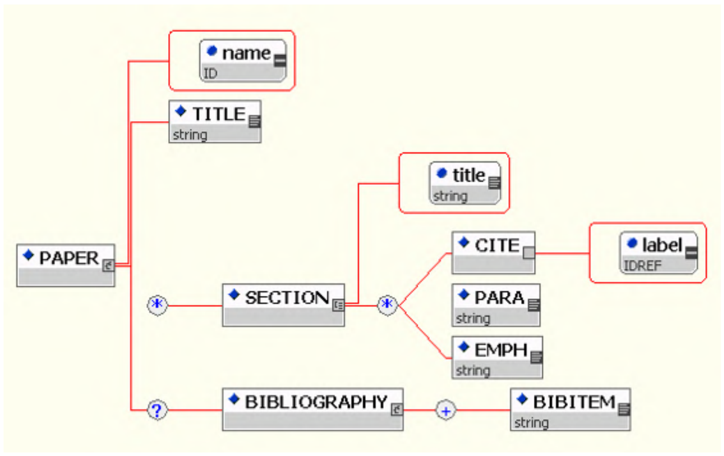
The Content Model pane uses a visual vocabulary to represent complex element content models. The boxes, containing element type names, data type indicators, and occurrence indicators, are the first part of this vocabulary, and the sequences as well as the choices between them are presented visually. Element types are displayed as objects (in boxes) within the Content Model pane. The content model may be composed of text, other elements, text and elements, data, or none of these (as defined in the element type definition). Each element type object may contain icons to indicate its contents, along with the element name. If an element type's content model includes other elements (that is, the [-] icon is displayed next to the element name), then the child elements may also be displayed in the content model pane.

Figure 10 shows the Visual representation of the structured paper DTD shown in Table 1 using XML Authority. This diagram was generated by importing the DTD into XML Authority.

XMLSPY

XMLSPY (Altova, 2004), is one of the most commonly used XML editors and features a visualization scheme to help manage XML. XMLSPY presents a consistent and helpful interface quite similar to that of the Visual Studio docking environment. XMLSPY supports the following different views for schema editing.

Figure 10. XML Authority - Visualization of structured paper DTD



1. The Text View includes code completion, syntax coloring, and validation as well as a check for well-formedness.
2. The Enhanced Grid View is a tabular view and helps perform XML editing operations on the document as a whole, instead of a line-by-line technique.
3. The Browser View helps provide previews for HTML outputs.
4. Authentic View provides developers with the ability to create customized views and data input forms.

In addition to the previous list, XMLSPY also supports a schema design/ WDSL view—an intuitive visual editing view that provides support for schema modeling. By default, the schema design/WDSL view displays as a list the various elements, complex types, and attribute and element groups. The Graphical view (the content model) can also be obtained for specific elements. This provides a global view of the chosen element. The element can be fully expanded by clicking on the "+" symbol displayed next to the elements name. XMLSPY provides a "Schema Navigator" to edit the schema in design view. Elements can be easily added to the content model by dragging them from the schema navigator window onto the desired position. Editing can be done by selecting the desired element and making the requisite changes in the properties window.

Figure 11. XML SPY — Visualization of the structured paper DTD

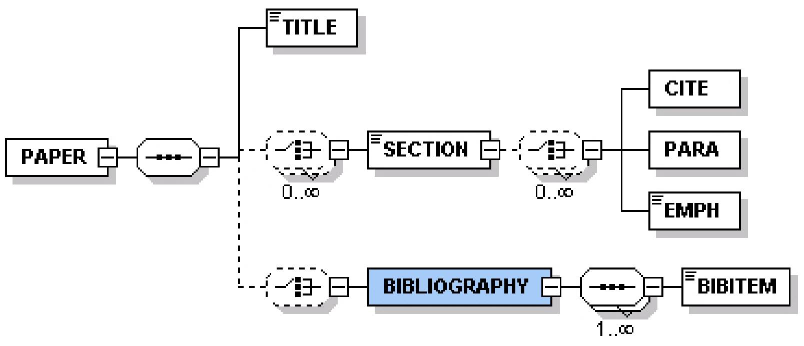


Figure 11 shows the Visual representation of the structured paper DTD shown in Table 1 using XMLSPY. This diagram was generated by converting the DTD into its equivalent XML schema using DTD2Xs (Rieger, 2003) and importing the resulting schema into XMLSPY.

Discussion

In this chapter, several research-based modeling approaches, as well as commercial tools for modeling XML applications were presented, all of which aim to visually represent XML structures, without the necessity of hand-editing the syntax-heavy DTD or XML schema. Table 3 summarizes the content-modeling capabilities of these techniques. Tools that support the XML schema (XSD) can also be used with XML DTDs, since all DTDs can be translated to a corresponding XML schema (the reverse is typically not true, since there are aspects of XML schema that cannot be modeled with DTDs). Based on the discussion of conceptual models earlier in this chapter, all of the research-based methods can be considered as potential conceptual models for XML, although among the commercial tools only HyperModel comes close to being a conceptual model. This chapter does not intend to provide an ordered comparison on the models. It is up to the XML community and the users to choose an appropriate modeling tool for their applications.

Table 3 rates all of the ten methods and tools discussed in this paper against nine different criteria. The first criterion *order* is one of the crucial elements in XML, and XGrammar is the only model which does not address this issue. All the models faithfully represent heterogeneity and complex XML content — two

*Table 3. Comparison of XML models (Legend: ✓ = full support, - = partial support, X = no support, * = Supported but not visually)*

	Order	Hetero	Complex	Mixed
ERX	-	✓	✓	-
XER	✓	✓	✓	✓
UML DTD	✓	✓	✓	✓
UML Schema	✓	✓	✓	X
Semantic	✓	✓	✓	-
XGrammar	X	✓	✓	X
VS. NET	✓	✓	✓	✓
HyperModel	✓	✓	✓	*
XML Auth	✓	✓	✓	✓
XML Spy	✓	✓	✓	✓

properties of XML applications which are uniformly represented in all the models. Mixed content is not a structural property of XML, and some of the models do not appropriately incorporate mixed content. Although all the models do not directly implement Document Type Definitions, that is not a major drawback since several tools exist that can translate DTDs into supported XML schema. Some of the models do not incorporate all the extra features of XML schema such as variations of mixed content, unordered entities, data types, and restrictions.

In Table 4, we refer to the process of translating an existing application up to the model as "up-translation" or reverse-translation, and the process of generating a logical schema from the conceptual model as the forward or "down-translation". Most models in this survey describe methods for generating the models from existing XML applications (and hence support up-translation), although not all the models can regenerate (or down-translate to) the original applications.

In summary, we believe that most of the models discussed here, with the exception of some of the commercial tools which simply represent a graphical representation of the hierarchy of a schema structure, can be successfully used for the purpose of designing conceptual models for XML-based applications.

Conclusion

In the title of this chapter, we asked the question — is conceptual modeling for XML a myth or a reality? Although we presented a number of models and tools

Table 4. Comparison of XML models (Legend: ✓ = full support, - = partial support, X = no support)

	DTD	Schema	Up- Tran	Down-Tran	Conceptual
ERX	✓	-	✓	✓	✓
XER	✓	✓	✓	✓	✓
UML DTD	✓	-	✓	✓	✓
UML Schema	✓	✓	✓	✓	✓
Semantic	✓	✓	✓	-	✓
XGrammar	✓	-	✓	-	✓
VS. NET	✓	✓	✓	✓	-
HyperModel	✓	✓	✓	✓	✓
XML Auth	✓	✓	✓	✓	X
XML Spy	✓	✓	✓	✓	X

that succinctly visualize an XML structure, visualizations of complex XML document structures typically are almost more overwhelming than the text-heavy schema or DTD. From 1986 when SGML was standardized, document authors have created DTDs manually, and it is the authors' viewpoint that for complex document model design with XML, manual input would continue to be the most frequently used method. However, XML is quickly becoming a method for data representation in Internet applications, and this is the domain where conceptual modeling tools would immensely assist in creating a good design. It is often debated whether data modeling is an art or a science. Although data models presented here can be automatically generated from existing applications, and new applications can likewise be created from the models, some of the component steps in modeling are often considered subjective (and hence artistic). Visually appealing models definitely aid this concept of data modeling. Therefore, visual conceptual models have been, and will remain, a crucial part of any project design. It is up to the XML community to choose one model that eventually gets accepted as a standard.

References

Altova Incorporated (2004). *XML Spy*. Retrieved from http://www.altova.com/products_ide.html

Apparao, V., Champion, M., Hors, A., Pixley, T., Byrne, S., Wood, L., et al. (1999). *Document object model level 2 specification* (Tech. Rep. No.WD-DOM-level-19990304). W3C Consortium.

Batini, C., Ceri, S., & Navathe, S. (1992). *An entity relationship approach*. CA: Benjamins Cummins and Menlo Park.

Booch, G., Rumbaugh, J., & Jacobson, I. (1998). *The unified modeling language user guide*. Boston: Addison Wesley.

Chen, P. (1976). The entity-relationship model — Toward a unified view of data. *Transactions of Database Systems (TODS), 1*(1), 9-36.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM, 6*(13), 377-387.

Conrad, R., Scheffner, D., & Freytag, J. (2000). XML conceptual modeling using UML. In *Proceedings of the International Conference on Conceptual Modeling 2000,* Salt Lake City, Utah (pp. 558-571). 1920. Springer LCNS.

Dia. (2004). *A drawing program*.

Feng, L., Chang, E., & Dillon, T. (2002). A semantic network-based design methodology for XML documents. *ACM Transactions on Information Systems, 4*(20), 390-421.

Fernandez, M., Malhotra, A., Marsh, J., & Nagy, M. (2003). *XQuery 1.0 and XPath Data Model* (W3C Working Draft).

ISO. (1986). *Information Processing Text and Office Systems — Standard Generalized Markup Language (SGML)*. International Organization for Standardization.

Kay, E. (2000). From EDI to XML. *ComputerWorld*. Retrieved October 1, 2005, from http://www.computerworld.com/softwaretopics/software/appdev/story/0,10801,45974,00.html

Layman, A. (1999). *Element-normal form for serializing graphs of data in XML*.

Malhotra, A. & Maloney, M. (1999). *XML schema requirements* (W3C Note). Retrieved October 1, 2005, from http://www.w3.org/TandS/QL/QL98/pp/microsoft-serializing.html

Mani, M., Lee, D., & Muntz, R. (2001). Semantic data modeling using XML schemas. *Proceedings of the International Conference on Conceptual Modeling* (pp. 149-163), London, Springer Verlag.

Microsoft. (2003). *The .NET Framework*. Retrieved October 1, 2005, from http://www.microsoft.com/net

Mohr, S. (2001). Thinking about e-business: A layered model for planning. *Unisys World White Papers*. Retrieved October 1, 2005, from http://www.unisysworld.com/webpapers/2001/02_xmlabs.shtml)

Murata, M., Lee, D., & Mani, M. (2001). Taxonomy of XML schema languages using formal language theory. In *Extreme Markup Languages*.

Naur, P., Backus, J. W. (1963). Revised report on the algorithmic language ALGOL 60. *Communications of the ACM, 1*(6), 1-23.

Nguyen, V., & Hailpern, B. (1982). A Generalized Object Model. *ACM SIGPLAN Notices, 9*(17).

Ogbuji, U. (1999). XML: The future of EDI? *ITWorld/Unix Insider*. Retrieved October 1, 2005, from http://uche.ogbuji.net/tech/pubs/xmledi.html

OMG. (2003). *XML Metadata Interchange (XMI) Specification — Version 2.0*. Retrieved October 1, 2005, from http://www.omg.org/docs/formal/03-05-02.pdf

Ontogenics. (2003). Ontogenics Web site on HyperModel.

Psaila, G. (2000). ERX: A conceptual model for XML documents. In *ACM Symposium of Applied Computing (SAC 2000)*. Como. (pp. 898-903). ACM Press.

Psaila, G. (2003). From XML DTDs to entity-relationship schemas. In *Lecture Notes in Computer Science, Vol. 2814*. Italy (pp. 278-289). Springer.

Rieger, J. (2003). *DTD2Xs v1.60 DTD to XML Schema Converter*. Retrieved October 1, 2005, from http://www.w3.org/2000/04/schemahack

Routledge, N., Bird, L., & Goodchild, A. (2002). UML and XML schema. In *Thirteenth Australasian Database Conference 2002*.

Sengupta, A., Mohan, S., & Doshi, R. (2003). XER: Extensible entity relationship modeling. In *XML 2003*, Philadelphia: Idea Alliance.

Sun. (2004). *Java 2 Platform, Enterprise Edition: Sun Microsystems*. Retrieved October 1, 2005, from http://www.sun.com

Tibco. (2003). Tibco's Web site on XML Authority. Retrieved October 1, 2005, from http://www.tibco.com/solutions/extensibility/

Endnote

1 http://www.omg.org/technology/documents/formal/xmi.htm

Chapter XI

Constraint-Based Multi-Dimensional Databases

Franck Ravat, Université Toulouse I, France
Olivier Teste, Université Toulouse III, France
Gilles Zurfluh, Université Toulouse I, France

Abstract

This chapter deals with constraint-based multi-dimensional modelling. The model we define integrates a constellation of facts and dimensions. Along each dimension, various hierarchies are possibly defined and the model supports multiple instantiations of dimensions. The main contribution is the definition of intra-dimension constraints between hierarchies of a same dimension as well as inter-dimension constraints of various dimensions. To facilitate data querying, we define a multi-dimensional query algebra, which integrates the main multi-dimensional operators such as rotations, drill down, roll up… These operators support the constraint-based multi-dimensional modelling. Finally, we present two implementations of this algebra. First, OLAP-SQL is a textual language integrating multi-dimensional concepts (fact, dimension, hierarchy), but it is based on classical SQL syntax. This language is dedicated to *specialists such as multi-dimensional database administrators. Second, a graphical query language is presented. This language consists in a graphical representation of multi-dimensional databases, and users specify directly their queries over this graph. This approach is dedicated to non-computer scientist users.*

Introduction

OnLine Analytical Processing (OLAP) has emerged to support multi-dimensional data analysis by providing manipulations through aggregations of data drawn from various transactional databases. This approach is often based on multi-dimensional databases. The multi-dimensional modelling (Kimball, 1996) represents data as points in multi-dimensional space. Data are viewed as a subject of analysis (fact) associated to axis of analysis (dimensions). Each dimension contains one or several viewpoints of analysis (hierarchies) representing data granularities. For example, *sale amounts* could be analysed according to *time*, *stores*, and *customers*. Along *store* dimension, a hierarchy could group *individual stores* into *cities*, which are grouped into *states* or *regions*, which are grouped into *countries*.

This approach induces topics of interests for the scientific community (Rafanelli, 2003). The main issues focus on technologies and tools that enable the business intelligence lifecycle from data modelling and acquisition to knowledge extraction. These problems are based on researches, which deal with design methods, multi-dimensional models, OLAP query languages, and tools that facilitate data extraction and data warehousing. Multi-dimensional data are crucial for the decision-making. Nevertheless, only a few researches focus on multi-dimensional data integrity (Hurtado & Mendelzon, 2002).

The confidence in a multi-dimensional database lies in its capacity to supply relevant information. A multi-dimensional model integrating constraints must provide an accurate model of the organisation activities, and it allows valid data restitution (Hurtado & Mendelzon, 2002). This chapter deals with constraint-based multi-dimensional modelling and querying.

The chapter outline is composed of the following sections. The second section gives an overview of related works. The third section defines a constellation model where dimensions support multiple instantiations as well as multiple hierarchies. The fourth section specifies a query algebra. We show the effect of constraints during multi-dimensional manipulations. The fifth section presents a tool named Graphic-OLAPSQL. It supports two languages: a SQL-like language for administrators, and a graphical language dedicated to casual users as decision makers.

Background

Efforts to model multi-dimensional databases have followed two directions; some models extend relational approaches to integrate multi-dimensional data, and others approaches model directly and more naturally multi-dimensional data. Multi-dimensional databases are manipulated through data cubes.

- In the relational context, the data cube operator (Gray, Bosworth, Layman, & Pirahesh, 1996) was introduced to expand relational tables by computing the aggregations over all the attribute combinations. Kimball (1996) introduces multi-dimensional models based on dimension tables and fact tables, whereas Li and Wang (1996) represent cubes through dimension relations and functions, which map measures to grouping relations. Barralis, Paraboschi, and Teniente (1997) consider multi-dimensional databases as a set of tables forming de-normalised star schemata.
- To integrate more naturally multi-dimensional data, Agrawal, Gupta, and Sarawagi (1997) introduce a model, which supports a symmetric treatment of dimensions and measures, and it provides a set of operators (manipulation of cubes). Several approaches were presented that support cubes with n-dimensions (Gyssen & Lakshmanan, 1997) and cubes integrating explicitly multiple hierarchies (Agrawal, Gupta, & Sarawagi, 1997; Lehner, 1998; Mendelzon & Vaisman, 2000; Vassiliadis & Sellis, 1999). Cabbibo and Torlone (1997) define a multi-dimensional database through dimensions, which are constructed from hierarchies of dimension levels, and f-tables, which store factual data. Lehner (1998) presents a model based on primary multi-dimensional objects, which represent cubes, and secondary multi-dimensional objects, which consist of all dimension levels. In Vassiliadis and Sellis (1999), cubes integrate explicitly multiple hierarchies.

We also find scientific literature devoted to OLAP query manipulations. The first dedicated operator was the data cube operator in the relational context (Gray et al., 1996). An important contribution is the definition of roll-up and drill-down operators (Agrawal et al., 1997; Lehner, 1998). Agrawal, et al. (1997) introduce pull for transforming measures into parameters and its reciprocal operation, named push, whereas Gyssen and Lakshmanan (1997) extend this idea through more complex operations such as fold for transforming dimensions into facts and unfold for converting facts into dimensions. Lehner (1998) defines rotations between dimensions. Cabbibo and Torlone (1998) extend Cabbibo and Torlone (1997) by defining a graphical language based on a query algebra, which integrates roll-up operator. Abello, Samos, and Saltor (2003) implement an algebraic set of operators on top of an R-OLAP database.

Although many database techniques have been revisited or newly developed in the context of multi-dimensional databases, little attention has been paid to the data integrity. This problem is crucial because decision-makings are based on multi-dimensional data. For example, geographic hierarchy which groups *cities* into *departments*, *regions,* and *countries* is consistent for French cities, whereas this hierarchy is irrelevant regarding to U.S. geography. Kimball (1996) refers this problem where incompatible hierarchies such as French geography and U.S. geography are modelled in a single heterogeneous dimension. Lehner (1998) proposes transforming dimensions into dimensional normal form. Pedersen and Jensen (1999) provides class dimensions, which are transformed into homogeneous dimensions by adding *null* information. These approaches do not integrate these inconsistencies between hierarchies. Hurtado and Mendelzon (2002) provide dimension constraints to reduce hierarchy inconsistencies; this approach introduces frozen dimensions, which are minimal homogeneous dimensions representing the structures that are implicitly combined in a heterogeneous dimension. Some problems are not taken into account in these approaches. For example, we consider the analysis of *sale amounts* according to *products* and *stores*. If a French taxonomy is defined along *product* dimension, it seems to be inconsistent to analyse *sale amounts* of these products according to U.S. *stores*, which are grouped into the U.S. geography.

Multi-Dimensional Modelling Under Constraints

This section defines a conceptual multi-dimensional model, which is based on facts, dimensions, and hierarchies. This model facilitates correlations between several subjects of analysis through a constellation of facts and dimensions. Along each dimension, various hierarchies are possibly defined. This model supports multiple instantiations of dimensions; for example, each dimension instance belongs to one or more hierarchies.

This model also integrates semantic constraints permitting the validation of data extraction and multi-dimensional data manipulation. These constraints are defined over one or several dimensions.

Fact

Presentation: A fact reflects information that has to be analysed through one or several indicators, called measures; for example, a fact data may be *sale amounts occurring in shops*.

Definition: A fact is defined as (N^F, M^F, I^F, $IStar^F$).

- N^F is a name of fact,
- M^F is a set of measures associated with an aggregate function; $M^F = \{f1(m^F_1), f2(m^F_2)\ldots\}$,
- I^F is a set of fact instances,
- $IStar^F$ is a function, which respectively associates fact instances to their linked dimension instances.

Formalism: A fact is represented as shown in Figure 1.

Figure 1. Fact graphical formalism

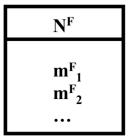
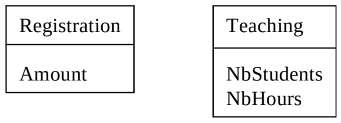


Figure 2. Example of facts



Example: The case we study is taken from the education domain. We define a multi-dimensional database, which allows users to analyse student registrations as well as teachings in French and American universities. Student registrations are analysed through the amount whereas teachings are analysed through a number of students and a number of hours.

According to the previous textual definition, we represent these needs through two facts defined as follows:

- F_1= (Registration, {avg(Amount)}, I^{F1}, $IStar^{F1}$),
- F_2= (Teaching, {sum(NbStudents), sum(NbHours)}, I^{F2}, $IStar^{F2}$),
- I^{F1}, $IStar^{F1}$, I^{F2}, $IStar^{F2}$ are depicted in the following sections.

The graphical representation of F_1 and F_2 is described in Figure 2.

Dimension

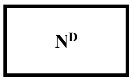
Presentation: A dimension reflects information according to which measures will be analysed. A dimension is composed of parameters, which are organised through one or several hierarchies; for example, dimensions may be *store, product, date…*

Definition: A dimension is defined as (N^D, A^D, H^D, I^D).

- N^D is a name of dimension,
- A^D is a set of attributes,
- H^D is a set of hierarchies,
- I^D is a set of dimension instances.

Formalism: A dimension is represented as shown in Figure 3.

Figure 3. Dimension graphical formalism



Example: We want to analyse the measures of "Teaching" through four axes of analysis respectively, representing:

- Universities where teachings are done (the analysis integrates both French Universities and U.S. Universities),
- Professors who insure teachings (each professor is classified according to one status),
- Dates (this dimension regroups temporal information),
- Course descriptions (the French courses are classified according to modules and degree, whereas the U.S. courses belong to areas).

In the same way, the measure of "Registration" is analysed through course descriptions, dates, and students.

The dimension, called "University", models geographic information. Each university belongs to a city, a zone (North, South,...), a country, and a continent. French geography groups cities in departments and regions whereas U.S. geography groups cities in states. The following expression represents the dimension D_1 named "University".

D_1 = ("University", {IdU, Uname, City, Department, Region, Country, Continent, IdZone, NameZone, State}, {FRGEO, USGEO, ZN}, $I^{University}$)

In the following, we describe some instances of the dimension "University":

$I^{University} = \{i_1, i_2, i_3, \ldots\}$ such as

- i_1 = ["U1", "Paul Sabatier University", "Toulouse", 31, "Midi-Pyrénées", "France", "Europe", "S_FR", "South of France", NULL],
- i_2 = ["U2", "Toulouse I University", "Toulouse", 31, "Midi-Pyrénées", "France", "Europe", "S_FR", "South of France", NULL],

- i_3 = ["U3", "Stanford University", NULL, NULL, NULL, "USA", "America", "SW_US", "South West of USA", "California"],…

Hierarchy

Presentation: In a same dimension, parameters are organised according to one or several hierarchies. Within a dimension, values represent several data granularities according to which measures could be analysed; along *store* dimension, a hierarchy could group *individual stores* into *cities*, which are grouped into *states* or *regions*, which are grouped into *countries*.

Definition: A hierarchy is defined as (N^H, $Param^H$, $Weak^H$, $Cond^H$).

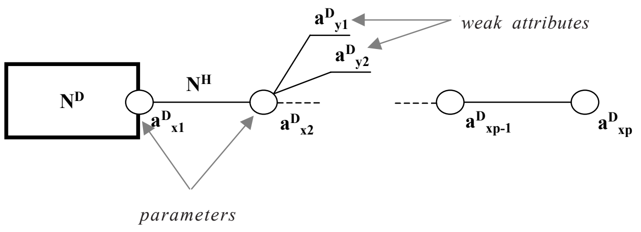
- N^H is a name of a hierarchy,
- $Param^H$ is an ordered set of attributes, called parameters, which represent a level of granularity along the dimension,
- $Weak^H$ is a function associating each parameter to one or several weak attributes, which complete the parameter semantic,
- $Cond^H$ is a condition, which determines a valid set of dimension instances related to the hierarchy.

This element ($Cond^H$) represents an important contribution of the model because it allows multiple instantiations of dimensions; for example, each dimension instance belongs to one or more hierarchies. Thus, a sub-set of dimension instances is associated to one hierarchy. This property confers a heterogeneous structure integration in one dimension.

Formalism: A hierarchy is represented as shown in Figure 4.

Example: Each dimension can be manipulated through various combinations of parameters. In the model, hierarchies describe these parameter combinations. For example, the dimension called "University" may be visualised through the French geography, the U.S. geography, and the zone decomposition. So, along this dimension, we define three hierarchies respectively, called FRGEO, USGEO, and ZN.

Figure 4. Hierarchy graphical formalism



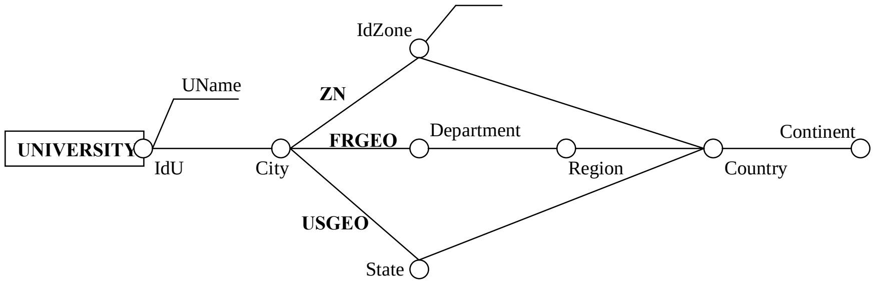
The textual definition of the three hierarchies of the dimension called "University" is defined as follows:

- FRGEO = (French_GEO, <IdU, City, Department, Region, Country, Continent>, {(IdU, Uname)}, $Cond^{FRGEO}$).
- USGEO = (US_GEO, <IdU, City, State Country, Continent>, {(IdU, Uname)}, $Cond^{USGEO}$).
- ZN = (Zone_GEO, <IdU, City, IdZone Country, Continent>, {(IdU, Uname), (IdZone, NameZone)}, $Cond^{ZN}$).

The graphical representation of the dimension called "University" is defined in the schema shown in Figure 5.

The model supports multiple instantiations of dimensions; for example, each dimension instance belongs to one or more hierarchies. This multiple instantiation property is defined through a condition associated to each hierarchy.

Figure 5. The dimension "University"



- $\text{Cond}^{FRGEO} = dom(\text{Country}) \in \{\text{“France”}\}$
- $\text{Cond}^{USGEO} = dom(\text{Country}) \in \{\text{“US”}\}$
- $\text{Cond}^{ZN} = dom(\text{IdZone})$ IS NOT NULL

We apply this property to the set of instances $I^{University} = \{i_1, i_2, i_3, \ldots\}$.

- $\{i_1, i_2\} \in$ FRGEO
- $\{i_3\} \in$ USGEO
- $\{i_1, i_2, i_3\} \in$ ZN

Constellation

Presentation: A constellation extends star schemas (Kimball, 1996), which are commonly used in the multi-dimensional context. A constellation regroups several subjects of analysis (facts), which are studied according to several axes of analysis (dimensions) possibly shared between facts.

Definition: A constellation is defined as (N^C, F^C, D^C, $Star^C$, $Cons^C$).

- N^C is a name of a constellation,
- F^C is a set of facts,
- D^C is a set of dimensions,
- $Star^C$ associates each fact to its linked dimensions, and
- $Cons^C$ is a set of constraints between the hierarchies.

This element ($Cons^C$) is the second main contribution of the model. These constraints insure the constellation consistence, for example, to prevent invalid analyses based on incompatible hierarchies.

Formalism: A constellation regroups a set of facts and dimensions possibly shared. Its representation is composed of fact, dimension, and hierarchy formalisms. Note that this formalism extends notations of (Golfarelli, Maio, & Rizzi, 1998).

Example: In the previous examples, we defined a constellation composed of two facts and five dimensions:

- The facts, called "Registration" and "Teaching" were defined previously.
- The dimension, called "University", is detailed in the top.
- The dimension, named "Course", is shared by two facts; it is composed of three parameters (Degree and module specific to French courses and area specific to U.S. courses) and one weak attribute named "Description", which complete the semantic of a parameter related to the course identification.
- The dimension, named "Date", is also shared by two facts, and regroups attributes related to the Gregorian calendar.
- The dimension, called "professor", models professors through its name, surname, and status.
- The dimension, called "student", represents students defined by name, surname, sex, and identifier.

In the schema shown in Figure 6, we represent a constellation schema responding to the needs defined in the previous examples.

Constraints

To maintain data relevance, the approach we present in this chapter deals with constraints.

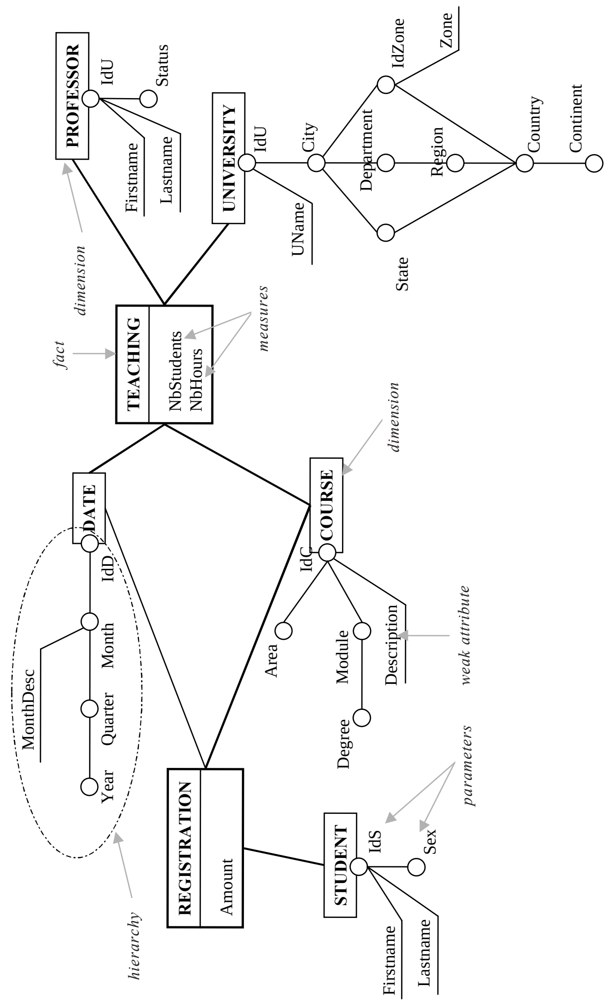
We express interactions between hierarchies (perspectives of analysis) in the same dimension. We solve this problem through **intra-dimension constraints**.

Another need concerns the association of fact instances to dimension instances. We solve this problem through **inter-dimension constraints**.

Intra-Dimension Constraints

The model we define supports five intra-dimension constraints. We define exclusion, inclusion, simultaneity, totality, and partition between hierarchies in one dimension.

Figure 6. Example of a constellation schema

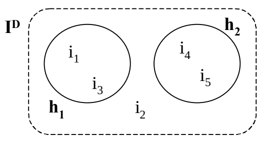
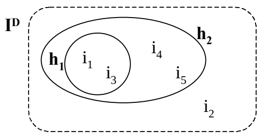
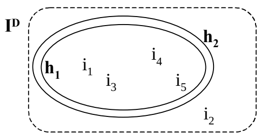
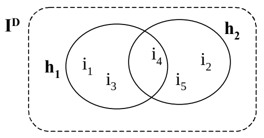
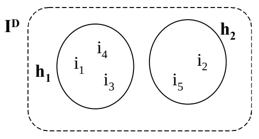


Let us considers one dimension denoted D, two hierarchies along the dimension denoted h_1 and h_2 ($h_1 \in H^D$, $h_2 \in H^D$), and several dimension instances such as $I^D = \{i_1, i_2, i_3, i_4, i_5, \ldots\}$.

Table 1 gives definitions of these constraints.

Example: We consider the dimension named "University". A French University is not located in the USA and vice versa. This semantic constraint is modelled through one exclusion constraint.

Table 1. Definitions of intra-dimension constraints

Constraints	Definitions	Illustrations
`Exclusion:` $h_1 \otimes h_2$	The dimension instances belong to h_1 do not belong to h_2 (and vice versa): $\forall i_1 \in h_1 \wedge \forall i_2 \in h_2 \Rightarrow i_1 \neq i_2$ Note that $h_1 \otimes h_2 = h_2 \otimes h_1$	
`Inclusion:` $h_1 \circledcirc h_2$	All the dimension instances belonging to h_1 belong to h_2: $\forall i_1 \in h_1 \Rightarrow i_1 \in h_2$ Note that $h_1 \circledcirc h_2 \neq h_2 \circledcirc h_1$	
`Simultaneity:` $h_1 \ominus h_2$	All the dimension instances belonging to h_1 belong to h_2 (and vice versa): $\forall i_1 \in h_1 \Leftrightarrow i_1 \in h_2$ Note that - $h_1 \ominus h_2 = h_2 \ominus h_1$ - $h_1 \ominus h_2 \Leftrightarrow h_1 \circledcirc h_2 \wedge h_2 \circledcirc h_1$	
`Totality:` $h_1 \ominus h_2$	All the dimension instances belong to h_1 and/or h_2: $\forall i_1 \in I^D, i_1 \in h_1 \vee i_1 \in h_2$ Note that $h_1 \ominus h_2 = h_2 \ominus h_1$	
`Partition:` $h_1 \oslash h_2$	All the dimension instances belong to h_1 or (exclusive) h_2: $\forall i_1 \in I^D, (i_1 \in h_1 \wedge i_1 \notin h_2) \vee (i_1 \notin h_1 \wedge i_1 \in h_2)$ Note that - $h_1 \oslash h_2 = h_2 \oslash h_1$ - $h_1 \oslash h_2 \Leftrightarrow h_1 \otimes h_2 \wedge h_1 \ominus h_2$	

- FRGEO $\otimes$ USGEO

Each University (French or U.S.) belongs to a unique zone ($\{i_1, i_2, i_3\} \in$ ZN).This semantic constraint is modelled through two inclusion constraints.

- FRGEO $\circledcirc$ ZN
- USGEO $\circledcirc$ ZN

In Figure 7, we represent the complete definition of the dimension called "University".

Figure 8 illustrates instance organisation of the dimension named "University". Each instance belongs to various sets according to the hierarchy conditions.

Figure 7. Schema of the dimension "University"

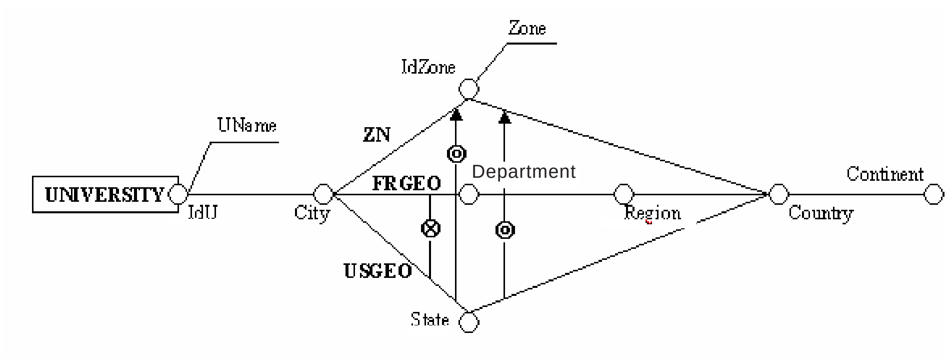
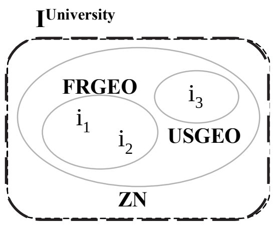


Figure 8. Instances of the dimension "University"



Inter-Dimension Constraints

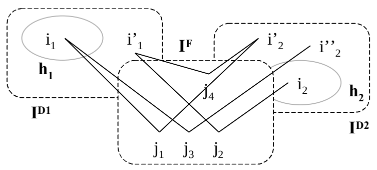
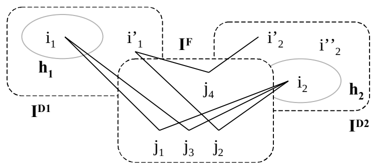
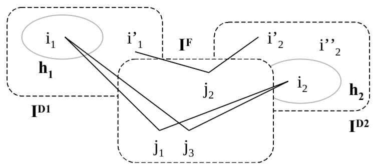
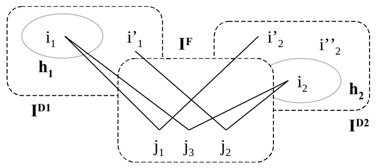
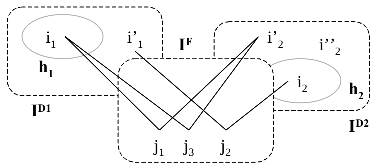
The model provides five inter-dimension constraints related to hierarchies belonging to different dimensions. We define exclusion, inclusion, simultaneity, totality, and partition between inter-dimension hierarchies.

Let us considers one fact denoted F, two dimensions denoted D_1 and D_2, which are associated to F ($Star^C(F)=\{\ldots, D_1,\ldots, D_2,\ldots\}$), two hierarchies along each dimension denoted h_1 and h_2 ($h_1 \in H^{D1}$, $h_2 \in H^{D2}$), and several instances such as $I^F = \{j_1, j_2, j_3, j_4,\ldots\}$, $I^{D1} = \{i_1, i'_1,\ldots\}$, $I^{D2} = \{i_2, i'_2, i''_2,\ldots\}$.

Table 2 gives definitions of these constraints.

Example: We complete the definition of our constellation schema. We define the dimension called "Course" by ("Course", { IdC, Description, Module, Degree, Area }, {FRCRS, USCRS}, I^{Course}). The hierarchies are defined as follows:

Table 2. Definitions of inter-dimension constraints

Constraints	Definitions	Illustrations
Exclusion: $h_1 \otimes h_2$	$\forall j \in I^F, \exists i_1 \in h_1 \mid i_1 \in IStar^F(j) \Rightarrow$ $\neg(\exists i_2 \in h_2 \mid i_2 \in IStar^F(j))$ Note that $h_1 \otimes h_2 = h_2 \otimes h_1$	 i1 i'1 IF i'2 i''2 h1 j4 i2 h2 ID1 ID2 j1 j3 j2
Inclusion: $h_1 \circledcirc h_2$	$\forall j \in I^F, \exists i_1 \in h_1 \mid i_1 \in IStar^F(j) \Rightarrow$ $\exists i_2 \in h_2 \mid i_2 \in IStar^F(j)$ Note that $h_1 \circledcirc h_2 \neq h_2 \circledcirc h_1$	 i1 i'1 IF i'2 i''2 h1 j4 i2 h2 ID1 ID2 j1 j3 j2
Simultaneity: $h_1 \ominus h_2$	$\forall j \in I^F, \exists i_1 \in h_1 \mid i_1 \in IStar^F(j) \Leftrightarrow$ $\exists i_2 \in h_2 \mid i_2 \in IStar^F(j)$ Note that - $h_1 \ominus h_2 = h_2 \ominus h_1$ - $h_1 \ominus h_2 \Leftrightarrow h_1 \circledcirc h_2 \wedge h_2 \circledcirc h_1$	 i1 i'1 IF i'2 i''2 h1 j2 i2 h2 ID1 ID2 j1 j3
Totality: $h_1 \ominus h_2$	$\forall j \in I^F, (\exists i_1 \in h_1 \mid i_1 \in IStar^F(j)) \vee$ $(\exists i_2 \in h_2 \mid i_2 \in IStar^F(j))$ Note that $h_1 \ominus h_2 = h_2 \ominus h_1$	 i1 i'1 IF i'2 i''2 h1 i2 h2 ID1 ID2 j1 j3 j2
Partition: $h_1 \oslash h_2$	$\forall j \in I^F, (\exists i_1 \in h_1 \mid i_1 \in IStar^F(j) \wedge$ $\neg\exists i_2 \in h_2 \mid i_2 \in IStar^F(j)) \vee (\neg\exists i_1 \in h_1 \mid$ $i_1 \in IStar^F(j) \wedge \exists i_2 \in h_2 \mid i_2 \in IStar^F(j))$ Note that - $h_1 \oslash h_2 = h_2 \oslash h_1$ - $h_1 \oslash h_2 \Leftrightarrow h_1 \otimes h_2 \wedge h_1 \ominus h_2$	 i1 i'1 IF i'2 i''2 h1 i2 h2 ID1 ID2 j1 j3 j2

- FRCRS = ("FR_COURSE", <IdC, Module, Degree>, {(IdC, Description)}, Module IS NOT NULL Ù Degree IS NOT NULL).
- USCRS = ("US_COURSE", <IdC, Area>, {(IdC, Description)}, Area IS NOT NULL).

The instances of the dimension called "Course" (A^{Course} = {IdC, Description, Module, Degree, Area}) are defined as follows:

$I^{Course} = \{i'_1, i'_2, i'_3\ldots\}$ such as

- i'_1 = ["C1", "BD-R", "Mod. BD", "Licence", NULL],
- i'_2 = ["C2", "BD-OR", "Mod. BD", "Licence", NULL],
- i'_3 = ["C3", "Relational DB", NULL, NULL, "Computer Science"],

The courses identified by C1 and C2 are courses of French Universities U1 and U2. This semantic constraint is modelled by the following expression:

- FRCRS $\otimes$ USGEO

In the same way, we can define the constraint USCRS $\otimes$ FRGEO.

Figure 9 illustrates instance organisation of the dimensions named "University" and "Course" as well as the fact named "Teaching". Each instance belongs to various sets according to the hierarchy conditions.

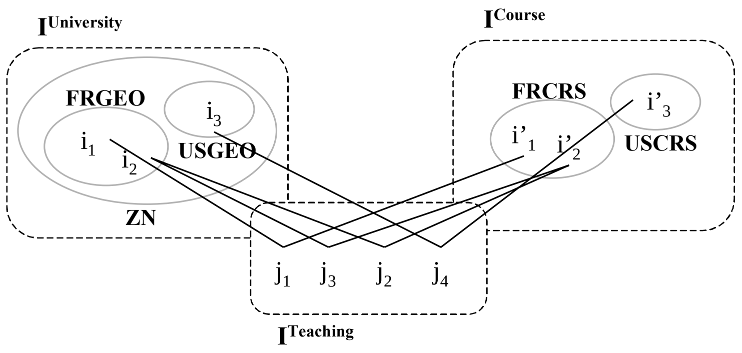
Multi-Dimensional Query Algebra

This section both provides a simple and elegant language close to the conceptual view rather than its formal definition; this algebra supports complex analysis through combinations of basic algebra operators. We also describe a multi-dimensional table structure to display query results. In the following, we study effects of the constraint-based multi-dimensional model that we presented in the second section.

Multi-Dimensional Table

A query result is displayed through a multi-dimensional table. In order to provide a reasonably simple way to querying multi-dimensional data, we define a multi-dimensional table allowing us to clearly separate structures and contents. This table is more complex than a relational query result because multi-dimensional tables are organised according to a non-clear separation between structural aspects and data contents (Gyssen & Lakshmanan, 1997).

Figure 9. Example of instances under inter-dimension constraints



Definition: A multi-dimensional table T is defined as (F, {f(m$_1$),f'(m$_2$),…}, D$_1$, HI$_1$, Att$_1$, D$_2$, HI$_2$, Att$_2$, pred).

- F represents a name of fact,
- {f(m$_1$),f'(m$_2$),…} is a set of displayed measures, which are associated to an aggregate function f,
- D$_1$ and D$_2$ are names of dimensions, which are linked to the displayed fact,
- HI$_1$={h$_{1\_1}$, h$_{1\_2}$…} and HI$_2$={h$_{2\_1}$, h$_{2\_2}$…} are non-empty sets of hierarchies along respectively D$_1$ and D$_2$,
- Att$_1$ and Att$_2$ are lists of displayed parameters along D$_1$ and D$_2$, according to HI$_1$ and HI$_2$, and
- pred is a predicate defining value domains of the table elements (fact, dimensions).

Definition: The display operator permits the generation of a first multi-dimensional table from a constellation. It is defined as DISPLAY(C, F, {f(m$_1$), f'(m$_2$),…}, D$_1$, HI$_1$, D$_2$, HI$_2$) = T$_R$ where:

- C is a constellation,
- F represents the displayed fact, and {f(m$_1$),f'(m$_2$),…} is a set of these measures, which are displayed according to an aggregate function,
- D$_1$ and D$_2$ are names of dimensions respectively displayed on rows and columns, and

- HI_1 and HI_2 represent current hierarchies of D_1 and D_2. Note that the displayed parameter of D_1 (respectively D_2) is the parameter representing the higher granularity of the current hierarchies. This parameter must be shared between all the hierarchies of HI_1 (respectively HI_2).

Example: In order to display a table showing the number of students by European Universities and year, users can define the following algebraic expression:

Display(SchoolConstellation, TEACHING, {*SUM*(NbStudents) }, DATE, HY, UNIVERSITY, FRGEO) = T_{R1}

In the following sub-sections, we describe a set of multi-dimensional operators. These basic operators use an entry multi-dimensional table T, and they build a multi-dimensional table result, denoted T_R. This closure property allows users to make complex queries by a combination of basic operators.

Rotation of Hierarchies

Presentation: The rotation between hierarchies consists in changing the current set of hierarchies along a dimension in a multi-dimensional table. By default, the analysis is based on the parameter of higher granularity along the new hierarchies of the studied dimension.

Definition: A rotation is defined by HRotate(T, D_i, HI_{i_new}) = T_R.

Figure 10. Multi-dimensional table T_{R1}

TEACHING (NbStudent)		**UNIVERSITY | FRGEO**	
		Continent	EUROPE
DATE | HY	**Year**		
	2001		600000
	2002		620000
	2003		610000
	2004		610500
PROFESSOR.All			
COURSE.All			

- $T = (F, \{f(m_1), f'(m_2), \ldots\}, D_1, HI_1, Att_1, D_2, HI_2, Att_2, pred)$ is an entry table,
- D_i is the dimension along the rotation is applied, and
- HI_{i_new} is the new set of hierarchies representing a new perspective of analysis; all these hierarchies must share the same parameter of higher granularity.

Remarks: The intra-dimension constraint specification implies the following treatments:

- If an exclusion constraint (and/or partition constraint) is defined between a hierarchy of HI_i and a hierarchy of HI_{i_new}, T_R represents a new analysis.
- If an inclusion constraint is defined from a hierarchy of HI_i to a hierarchy of HI_{i_new}, T_R represents an extended analysis.

Example: Users complete the previous analysis (T_{R1}); now they only want to analyse the number of American students by year. They apply a hierarchy rotation defined as follows:

HRotate(T_{R1}, UNIVERSITY, {USGEO}) = T_{R2}

The result of this operation is illustrated by Figure 11. Note that an exclusion constraint is defined between the hierarchies named FRGEO and USGEO. The resulting multi-dimensional table is a new analysis integrating only measures related to American students.

Users want to extend the analysis by integrating French students and American students in the same multi-dimensional table. Users can define several expressions.

- The first solution consists in choosing a hierarchy, which groups dimension instances belonging to USGEO and dimension instances belonging to FRGEO.

HRotate(T_{R2}, UNIVERSITY, {ZN}) = T_{R3}

Figure 11. Multi-dimensional table T_{R2}

TEACHING (NbStudent)		**UNIVERSITY | USGEO**	
		Continent	AMERICA
DATE | HY	**Year**		
	2001		2000000
	2002		2050000
	2003		2200000
	2004		2300000
PROFESSOR.All			
COURSE.All			

Figure 12. Multi-dimensional table T_{R3}

TEACHING (NbStudent)		**UNIVERSITY | ZN**		
		Continent	EUROPE	AMERICA
DATE | HY	**Year**			
	2001		600000	2000000
	2002		620000	2050000
	2003		610000	2200000
	2004		610500	2300000
PROFESSOR.All				
COURSE.All				

- The second solution consists in specifying the two dimensions. Note that they must share the same parameter of higher granularity (Continent).

HRotate(T_{R2}, UNIVERSITY, {FRGEO, USGEO}) = T_{R3}'

Rotation of Dimensions

Presentation: The rotation between two dimensions consists of changing the current dimension in a multi-dimensional table. By default, the analysis is based on the parameter of higher granularity along the new dimension.

Definition: A rotation of dimensions is defined as DRotate(T, D_1, D_2, HI_2) = T_R.

- D_1 is the dimension of T which is replaced,
- D_2 is the new dimension of T_R,
- HI_2 is a set of hierarchies; these hierarchies must share the same parameter of higher granularity.

Remarks: The inter-dimension constraint specification implies the following treatments:

- If an exclusion constraint (and/or partition constraint) is defined between a hierarchy of HI_i and a hierarchy of HI_{i_new}, T_R represents a new analysis.
- If an inclusion constraint is defined from a hierarchy of HI_i to a hierarchy of HI_{i_new}, T_R represents an extended analysis.

Example: Decision makers use the multi-dimensional table called T_{R2}. These users want to modify the analysis.

- First, they analyse the number of students by course areas and years. The resulting table displays a new distribution of American students according to years and course areas. Users apply the following algebraic operation:

DRotate(T_{R2}, UNIVERSITY, COURSE, {USCRS}) = T_{R4}

- Second, they analyse the number of students by course degrees and years. Note that this parameter belongs to a hierarchy (FRCRS), which is an exclusion between the initial hierarchy (FRCRS $\otimes$USGEO). The resulting table represents a new analysis corresponding to a new distribution of European students according to years and course degrees. Users apply the following algebraic operation:

DRotate(T_{R2}, UNIVERSITY, COURSE, {FRCRS}) = T_{R5}

Rotation of Facts

Presentation: The rotation between two facts (also called drill across) consists of changing the current fact in a multi-dimensional table. The two facts must at least share the displayed dimensions.

Definition: A rotation of fact is defined as FRotate(T, F_x, {$f(m_1)$, $f'(m_2)$,…}) = T_R.

Figure 13. Multidimensional table T_{R4}

TEACHING (NbStudent)		**COURSE | USCRS**		
		Area	Computer Science	Economy
DATE | HY	**Year**			
	2001		120000	80000
	2002		125000	80000
	2003		110000	110000
	2004		120000	110000
PROFESSOR.All				
UNIVERSITY.All				

Figure 14 Multi-dimensional table T_{R5}

TEACHING (NbStudent)		**COURSE | FRCRS**		
		Degree	Licence	Master
DATE | HY	**Year**			
	2001		40000	20000
	2002		42000	20000
	2003		40000	21000
	2004		40500	20000
PROFESSOR.All				
UNIVERSITY.All				

- F_x is the new fact analysed with dimensions previously defined in the table T, and
- $\{f(m_1), f'(m_2), \ldots\}$ is a set of measures, which are displayed according to an aggregate function.

Example: Decisionmakers use the previous table T_{R5}, and they change the subject of analysis.

This operation is valid if and only if displayed dimensions in the initial table are shared between the initial fact and the new displayed fact. Users apply the following algebraic operation:

FRotate(T_{R5}, REGISTRATION, {Sum(Amount)}) = T_{R6}

Drill Down

Presentation: The drill down consists in "disaggregating" data by moving down a hierarchy. This operation adds detailed parameters along a hierarchy

Figure 15. Multi-dimensional table T_{R6}

REGISTRATION (Amount)		**COURSE | FRCRS**		
		Degree	Licence	Master
DATE | HY	**Year**			
	2001		2000000	1800000
	2002		2100000	1800000
	2003		2200000	1820000
	2004		2300000	1840000
STUDENT.All				

in a multi-dimensional table. This operation allows users to analyse measures with more detailed parameters.

Definition: A drill down is defined as DrillDown(T, D_i [, HI_i], p_{i_x}) = T_R.

- D_i is the dimension along which the drill down operation is applied,
- HI_i is a set of hierarchies, defined by user. These hierarchies share all the parameters belonging to Att_i of T and p_{i_x}, and
- p_{i_x} is the new parameter added in the list of displayed attributes in T_R.

Remarks:

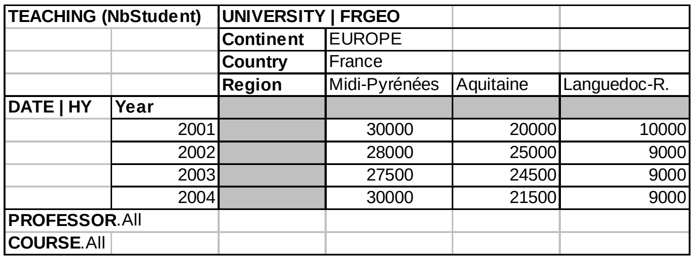
- When HI_i is specified by users, the operation is valid if the displayed parameters are shared by all the hierarchies of HI_i.
- When HI_i is not defined, the initial set of hierarchies in T is not modified. The operation is valid if all the hierarchies share the displayed parameters.

Example: Decisionmakers want to display more detailed information from the table called T_{R1}. The new analysis displays the number of French students by continent, by country, by region, and by year.

DrillDown(DrillDown(T_{R1}, UNIVERSITY, {FRGEO}, Country), UNIVERSITY, {FRGEO}, Region) = T_{R7}

Note that the algebra we present allows the combination of several operators to express complex queries (closed algebra).

Figure 16. Multi-dimensional table T_{R7}



TEACHING (NbStudent)		UNIVERSITY | FRGEO			
		Continent	EUROPE		
		Country	France		
		Region	Midi-Pyrénées	Aquitaine	Languedoc-R.
DATE | HY	**Year**				
	2001		30000	20000	10000
	2002		28000	25000	9000
	2003		27500	24500	9000
	2004		30000	21500	9000
PROFESSOR.All					
COURSE.All					

Roll Up

Presentation: The roll up consists of aggregating data by moving up along a hierarchy. This operation deletes detailed parameters along a dimension in a multi-dimensional table.

Definition: A roll up is defined as RollUp(T, D_i [, HI_i], p_{i_x}) = T_R.

- D_i is the dimension along which the roll up operation is applied,
- HI_i is a set of hierarchies, defined by user. These hierarchies share all the parameters belonging to Att_i of T, which represent higher granularities than p_{i_x}, and
- p_{i_x} is the parameter representing the lowest granularity in the list of displayed attributes in T_R.

Remarks:

- When HI_i is specified by users, the operation is valid if the displayed parameters are shared by all the hierarchies of HI_i.
- When HI_i is not defined, the initial set of hierarchies in T is not modified. The operation is valid if all the hierarchies share the displayed parameters.

Example: Decisionmakers modify the previous analysis (T_{R7}). They analyse the number of students by continent, by country, and by year. So, they apply the following roll-up operation:

Figure 17. Multi-dimensional table T_{R8}

TEACHING (NbStudent)		**UNIVERSITY | FRGEO**	
		Continent	EUROPE
		Country	France
DATE | HY	**Year**		
	2001		600000
	2002		620000
	2003		610000
	2004		610500
PROFESSOR.All			
COURSE.All			

RollUp(T_{R7}, UNIVERSITY , {FRGEO}, Country) = T_{R8}

Nest

Presentation: The nest operation consists of changing the hierarchical order of two parameters.

Definition: A nest is defined as Nest(T, D_i, p_{i_1}, p_{j_2}) = T_R.

- D_i is the dimension along which the nest operation is applied, and
- p_{i_1} and p_{j_2} are the permuted parameters.

Example: Decisionmakers extend the previous analysis (T_{R8}). They add quarters along the temporal dimension, but they want to nest years into quarters.

Nest(DrillDown(T_{R8}, DATE, {HY}, Quarter), DATE, Year, Quarter) = T_{R9}

Implementations

In order to validate our propositions, we have implemented a prototype, called Graphic OLAPSQL. This tool is developed using Java 1.4, JavaCC and Java JDBC over Oracle9i Application Server.

Figure 18 Multi-dimensional table T_{R9}

TEACHING (NbStudent)			**UNIVERSITY | FRGEO**	
			Continent	EUROPE
			Country	France
DATE | HY	**Quarter**	**Year**		
	Q1	2001		20000
		2002		22000
		2003		21000
		2004		21000
	Q2	2001		10000
		2002		10000
		2003		15000
		2004		20000
	Q3	2001		20000
		2002		20000
		2003		15000
		2004		10000
	Q4	2001		10000
		2002		10000
		2003		10000
		2004		10500
PROFESSOR.All				
COURSE.All				

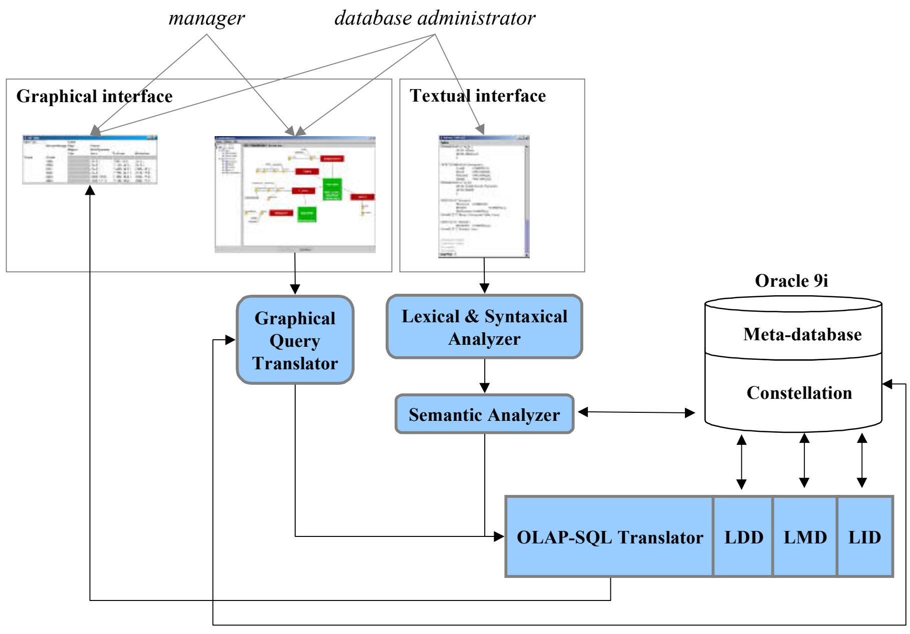
Graphic-OLAPSQL Tool

Figure 19 describes the Graphic-OLAPSQL architecture.

This tool is composed of several components:

- The **textual interface** supports a textual language called OLAP-SQL. This language allows decisional administrators and computer science users to manipulate multi-dimensional databases. This language is based on the SQL syntax using multi-dimensional concepts.
- The **graphical interface** provides a representation of the database as a graph of nodes and links. Expressing a query consists in selecting nodes and performing operations. The query result (textual and/or graphical queries) is composed of two parts: a dimensional table representing data analysed, and a new set of nodes visualised with specific colors. The user may also go on his query by selecting new nodes or performing new operations from nodes.
- The **graphical query translator** translates each graphical query into an algebraic query. This algebraic query is represented by a list of operators related to the algebra that we presented in the third section.

Figure 19. Graphic-OLAPSQL architecture



- The analyser components (lexical, syntactical, semantic) validate each textual query according to the OLAP-SQL syntax. The textual query is also translated into the algebraic format.
- The **OLAP-SQL Translator** converts each algebraic query into a set of SQL queries. This set of queries is applied on an R-OLAP database, which stores the constellation elements through relational tables and meta-tables.

The multi-dimensional query algebra that we presented is used as a pivot format for two kinds of languages; for example, the graphic-OLAPSQL tool supports textual and graphical languages.

Textual Language

OLAP-SQL is an SQL-like language integrating multi-dimensional concepts. We argue that SQL syntax is very simple for administrators. OLAP-SQL allows:

- The definition (create, drop, alter) of facts, dimensions, hierarchies, and constraints,
- The manipulation (insert, delete, update) of multi-dimensional data, and
- The querying of data using extended select orders.

The Appendix describes the main OLAP-SQL orders. The following example focuses on the SELECT order.

Example: For example, users can display the first multi-dimensional table T_{R1} by expressing the next OLAP-SQL order.

```
select SUM(NbStudents)
according to rows Year of DATE with HY,
             columns Continent of UNIVERSITY with FRGEO
from TEACHING;
```

Note that the Graphic-OLAPSQL textual order is translated into the following algebraic expression. Figure 10 illustrates the resulting table:

Display(SchoolConstellation, TEACHING, {*SUM*(NbStudents) }, DATE, HY, UNIVERSITY, FRGEO) = T_{R1}

From the algebraic expression, the OLAP-SQL translator sends a set of queries to the R-OLAP, which stores the constellation. OLAP-SQL facilitates multi-dimensional manipulations. It is complete regarding to our algebra; all algebraic operation can be defined using OLAP-SQL.

Example: The rotation of hierarchies, corresponding to the algebraic expression, HRotate(T_{R1}, UNIVERSITY, {USGEO}) = T_{R2}, is written as follows:

```
select SUM(NbStudents)
according to rows Year of DATE with HY,
             columns Continent of UNIVERSITY with USGEO
from TEACHING;
```

Figure 11 illustrates the resulting table.

Graphical Query Language

We provide a graphical query language dedicated to managers. This language is based on a graphical multi-dimensional view of constellations as shown in Figure 21, and decision- makers can build their queries directly using graphical facts, dimensions, and hierarchies.

Decisionmakers can express their queries using the graph, which describes the multi-dimensional database schema. The operations, which may be triggered from the nodes, permit a user to display subjects of analysis according to one row dimension and one column dimension. The operations performed from graph nodes are related to the type of the nodes:

- If the node is a fact, possible operations are rotation of fact, send query (display), and selection of measures,
- If the node is a dimension, possible operations are selection of hierarchy, drill-down, roll-up, rotation of dimensions, rotation of hierarchies, and nest.

Example: We consider the previous example. To display the first multi-dimensional table T_{R1} (Figure 10), users express the query as the following:

As textual queries (OLAP-SQL queries), the Graphic-OLAPSQL tool translates each graphical operation into its corresponding algebraic operation. From this internal format, the OLAP-SQL translator triggered a set of SQL queries to the R-OLAP constellation.

Example: Figure 22 illustrates the graphical operation using for displaying T_{R2}.

Future Trends

We intend to extend these works through two ways. First, existing models must be extended in order to support complex and time-variant decisional data.

Figure 20. Graphical multi-dimensional view

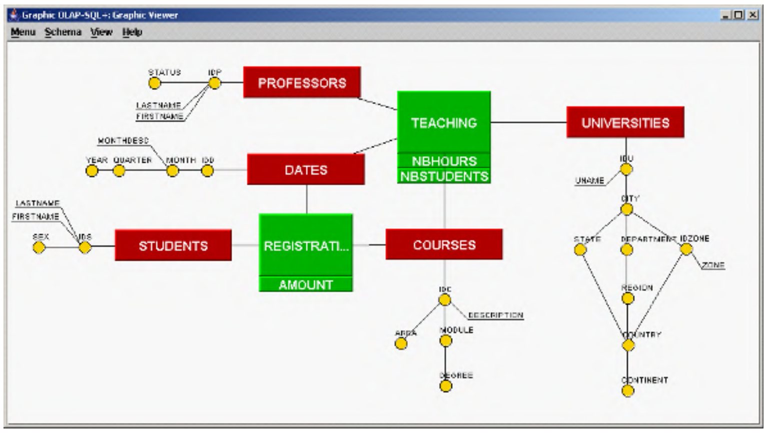
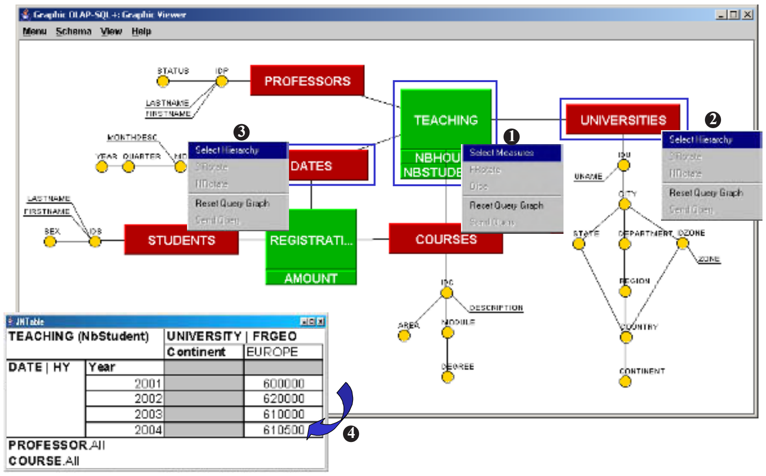


Figure 21. Example of graphical query



TEACHING (NbStudent)		UNIVERSITY | FRGEO	
		Continent	EUROPE
DATE | HY	Year		
	2001		600000
	2002		620000
	2003		610000
	2004		610500

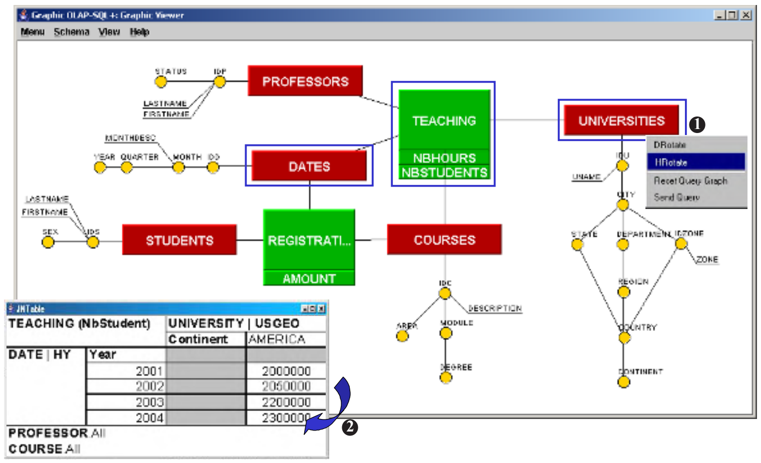
PROFESSOR.All
COURSE.All

Second, the multi-dimensional database designing is a complex task, which must be based on a whole design method.

Existing models support only factual data whereas more complex information may be integrated.

- Internet is a significant source of information and knowledge. To take advantage of the Web content in a decision support framework, multi-dimensional models must be able to integrate semi-structured data such as

Figure 22. Example of graphical HRotate operation



TEACHING (NbStudent)		UNIVERSITY | USGEO	
		Continent	AMERICA
DATE | HY	Year		
	2001		2000000
	2002		2050000
	2003		2200000
	2004		2300000

PROFESSOR All
COURSE All

XML documents. Some researches provide systems allowing the storage and interrogations of semi-structured and/or structured documents in database management systems (Abiteboul, Segoufin, & Vianu, 2001; Faulstish, Spiliopoulou, & Linnemann, 1997; Gardarin, Mensch, & Tomasic, 2002). These systems, called data Web, are mainly based on data mining techniques, but they do not support a multi-dimensional approach; few works allow multi-dimensional manipulations of documents by their structures and their contents (Khrouf & Soulé-Dupuy, 2004). These works may be completed by supporting constraints.

- Source data and decision maker needs are time-variant. To support inefficiently data evolutions, multi-dimensional models must integrate temporal aspects as temporal databases and/or versions (Body, Miquel, Bédard, Tchounikine, 2003; Mendelzon & Vaisman, 2000) . Nevertheless, these approaches do not integrate constraint specifications as well as constraint evolutions. Another future work consists of defining evolutions of the extraction process as regard to data evolutions; for example, when a fact is modified, its related refresh process must be updated.

The multi-dimensional database designing is a complex task but only a few design methods are provided in the multi-dimensional modelling field. In the scientific literature, we distinguish three categories: (1) top-down methods, which define multi-dimensional schema from decision maker needs (Kimball & Ross, 2002), (2) bottom-up methods, which use the data sources to define decisional needs and to design the multi-dimensional schema (List, Bruckner, Machaczek, & Sciefer, 2002; Moody & Kortink, 2000) and (3) mixed methods based on the data sources to define the decisional schema and integrating needs of decision makers (Luján-Mora, Trujillo, & Vassiliadis, 2004). The design methods are always based on a logical or conceptual multi-dimensional model. None of these models integrate semantic constraints between multi-dimensional data even when these constraints allow the definition of consistent multi-dimensional schemata by eliminating inconsistencies and the improvement of analysis by offering reliable information to decision-makers.

Concluding Remarks

This chapter presented a constraint-based multi-dimensional model. This model describes multi-dimensional data as a constellation of facts and dimensions, which are organised through hierarchies. Each hierarchy is associated to a sub-set of dimension instances; the model integrates heterogeneous data in one dimension through this multiple instantiation property. In order to insure data consistency and reliable data manipulation, the model integrates several kinds of semantic constraints; the intra-dimension constraints are defined between hierarchies of one dimension, whereas the inter-dimension constraints are related to hierarchies of two dimensions.

A tool integrating textual and visual query languages, called Graphic-OLAPSQL, is described. The tool is based upon the constraint-based model allowing the description of constellations.

- The interface represents the data stored in an R-OLAP database as a graph of nodes and links; the nodes correspond to multi-dimensional concepts while the links associate these concepts. The graphical language is dedicated to casual users because it provides a global view of the constellation, an incremental querying process, and a contextual help system by displaying operations, which are consistent in regard to selected elements.
- The textual language, called OLAPSQL, is dedicated to administrators because it provides an SQL-like syntax integrating multi-dimensional concepts; administrators can define, manipulate, and query constellations without relational considerations.

Our query algebra formalises these languages; each graphical query operation or textual query order is translated into an algebraic operator. This well-formalised algebra integrates the main multi-dimensional operations, and it is a closed algebra (to express complex queries).

References

Abello, A., Samos, J., & Saltor, F. (2003, November). *Implementing operations to navigate semantic star schemas.* In The 6th International Workshop on Data Warehousing and OLAP (DOLAP'03), New Orleans, LA (pp. 56-62). ACM.

Abiteboul, S., Segoufin, L., & Vianu, V. (2001). *Representing and querying XML with incomplete information*. In PODS 2001.

Agrawal, R., Gupta, A., & Sarawagi, S. (1997, April). *Modeling multi-dimensional databases.* In The 13th International Conference on Data Engineering (ICDE'97), Birmingham, UK (pp. 232-243).

Baralis, E., Paraboschi, S., & Teniente, E. (1997, August 25-29). Materialized view selection in a multi-dimensional database. In *The 23rd International Conference on Very Large Databases* (VLDB'97), Greece (pp. 156-165).

Body, M., Miquel, M., Bédard, Y., & Tchounikine, A. (2003, November 8). *A multi-dimensional and multi-version structure for OLAP applications.* ACM Fifth International Workshop on Data Warehousing and OLAP (DOLAP 2002), McLean, USA.

Cabibbo, L., & Torlone, R. (1997). Querying multi-dimensional databases. In *Proceedings of the 6th DBPL Workshop* (pp. 253-269).

Cabbibo, L., & Torlone, R. (1998, March). A logical approach to multi-dimensional databases. In *The 6th International Conference on Extending Database Technology (EDBT'98)*, *Lecture Notes in Computer Science 1377,* Valencia, Spain (pp. 183-197). Springer.

Codd, E. F. (1993). *Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate.* Technical Report, E. F. Codd and Associates.

Faulstich, L. C., Spiliopoulou, M., & Linnemann, V. Wind (1997). A warehouse for Internet data. In *Advances in Databases, Proceedings of*

British National Conference on Databases (BNCOD'97) 15, # 1271 in LNCS (pp. 169-183). Springer.

Gardarin, G., Mensch, A., & Tomasic, A. (2002, March 24-28). *An introduction to the e-XML data integration suite.* In The 8th Conference on Extending Database Technology, Prague.

Golfarelli, M., Maio, D., & Rizzi, S. (1998). *Conceptual design of data warehouses from E/R schemes*. In The 31st Hawaii International Conference on System Sciences.

Gray, J., Bosworth, A., Layman, A., & Pirahesh, H. (1996, March). *Data cube: A relational aggregation operator generalizing group-by, cross-tabs, and subtotals.* In The 12th International Conference on Data Engineering (ICDE'96), New Orleans, LA (pp. 152-159).

Gyssen, M., & Lakshmanan, L. V. S. (1997, August). *A foundation for multi-dimensional databases.* In The 23rd International Conference on Very Large Data Bases (VLDB'97), Athens, Greece (pp. 106-115).

Hurtado, C., & Mendelzon, A. (2002, June). *OLAP dimension constraints.* In the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'02), Madison (pp. 169-179).

Kimball, R. (1996). *The data warehouse toolkit*. New York: John Wiley & Sons.

Kimball, R., & Ross, M. (2002). *The data warehouse toolkit* (2nd ed.). New York: John Wiley & Sons.

Krouf, K., & Soulé-Dupuy C., (2004). A textual warehouse approach: A Web data repository. In M. Mohammadian (Ed.), *Intelligent agents for data mining and information retrieval* (pp. 101-124). Hershey, PA: Idea Group Publishing.

Lehner, W. (1998, March). *Modeling large scale OLAP scenarios*. In The 6th International Conference on Extending Database Technology (EDBT'98), Valencia, Spain (pp. 153-167).

Li, C., & Wang, X. S. (1996, November). *A data model for supporting on-line analytical processing*. In The 5th International Conference on Information and Knowledge Management (CIKM'96), Rockville, MD (pp. 81-88).

List, B., Bruckner, R., Machaczek, K., & Sciefer, J. (2002). *A comparison of data warehouse development methodologies case study of the process* warehouse. In The 13th International Conference on Database and Expert Systems Applications (DEXA'02), Aix-en-Provence, France, LNCS 2453 (pp. 203-215).

Luján-Mora, S., Trujillo, J., & Vassiliadis, P. (2004). *Advantages of UML for multi-dimensional modeling.* In 6th International Conference on Enterprise Information Systems.

Mendelzon, A., & Vaisman, A. (2000, September). *Temporal queries in OLAP.* In 26th International Conference on Very Large Data Bases (VLDB 2000), Cairo, Egypt (pp. 242-253).

Moody, D. L., & Kortink, M. A. R. (2000, June). *From enterprise models to dimensional models: A methodology for data warehouse and data mart design.* In The 2nd International Workshop on Design and Management of Data Warehouses, Stockholm, Sweden.

Pedersen, T. B., & Jensen, C. S. (1999, March). *Multi-dimensional data modeling for complex data.* In International Conference on Data Engineering (ICDE'99), Sydney, Australia (pp. 363-365).

Rafanelli, M. (2003). *Multidimensional databases: Problems and solutions*. Hershey, PA: Idea Group Publishing.

Vassiliadis, P., & Sellis, T. (1999, December). A survey on logical models for OLAP databases. *SIGMOD Record, 28*(4), 64-69.

Appendix: OLAP-SQL

DDL

The Data Definition Language consists of creating facts and dimensions of a constellation. The hierarchies are defined through the dimension creation. The language also permits the deletion of the elements of a constellation. Due to space limitation, we do not detail update orders.

Create orders

```
create fact < fact_name >
 ( <measure1> <format1> [<option1>],
   < measure 2> <format2> [<option2>], ...)
[ connect to <dimension1>, <dimension2>, ... ];

create dimension <dimension_name>
 ( <parameter1> <format1> [<option1>],
   <parameter2> <format2> [<option2>],...)
 [ as (<select_query>) ]
 [ with hierarchy <hierarchy_name1>
 (level <parameter1> [(<weak_attribute1_1>, < weak_attribute1_2>,...)],
  level <parameter2> [(<weak_attribute2_1>, < weak_attribute2_2>,...)],...),
  with hierarchy < hierarchy_name 2> … ] ;
```

Drop orders

```
drop dimension < dimension_name>;

drop fact <fact_name> [cascade];
```

DML

The Data Manipulation Language consists in inserting, deleting, updating, or querying multi-dimensional databases.

Insert orders

insert into dimension < dimension_name >
[**(**<parameter1>**,** <parameter2>**,...)**]
values { **(**<value1>**,** <value2>**,**…**)** **|** **(**<select_query>**)** } **;**

insert into fact <fact_name>
[**(**<measure1>**,** <measure2>**)**]
values (<value1>**,** < value2>**,**…**)**
according to ref(<dimension_name 1>**,** <predicate1>**),**
ref(<dimension_name 2>**,** <predicate2>**),**…**;**

Update orders

update dimension < dimension_name >
set <attribute1> **=** <expression1>**,**
[**set** <attribute2> **=** <expression2>**,**…]
[**where** <predicate>]**;**

update dimension < dimension_name >
set (<attribute1>**,** <attribute2>**,**…**) = (**<select_query>**)**
[**where** < predicate >]**;**

Delete orders

delete from dimension < dimension_name > **where** < predicate >**;**

delete from fact < dimension_name > **where** < predicate >**;**

Select order

select <aggregate_function>**(**<measure1>**)**, < aggregate_function >**(**<measure2>**)**,
        …
[**according to** [**rows** <parameter1_row>, <parameter2_row>,…,
                        of < dimension_name _row > **with** <hierarchy_name_ ow >,]
                          [**columns** <parameter1_col>, <parameter2_col>,…,]
                        of < dimension_name_col> ***with*** <*hierarchie*_name_col>]]
from <fact_name>
[**where** <predicate>]
[**order by** <parameter1> **values(**<value1_1>**,** <value1_2>**,**…**)**
        […[,<parameter2> **values(**<value2_1>**,** <value2_2>**,**…**)],**…]]**;**

About the Authors

Zongmin Ma (Z. M. Ma) received his PhD from the City University of Hong Kong in 2001, and is currently a full professor at 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 more than 40 papers in international journals, conferences, edited books, and encyclopedias in these areas since 1998. He has also edited and authored several scholarly books published by Idea Group Publishing and Springer-Verlag, respectively.

***

Indranil Bose is associate professor of information systems at the School of Business, University of Hong Kong. His degrees include BTech (electrical engineering) from Indian Institute of Technology, MS (electrical and computer engineering) from University of Iowa, MS (industrial engineering) and PhD (management information systems) from Purdue University. He has research interests in telecommunications — design and policy issues, data mining and artificial intelligence, electronic commerce, applied operations research, and supply chain management. His teaching interests are in telecommunications, database management, systems analysis and design, and data mining. His publications have appeared in *Communications of the ACM*, *Computers and Operations Research*, *Decision Support Systems and Electronic Commerce*, *Ergonomics*, *European Journal of Operational Research*, *Information and Management*, *Operations Research Letters*, and in the proceedings of numerous international and national conferences.

Stefan Brecheisen is a teaching and research assistant in Professor Hans-Peter Kriegel's group, University of Munich, Germany. He received his diploma in computer science at the University of Munich in 2003. The diploma thesis is entitled "Using Sets of Feature Vectors for Similarity Search on Voxelized CAD Objects." Currently, he works in the field of similarity search in high-dimensional objects, spatial objects, and complex objects.

T. W. Carnduff is a principal lecturer in the School of Computing at the University of Glamorgan, UK. He was awarded an MSc in computer science in 1986, and a PhD in computer science in 1993, both from the University of Wales. His research interests include concurrent engineering, the use of object-oriented databases in engineering design, and aspect-oriented databases.

Lam Albert Kar Chun is an undergraduate student of School of Business in the University of Hong Kong, majoring in information systems and software engineering. Throughout his university years, he has been actively involved in various course projects in the fields of database management, information systems and project management, analysis and design of software systems, and enterprise resources planning systems. Furthermore, he has obtained a higher diploma degree in information systems.

J. S. Goonetillake graduated from University of Colombo, Sri Lanka, in 1995 in computer science. In 1998, she obtained her master's degree in data engineering from Keele University, UK. Then she continued her studies at University of Wales Institute, Cardiff, UK, for a PhD. Goonetillake obtained her PhD from the University of Wales in 2004 for her research in the management of evolving engineering design constraints. Currently she is working as a senior lecturer at School of Computing, University of Colombo, Sri Lanka. Her research interests are in the areas of object-oriented database systems, engineering design databases, and distributed database systems.

Wong Oi Ling Helen is a final-year undergraduate student at the School of Business in the University of Hong Kong. She is studying business administration with a major in information systems. She completed several database projects during her secondary school years. For her undergraduate study, she has been involved in projects in management information systems, database development and management, advanced database management and data mining, enterprise resources planning systems, and information systems development and project management.

Li Hoi Wan Ines is an undergraduate student of School of Business in the University of Hong Kong. She majors in information systems and software engineering. She has been actively involved in several projects: system integration project, database project, and analysis and design of software projects.

Hans-Peter Kriegel is a full professor at the University of Munich, Germany, and head of the database group since 1991. He studied computer science at the University of Karlsruhe, Germany, and finished his doctoral thesis there in 1976. He has more than 200 publications in international journals and reviewed conference proceedings. His research interests are database systems for complex objects (molecular biology, medical science, multimedia, CAD, etc.), in particular, query processing, similarity search, high-dimensional index structures, as well as knowledge discovery in databases and data mining.

Peer Kröger has been working as a research assistant in the group of Professor Hans-Peter Kriegel at the University of Munich, Germany, since October 2001. In July 2004, he achieved his PhD with a thesis on "Coping with New Challenges for Density-Based Clustering." His research interests include knowledge discovery in large standard, spatial, and multimedia databases, data mining for molecular biology data analysis, and similarity search in spatial databases.

Chang-Tsun Li completed his PhD under Professor Roland Wilson's supervision the University of Warwick, UK, in 1998. He is currently a lecturer in the

Department of Computer Science at the University of Warwick. He has published some 40 refereed book chapters and papers in the areas of image processing, computer vision, and multimedia security. He was the organizer of the Special Session on Image and Volume Segmentation, the 5th IASTED International Conference on Visualisation, Imaging, and Image Processing, 2005. He is also a member of the Technical Committee on Image Processing of The IASTED for the term 2004-2007, and has been serving on the International Program Committee of the IASTED International Conference on Internet & Multimedia Systems and Applications (IMSA), and Computer Graphics and Imaging (CGIM) since 2004.

Elvira Locuratolo graduated in mathematics at the University of Naples. Since 1982, she has been a researcher at Institute of Information Science and Technologies (ISTI) of the Italian National Research Council in Pisa, Italy. Her research interests include quality modeling, database design methodologies, formal methods, and re-engineering methods. She is the author of numerous articles in books and journals. She has collaborated with the Rutherford Appleton Laboratory at Oxford and other foreign institutions. She also carries out teaching activity in mathematics and computer science at the University of Pisa.

Sriram Mohan is a PhD candidate in computer science specializing in the area of databases at Indiana University, USA. His research interests primarily center around XML with an emphasis on conceptual modeling, security, and access control. Sriram is currently involved in the development of AXCESS — a framework for implementing and enforcing access control for XML. He has also been involved in the development of XML database engines and XML benchmarking.

Yannis Panagis was born in 1978. Yannis is a PhD candidate at the Computer Engineering and Informatics Department, University of Patras, Greece, and a member of the Research Unit 5 of the Computer Technology Institute. Yannis holds an MSc from the same department since 2004, where he also completed his undergraduate studies in 2001. His interests span the areas of data structures, string processing algorithms, Web engineering, and Web reorganization. He is the author of papers in international journals and conferences in these areas.

Martin Pfeifle is a teaching and research assistant in Professor Hans-Peter Kriegel's group, University of Munich, Germany. He finished his doctoral thesis on "spatial database support for virtual engineering" in the spring of 2004. His research interests include data mining on moving objects, distributed and parallel data mining, spatio-temporal query processing, and geographic information systems.

Marco Pötke is head of the PLM Solutions Group at sd&m AG Germany. His research interests include PLM services, part-data management and virtual engineering solutions, in particular, shape-based search engines and online methods for digital mockup. In this context, the field of relational indexing plays a vital role to leverage spatial technology for commercial usage. In 1998 he received his diploma, and in 2001, he received his PhD, from the University of Munich, Germany.

Franck Ravat is an assistant professor of computer science at the Université Toulouse I, France. He is a research staff member at the IRIT research center (CNRS UMR 5505) in the SIG/ED research group. His research interests include multi-dimensional database design, OLAP manipulations, and distributed databases.

Evangelos Sakkopoulos is a PhD candidate at the Computer Engineering and Informatics Department, University of Patras, Greece, and a member of the Research Unit 5 of the Computer Technology Institute. He has received the MSc with honors and the diploma of computer engineering and informatics at the same institution. His research interests include Web usage mining, Web engineering, Web-based education, Web Services, Web searching, and intranets. He has more than 20 publications in international journals and conferences in these areas.

Matthias Schubert is currently a teaching and research assistant at the database group at the University of Munich, Germany. He received his diploma in computer science in 2000 and achieved the doctoral degree in 2004 at the University of Munich. His thesis is entitled "Advanced Data Mining Techniques for Compound Objects." He is currently continuing his research on complex object representations for non-standard database systems and data mining.

Thomas Seidl is a full professor for computer science and head of the data management and data exploration group at RWTH Aachen University, Germany, since 2002. His research interests include data mining and data management in multimedia databases, spatial databases, computational biology, medical imaging, mechanical engineering, computer graphics, and so forth. One focus is on effective and efficient similarity retrieval based on content, shape, or structure of complex objects in large databases. His research in the field of relational indexing aims at exploiting the robustness and high performance of relational database systems for complex indexing tasks. Having received his MS in 1992 from the Technische Universität München, Seidl received his PhD in 1997, and his venia legendi in 2001, from the University of Munich, Germany.

Arijit Sengupta has a PhD in computer science from Indiana University, USA. He served as the director of Educational Development in Computer Science at Indiana University, and an assistant professor in Computer Information Systems at Georgia State University and at the Kelley School of Business before joining the Raj Soin College of Business at Wright State University, USA. His main area of research is databases for complex structured documents. He has authored and published many papers in this area. Dr. Sengupta is also the developer of DocBase, a system for document querying, as well as query languages DSQL (Document SQL) for querying XML documents using SQL, and QBT (Query By Templates) for querying XML documents using a graphical form.

Spyros Sioutas obtained his diploma, MSc, and PhD from the Department of Computer Engineering and Informatics of the University of Patras, Greece. He currently is a research associate at the same department. His current research interests include fundamental data structures, algorithms and complexity, spatio-temporal and multimedia databases, computational geometry, peer-to-peer networks, Web Service discovery, knowledge management and advanced information systems. He has published 40 papers in international conferences and journals.

Olivier Teste is an assistant professor of computer science at Paul Sabatier University, France. He is a research staff member at the IRIT research center (CNRS UMR 5505) in the SIG/ED research group. His research interests include multi-dimensional databases, database modeling, query language design, and temporal databases.

Athanasios Tsakalidis was born in 1950, obtained his diploma in mathematics from the University of Thessaloniki, Greece, in 1973, his diploma in computer science in 1981, and his PhD in 1983, from the University of Saarland, Saarbuecken, Germany. He is currently the R&D coordinator and vice-director of the Computer Technology Institute (RACTI, Patras, Greece) and a full professor and chairman in the Department of Computer Engineering and Informatics, University of Patras, Greece. His research interests include data structures, graph algorithms, computational geometry, expert systems, GIS, medical informatics, databases, multimedia, information retrieval, and bio-informatics. He has been one of the co-authors to one of the most significant books of computer science *Handbook of Theoretical Computer Science* (published by Elsevier Science Publishers and MIT-Press).

Chia-Hung Wei is currently studying for a PhD in the Department of Computer Science at the University of Warwick, UK. He obtained a master's from the University of Sheffield, UK, in 2000, and a bachelor's from Tunghai University, Taiwan, in 1996. His research interests include content-based image retrieval and image processing.

Roland Wilson was appointed to a senior lectureship at Warwick University, UK, in 1985, and was promoted to a readership in 1992 and a professorship in 1999, respectively. He has published some 100 papers, the bulk of which have been in the area of image processing and coding. He acted as an associate editor for *IEEE Transactions on Image Processing*, with responsibility for image coding, for the *Journal of Pattern Recognition*, and as a reviewer for several journals. He has served on the E4 Professional Group Committee of the IEE. A major theme in his research over many years has been the development of signal modeling and analysis using multi-resolution methods, ranging from grey-level pyramids to wavelet transforms.

Xun W Xu received a BSc and MSc from Shenyang Jianzhu University and Dalian University of Technology, P. R. China, in 1982 and 1988, respectively. In 1996, he received a PhD from the Department of Mechanical Engineering, University of Manchester Institute of Science and Technology (UMIST), UK. He is now a senior lecturer in the Department of Mechanical Engineering, the University of Auckland, New Zealand. Dr. Xu is a member of ASME and

IPENZ. He heads the Manufacturing Systems Laboratory and the CAD/CAM Laboratory in the University of Auckland. His main interests lie in the areas of CAD/CAPP/CAM.

Jun Yuan received his PhD in computer science from Southeast University, China, in 1995. He joined M&CT, Phantom Works, the Boeing Company, USA, in 2001, and is currently an advanced computing technologist. Prior to that, he has held faculty positions in Florida International University, Hong Kong University of Science and Technology, and Southeast University in China. His research interests are in areas of database systems and information management. His recent work has been focusing on semantic information integration and distributed query processing. He is a senior member of IEEE and IEEE Computer Society.

Leung Vivien Wai Yue is an undergraduate student of School of Business in the University of Hong Kong. She is majoring in information systems and software engineering. She has been involved in various course projects in the fields of database management, information systems and project management, analysis and design of software systems, and enterprise resources planning systems.

Gilles Zurfluh is a professor of computer science at the Université Toulouse I, France. He is a research staff member at the IRIT research center (CNRS UMR 5505), where he manages the SIG/ED research group. His research interests span many aspects of non-traditional data management in databases, information systems, and decisional systems.

Index