AN ORGANIZATION MODEL TO SUPPORT DYNAMIC COOPERATIVE WORK AND WORKFLOW

by

Edward C. Cheng

A Thesis Submitted in Fulfilment of the Requirements for the Degree of

Doctor of Philosophy

in the

University of London

March, 2004

Department of Computer Science

Birkbeck College

ABSTRACT

The rapid growth of computer networks, along with the wide acceptance of the Internet and Intranet (Cole 2003; Whinston et al. 2001), has pushed distributed computing into a new era. Software developers who used to focus solely on atomic and isolated computational models are now expanding their scope to cover group-computing solutions as well. Numerous research efforts in collaborative computing and process management have resulted in workflow and other automation techniques. These attempts emphasize mainly theories regarding process modelling and concurrent activity coordination. However, applications of these technologies in a large production environment have faced major setbacks due to the disconnection between the conceptual process definition and the actual complex operations of the real operating environment. The principal problem is that organizational information is rarely, if ever, comprehensively modelled, let alone integrated, into the process management system. Indeed, without a flexible and extensible organization model to describe the global resource information, it is impossible for a rapidly changing company to apply an enterprise-wide workflow system or groupware solution to handle global assignment, authorization, authentication, notification, and role migration.

In this thesis, we define the issues in organization modelling and review the existing systems that are used to address these issues. We then propose a formal organization model, namely OMM (Organization Modelling and Management), which uses a hybrid approach (a cross between ER and OO) to capture all organizational resources available to the enterprise. With the OMM approach, thousands of dynamically changing complex inter-relationships between resource objects are abstracted into a handful of business rules. Under the proposed model, an enabling service is defined to support dynamic role resolution, task assignment, process authorization and role-based access control. OMM not only enables large enterprises to take command of their various resources, but also allows employees, business partners, vendors and even customers to truly collaborate and interact with one another over an expanding and self-maintained communication network.

A prototype of OMM has been developed and tested against a real operating environment, namely Hitachi America, which has a total of 6,000 employees distributed in several cities of the United States. The flexibility of OMM is further demonstrated by

modifying the said prototype to obtain a new prototype, called OMM/PS (Publish-and-Subscribe) to support PS in an e-commerce environment. This research prototype is used to enable an Internet-based commercial insurance company, called InsurePoint, to conduct interactive e-commerce on the Web.

In Remembrance of My Parents Chung-Yin and Tak-Ying Ho Cheng

Acknowledgements

I am very thankful to my supervisor Professor George Loizou, who has encouraged me from the first day we met to complete my study and research at Birkbeck College. His consistent guidance and suggestions have been key to the success of this work. My thanks to Mrs. Diane Loizou for using various means to facilitate the remote communications between Professor Loizou and myself.

My thanks to Professor Dieter Gawlick and Professor Hector Garcia-Molina who have encouraged me to go through the PhD programme at Birkbeck College. Their numerous discussions with me on the topic of e-commerce, publish-and-subscribe and workflow have given me tremendous insight. I am grateful to Drs. K. L. Mannock and J. Crampton who tirelessly read part of this manuscript, especially Chapters 4, 5 and 6, and provided me with invaluable and much-needed suggestions.

Thanks to my wife Gem, my daughter Faith and my sons Shem and Jeshua for their patience, understanding and prayers. Without their support I would not be able to complete this work.

I also wish to thank my colleagues at OCT Research Laboratory, for their contribution in refining the ideas and application of this work. My thanks are due to Thomas To, Brutus Lo, Eddie Lo and Johnny Lo.

I gratefully acknowledge the financial support from OCT Research Laboratory and Eguanxi, Inc. Thanks be to God for giving me creativity.

TABLE OF CONTENTS

TABLE OF CONTENTS ..... **v**
LIST OF FIGURES ..... **ix**
LIST OF TABLES ..... **xii**
LIST OF ABBREVIATIONS ..... **xiv**

CHAPTER 1 INTRODUCTION **15**

1.1 A Dynamic and Collaborative Computing Environment 16
1.2 Motivation and Aims 17
1.3 Definition of Terms 20
1.4 Structure of the Thesis 22

CHAPTER 2 BACKGROUND **24**

2.1 Introduction 24
2.2 Organization Modelling and Reengineering 25
2.2.1 Organization Analysis 26
2.2.2 Organization Conceptual Design 27
2.2.3 Organization Design Implementation 30
2.3 Organization Modelling Principles 31
2.4 Goals of Organization Modelling 33
2.5 Scope of Organization Modelling 35
2.6 Success Criteria in Organization Reengineering 36
2.7 Conclusion 38

CHAPTER 3 A CRITICAL ASSESSMENT OF ORGANIZATION MODELLING APPROACHES IN EXISTING WORKFLOW MANAGEMENT SYSTEMS **40**

3.1 Introduction 40
3.2 Workflow Management Systems 41
3.3 Running Example: A Sample Workflow Process 43
3.4 Previous Work in Workflow Organization Modelling 45
3.4.1 ARIS 46
3.4.2 CIMOSA 47
3.4.3 EMS 49
3.4.4 M*OBJECT 50
3.4.5 Objectflow 51
3.4.6 SAM* 53
3.5 Directory Service Based Organization Model 55
3.5.1 X.500 and LDAP 56
3.5.2 Novell NDS 59

3.5.3 Microsoft Active Directory 60
3.6 STANDALONE ORGANIZATION MODELLING SYSTEMS 61
3.6.1 ORM 61
3.6.2 OVAL 61
3.6.3 Other Organizational and Office Systems 63
3.7 WEAKNESSES OF EXISTING OM SYSTEMS 63
3.8 CONCLUSION 65

CHAPTER 4 OMM: A HYBRID MODEL FOR ORGANIZATION MODELLING 66

4.1 INTRODUCTION 66
4.2 AN ENTERPRISE AND ITS RESOURCES 67
4.3 ORGANIZATIONS 70
4.3.1 OMM Organization Partitioning 71
4.3.2 Relationships between Organizations 74
4.4 MEMBERS AND THE INFORMATION MODEL 75
4.4.1 Object-Orientation of OMM 75
4.4.2 State Transition of OMM Members 80
4.4.3 Transfer of Member Objects between OMM Organizations 81
4.5 VIRTUAL LINKS AND THE RELATIONSHIP MODEL 82
4.6 AN EXAMPLE 84
4.7 SUMMARIZED FEATURES OF OMM AND OTHER OM SYSTEMS 87
4.8 OMM VERSUS OTHER OM SYSTEMS 90
4.9 CONCLUSION 92

CHAPTER 5 THE POLICY-BASED RELATIONSHIP MODEL IN OMM 94

5.1 INTRODUCTION 94
5.2 VIRTUAL LINK DEFINITION 97
5.3 APPLYING VIRTUAL LINKS TO MODEL DYNAMIC ROLES 98
5.4 APPLYING VIRTUAL LINKS TO MODEL DYNAMIC RELATIONSHIPS 101
5.5 BI-DIRECTIONAL RELATIONSHIPS 102
5.6 TRANSITIVITY OF VIRTUAL LINKS 103
5.7 PUBLISH-AND-SUBSCRIBE 104
5.8 AN OMM/P&S PROTOTYPE 104
5.9 CONCLUSION 107

CHAPTER 6 ROLE RESOLUTION IN WORKFLOW MANAGEMENT SYSTEMS AND OTHER COOPERATIVE APPLICATIONS 109

6.1 INTRODUCTION 109
6.2 ROLE RESOLUTION IN WORKFLOW MANAGEMENT SYSTEMS 110
6.2.1 Task Assignment 111
6.2.2 Task Authorization 112
6.2.3 Routing Decision 114

6.3 ROLE RESOLUTION WITH OMM 115
6.4 ROLE-BASED ACCESS CONTROL 117
6.5 CONCLUSION 119
APPENDIX 6A. JAVA CODE SEGMENT FOR ROUTING CONTROL BY USING DYNAMIC ROLES 120

CHAPTER 7 ORGANIZATION MODELLING AND REENGINEERING BY USING OMM 121

7.1 INTRODUCTION 121
7.2 BUSINESS PROCESS REENGINEERING AND ORGANIZATION REENGINEERING 122
7.3 ORGANIZATION MODELLING APPROACHES AND OMM 123
7.3.1 Information System Approach 123
7.3.2 Object-Oriented Approach 124
7.3.3 Petri Nets Approach 125
7.4 COMMON TECHNIQUES IN ORGANIZATION REENGINEERING 126
7.4.1 OR Modularization 126
7.4.2 OR Decentralization 128
7.4.3 Bottom-Up Analysis in Organization Reengineering 129
7.4.4 Top-Down Analysis in Organization Reengineering 130
7.5 CONCLUSION 132

CHAPTER 8 CONCURRENCY CONTROL 133

8.1 INTRODUCTION 133
8.2 BACKGROUND 134
8.2.1 Database Systems 134
8.2.2 Transactions 135
8.2.3 Commit and Abort 136
8.3 CONCURRENT ACCESS TO SHARED OMM OBJECTS 138
8.4 DEADLOCK DETECTION 140
8.5 DEADLOCK AVOIDANCE AND RESOLUTION 142
8.6 CRASH AND RECOVERY 143
8.7 CONCLUSION 145

CHAPTER 9 IMPLEMENTATION EXPERIMENTS AND APPLICATIONS OF THE OMM PROTOTYPE SYSTEM 147

9.1 THE OMM PROTOTYPE SYSTEM ARCHITECTURE 148
9.2 USER INTERFACE FOR DEFINING AND MANIPULATING ORGANIZATIONAL OBJECTS 150
9.2.1 Application Programming Interface 150
9.2.2 Graphical User Interface 151
9.3 DOMAIN UUID AND NAMING CONVENTION 153
9.4 INFORMATION EXCHANGE WITH EXISTING DATABASES 154
9.4.1 Generic Database Schema for Mapping Existing Databases to OMM 154
9.4.2 Mapping Agent Applications 155
9.5 MAPPING THE OMM DATABASE SCHEMA TO OTHER RELATIONAL DATABASE SCHEMAS 156

9.6 MAPPING OMM OBJECTS TO X.500 DIRECTORY OBJECTS 157
9.6.1 The X.500 Directory Model 158
9.6.2 Using OMM Organizations to Model DIT 160
9.6.3 Using OMM Objects to Implement X.500 Objects 161
9.6.4 Using X.500 to Implement OMM Conceptual Entities 163
9.6.5 Other Considerations 165
9.7 APPLICATION OF THE OMM PROTOTYPE SYSTEM IN INDUSTRY 166
9.7.1 An OMM Prototype System to Support Enterprise Modelling in Hitachi, America 166
9.7.2 OMM Organization Definitions 167
9.7.3 Organizational Relationship Modelling 169
9.7.4 Using Virtual Links to Support Workflow 169
9.8 CONCLUSION 170

CHAPTER 10 CONCLUDING REMARKS AND FURTHER RESEARCH 172

10.1 ORGANIZATION MODELLING PRINCIPLES 173
10.2 REVIEW OF AIMS AND ACCOMPLISHMENTS 175
10.2.1 Scalability 175
10.2.2 Extensibility 176
10.2.3 Flexibility 176
10.2.4 Performance 176
10.3 FURTHER WORK 177

APPENDIX A: JAVADOC LISTING OF THE OMM API 180

A. 1 OMSOBJECT 180
A. 2 OMSORGANIZATION 182
A. 3 OMSMEMBER 190
A. 4 OMSVIRTUALLINK 200

APPENDIX B: CLASS DIAGRAMS OF OMM CONCEPTUAL ENTITIES 204

REFERENCES 211

LIST OF FIGURES

Figure 2-1 The Organization Reengineering Cycle 25

Figure 2-2 Implementation Design Phase Overall Architecture 31

Figure 3-1 Different Components of a Workflow Management System 42

Figure 3-2 Electronic Parts Ordering Process 43

Figure 3-3 Company Resources Exist in a Hierarchy 44

Figure 3-4 ARIS Architecture 46

Figure 3-5 User's View of the X.500 Directory Service 57

Figure 3-6 Distributed Directory Service 57

Figure 3-7 A Sample Directory Information Tree 58

Figure 4-1 The ER Component of the OMM Model 69

Figure 4-2 The OO Component of the OMM Model 69

Figure 4-3 Diagram of the OMM Model 70

Figure 4-4 Horizontal and Vertical Partitioning of Enterprise Resources in OMM 72

Figure 4-5 Relationships between OMM Organizations 74

Figure 4-6 OMM Conceptual Entities Correspond to OO Concepts 76

Figure 4-7 Hierarchy of the OMM Member 77

Figure 4-8 Supported Value Constraints in OMM 80

Figure 4-9 State Diagram of OMM Members 81

Figure 4-10 Digraph Showing Relationships Between Resources Using Virtual Links 84

Figure 4-11 An Enterprise Composed of Three OMM Organizations 85

Figure 4-12 Relationships Between OMM Organizations 85

Figure 4-13 Three Members Exist in the EMPLOYEE OMM Organization.....85

Figure 4-14 OMM Separates the Role and Organization Models 91

Figure 5-1 OMM Relationship Graph Showing Reverse Relationships.....103

Figure 5-2 The Supervisor Relationship Showing the Transitivity Property 103

Figure 5-3 OMM/P&S Organizations for InsurePoint 105

Figure 6-1 Life Cycle States and State Transitions of Workflow Objects 111

Figure 6-2 A Workflow Routing Decision Which Requires Role Resolution 114

Figure 7-1 The Business Process Reengineering Cycle 123

Figure 7-2 An Example of a Petri Net 125

Figure 7-3 An Example of Merging Two OMM Organizations.....127

Figure 7-4 Different Ways to Logically Connect OMM Organizations.....128

Figure 7-5 Bottom-Up Organization Reengineering Method.....130

Figure 7-6 Top-Down Organization Reengineering Method 131

Figure 8-1 A Database Program to Withdraw Money from a Bank Account 137

Figure 8-2 An Example of Conflict on the Object Level but not on Data Level.....139

Figure 8-3 An Example of Concurrent Accesses without Conflict on Object Level 139

Figure 9-1 The OMM Run-time System Architecture.....149

Figure 9-2 The OMM Prototype System Software Architecture.....149

Figure 9-3 A Relationship Graph Displayed in Organization Chart Format.....152

Figure 9-4 Pop-up Window Showing Member Attributes with the Web Interface.....152

Figure 9-5 An OMM Mapping Agent Application.....156

Figure 9-6 Mapping RDBMS Column Names to OMM Attributes 157

Figure 9-7 A Sample DIT 159

Figure 9-8 Sample Corporation Tree Modelled by OMM.....161

Figure 9-9 Partitioned OMM Organizations.....163

Figure 9-10 A Sample Corporation with Groups and Relationships Modelled by OMM 164

Figure 9-11 OMM Organizations Defined for Hitachi America 167

Figure 9-12 Vacation Request Process 170

Figure B-1 Class Diagram of OMM Classes in the Client Package.....206

Figure B-2 Class Diagram of OMM Classes in the Common Package.....207

Figure B-3 Class Diagram of OMM Classes in the Db Package.....208

Figure B-4 Class Diagram of OMM Classes in the DbRel Package 209

Figure B-5 Class Diagram of OMM Classes in the Server Package 210

LIST OF TABLES

Table 4-1 Sample Attributes for OMM Organizations 71

Table 4-2 Syntax of OMM Member Attribute 79

Table 4-3 Supported Data Types in OMM 79

Table 4-4 Examples of Organizational Resources Going Through Life Cycle 81

Table 4-5 Attribute Definition of the DIVISION OMM Organization 86

Table 4-6 Attribute Definition of the DEPARTMENT OMM Organization 86

Table 4-7 Attribute Definition of the EMPLOYEE OMM Organization 86

Table 4-8 Organization Modelling Features of OMM 87

Table 4-9 Organization Modelling Features of WFMS 89

Table 4-10 Organization Modelling Features of Directory Services Systems 89

Table 4-11 Organization Modelling Features of Stand-alone OM Systems 90

Table 5-1 The Syntax of an OMM Virtual Link 97

Table 5-2 Attribute Definitions of InsurePoint Organizations 106

Table 6-1 Operations Available on Workflow Objects 113

Table 6-2 Pseudo-code Program Segment for Routing Decision of *Process Order* 115

Table 6-3 Syntax of Control Statement in Workflow Step Definition 116

Table 6-4 Expanding Roles to Cover Relationships 116

Table 9-1 OMM Application Programming Interface Categorized by Class 151

Table 9-2 The Database Schema Used in OMM to Store an OMM Organization 155

Table 9-3 The Rules for Mapping X.500 Objects to OMM Objects 161

Table 9-4 Examples for Applying Rules to Map X.500 to OMM 162

Table 9-5 The Rules for Mapping OMM Conceptual Entities to X.500 Objects 165

Table 9-6 Estimated Number of Resource Objects in Hitachi America 167

Table 9-7 Attribute Definition of the COMPANY Organization 168

Table 9-8 Attribute Definition of the REGION Organization 168

Table 9-9 Attribute Definition of the DIVISION Organization 168

Table 9-10 Attribute Definition of the DEPARTMENT Organization 168

Table 9-11 Attribute Definition of the EMPLOYEE Organization 168

Table 9-12 OMM Virtual Links Representing Relationships in Hitachi America 169

Table 10-1 Elapsed Time of the OMM API Categorized by Object and Operation Type 177

LIST OF ABBREVIATIONS

ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programming Interface
BP	Business Process
BPM	Business Process Management
BPR	Business Process Reengineering
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CIM	Computer Integrated Manufacturing
DAP	Directory Access Protocol
DBMS	Database Management System(s)
DIB	Directory Information Base
DIT	Directory Information Tree
DS	Directory Service
DSA	Directory Service Agent
DSP	Directory System Protocol
DUA	Directory User Agent
ER	Entity-Relationship
ERP	Enterprise Resource Planning
GUI	Graphical User Interface
HR	Human Resources
ID	Identifier
JDBC	Java Database Connection
JSP	Java Server Page
LDAP	Light Weight Directory Access Protocol
NDS	Novell Directory Services
OLTP	Online Transaction Processing
OO	Object-Oriented
OM	Organization Modelling
OMM	Organization Modelling and Management
OMS	OMM Services
OR	Organization Reengineering
OSI	Open System Interconnection
OU	Organizational Unit
P&S	Publish-and-Subscribe
R&D	Research and Development
RBAC	Role-based Access Control
RDBMS	Relational Database Management System(s)
RDN	Relative Distinguishing Name
RM	Resource Manager
ROI	Return On Investment
SADT	Structured Analysis and Design Technique
UML	Unified Modelling Language
UUID	Universal Unique Identifier
WFMS	Workflow Management System(s)

CHAPTER 1 Introduction

About 619 million people now access the Internet worldwide and access is growing at the rate of 50 per cent per year. The majority of United States households, approximately 60%, had personal Internet access in 2002 (Cole 2003). The trend is the same in Europe and other parts of the world. According to Global Reach's Internet usage research, in 2002, 224 million people used the Internet daily in European languages excluding English, and 179 million people in Asian languages (www.glreach.com/globstats/, 2003). The Internet/Web explosion has swept across our world in the past few years and revolutionized our thinking and practice in business as well as our life-style. Electronic commerce and Internet transactions have caught the attention of most major international corporations. According to the global Internet Economy Indicators (www.InternetIndicators.com, 2003), the revenue of the Internet economy was more than $800 billion in 2001. Internet economy revenue is growing faster than any other traditional economy. For example, Internet economy companies' Internet-related revenue grew $68 billion from the first quarter of 1999 to the first quarter of 2000 - an astounding 64 percent. That compares to a $23 billion growth in non-Internet revenue during the same period — a growth rate of 3.6 percent. Even as the overall economy experiences fluctuations, Internet economy forces continue to reshape the economy in unprecedented ways, producing savings for businesses and consumers alike. This revenue growth pattern would make it larger than the car ($728 billion) and life insurance ($724 billion) industries. Internet-related revenue growth was 15 times the growth rate of the US economy (Whinston et al. 2001). The Gartner Group estimated that the growth of electronic commerce would reach $7.3 trillion by the year 2004. The vast acceptance of and participation in the Internet create new opportunities in collaborative computing. They also give rise to new challenges in information and resource management. Companies have come to realize that they cannot afford not to have an e-commerce strategy. Unfortunately, e-commerce strategy has been focusing overwhelmingly on electronic catalog management and payment processing. There has been relatively little effort in bringing the entire enterprise onto the Internet to perform true e-business. This thesis is mainly concerned with modelling and organizing a large amount of enterprise resources on the Internet, and the management of their rapidly changing relationships in a dynamic environment. We will show that indeed it is

necessary to have a connected enterprise resource environment, in order for collaborative software to run over the entire enterprise. This chapter gives the motivation of this thesis and an outline of the main goals achieved.

Hereafter the terms enterprise, corporation, company and firm will be used interchangeably.

1.1 A Dynamic and Collaborative Computing Environment

Computer integrated manufacturing and office automation rely on process modelling and the management of business processes over a distributed and collaborative computing environment. Since business processes often involve different types of corporate resources, the underlying information system must maintain knowledge of and support access to the rapidly changing organizational structure and resource relationships to ensure compliance with company business policies and overall system consistency and efficiency.

The rapid growth of the Internet and e-commerce has pushed collaborative computing to a new era. The walls of a corporation no longer bound access to business processes; early adopters of technologies are looking into ways to run inter-enterprise business processes. Resources from business partners, vendors and customers may all take part in collaborative computing to complete a business goal. This is reflected by the growth of the Extranet, which is a form of private Internet that is authorized to be used only by a number of partner companies forming that private network.

Overall, an enterprise level collaborative computing environment is characterized by the following properties:

- *Having a large amount of enterprise resources*. These are not just human resources; an enterprise is dealing with many different types of resources, some tangible and some intangible. Examples of enterprise resources include employees, products, services, vendors, business partners, facilities, equipment, R&D projects, financial plans, sales regions and offices.

- *Complex and rapidly changing relationships between various enterprise resources*. Assuming every person in a corporation is dealing with 5 other resources (for

example, reporting hierarchy, project responsibility, HR, administrative functions and machine utilization), a 5,000 person company will have over 25,000 relationships between various resources. To record these relationships and keep them up to date can be an overwhelming administrative task. Since knowledge of these relationships is not centralized, it is almost impossible for a single administrative function to maintain and manage this information.

- *Automating the enterprise business processes requires knowledge of resources and the inter-relationships between them.* To streamline operations and to tune up productivity, companies need to reengineer the business processes operating among the various resources. To allow business processes to be automated in the enterprise or across enterprises, the enterprise must be a well-organized entity, with its resources and relationships properly defined.

These properties of enterprise-wide collaborative computing bring us to our motivation and aims of this thesis.

1.2 Motivation and Aims

Why do we need to perform organization modelling? Is there a need for an enterprise to engage in the cycle of organization analysis and organization reengineering? Can an enterprise afford not to do it? In this section, we take a look at today's business challenges in the business world, and how organization modelling is critical to an enterprise's growth.

The business world is constantly changing. Today, our world is moving from an economy of scope to an economy of scale under a global economy for mass customization. For many companies around the world, staying in business means:

- meeting customer requirements,
- reducing the time-to-market of their products, and
- manufacturing products or providing services at a lower cost, in a shorter time, and with increased quality.

This is also commonly known as the "*good, fast* and *cheap*" principle. It is relatively easy to produce low quality, high cost products with long delays. It is much more difficult to produce high quality, low cost products in a short time because these are conflicting requirements calling for a high degree of automation, sometimes high precision, and in most cases excellence in management. The introduction of ERP solutions brings the world's attention to process automation and BPR. Many believe that the answer behind an ideal corporation lies in a continuous enterprise-wide BPR environment (Howard 1991). However, the experience of deploying a BPR solution shows that, up until now, BPR has had its success mainly as a limited departmental solution. When we attempt to automate business processes on an enterprise level, it turns out to be virtually impossible because conventional BPR lacks a clear model to describe the underlying enterprise, its resources and the inter-relationships between them. As a result, true enterprise-level organization modelling and coordination of workflow between and across the various resources within an enterprise is deemed impossible.

The trend of Internet growth and global economy create new opportunities but at the same time also create additional challenges for corporations. These challenges include:

- *Globalization of Resources:* within the global economy, resources such as people, facilities, projects, service centres and manufacturing sites are distributed all over the world. It is difficult to coordinate the activities involving all these distributed resources. Sometimes one branch of the company may not even know that certain valuable resources exist in another part of the company.

- *Mobility of Resources:* people within a global enterprise often travel from one site to another. This creates problems especially for the management of personnel who have multiple responsibilities covering a wide geographic region, such as the entire Asia Pacific or a large country like China; it is often difficult to keep track of where such persons are located at any moment. Moreover, for large enterprises, it is very common for projects or departments to move from one location to another. This creates a communication breakdown between different groups of people.

- *Accessibility vs. Security:* once information is available on the Internet, it is virtually accessible by anyone, anywhere, anytime. Current security models used in Internet applications are either password-based or key-based (Yialelis and Sloman 1996).

These security methods, although relatively straightforward to implement, do not have the flexibility to allow large numbers of selected users to access different types and levels of information based on their roles (Cheng 1999a). As a result, it is difficult to increase accessibility and at the same time enforce security.

- *Information Overload and Information Undermine:* the Internet is a wide-open cyberspace which allows any user to post information and to search for information. Between net casting and "spam" distribution of electronic mail, users on the Internet either find themselves overloaded by a huge amount of information, or they are desperate to find the information that they know is out there, but cannot find, buried as it is in the gigantic network.

- *Customer Satisfaction:* As the market expands, requirements coming from different customers (many times from different countries with a different background and culture) for the same product can be slightly different. This requires that products be customized to individual specifications and often requires that they must be delivered according to multiple imposed due dates. To accomplish this, the whole company must be well connected and coordinated to achieve common goals.

- *Parallelization:* Due to time-to-market competition, companies must coordinate the simultaneous execution of cooperative tasks to reduce time of design, engineering, manufacturing, and delivery. Coordination and collaboration between departments are critical for a company's overall success.

- *Flexibility:* Today's markets change rapidly and constantly. Companies must be quick to adapt in reaction to their changing environment. More importantly, the decision to change, which often comes down from upper management must be filtered down to all levels of employees in a very short time.

- *Virtual Enterprise:* Outsourcing of non-core activities to business partners and third party vendors is a common business practice today. As a result, an enterprise is more than just the core company: the virtual enterprise truly includes its business partners and vendors as well. The better this virtual enterprise communicates and coordinates, the stronger it is in the competitive market.

- *Total Quality Management:* Product and service quality is everyone's business, from product design to manufacturing, shipment, and customer services. To achieve total

quality management, the company must be well connected, from requirement gathering, product design, manufacturing, testing, packaging and shipping, to customer services.

The work in this thesis solves these issues by having a flexible, generic, object-oriented model to help organize different resource objects of the enterprise, bringing them together through a rule-based relationship model, and finally allowing users to securely access and maintain the information with a fine-grained security model. We have implemented a run-time Prototype System according to this organization model. Collaborative software such as workflow management systems can interact with this Prototype System by calling its open API. As we apply this organization model to the modelling of an enterprise, and to the support of the underlying collaborative software with the associated organization modelling service, the enterprise can become more connected, flexible and dynamic.

1.3 Definition of Terms

In this section, we will define some keywords and terms that are used throughout the thesis.

- Organization model

  A generic organization model is a reference model which can be applied to describe the different aspects of an organization. A specific organization model is the output of applying the generic model to a particular view of an organization.

- Organization modelling

  Organization modelling is a practice companies adopt to refine their structure aiming at improving productivity. In general, organization modelling has three phases; first is the organization analysis phase, second is the conceptual design phase, and third is the design implementation phase (Vernadat 1996). These three phases involve applying some selected reference models (Bussler 1994; Berio et al. 1995; Di Leva et al. 1997; Kosanke et al. 1997) to define the resources that make up the organization and their inter-dependent relationships. Organization resources include various types of entities such as human resources, products, services, machines, robots, projects,

assets and facilities, business processes and others. In addition, the design implementation phase concerns also the implementation of the organization design in an information system, allowing definition and manipulation of organizational resource information. Note that business process modelling which involves the definition of workflow and data-flow for each business process is outside the scope of organization modelling. However, organization modelling is a pre-requisite to business process modelling.

- Organization reengineering

  As business conditions change, a company has to adjust itself for survival and to gain a competitive edge. This cycle of analysis and adjustment that organizations periodically need to undertake is termed organization reengineering. Organization reengineering is therefore a cycle of applying the practice of organization modelling.

- Process model

  A process model, also known as a flow model, describes how a business process is routed through different agents. Agents in this context are resources within the company responsible for carrying out certain tasks which make up the business processes. An agent can be a human being, a machine or a software program.

- Role model

  A role model is a generic reference model used to describe how roles are defined and how different resources are assigned to take on one or more roles within the company. Traditionally, roles are defined simply as static labels; the actual semantic meaning and business policies behind the roles are often hidden as unstructured knowledge embedded in human minds.

- Role resolution

  Role resolution refers to the mechanism of identifying the resources that are playing a certain role at a certain time. Role resolution is critical to a workflow management system in that the workflow engine needs to resolve the role in order to forward an active work task to the right agent, i.e. a person.

1.4 Structure of the Thesis

This chapter gives the motivation of our work. It also provides the context of our research area by describing the kind of computing environment we face today. The challenges and requirements of this distributed and collaborative world of computing are also discussed.

Chapter 2 provides the background of the thesis. It begins by defining organization reengineering and organization modelling, and the relationships between the two. It describes the different phases of organization modelling and how they support a dynamic enterprise able to adapt quickly to change. The principles, goals, and scopes of organization modelling are discussed. It also provides a set of criteria to measure the success of organization reengineering.

Chapter 3 presents a critical survey of existing organization modelling systems. It describes some prior research and industry efforts in organization modelling in the context of workflow and collaborative computing. It presents the tightly coupled approach in which the workflow management system and the organization modelling component are architected together; the loosely coupled approach of a directory-based organizational system; and finally the stand-alone approach of some organizational modelling components. The chapter concludes by pointing out the weaknesses in these approaches.

Chapter 4 provides a detailed description of our organization model, called OMM. The chapter starts with a detailed description and definition of an enterprise and its resources, and then presents the information aspects and relationship aspects of OMM. A comparison between OMM and existing organization modelling systems is included at the end of this chapter. The comparison is based on four salient characteristics of such systems.

Chapter 5 provides a formal definition of the relationship model in OMM. It also includes a number of examples to exhibit the dynamic and flexible nature of our approach to model both relationships and roles.

Chapter 6 discusses the role-resolution function of OMM as applied to workflow and collaborative software. The concept of role resolution, task assignment and

authorization, and workflow routing control will be visited. By using some examples that are developed through the thesis, we will point out how our approach can solve some of the difficult problems in enterprise-wide role resolution that do not yet have a satisfactory solution. The concept and application of role-based access control using OMM will also be discussed in this chapter.

Chapter 7 focuses on using OMM to perform organization modelling. We describe the various techniques used in organization modelling and how OMM can be used effectively to support each of these techniques.

The OMM model is not limited to performing analytical modelling — it also allows for continuous user maintenance and updating of enterprise information. We therefore have to concern ourselves with features of concurrent access and related issues in accessing the OMM (organizational) objects. Chapter 8 discusses our approach to handling concurrent access to enterprise resources. It also deals with the concept of deadlock and our approach to preventing and resolving this issue.

Chapter 9 details our implementation experience in building the OMM Prototype System. It lays out the software architecture of this Prototype System and describes the interfaces to OMM. Software developers can interact with the OMM Prototype System through an openly defined application programming interface. As a result, WFMS can easily call upon OMM to perform role resolution. In addition, we have also built a simple graphical administrative tool to allow manipulation of OMM objects easily over the Internet. Finally, we provide a detailed discussion of the mapping of OMM to RDBMS and X.500 directories.

We conclude in Chapter 10 by reviewing our work and evaluating how we have achieved the goals we set out in Chapter 1. Finally, at the end of this chapter we provide some future research directions stemming from our work.

CHAPTER 2 Background

Organization modelling and reengineering have existed for decades, especially for the support of CIM. The principles of organization modelling as well as the different approaches to the modelling practice have been established over a long period of time.

In this chapter we describe the current methodological concepts and the different phases involved in organization modelling. Section 2.2 describes the concepts and models behind organization modelling and reengineering. In Section 2.3, we present the principles of organization modelling. In Section 2.4, we define the goals of organization modelling. As organization modelling can be an enormous project, it is necessary to understand the scope of organization modelling. We describe the scope of organization modelling in Section 2.5. Finally, in Section 2.6, we outline the criteria of success in the effort of organization reengineering. This chapter mainly concerns the background and principles behind organization modelling. The practice and the common techniques of organization modelling will be discussed in Chapter 7 in the context of OMM.

2.1 Introduction

Organization modelling is not new. Throughout history people have been using various means and tools to represent different organizational resources and structures. However, up until now, organization modelling had always been considered an administrative overhead in large corporations. Every year companies publish hard copies of employee directories and organizational charts. This static information has several problems. Firstly, companies are changing constantly; the information is often outdated even before it gets propagated to the people for whom it is intended. Secondly, the picture of the organizational structure that gets published is usually inflexible and represents only the very highest level; these published structures do not represent the true operational relationships of the enterprise at the level where most actions actually take place. Indeed, this rigid and high-level structural picture of the company sometimes even inhibits the creative thinking of the executives in their strategic planning. Moreover, such published organizational information is often focused on a single dimension of the enterprise, namely human resources. It only captures employee information or at most some portion of the enterprise's functional groups. As a result, no one has a complete

picture of all the resources within the enterprise and how these resources interact with one another.

As people within a company are networked together, the need for a well-defined reference model to capture the different resources and their inter-relationships becomes more significant. Such an organization model will become the foundational architecture for enterprise-wide cooperative software such as workflow applications and other groupware.

2.2 Organization Modelling and Reengineering

Organization Reengineering (OR) is a continuous analysis and refinement cycle in an enterprise. It is a cycle because business conditions are constantly changing; a company needs to continually refine its resources and organizational structure to maintain a competitive edge. The OR cycle has 3 stages; namely organization analysis, organization conceptual design, and organization design implementation. These three stages are also referred to as Organization Modelling (OM). OR is a cycle of performing OM repeatedly throughout the life of the enterprise. Figure 2-1 shows the OR cycle.

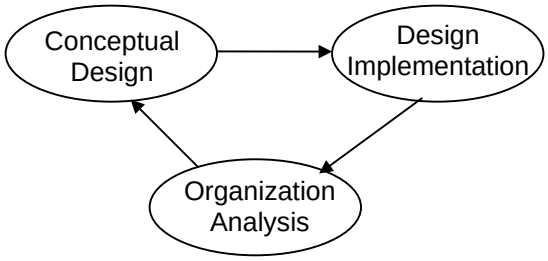


Figure 2-1 The Organization Reengineering Cycle

The process of organization modelling is composed of three phases. The first phase involves analysis of the current organization. The second phase involves applying certain reference models or methodologies to conceptually design a desired organization. The third phase is the implementation of that design by representing it in some organizational information system. We now discuss these three stages in more detail.

2.2.1 Organization Analysis

The organization analysis phase is aimed at supporting a fine structural analysis of the enterprise in terms of the analysis of the current status (AS-IS Analysis) and the analysis of the future potential status (TO-BE Analysis) (Vernadat 1996). The goal of the AS-IS Analysis is to provide managers and workers with an accurate model of the enterprise with which they can make a useful assessment of the current status of the enterprise. The aim of the TO-BE Analysis is to allow managers and decision-makers to assess the potential future status of the enterprise based on some assumptions. The analysis is usually carried out through a number of interviews with the various department managers. According to the survey of Vernadat (Vernadat 1996), most of the organization analysis can be accomplished by 1-3 one-hour long interview meetings with the department managers or decision-makers. The ultimate goal of organization analysis is to get a clear understanding of the functions and operations of the enterprise expressed in terms of a functional model and an information model.

2.2.1.1 Functional Model

The functional model is based on well-known software engineering practices using functional specification, such as SADT (Ross and Schoman 1977) and Petri Nets (Di Cesare et al. 1993; Murata 1989; Peterson 1993). The functional model provides a description of the enterprise in terms of a hierarchy of functions. A function is a set of activities which combine to accomplish a business goal in the global operation of an enterprise. In order to precisely specify a function, the organization modeller is required to identify what objects are input to or output from the function, what objects control or influence the function, and what is necessary to carry out the function. *The word "object" here is used in its broad sense and may represent every different type of entity* (e.g. data, people, documents, and materials) *in an organization as distinct from its meaning in the object-oriented paradigm*. The output objects of some functions may be used as input objects of other functions.

Enterprise functions are also referred to as business processes. A business process includes a number of tasks (activities), each of which is performed by one or more persons in the company. However, there may also be automated tasks, which are performed by software programs, robots, or electronic equipment. Workflow

management systems that we discuss in Chapter 3 are mainly concerned with the management and routing of a business process from task to task.

2.2.1.2 Information Model

The information model provides a description of the information circulating within the enterprise. It deals only with the analysis of information objects, which appear in the functional model and constitute the interfaces among the organization functions. The purpose of this analysis is to identify and classify information objects into the following classes:

1. *Data*: Data are information items relevant to the organization. These are pieces of information used by the functions within the organization environment.

2. *Messages*: A message is a piece of information concerning the fact that something has happened. According to this definition, a message is the organization concept related to the fact that an event has happened in the real world (e.g. an order has arrived or someone has requested product information by visiting a certain web page).

3. *Files*: A data file is a structured set of records composed of data items. Records represent real-world objects as manipulated by computers. A data file may or may not be managed by a DBMS. In the case of database files, further analysis will be performed to identify the database schema.

4. *Forms*: A form is a document that is used to collect, display, or communicate data within the organization environment. Forms may be in paper or electronic format.

5. *Agents*: An agent consumes and produces data, messages, files, or forms. An agent may be a human being, a software program, or a unit of workers.

2.2.2 Organization Conceptual Design

Once the modeller has finished the organization analysis phase, s/he is ready to move into the conceptual design phase. In the Conceptual Design phase, a set of *local views* and a *global view* of the enterprise are constructed. A *view,* either local or global, is the complete and formalized representation of the static (data) and dynamic (operational and functional) environment requirements. A view is created based on the data collected

from the previous analysis phase, and the projection, direction, and expectation formulated by the various inputs received from management. The description of a view contains: (a) a data schema, (b) a set of operational schemas, and (c) a functional net.

As a first step in the Conceptual Design phase, formatted and non-formatted requirements collected in the previous phase are analyzed. Static and dynamic requirements are expressed in terms of the functional model and information model introduced at the previous analysis stage (refer to Section 2.2.1 and its subsections). All relevant resources in the enterprise will be formally modelled in this phase. In addition, relationships and functional interactions between the various resources are also captured.

At the conceptual level, a complete formal description of both static and dynamic aspects of the organization must be produced. These descriptions are specified by means of two models: a data model and a process model. Using the Entity-Relationship (ER) approach (Levene and Loizou 1999) to capture the data model, we can also describe the relationships between the enterprise objects.

1. *Data Model*: Entity-Relationship models can be used to describe the information of the resource objects of interest for the organization. The ER data model provides the designer with four abstraction mechanisms: entity types, relationships, attributes, and abstraction hierarchies.

   Definition 2.1 (Entity) An *entity* is a "thing" that exists in the real world and can be uniquely identified.

   Definition 2.2 (Entity type) An *entity type* is a set of entities, which share common characteristics.

   Definition 2.3 (Relationship type) A (binary) *relationship type* among entity types is an association between two entity types.

   Definition 2.4 (Relationship) A *relationship r* is an instance of a relationship type R between entity types E_1 and E_2, where r is an association between two entities $e_1 \in E_1$ and $e_2 \in E_2$.

Definition 2.5 (Recursive relationship) A relationship type between two occurrences of the same entity type is called a recursive relationship type (or cyclic relationship type).

Note that an entity in an entity type may or may not participate in a relationship. In other words, the participation in a relationship of entities is optional (or partial).

Let r be a relationship over R and let e_1 and e_2 be entities belonging to instances of E_1 and E_2, respectively, such that e_1 and e_2 participate in r. Then we say that R is

- *many-to-one* if every entity e_1 as defined above is associated in r with at most one entity belonging to an instance of E_2.
- *one-to-many* if every entity e_2 as defined above is associated in r with at most one entity belonging to an instance of E_1.
- *one-to-one* if every entity e_1 as defined above is associated in r with at most one entity belonging to an instance of E_2, and correspondingly every entity e_2 as defined above is associated in r with at most one entity belonging to an instance of E_1.
- *many-to-many* if every entity e_1 as defined above is associated in r with zero or more entities belonging to an instance of E_2, and correspondingly every entity e_2 as defined above is associated in r with zero or more entities belonging to an instance of E_1.

Definition 2.6 (Attribute) *Attributes* (or attribute names) are properties of entity types.

Entities of the same entity type therefore share common attribute names although they can carry the same or different attribute values.

2. *Process Model*: Process modelling is used to represent organizational behaviour, that is, the possible sequences of operations of the organization functions within a given environment, and the communications among them. With workflow technology, organization behaviour is represented by means of business processes, which are in turn described by Petri Nets (Di Cesare et al. 1993) or similar network diagrams, such as Predicate Transition Net models (Belli and Dreyer 1994). With business

processes, a set of operations (or tasks) are linked together by execution conditions to form a network. The sequence of executions represents the progressive behaviour of the business process. Execution conditions are usually related to messages coming from the external world (e.g. a purchase order is submitted) and to the existence of some objects in a particular state (e.g. an unpaid account has been overdue for more than 90 days).

The data model and the process model are related in the sense that the objects consumed and produced by the business processes are described in the data model. In addition, the agents responsible for executing the tasks within a process are also captured in the data model. The work in this thesis focuses on proposing a data model to support the organization conceptual design phase. However, once a company starts to adopt our modelling methodology, it can perform organization analysis and refinement using the same approach.

Upon completion of the conceptual design of an organization, we are ready to implement this conceptual design into the enterprise. This moves us into the third and final phase of organization reengineering.

2.2.3 Organization Design Implementation

This phase in organization reengineering is concerned with implementing the design output from the second phase into the enterprise. This includes two areas: firstly, database implementation using a DBMS to reflect the design is necessary. Secondly, depending on the current status of the organization and the new architecture as outlined in the conceptual design, this phase may also require changing the actual organization. This may mean adding or removing resources from the company, as well as changing the way the resources relate to one another.

The actual work of database implementation to reflect the new enterprise is dependent on the kind of DBMS used (object-oriented, object-relational, relational, hierarchical, or network database system) and on the physical system itself. This is a typical problem for database designers and administrators. Our proposal simplifies the work of implementing the database representation of the enterprise; the conceptual model maps directly into a database design. We will discuss our experience in implementing an

organization design in Chapter 9. The organization design implementation phase is composed of two major tasks: Logical Design and Physical Design.

In the Logical Design task, the global data schema is transformed into a relational schema in which data structures are expressed according to the logical data model of the DBMS. In the same way, the global operation schema is converted into program fragments.

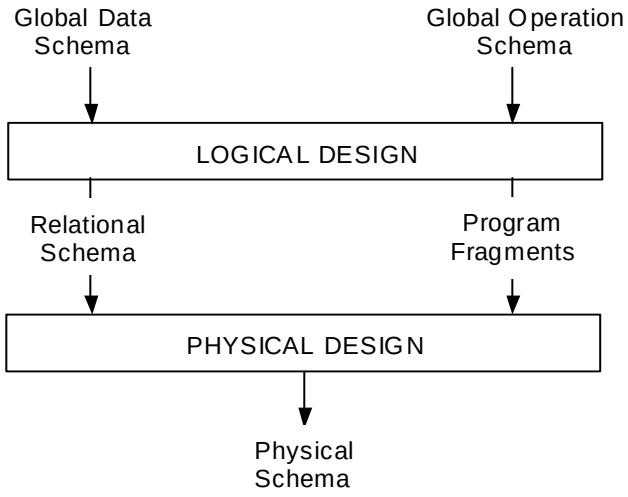


Figure 2-2 Implementation Design Phase Overall Architecture

The Physical Design task constructs a physical schema, which specifies the access paths. Criteria are given in the program fragments for choosing an "efficient" logical-physical representation among several alternatives that can be derived form the conceptual schema. The overall architecture of the Implementation Design phase is shown in Figure 2-2.

2.3 Organization Modelling Principles

Before we survey the current organization modelling techniques in Chapter 3, and propose our own approach to organization modelling in Chapter 4, let us review some basic principles applicable to organization modelling. According to Ross and Schoman (Ross and Schoman 1977), any modelling technique is characterized by the following four principles:

- *The purpose of the model:* Modelling must have a final goal in mind. This creates a direction and a framework for the process of modelling. We will discuss further the goals of organization modelling in Section 2.4.

- *The scope of the model:* This is also called the universe of discourse, defining the range or domain covered by the model. We will further discuss the scope of organization modelling in Section 2.5.

- *The viewpoint of the model:* We must define which aspects of the enterprise are covered by the model and which are left out and the viewpoint from which the modelling is drawn. With the organization model we propose in this thesis, we focus on modelling the organizational resources and the relationships between them from the perspective of management and decision-making. The functional, or business process aspect is left out in our model. We rely on the workflow model to describe the functional aspects of the enterprise.

- *The detailing level of the model:* This defines the level of precision or granularity of the model regarding the reality being modelled. Obviously, the degree of detail that is possible in a given context depends on the depth to which the analyst understands the reality of the organization. However, the underlying modelling technique used should be flexible enough to allow the analyst to model the organization to any desired level of detail. In Chapter 4, where we present our proposed organization model, it will be clear how our entity-relationship and object-oriented hybrid approach allows an analyst to model an enterprise to any level of granularity.

These essential principles outlined by Ross and Schoman (Ross and Schoman 1977) apply to organization modelling. Furthermore, Vernadat (Vernadat 1996) also suggests additional principles to be considered particularly for organization modelling:

- *Principle of modularity:* To facilitate management of change, models must be modular, i.e. be made up of an assembly of compatible building blocks so that the model can be built on a 'plug-and-play' basis. This is a way of dealing with enterprise model complexity, and it makes model maintenance much easier to realize than do other types of modelling architectures.

- *Principle of model genericity:* Many activities or components of an enterprise exhibit identical or similar properties even though they may support quite different

functionalities. It is therefore important to define standard building blocks as generic classes to factor common descriptive attributes and behaviours. These classes can then be adapted or specialized in the modelling of peculiar components or applications. Key concepts of the object-oriented paradigm such as classes, objects and inheritance provide the necessary underlying principles and guidance in this respect. This is another way of handling complexity in organization modelling.

- *Principle of reusability:* To reduce modelling efforts and increase model modularity, predefined building blocks or partial models must be reused and customized to specific needs as much as possible when modelling new parts of the system. This is referred to as ease of customization.

- *Principle of process and resource decoupling:* In order to preserve operational flexibility it is important to separately consider the actions that are being performed (the business processes) and the agents performing them (the resources). The mapping between the two is a scheduling problem particularly critical in manufacturing systems and project management.

As we will see in Chapter 3, very few existing organization modelling systems satisfy all these principles. However, to support flexible organization modelling in the OR cycle and to continually streamline an enterprise, it is important to observe these principles. It is our goal that our proposed methodology covers all these principles. In Chapter 10, we will review how our work fulfils these principles.

2.4 Goals of Organization Modelling

Any modelling method must have an understanding of its finality, i.e. the goal of the modeller. This finality usually has a direct influence on the modelling method definition.

In this thesis, we have adopted the firm position that an enterprise is made up of a large collection of resources that communicate, coordinate, and collaborate with one another to accomplish a set of concurrent business processes. Through the execution of these business processes, the functional resources contribute to the enterprise's business objectives. Organization modelling is essentially a matter of modelling and connecting these resources.

Business processes represent the flow of control of events happening in the enterprise. They are the means to accomplish the objectives of the enterprise. They materialize in management policies, flows of documents, operating procedures, manufacturing processes, administrative procedures, regulations, rules and the like. They are highly nested in order to represent the integrated activities of the company. Hence, organization modelling is often driven by business process modelling and management.

The goal of an organization modelling effort is not to model the entire enterprise in all of its details, although this might be theoretically possible at various levels of abstraction. The term "organization" here means a part of a company, the entity that needs to be represented; its size, scope and type are defined by the business users.

The primary goal of organization modelling is to support analysis and refinement of an enterprise. Another goal is to support modelling business processes that need to be automated through computer control. For instance, a person needs to be modelled if he or she is involved in an automated business process.

Broadly speaking, organization modelling aims to provide:

- a better understanding and uniform representation of enterprise resources;
- support for expansion and reduction in the size and structure of the enterprise; and
- a model used to support the control and monitoring of enterprise operations.

The main motivations for performing organization modelling are:

- managing system complexity;
- better management of all types of resources;
- capitalizing on enterprise knowledge and resources;
- support for business process reengineering; and
- enterprise integration (such as in the event of mergers-and-acquisitions).

2.5 Scope of Organization Modelling

Organization modelling is concerned with modelling the *what, how, when,* and *who* aspects of an enterprise. The "what" essentially refers to operations performed and the objects processed in the operations of the enterprise. The "how" defines the enterprise behaviour, i.e. the way things are done. The "when" enforces the notion of time as being an essential component of the model. It can be associated with events representing a change in the state of the enterprise at a certain time. The "who" concerns the resources or agents of the enterprise performing operations of the business processes. Of course, the *how much* (economic aspects) and *where* (logistical aspects) are also important aspects of an enterprise to be considered.

Based on these assumptions, Curtis et al. (Curtis et al. 1992) define four basic aspects to be modelled in an enterprise:

- functional aspects describing what has to be done;
- behavioural aspects defining how and when something has to be done;
- informational aspects defining what data are used or produced and their relationships; and
- organizational aspects indicating who has to do something and where.

An enterprise is by nature a complex, dynamic system. From the point of view of integration, various essential aspects of an enterprise need to be modelled, either to analyze or to control the enterprise. These include but are not limited to:

- organization functionality and behaviour in terms of processes, activities, basic functional operations, and triggering events;
- decision-making processes, decision flows, and decision centres;
- products, their logistics, and their life cycles;
- physical components or resources, e.g. machines, tools, storage devices, or transportation means, their logistics, capabilities, capacities, and layout;
- applications (i.e. software packages) in terms of their basic functional capabilities;

- business data and information and their flow in the form of orders, documents, data items, data files, or complex databases;
- enterprise knowledge and know-how, i.e. domain-specific knowledge, rules of thumb, specific decision-making rules, internal management policies, internal regulations, and so on;
- human individuals, especially their qualifications, skills, roles, and availability;
- organizational structure, i.e. organization units, decision levels, decision centres, and their relationships;
- responsibility and authority distribution over each of the previous elements;
- exceptional events and reaction policies; and
- the time aspect of all of the above, because an enterprise is a dynamic system.

Since the description of all these enterprise elements cannot be fully represented in just one model, it usually results in different, partially interconnected, overlapping models. These may include: product model, process model, functional model, information model and its databases, knowledge base, resource model, relationship model, configuration model, decision model, economic model, organization model, and role model. However, all these different aspects of the enterprise are based on the enterprise resources and the relationships between them. Even the functional model, which defines the activities and processes in the corporation, is defined on top of the various resources. In this thesis, we consider mainly the resource model, the relationship model and the role model, but we will also apply our work to business process modelling, represented by the functional model.

2.6 Success Criteria in Organization Reengineering

Since organization reengineering (OR) is a continual process in the life of an enterprise, it is important for the enterprise to evaluate and to refine the process. The tangible result of the OR process is the major indicator reflecting how well the corresponding organization model actually performs its task of modelling the enterprise. In this section, we identify four criteria that are essential in measuring the success of an organization

model. (In Chapter 10, we will assess how successfully our work has met the requirements emanating from them.)

1. Scalability

   It is important that the model is scalable from modelling hundreds of objects to millions of objects, which may correspond to terabytes of data in the repository. The implementation of the resulting design model should also support the capture of a large number of objects in the enterprise. Consider a company of 100,000 people, each of whom manages 20 resource objects (such as projects, machines, products, customer accounts, and so on); the total number of objects will be 2 million. This is even worse when we consider maintaining the relationships that exist amongst these 2 million objects.

2. Extensibility

   It will be impossible to include in the conceptual design phase an exhaustive list of all possible types of resources. Indeed, as a company grows, it will often accrue new types of resources. Practically, an organization modeller begins with a few types of objects, such as people, departments and machines, to be included in the design. This way, both the design and the implementation phases are kept to a manageable size. Once these types of objects are covered and the design is implemented, the organization designer may then add other types of objects such as documents, equipment, products, and many others. This means that an extensible model and methodology are critical for successful organization modelling.

3. Flexibility

   To allow the methodology to be applied to different types of organizations, the reference model (Cheng 1998) must be flexible. A successful organization model should be applicable to not only some types of commercial companies, such as hi-tech firms or manufacturing enterprises, but also to other enterprises including non-profit organizations, governmental structures or even an entire country.

4. Performance

   Through the conceptual design and design implementation phases, a composite organizational information system will be deployed in the enterprise. Workers, managers, administrators and executives will rely on this system to access the most

up-to-date organizational information in making decisions, performing analysis and deriving future plans. They will also run collaborative applications on top of this organizational architecture to enhance communication and coordination amongst resources. Consequently, such an enterprise information system must have a quick response time that scales favourably in the various application environments. Unfortunately, as discussed earlier, today there is not enough research focus in this area, let alone a benchmark standard to measure performance. In this thesis, although we concentrate on the proposal of an organization model rather than on a specific implementation, we have implemented a prototype of our model in order to demonstrate aspects of our ideas. This prototype has been used to perform organization modelling design for Hitachi America. We will review the performance of our system in Chapter 10.

2.7 Conclusion

In this chapter, we have introduced the concepts of organization modelling and reengineering. Organization modelling is composed of three phases; namely organization analysis, organization conceptual design, and organization design implementation. Organization reengineering concerns the analysis and refinement of an enterprise by applying organization modelling repeatedly in a cycle.

We also discussed the principles behind organization modelling. A strong organization model should observe these principles as much as possible. They therefore lay the background of our proposed organization model; we elaborate in Chapter 10 how our model addresses all of these principles.

The ultimate goal of organization modelling is to provide a uniform representation of the enterprise, so that decision-makers not only understand how the enterprise operates, but are able to make changes to it quickly. In addition, all employees can be part of a common organizational architecture in order to automate their business processes. Indeed, we show that organization modelling is mainly driven by business process reengineering (Cheng 1995), which concerns the automation and streamlining of business processes among company resources.

We have also discussed the scope of organization modelling in this chapter. Although there is a long list of models which might be included in organization modelling, we highlight the importance of the resource model, the relationship model and the role model. Our work is mainly concerned with the realization of these models.

Finally, we presented the criteria of a successful organization modelling effort. The four criteria that are essential to the success of organization modelling are scalability, extensibility, flexibility, and performance. We will measure our work vis-à-vis these criteria in Chapter 10.

CHAPTER 3 A Critical Assessment of Organization Modelling Approaches in Existing Workflow Management Systems

In this chapter, we will review the existing efforts in organization modelling. We are particularly interested in focusing on the organization modelling approaches that are used to support workflow and cooperative works. A running example of a workflow process will be described in Section 3.3. This example will be used throughout this thesis to illustrate the novel approach we have taken with OMM and the contribution that our proposal makes to the current state of organization modelling. In Section 3.4, we will survey the organization modelling systems used by the current state-of-the-art workflow management systems. These systems include ARIS, CIMOSA, EMS, M*-OBJECT, Objectflow, and SAM*. We will also describe and compare a common practice of using corporate directory services to support role resolution in some other cooperative computing systems. Finally, in Section 3.6, some stand-alone organization modelling and management systems will be surveyed and reviewed. Based on this assessment of all current efforts, we will summarize the shortcomings of existing systems in supporting a collaborative application environment.

3.1 Introduction

Corporations rely on business processes to allow various enterprise resources to work together in order to carry out their business functions. To ensure market competitiveness, corporations have to constantly evaluate and streamline their business processes. Workflow technology emerged in the late eighties to enable automation and coordination of business processes through computational models (Howard 1991). Workflow concerns the routing of work amongst designated organizational resources. Although it is possible to define and manage business processes on a smaller scale, such as running a departmental workflow application, without a carefully designed organization model it is impossible to run production-level and enterprise-wide workflow. A survey on the workflow architectures in use today reveals that all production-level workflow systems have an organization and role modelling component embedded in WFMS.

3.2 Workflow Management Systems

Workflow technology supports business process integration and automation (Medina-Mora et al. 1992; Cheng 1995). It provides a framework on which multiple tasks and applications are integrated to form a network of computational steps to accomplish a business process (Vanderaalst and Vanhee 1996). A workflow process can be formulated as a set of nodes, representing tasks or steps, connected by directed edges which are condition arcs governing the route of the process (see Figure 3-2). To ensure that a model has a consistent flow behaviour, a process always has a BEGIN and an END step. The BEGIN step only has outgoing arcs and the END step only has incoming arcs. The steps that exist between BEGIN and END have one or more incoming arcs and outgoing arcs (Hsu and Kleissner 1996). We note that the outgoing arcs may represent a *split*, in which case the process branches out and routes to more than one node. A split can be an "AND" or an "OR" condition. The "AND" split results in parallel execution of multiple steps. The "OR" split maintains a single thread of execution, but provides a means to choose different routes depending on the condition of the flow. Parallel routes will always merge back to a single node at some later point in the flow; this constitutes a *join* in the workflow process. Thus the nodes define the atomic steps, or units of work, of the business process, while the arcs define the flow logic of the process. Each step is "atomic" because in order to ensure process consistency, the workflow system must guarantee that the state transition before and after a step be completely done or have no effect at all. The atomicity property is also referred to as the "all-or-nothing" behaviour in transaction processing. The model used to represent a business process is termed the flow model or process model. Petri Nets are one of the most common representations for the flow model (Murata 1989; Vanderaalst 1996). Workflow systems usually include a graphical editor to allow administrators to define business processes using a Petri-Net approach.

As a process progresses in time, different tasks are created and assigned to various agents in the company. An agent is a kind of organizational resource responsible for the execution of different tasks. Sometimes an agent can be a person (an employee) executing a software program to complete a task. Occasionally, a software program can be triggered by the workflow system to run automatically without any human participation. There are two possible ways an employee becomes involved in the workflow. In some situations, a particular employee may be chosen to execute a step

(the *push* model). In other situations, a group of employees may be identified as potential candidates to perform a task; anyone of this group of employees will pick up the task from a work list on their own initiative (the *pull* model). In both cases, authorization checking must be performed when someone attempts to open and execute a workflow step. To allow flexibility in workflow authorization, the workflow system often adopts a certain *role model*. A role model describes how roles are defined and how different resources may play one or more roles within the company in order to perform certain tasks. With the abstraction of a role model, the workflow system simply performs task assignment and task authorization over roles (Cheng 1995).

A WFMS is made up of a number of software components, each of which is implemented based on a conceptual model. These conceptual models include the process (or flow) model, the role model, the organization model, the application model and finally the data model. The data model is concerned with how workflow specific data flow from one step to another, and how it is shared and controlled by various users who participate in the flow. The flow model defines the flow logic of the workflow process. It is responsible for moving the process from beginning to completion through the various state transitions of the workflow steps. The application model defines how a software application program relates to a step. It also defines the data exchange, if any, between the workflow system and the application software. Again, the role and organization models together deal with task assignment and task authorization in the workflow process. Figure 3-1 shows how these different components interface with one another.

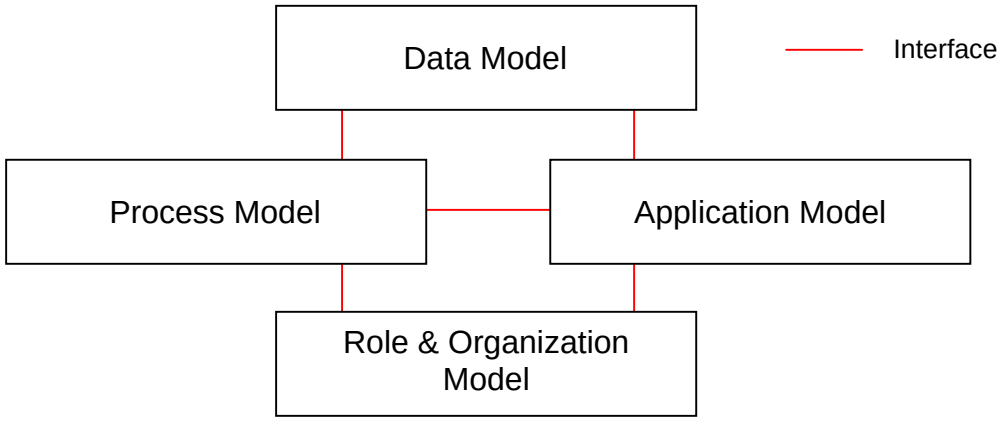


Figure 3-1 Different Components of a Workflow Management System

3.3 Running Example: A Sample Workflow Process

Within the workflow approach, process routing control is abstracted out from the logic of the application; it thus results in a flexible design and implementation of the flow logic without interfering with the implementation of the associated applications. The flow logic concerns mainly the routing decisions throughout the life of a process instance. The Petri-Net like representation in Figure 3-2 illustrates a flow description of a business-to-business electronic commerce application. This example captures a workflow, which allows business partners to order electronic parts over the Internet.

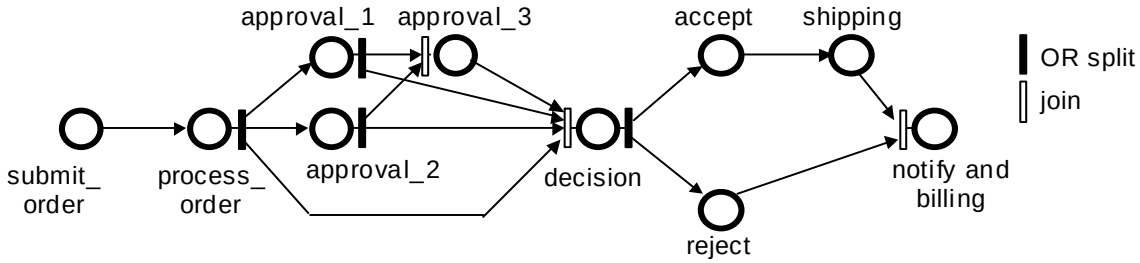


Figure 3-2 Electronic Parts Ordering Process

In this example, an electronic parts manufacturing company is offering its products online to business customers. The company divides its product lines into divisions, and different departments within a division manage the production and sales of individual products. Employees of the company work in different departments; they are further classified into permanent workers, temporary workers and contractors, for example. Conceptually the company resources can be represented as in Figure 3-3. In Chapter 4, we will use our model to represent the resources of such a company.

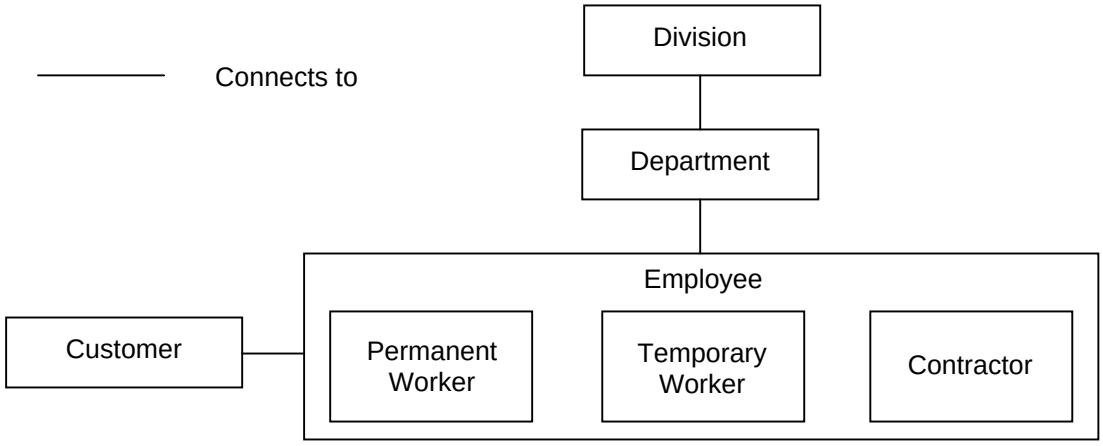


Figure 3-3 Company Resources Exist in a Hierarchy

A sales representative (rep) works in a department and reports to his or her corresponding manager. To complete an electronic ordering process, a number of employees are involved. The business process is initiated by an employee (flow-initiator) in the customer company, most likely over the Internet. This employee will execute the *submit_order* step. The WFMS will then move the process forward, and the sales representative who is serving the flow-initiator's company must execute the *process_order* step. If, for example, the dollar amount of the order is within the customer's credit limit, the flow will immediately move to the *decision* step. However, if the dollar amount of the order exceeds the customer's credit, the business process will require managerial signature; this corresponds to the *approval_1* step in Figure 3-2. In general, only the manager of the sales rep of a process instance can grant such authorization. Unfortunately, since the manager-to-sales-rep ratio is usually high (1:10) and in some cases can grow to 1:20, this step often creates a performance bottleneck. To reduce the occurrence of such a bottleneck, a policy is established such that the approval signature can be sought from an alternative source under the following condition. If the sales rep's department currently has more than *n* managerial approval cases pending (where *n* can be any positive number set by the administrator), then managers of any other departments within the same division can authorize the process. This corresponds to the *approval_2* step in Figure 3-2. Once a managerial signature is obtained, the process will move forward. If the difference between the customer's credit and the order's value is over a certain amount $*m* (where *m* can be any dollar amount set by the administrator), then an additional signature from the vice president of the division is required. This corresponds to the step *approval_3* in Figure 3-2. Note that only the vice

president of the division to which the sales rep of this process instance belongs can process step *approval_3*. The next three steps in this example, *decision, accept,* and *reject,* can be automated steps, which are accomplished by software programs without any human intervention. The *shipping* step can be accomplished by employees in the shipping department. The final step, *notify-and-billing,* can again be an automated step.

A sophisticated workflow management system supports the definition of this process by allowing an administrator to define this flow-map using a graphical or scripting interface. The workflow data, which impacts the routing decision of the flow, is also defined as part of the flow definition. Agent applications, the applications associated with individual steps, are connected to the workflow management system through a workflow programming interface. Finally, roles are defined to control *task assignment* and *task authorization* as well as to control access to the underlying workflow data. Since the organization model of the current WFMS does not support dynamic relationships between corporate entities, it is not possible to adequately model task assignment and task authorization of a practical flow such as the one shown in Figure 3-2.

3.4 Previous Work in Workflow Organization Modelling

Resource Manager implementations have historically focused on technologies surrounding access methods, concurrency control, and logging and recovery. The security model and access control systems usually assume a simple and static model which is based on user and group identifiers. In general, cooperative computing software applications, such as workflow, group scheduling, and electronic commerce applications, simply adopt the user and security model of RM's, e.g. an RDBMS, as their access control model. However, the user model in an RDBMS is designed primarily to support access control in processing isolated transactional operations rather than integrated process activities. It is thus not adequate to model the flexible resource relationships that are required to support collaborative computing. Other researchers have proposed specific role models and methodologies for concurrent engineering such as ARIS (Scheer 1993), CIMOSA (Kosanke et al. 1997), EMS (Graefe and Chan 1993), M*-OBJECT (Di Leva et al. 1997), Objectflow (Hsu and Kleissner 1996) and SAM* (Su 1986). These researchers all start from the process view and tightly couple the

organization model with the role model and the process model. We next review these systems briefly.

3.4.1 ARIS

ARIS (ARchitecture of Integrated information Systems) was designed to be used as a foundation for the creation and evaluation of methods for information systems design to support CIM environments (Scheer 1993). The ARIS architecture constitutes a framework in which integrated applications systems can be developed, optimized and converted into process-oriented implementations. At the same time, it demonstrates how businesses can examine and analyze information systems in order to translate their contents to support process-oriented applications. The ARIS architecture aims to define four major components, namely organization, functions, data, and control. All the components are broken down in terms of their proximity to the information technology resources into three descriptive levels: business-level concept, electronic data processing concept, and implementation. Figure 3-4 shows the ARIS architecture.

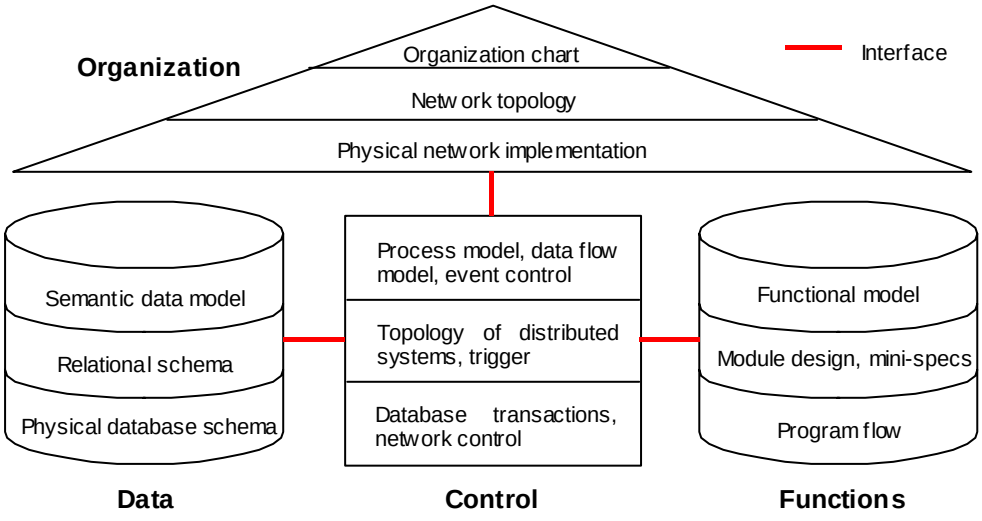


Figure 3-4 ARIS Architecture

In ARIS, support of business processes is an important goal for business information systems. The components of an information system to be described from the business economics standpoint are conditions, events, processes, human labour (employees), equipment, production materials, and organizational units. Since each component can be related to every other component, this situation generates a complex structure. Although

ARIS explicitly describes inter-object relationships, it uses a hardwired model to capture these relationships.

ARIS has an organization model which covers mainly human labour (employee resources). Since ARIS focuses on the CIM environment, human labour in the model refers to only those employees involved in the production process, and those users who are directly involved with the information system. Users are assigned to organizational units, which are constructed on the basis of criteria such as "same job function" or "same work object". This view is referred to as the organizational view. The central concept in the model of the organizational view is the organization unit. This can be defined as a department, position, or a larger unit such as an operational area within the enterprise. Here the boundaries between the meta-levels and the description of an application are not always clear. For example, the organizational view is on the one hand described as part of the ARIS architecture in the meta-information model, and on the other hand "organization" is a business application which can be represented in an application-specific "organization" data model (Scheer 1993).

3.4.2 CIMOSA

The AMICE Consortium through a series of ESPRIT projects has developed CIMOSA, the European Open Systems Architecture for CIM. It structures a CIM system as a set of concurrent communicating processes executed by a finite set of functional entities (Kosanke et al. 1997).

CIMOSA consists of:

1. A modelling framework based on a process model.

2. An integrated infrastructure consisting of information technology services to support enterprise integration, application interoperability and model execution.

3. A methodology, called the CIMOSA modelling process, which follows the CIM system life cycle.

In CIMOSA, an enterprise is considered to be a set of communicating concurrent processes governing the execution of basic actions, called *functional operations*. An enterprise is also a finite set of agents, called *functional entities*, executing the functional

operations required by business processes and processing *enterprise objects*. Functional entities are any resources capable of performing actions. They range from programmable agents to intelligent autonomous agents, such as robots, computers, software applications or human operators. The link between functional entities and business processes is made by functional operations, which are atomic functional units in the model. A functional operation is any message sent or received by an agent. Enterprise objects are entities used, transformed, produced or consumed by functional entities of the enterprise. They are characterized by a set of properties. They exist in the real world under different appearances or states.

Like other WFMS, processes are used to define the control flow in CIMOSA. Processes are triggered by events and can be structured into basic steps, called *enterprise activities*. An enterprise activity is a task to be performed by one or more functional entities allocated to the activity for the entire duration of the task. In other words, a task is a sequence of functional operations. CIMOSA also has the concept of *domains* to group different processes into logical units. This helps in managing different classes of processes in an enterprise.

CIMOSA uses *resources, organization units and organization cells* to capture the organization model. *Resources* can be of two types: passive resources (not able to execute functional operations by themselves such as tools and carts) and active resources, resources having a control device (able to execute functional operations). Active resources are the functional entities introduced above. In other words, organizational resources in CIMOSA are classified mainly from the viewpoint of business processes.

An *organization unit* in CIMOSA is an enterprise object (usually a resource) assuming responsibility and authority for one or more elements of the model. For instance, a foreman can be the supervisor of an operation floor where some process-oriented activities are performed to produce some product. Task assignment and task authorization therefore can be applied not only to individuals but also to an entire organization unit. There is a close connection between the definition of organization unit and the definition of process authorization.

In CIMOSA, *organization cells* are used to group organization units into larger entities at different responsibility levels in order to model the organization structure of the enterprise. Organization cells may also be mapped logically to the process domains.

The CIMOSA model is a powerful model for implementing an event-driven, process-based approach using generic constructs. These constructs can be specialized or customized to build partial or particular models. However, the organization model assumes the constructs of the process model (actualized in such things as operations, processes and domains), and as a result it will only work within the context of CIMOSA. In other words, a corporation adopting the CIMOSA organization model must also assume the CIMOSA process model and WFMS.

3.4.3 EMS

The Enterprise Modelling System (EMS) has been jointly developed by the Institute for Advanced Manufacturing Technology and SIMCON, a consortium of companies engaged in collaborative research with the National Research Council of Canada. The main objective of EMS is to provide a comprehensive set of tools for the creation of structural and process models of the business and production operations within an enterprise, with capabilities specifically aimed at continuous process improvement and evaluation of decision-making alternatives (Graefe and Chan 1993). EMS is developed on the premise that business process reengineering involves the analysis and design of workflow within and between organizations. As a result, its focus is specifically on having an organization model to go hand-in-hand with its process model.

In EMS, the organizational structure of an enterprise is represented as a hierarchy of generic *business units*. A business unit is any type of organizational or functional unit consisting of a set of resources for performing either manufacturing or non-manufacturing operations. These business units are described through a set of processes. Resources in EMS can be human or process-related equipment categorized into different types such as managers, IT workers, clerical personnel for human resources, material handlers, manufacturing processors, and storage buffers for equipment. Examples of business units are a production department and a work cell or work centre. The user enters the hierarchical relationships of the business units in the form of a tree structure, and specifies the resources, processes and activities for each business unit via a graphical user interface.

The organization model of EMS is developed solely for the support of BPR. It concerns only resources that are related to the definition of processes in EMS. It lacks a generic model to comprehensively represent all the other enterprise resources and their interactions. As a result, it lacks flexibility in performing organization modelling. Moreover, EMS does not have a flexible relationship model to define the dynamic connections between the resources of an enterprise.

3.4.4 M*OBJECT

M*OBJECT, developed at the University of Turin, Italy, is a methodology for information system analysis, design and implementation for CIM (Berio et al. 1995; Di Leva et al. 1997). M*OBJECT covers all the three major phases in organization modelling, namely organization analysis, conceptual design, and implementation design (refer to Chapter 2 for a detailed discussion on organization modelling and its three phases). It is an extension to its predecessor, M*, which is a simpler organization model for CIM (Di Leva et al. 1987). M*OBJECT differentiates from M* by having a concept of life cycle for enterprise objects and an object-oriented paradigm for describing static and behavioural aspects.

Although M*OBJECT focuses on the information system aspect, its methodology heavily emphasizes organization modelling. For this reason, it is more appropriate to consider M*OBJECT to be a methodology for analysis, design and implementation of organizational databases. For instance, the model is used to describe three different analysis levels, namely: (1) the organization level or management level representing the enterprise functionality and behaviour, (2) the conceptual level, which provides a common view between managers, engineers, and other employees, and (3) the application development level.

The organization analysis phase of M*OBJECT concerns global enterprise modelling in general, and information system requirements definitions in particular. It uses a simple 3-level organizational architecture to describe an enterprise and its components. Firstly, an object *enterprise* is a real-world system, such as a company, production system, or cost centre, which is the focus of the analysis. Secondly, an *enterprise environment* is a subset of the enterprise made up of users and functions sharing a common view of the information system of the enterprise. Finally, an enterprise consists of *organization units*, which in turn can be broken down into *work centres.* These hierarchical

organizational elements are defined in terms of business objectives, business constraints, and clusterings of system functionalities at different organization levels.

Besides their organizational architecture, enterprises are structured according to a functional architecture, which must be analyzed and modelled. At the work centre level, a user *activity*, or work step, can be defined as a homogeneous set of actions. Activities are then clustered into *processes*, which can be defined as a coordinated and partially ordered set of activities that fulfils an objective of an organization unit. This constitutes the process model of M*OBJECT.

In M*OBJECT, the intrinsic behaviour of components can be expressed via the concept of a *life cycle*, i.e. a state-transition diagram used to specify the possible sequences of execution of basic activities for the given component.

Overall, M*OBJECT offers an organization model which describes functional, information, organization, and even resource aspects in one model. Control flow, information flow, and material flow can be defined collectively or separately with this process model. At the conceptual level, M*OBJECT takes advantage of object orientation to model complex enterprise objects. However, it does not push the object-oriented approach to its full extent; it still has a specific view of enterprise architecture. Companies using M*OBJECT must adopt this view in organization modelling. This can sometimes become a limitation. Furthermore, the tightly coupled architecture between the workflow component and the organization modelling component, and the fact that it lacks an open interface, have blocked other WFMS from integrating with M*OBJECT. It also lacks a relationship model to define dynamic interactions between resources.

3.4.5 Objectflow

Objectflow was developed by the Activity Management Group at Digital Equipment Corporation based on the prior research work of Reliable Flow Manager (RFM) (Hsu et al. 1991; Hsu and Kleissner 1996). Objectflow is a WFMS, which executes long-lived and multi-user computations, called *activities*. An activity is composed of a set of application routines, each necessary for accomplishing a business function, and is inter-related by information flow and control flow. An activity may involve multiple individuals or organizations. Its execution may last for hours, days, or months as compared to microseconds, as is common for short-lived transactions in OLTP.

Compared to other CIM modelling systems, Objectflow is a full-function, comprehensive WFMS. By that we mean Objectflow consists of a set of tools for defining business processes and the underlying organizational database, a workflow engine to manage run-time execution, and an administrative tool to monitor and audit process instances and process history. It provides various advanced services such as reliable forward execution, exception handling (including step-retry and step-compensation), system integration (i.e. invocation of services in distributed and heterogeneous systems), user notification by an event-driven mechanism, scheduling and load balancing among users, and administrative security. It also allows for tracking of status of current activities, and inquiry on the history of activity execution to support auditing functions. In addition to the workflow engine, which is the centrepiece of the system for controlling and managing the flow of business processes, Objectflow also has a set of tools to support development of workflow applications. This includes the Workflow Designer (a GUI tool to construct a business process in a Petri-Net like graph), the Organization Editor (to define organizational entities and populate the organizational database), the Inspector (for inspecting status and history of activities), and the Management Utility (to perform startup, shutdown, abort flow or step, and other administrative functions).

In order to accomplish user notification, scheduling and load balancing, Objectflow uses a *Policy Resolution Model* (PRM), which is built on top of its organization model (Bussler and Jablonski 1995). PRM is implemented as a tightly coupled component in Objectflow. It allows users to define organizational entities and relationships, which in turn support the specification of business policies to control task assignment and notification in the workflow system. In PRM, an administrative user defines an organization by specifying object types and relationship types. Object types are basically roles that exist in the company; examples include *Manager, Secretary,* and *Agent*. Relationship types define organization structures over the roles, such as *Member_of, Manager_of,* and *Responsible_for*. Both object types and relationship types belong to a conceptual schema of an organization structure. They have to be implemented and populated in a system implementing PRM to capture all instances of a real organization in the system.

3.4.6 SAM*

Researchers at the University of Florida, funded by the Department of Energy, have investigated the requirements of a scientific database and integrated manufacturing system. The result of this investigation was the creation of a semantic association model, called SAM*. SAM* can be used to model semantic properties of data in integrated manufacturing environments. It focuses mainly on the data model aspect; however, with its object-oriented approach, the model can be applied to cover modelling of different types of enterprise objects. Compared to other systems, SAM* is strong in being able to model the various types of enterprise resources.

In the SAM* semantic association model, an integrated CAD/CAM database can be modelled by a network of inter-related *concepts*. A concept can be a physical object, abstract object, or event. A concept can be defined by a set of attributes or characteristics. There are two types of concept: atomic and non-atomic. Atomic concepts are those which cannot be decomposed; these are observable objects, that users regard as fundamental information units. Examples are an employee, a workstation, and a project. A non-atomic concept is an object that can be decomposed. Its meaning is described in terms of other atomic or non-atomic concepts. For example, the concept of "Project Team" can be described by the concepts Employee, Robot and Project.

The grouping of atomic or non-atomic concepts to describe another non-atomic concept is called an *association*. Different types of association can be distinguished according to the different structural properties, operational characteristics, and semantic constraints that users or the database administrator associates with these concept groupings. The seven association types in SAM* are listed hereafter:

1. *Membership association*: grouping of homogeneous atomic concepts to form a class. This creates different categories of concepts or objects.

2. *Aggregation association*: grouping of a set of attributes or characteristics to define another concept.

3. *Interaction association*: independent entity types defined by aggregation associations can be grouped together to describe a set of events or facts that are the result of some

interactions among the occurrences of these entity types. With interaction association, one can define relationships between atomic and non-atomic concepts.

4. *Generalization association*: grouping of concept types according to their generic nature to form a more general concept type. For example, two aggregation concepts, *foreign parts* and *domestic parts*, define two sets of parts used in a factory. They can be grouped together to form a more general concept type, called *parts.*

5. *Composition association*: grouping of similar or dissimilar concept types, each of which is a part of the whole modelled by the composite concept. It has a single occurrence that is a set of sets. Each member of the set is a set of occurrences associated with one of its component concept types. A composite association can be formed by any type of association, including the composite association itself.

6. *Cross-product association*: grouping of some concept types whose occurrences are the result of taking the cross product of the occurrences of its component concept types.

7. *Summarization association*: similar to cross-product association, summarization association is a grouping of concept types whose occurrences are the results of taking the sum of the occurrences of its component concept types.

We observe that association types 2, 3, 4 and 5 correspond closely to concepts encountered in the OO paradigm, namely aggregation, coupling, inheritance and composition.

Using the SAM* model, integrated CAD/CAM databases and applications, as well as enterprise resources, can be abstracted and represented graphically as a network of association nodes whose types are explicitly labelled. A given node can be easily traced to all nodes that are semantically associated with it (that is, described by one of the seven semantic associations). Each association type has its own structural properties, constraints, and operations that are, in general, different from the other types.

SAM* has its strength in generic resource definition using an object-oriented approach. Its association types also provide a rich semantics and well-classified types to define groupings and relationships between the various objects. However, relationships in SAM* are limited to simple association, summation, and cross product. For more

complex relationships, which can only be expressed by a computable expression, SAM* may have some difficulties in nesting the various associations to establish a relationship. Similar to other systems, SAM* also lacks an open API.

3.5 Directory Service Based Organization Model

In contrast to the strategies used by the integrated BPM systems just reviewed, other efforts to model an enterprise have attempted to address the role management issue through directory services. Role management in corporations using DS is sometimes referred to as position management. With the integration of Internet, Intranet and corporate networks, a typical business network is becoming location-independent. The directory service provides information, applications, and communications to people wherever they might be, inside and outside the organization – often from physical servers scattered around the globe. Directory services and other naming services, such as DEC's cell service, aim to support distributed object lookup with a naming convention (CCITT 1988; Jia and Maekawa 1999).

A recent survey indicates that Fortune 500 US companies each have an average of 190 directories, and supporting them is becoming increasingly complex due to the number of formats to be supported and the disparate systems available to collaborating users (McFadden 1999). Such problems may be eased by the use of meta-directories, which integrate directories by linking various different schemas and attributes into a universal, logical view. Directories can be used as the basis for e-commerce and collaborative software as well, such as in the case with the Automotive Network Exchange, which provides an Extranet for e-commerce between suppliers and the automotive industry.

A directory service is a database of objects, which may include users, applications, network devices, and other resources that users might find on a network. It helps to manage relationships between people and networks, network devices, network applications, and information on the network. Under this type of modelling scenario, each object, including persons, in a company is assigned a static and universally unique identifier. Roles can be created as objects in the DS and users are assigned to these roles. This approach yields an efficient solution for simple point-to-point interactions in collaborative software; it resolves static roles efficiently. Nevertheless, DS lacks a generic organization model to support modelling the different enterprise resources. It

also lacks a model to define dynamic relationships between resources. Consequently, DS fails to support advanced applications such as those involved in publish-and-subscribe scenarios where the publisher is not interested in a list of static roles but would like to identify privileged subscribers based on some correlation between the publishing context and the characteristics of target customers (Cheng 1999b).

Hereafter, we will survey the three most commonly deployed corporate directory standards; these are the X.500 and LDAP, Novell's NDS, and Microsoft's Active Directory.

3.5.1 X.500 and LDAP

The OSI standard architecture is composed of two portions, namely the application layer and the communication layers; X.500 is an application-layer protocol in the OSI architecture (CCITT 1988; Radicati 1994). The physical representation of the X.500 directory model consists of three functional components:

1. The Directory Information Base (DIB),

2. Directory System Agents (DSA), and

3. Directory User Agents (DUA).

The DIB contains a collection of information about users, resources, and the network that is maintained by the directory. The DIB resides physically within and is managed by network servers, known as Directory System Agents or DSAs. DSAs provide the actual directory service and implement the service side of the directory operations. DUAs represent the "client" side of the directory service. They represent the user in activities involved with accessing the information stored in the directory. Figure 3-5 shows the functional picture of the X.500 model.

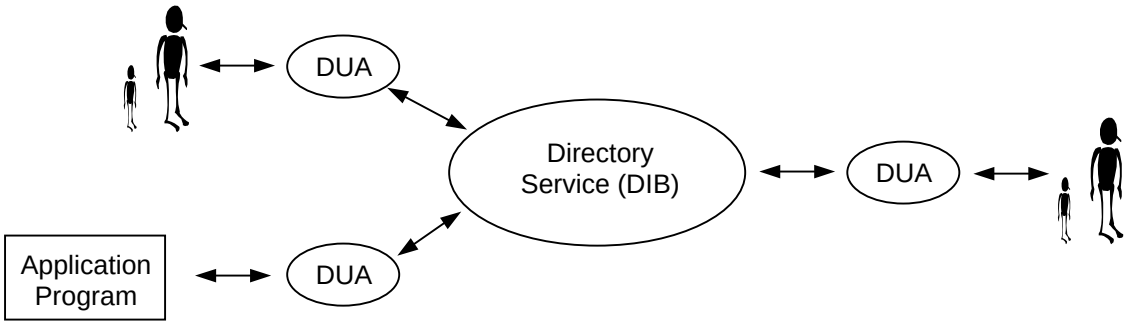


Figure 3-5 User's View of the X.500 Directory Service

As the information contained in the directory grows, it is usually necessary to partition the DIB among multiple DSAs, called cooperating DSAs. This increases the availability of the information and improves overall system performance by ensuring that information is maintained close to the network users who need to access it most often. From the view of the DUA, however, such a collocation of DSAs must continue to behave as a single unified database. To accomplish this, X.500 directories comprise two distinct protocols:

1. A Directory Access Protocol (DAP), which is used by DUAs to access the information stored in DSAs.

2. A Directory System Protocol (DSP), which is used between DSAs to service user queries that require information, possibly distributed over multiple DSAs.

Figure 3-6 shows a set of DUAs that access the directory service as a whole by using the DAP protocol, and a distributed directory service made up of multiple DSAs that interact with one another using the DSP protocol.

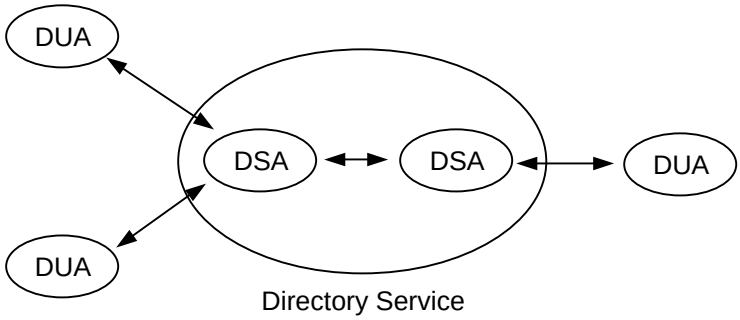


Figure 3-6 Distributed Directory Service

The DIB is the physical representation of the X.500 directory. The logical representation of the X.500 directory is referred to as the Directory Information Tree (DIT). Each DIB entry corresponds to a vertex of the DIT. The information stored in a particular DSA is referred to as its DIB. Figure 3-7 shows a sample DIT, made up of entries from three countries: the United States, the United Kingdom, and Japan. Each country name space is further subdivided into an organizational-level name space for company names (NC Corp, Oracle and Enqueue). The company is further divided into organizational units. Finally, people resources are listed under each organizational unit name.

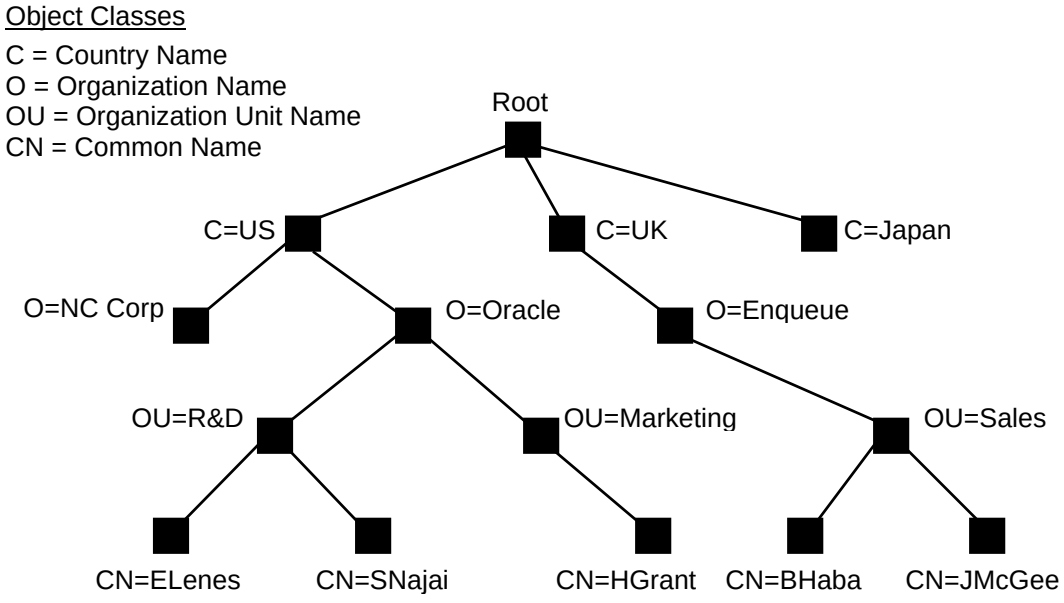


Figure 3-7 A Sample Directory Information Tree

Since X.500 is intended to serve as a global directory service, this implementation of X.500 is commonly used by many corporations in today's world (CCITT 1988). Section 9.1.6 provides a more detailed discussion on X.500 and how it is compared to our proposed organization model.

LDAP, a lightweight implementation of the X.500 DAP protocol, was developed by the University of Michigan (Howes 1995) as a simple protocol on X.500 directory systems. It has recently proliferated with the growth of the Internet, and is being used in a wide variety of network-based applications to store data such as personal profiles, address books, and network and service policies. Such systems provide a means for managing heterogeneity in a way far superior to what conventional relational or object-oriented databases can offer (Cluet et al. 1999). As an X.500-based system, LDAP uses the same directory and information model as X.500.

3.5.2 Novell NDS

Novell's NDS is a fully functional directory service that is based on the X.500 international standard (Andrew and Shropshire 1998). As an X.500-based system, NDS organizes objects in a hierarchical tree structure, called the directory tree. A company or organization can arrange objects in the directory tree according to its organizational structure, which usually represents how people access and use company resources.

An NDS object is contained in an Organizational Unit (OU), which in turn can be contained in another OU. Just as individuals have unique identities within an organization based on their individual needs, departmental needs, and organizational needs, identities are created in NDS to provide users with access to the resources they need to fulfil their responsibilities. For example, all employees in an organization are given an office, desk, and telephone just because they are employed; similarly, NDS allows every individual in an organization to have access to specific resources, such as employee benefits and telephone databases. Beyond this, individuals within a particular division or group should automatically be given access to network resources associated with their division or group. NDS uses the inheritance property in access rights authorization to make this possible. Administrators may grant access rights to an individual; alternatively, rights can be granted to an OU. Users may have requirements that are specific to their individual responsibility as well as organizational needs. For example, a user may be given access to specific network resources associated with the user's role in the Finance Department. Additional network access may be granted because the user is a member of a task force that is working on a specific issue. Similarly, privileges may be granted based upon location as well.

Overall NDS is strong in providing a comprehensive directory service environment to enable network and application management. It supports distributed services, replication, and has a fault-tolerant feature. It has an open API for any collaborative software to interface with the directory. However, as a practical DS that focuses on supporting network management, it lacks a generic organization model to include resources and relationships outside of the computer network environment, such as projects, assets, sales regions, etc. It also lacks a relationship model to define complex relationships amongst resources.

3.5.3 Microsoft Active Directory

Microsoft's Active Directory is bundled with Windows 2000. Some industry analysts believe that the Active Directory could be the most significant determining factor in Microsoft's quest to make Windows 2000 a true enterprise server solution (Burns 1998). With scalability and Internet-based standards, Active Directory provides much more than Microsoft's older Security Account Manager service directory. Active Directory is a completely extensible and scalable directory service, with hierarchical and scalable name space; directory partitioning for global enterprise scalability; multi-master replication of data store; dynamically extensible schema; and LDAP support for interoperability. The Active Directory software development kit also includes an API software library for linking other directory services, including both NDS and LDAP, and to support integration with collaborative software. Windows 4.0 has one master replication model, the Primary Domain Controller, which has the only write-enabled copy of the directory database. Windows NT 5.0 with Active Directory uses a multi-master replication model that refers to servers as Domain Controllers. This increases the reliability and availability of directory services in NT.

With Active Directory, a *domain* is a functional or administrative grouping of resources. One domain tree is a full partition of the directory, and Active Directory can be separated into organizational units for functional or administrative groupings. Although Active Directory was not developed based on the X.500 model, with the domain tree, which is very similar to the DIT in X.500, the information model of Active Directory is indeed very similar to that of the X.500 directory.

Despite Microsoft's intentions of making Active Directory the de facto industry standard, analysts believe that it is unlikely that the wide and deep penetration of NDS in the PC directory market will be replaced by Active Directory in the near future (Gaudin 1999). In fact, Microsoft, in the Windows 2000 release, provides a bi-directional synchronization tool that links NDS with Active Directory. At some point in time, users will want one directory infrastructure, but given the existing penetration of NDS, that will take some time to become a reality. This is a significant feature for developers of collaborative software; building applications on top of the Active Directory means they can leverage the organizational databases that are managed by LDAP and NDS.

Similar to NDS, Active Directory is strong in providing a complete set of network directory services such as data replication, directory synchronization, and administrative tools. Nevertheless, just like NDS, it is feeble in supporting enterprise-wide collaborative software that must coordinate activities among various types of dynamically related organizational resources.

3.6 Standalone Organization Modelling Systems

In addition to completely integrated BPM systems, and the DS-based approaches we have just seen, there exist systems which are essentially standalone organization modelling systems. Some implementations attempt to isolate an organization component from the workflow engine. Some prominent works in this area include ORM and OVAL. This category also includes organizational and office systems whose purpose is to provide a graphical modelling capability for describing organizational structures. This section will survey the said two systems by outlining their salient features, strengths, and weaknesses.

3.6.1 ORM

WorkParty, a workflow system developed by SIEMENS, has an organization component called ORM, which is a standalone client-server database application to support organization modelling (Bussler 1994; Rupietta 1994). ORM has an application programming interface and a graphical user interface to allow users to define and populate the organizational database. Although ORM separates the organization model from the process model, it does not separate the organization model from the role model; roles are defined as a resource type in the organization. As a result, it falls short in supporting resources playing different roles under different contexts, such as when they are involved in different business processes. Limitingly, the organization definition and the role definition of ORM are static, as is the case with many other systems, and it suffers from the lack of a dynamic relationship model.

3.6.2 OVAL

OVAL is the acronym for *Object, View, Agent,* and *Link*. It is a "radically tailorable" tool for constructing a cooperative work environment. The *OVAL* project was led by

Malone at Sloan Business School of MIT; its purpose is to provide a handy GUI tool for forming organizations within a corporation (Malone et al. 1993; Malone et al. 1995; Oval 1992). OVAL has an object model for constructing organization information and structure. Through user-specified rules, it can process message objects such as notification or customized information flows according to a user's needs. OVAL also supports adding hard links between resource objects but not dynamic links. OVAL was developed upon four key building blocks:

1. Semi-structured *objects* represent real-world entities such as people, tasks, messages, and meetings. Each object includes a collection of attributes and attribute values and a set of methods that can be performed upon it. The object types are arranged in a hierarchy of increasingly specialized types with each object type inheriting attributes and methods from its parents in the hierarchy. The objects are semi-structured in the sense that users can fill in as much or as little information in different attributes as they desire and the information in an attribute is not necessarily of any specific data type (e.g. it may be text, integer, or a pointer to another object).

2. User customizable *views* summarize collections of objects and allow users to edit individual objects. For instance, users can select the fields to be shown in a table display of a collection of objects, or they can select the links to be used to create a network display of the relationships between objects. A calendar display can be used to summarize objects with dates in one of their attributes. Any appropriate display format can be used to show any collection of objects and any attributes of those objects. Currently OVAL implements the table, network, and calendar views.

3. Rule-based *agents* perform active tasks for users without requiring the direct attention of those users. Agents can be triggered by events such as the arrival of new mail, the appearance of a new object in a folder, or the arrival of a pre-specified time. When an agent is triggered it applies a set of rules to a collection of objects. Rules contain descriptions of the objects to which they apply and actions to be performed on those objects. Actions include general actions such as moving, mailing, and deleting objects or object-specific actions such as loading files or responding to messages.

4. *Links* represent relationships between objects. For example, users can use links to represent relationships between a message and its replies, between people and their

supervisors, and between different parts of a complex product. The links are in hypertext; users can follow these hypertext links by clicking on them, and the knowledge represented by the links can be used by rules or in creating displays.

Using a GUI tool, OVAL users create and populate organizational objects, define relationships between specific objects, customize views to manipulate organizational data, and instantiate agents to watch over certain events. Although OVAL provides a handy object tool to create and manage organizational objects, it does not have a formal relationship model to construct roles and dynamic relationships; it therefore has difficulty in managing a large number of linked objects in a shared, distributed network. It also lacks an open interface for other cooperative software to take advantage of the organizations created.

3.6.3 Other Organizational and Office Systems

Other researchers have proposed visual and programming languages for organizational and office systems such as Officeaid-VPE (Di Felice and Clementini 1991), HI-VISUAL (Hirakawa et al. 1990), and Regatta VPL (Swenson 1994). Officeaid-VPE and HI-VISUAL were limited to the description of single office tasks. They are therefore not adequate for describing the integration and collaboration across multiple offices in an enterprise. Regatta VPL has a comprehensive process model and an abstract view of organizations; however, the coupling of the process model with the organization model limits its flexibility in organization and role design.

3.7 Weaknesses of Existing OM Systems

Overall, the existing approaches to the support of organization modelling and role management suffer from the following weaknesses:

- Lack of a conceptual organization reference model. We need a generic solution so that we can apply the model to the access control needs of different concurrent engineering environments.
- Tight integration with the BPM's process and application models. As a result of this tight integration, current solutions are only adequate for the support of those BPM systems that observe the specific models.

- Support for only some predefined resource types. Network DS systems focus on machine nodes, users and applications; messaging DS systems focus on user addresses; and BPR organization sub-components focus on users, groups and roles. To support integration and collaboration between different applications and users, the role model must be extensible and flexible enough to be able to cover all different resource types, including employees, departments, products, machines, projects, business partners, customer accounts, and many others. A model with this type of extensibility would allow us to model context-rich access control; for example, the user who plays role X is allowed to access relevant data only if he is executing from a machine node which is certified to be a secured node.
- Assumption of only static and hardwired relationships between resources. In reality, relationships between resources are changing rapidly. Relationships exist not only amongst resources of the same type, but also amongst different types of objects. For instance, there is a many-to-many relationship between the *projects* and *employees* of a company. Similarly, a three-way relationship can be defined between *users, machines,* and *projects.*
- Lack of ability to integrate with other organizational management systems. In an enterprise environment, it can be assumed that there will be existing directories and organizational resource information systems. A comprehensive architecture must take into consideration such directories and systems in order to exchange information with them whenever it is appropriate to do so.

In view of the above identified deficiencies, herein we present a new organizational and role model, called the OMM model, in order to overcome these deficiencies. The OMM model supports the organization analysis, conceptual design and the design implementation phases of the organization reengineering cycle (Berio et al. 1995). It has a conceptual and reference model for organization and role modelling. It also includes a relationship model to define the changing relationships between organizational resources. OMM does not assume a particular process or application architecture. With this generic approach, OMM is able to map its object types to other organizational data schemas, and to present an integrated and multidimensional view of an enterprise to the user. OMM consists of an open API to allow any external systems, such as WFMS and other collaborative software, to leverage the organizational information managed by OMM. A detailed description of the OMM model is presented in Chapter 4. Therein we

will also show how OMM can support WFMS to model flexible business processes such as the one shown in Figure 3-2. In Chapter 5, we will illustrate how the OMM model is used to support flexible task assignment and authorization in a cooperative computing environment.

3.8 Conclusion

In this chapter, we have surveyed the existing organization modelling (OM) systems that are used to support WFMS and other groupware. These OM systems fall into three categories. First is the OM component that is embedded in the WFMS. This includes ARIES, CIMOSA, EMS, M*OBJECT, Objectflow, and SAM*. Second is the directory service, which is used particularly by groupware such as communication software, e-commerce applications and conferencing applications. We reviewed the model and services of three most commonly deployed corporate directories: X.500 and LDAP, Novell's NDS, and Microsoft's Active Directory. Third is the standalone office and organization systems. They are mainly tools used to define organizational resources. We discussed briefly ORM, OVAL, Officeaid-VPE, HI-VISUAL, and Regatta VPL. The first two are tools that consist of a service to create and manage organizations, while the last three are mainly GUI ideas to represent organizational resources and structure.

In Section 3.3, we provided a running example of a business process, which is an electronic parts ordering process. Due to the requirement of relating the roles involved in executing this process, existing organization models fail to model the roles and therefore will not be able to support WFMS to model the task assignment and authorization of this workflow. This example will be used throughout this thesis to point out how OMM can satisfy such requirements and support the modelling of processes in general.

The ensuing chapter is devoted to the presentation of OMM. Therein we present the organization model and information model of OMM and compare OMM to its main competitors.

CHAPTER 4 OMM: A Hybrid Model for Organization Modelling

In this chapter, we will introduce our approach to organization modelling. Our model is called OMM, which stands for Organization Modelling and Management. OMM is a hybrid of the object-oriented model and the entity-relationship model. Through the object-oriented model, OMM can be applied as a reference model to represent different types of resources within an enterprise. The entity-relationship model allows OMM to define the complex relationships between resource objects.

In Section 4.2, we will describe the OMM organization model by using a combined entity-relationship and object-oriented diagram. In Sections 4.3 through 4.5 we will define each of the fundamental concepts of OMM in detail. OMM uses an object-oriented approach to describe the organizational information of the enterprise. With the OO approach, we are able to flexibly apply OMM to capture the various types of resources in different kinds of enterprises. Some of these resources are tangible in nature; these may include people, robots, equipment, products, facilities, documents and other types of resources. Other resources are intangible but of equal importance to the success of the corporation; these include projects, tasks, business objectives, long range plans, and many others.

4.1 Introduction

A business process may involve different corporate resources, which may in turn relate to one another indirectly through other resources; therefore, being able to represent any resources of an enterprise with a common organization model like OMM is a fundamental requirement in facilitating a cooperative working environment. Using the OMM model to manage the information pertaining to enterprise resources will serve as a common ground of collaboration between all employees. Administrative and management tools can be built based on the organization model and used to support different types of enterprises. In this sense, the idea is similar to what Javasoft's JavaBeans™ industry standard is attempting to accomplish (Voss 2002). All Java objects implemented based on the JavaBeans™ standard adopt a common calling convention and implement a set of required APIs. As tools are created to observe the JavaBeans™ standard, they can be used to handle all JavaBeans™ compliant objects.

However, modelling and storing organizational resource information is only the first step in organization modelling. Once we can describe different types of resources, we need to also flexibly describe the relationships between the different resource objects. For instance, in our case study, when modelling a company employing 25,000 people across 1,500 groups, over 82,500 relationships were estimated to exist between the people and the departmental infrastructure of the company. To identify the tens of thousands of relationships is a complex challenge, and is even more complex as we realize that these relationships are rapidly and constantly changing in an enterprise; once we have defined a relationship, there is a high chance that it could be outdated within days.

In OMM, we solve the issue of modelling numerous and dynamic relationships by using a policy-based approach. Instead of describing a relationship in a "hardwired" fashion, we abstract the relationship into a computable expression. We term such relationship definitions *virtual links* or *virtual relationships,* in contrast to the static, hardwired relationships defined in traditional organization modelling systems. In OMM, some of these virtual links exist between objects of the same type, such as the reporting hierarchy within a corporation which involves the relationships between all employees of the corporation. Other virtual links exist amongst different types of resource objects. For example, the relationships between people, departments, and facilities describe which departments and who occupy what offices. In our case study of applying OMM to Hitachi America, by abstracting relationships into computable expressions, we are able to describe over 2,000 relationships of the company by using four business policies. The results of this case study will be discussed fully in Chapter 9.

4.2 An Enterprise and Its Resources

An enterprise is composed of different types of resources. Resources include more than people. Human resources form only one category of resource objects that an enterprise depends on to accomplish its goals. Other types of resources include such things as machines, products, business units, R&D projects, assets, facilities, software applications, business processes and vendors. The list of resource types can be significantly different from one business to another. It may also grow and change with the growth of the company. The challenge in this area is that although it is possible to list all resource types within an enterprise at any moment, we cannot employ a rigid

approach to describe the resources of a changing enterprise, nor can we apply such a restricted model to different types of enterprises.

OMM is a hybrid model comprising aspects of the object-oriented model and the entity-relationship model. It is a generic reference model to define enterprise resources. As a generic model, OMM can be applied flexibly to define different types of resources. Furthermore, with a business rule-based relationship model, OMM can define flexibly the roles the resources play, as well as the inter-relationships between the resources. As discussed in Chapter 2, modelling of an enterprise involves defining its resource types and the dynamic relationships between the resource objects. OMM, as it will be seen subsequently, is able to cover both aspects.

Moreover, since business processes are executed by company resources that are related to one another, once the resource relationships are formally modelled, we have the opportunity to automate and improve the operational efficiency of the company by focusing on reengineering its business processes. Workflow technology provides the model, services, and tools to automate and refine business processes on top of a well-defined enterprise. A workflow system assigns tasks and privileges to a subset of corporate resources such as employees, robots or customers. It may also associate a workflow step with a certain application, or identify a set of trusted machines on which a workflow step should be running. OMM, as it will be seen subsequently, is able to support the stringent requirements of task assignment and resource allocation in workflow systems.

There are three fundamental conceptual entities that make up an enterprise in the OMM model, namely

- *organizations*
- *members* and
- *virtual links*.

An enterprise is composed of a number of OMM *organizations*. Each organization represents a type of corporate resources such as employees, departments, products or projects. Each object instance within an organization is termed a *member object*, or simply a member, in OMM. Each *member* within an organization maps to an actual

resource of the corporation. Members of the same type share a common set of attributes and methods that is extensible by the user. A member can relate or link to other members through *virtual links*. Contrary to static connections, a virtual link only has a relationship definition stored as a computable expression, which is evaluated and resolved at runtime. We shall define each of these OMM conceptual entities and discuss them in greater detail hereafter.

As stated earlier the OMM is a hybrid ER/OO model. In Figure 4.1 we can see the ER aspects of the model. In Figure 4.2 we see the OO aspects of the OMM model where the "isa" relationship should be interpreted as subtyping in accordance with the OO model (Booch 1999).

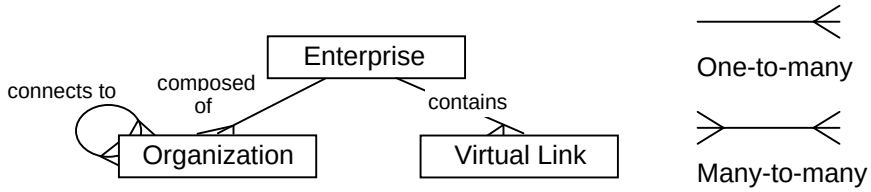


Figure 4-1 The ER Component of the OMM Model

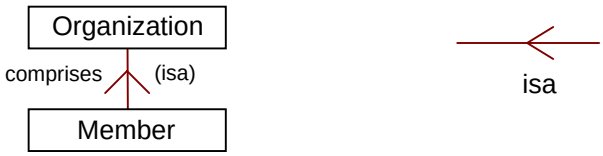


Figure 4-2 The OO Component of the OMM Model

From the diagram in Figure 4.2 we can see that it is possible to specialize the conceptual entity of organization (should it be necessary) but extension can also be achieved through the aggregation process of Figure 4.1. For example, a modeller may focus on a specific resource type, such as EMPLOYEE, in a company and apply the OO component of OMM to capture the properties of this resource type as the attributes of an OMM organization. In addition, s/he may define another resource type, say DEPARTMENT, and again represent its properties using the OO component of OMM. Finally, the EMPLOYEE organization and the DEPARTMENT organization may be connected using the ER component of the OMM model (through the *connects to* relationship). This *connects to* relationship between the two organizations is used to represent the overarching fact that all employees work in departments. These two organizations are

further aggregated to be put under the same Enterprise (through the *composed of* relationship).

Figure 4.3 combines the ER and OO components into a single diagram; the yellow box contains the ER aspects of the model while the green box contains the OO aspects of the model.

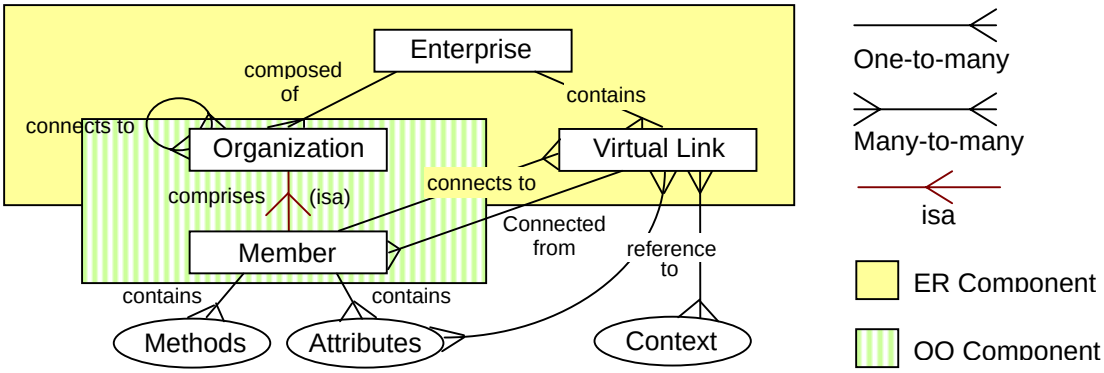


Figure 4-3 Diagram of the OMM Model

4.3 Organizations

When a modeller performs organization modelling over an enterprise, three phases are involved (Vernadat 1996). The first phase is to analyze the current state of the enterprise, and to project the expected state based on the input from management. The second phase deals with the conceptual design of various desired groups or units that make up the corporation. Both the analysis phase and the conceptual design phase are concerned with how a corporation is broken down logically into different components based on resource types, functions, management responsibilities or geographic locations. The third phase is known as design implementation. It takes the conceptual design from the second phase and implements the design with the underlying information system (Berio et al. 1995). We have discussed the concept of organization modelling and the details of these three phases in Chapter 2. In the conceptual design phase, OMM organizations give administrators flexibility in defining the various components of a corporation.

The conceptual entity of OMM organization is next defined.

Definition 4.1 (Organization) An *organization* in OMM is a resource type that exists in the enterprise. It represents a set of resources which share common characteristics.

Each OMM organization has a unique name across the entire enterprise. This name serves as an organization ID and is specified by the user at the time the organization is created. Examples of organization names include EMPLOYEE, DEPARTMENT, CONTRACT and MACHINE. The organization properties are described by attribute definitions, which can be mapped to column names in relational database tables, for example. The following table shows some sample attributes of these organizations.

OMM Organization	Organizational Attributes
EMPLOYEE	First_Name, Last_Name, Employee_Number, Department, Manager, Title
DEPARTMENT	Manager, Parent_Department
CONTRACT	Company, Type, Duration, Date_Signed, Value
MACHINE	Serial_Number, Make, Model, Last_Service_Date, Next_Service_Date

Table 4-1 Sample Attributes for OMM Organizations

In this example, each of the attributes can be implemented as a column in a relational database table. For example, in the case of the EMPLOYEE organization, First_Name, Last_Name and Department can be represented by three table columns, respectively.

As database tables can easily represent OMM organizations, our approach also simplifies the design implementation phase. Once we have applied OMM organizations to model the different resource types of an enterprise, the organization analysis phase can be performed easily by analyzing the OMM organization definitions and class hierarchy.

In our prototype, OMM organization is implemented in the `OmsOrganization` Java class (see Appendix A). Its properties include the organization ID, organization name, a vector that contains a list of other organization IDs that it relates to, and a vector that points to its organizational attribute definitions.

The next two subsections describe how the conceptual entity of OMM organization is applied to model various resource types of an enterprise.

4.3.1 OMM Organization Partitioning

In OMM, an enterprise is composed of a number of OMM organizations. These organizations are created to map the different resource types and components of a company. Each OMM organization has an identifier that is uniquely defined across the global enterprise. Using the OMM organizations, a company is partitioned first *horizontally* and then *vertically*. Horizontal partitioning divides the enterprise into

different resource types, such as employee, division, department, product and customer. Vertical partitioning is applied to break resources of the same type into smaller units, such as breaking employees into permanent workers, temporary workers and outside contractors. Figure 4-4 shows pictorially these two ways to partition an enterprise's resources using OMM organizations. For example, to support the workflow example described in Section 3.2, we need to model the enterprise resources of customer, employee, division and department.

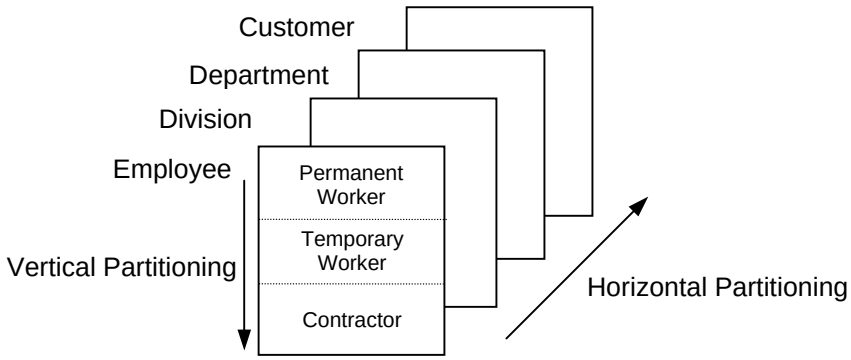


Figure 4-4 Horizontal and Vertical Partitioning of Enterprise Resources in OMM

During the conceptual design phase of organization modelling, one will first apply horizontal partitioning to define the different types of resources, followed by vertical partitioning to further refine the organization model. Referring to the running example in Chapter 3, we can define company resources using the OMM organization concept. For example, an OMM organization may be defined to represent the *employees* of the company, another to represent the *divisions*, and still another to represent the *departments* within each division.

Thereafter, we further divide the organization representing a resource type vertically. For example, employees who are permanent workers may be included in one organization, while temporary workers are placed in another. In summary, horizontal partitioning helps to define the different types of resources within the enterprise, while vertical partitioning allows users to logically divide resources of the same type into smaller sub-components.

To further illustrate how partitioning maps to the OMM model, we present another example. Suppose a company wants to capture all its documents as a resource type. The simplest way a modeller may adopt is to define an OMM organization, called *documents*, with attributes such as create_date, owner, type and file (which is an attribute pointing to

the path and filename of the actual document image for retrieval). However, since there are different types of documents in the company, the modeller may also perform vertical partitioning and define a number of OMM organizations such as *financial documents, performance review, business plans, customer contracts* and *legal documents.* Each of these organizations may share the common attributes as defined for *documents*, but they also uniquely possess attributes that are meaningful to them. For instance, *financial documents* has attributes like tax_code and fiscal_year; *performance review* has review_date and employee_ID; *business plans* has project_ID and department_code; *customer contracts* has company_name and contact_value; and *legal documents* has security_level. (The same obtains for methods.)

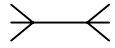
Note that partitioning is a methodology used in the organization conceptual design phase to model different components of the enterprise. Once partitioning is done, the result is a number of distinctly defined OMM organizations, each of which represents a certain component of the enterprise.

The idea of organization partitioning gives a high degree of flexibility to the modeller in describing the logical structure of an enterprise. It also gives greater ownership and autonomy to management. For instance, different divisions in a company may own their individual organizational definition; as a result, a much greater level of autonomy in defining and managing their own organizational information is granted to them. They can update, delete, or extend their organizational definition without impacting others. For major restructuring, administrators may alter the organization definition which corresponds to their units only. In addition, the granularity of partitioning is controlled entirely by the modeller; s/he has the flexibility to decide how fine s/he wants to divide the organization. When the business conditions change, a company may choose to either merge or further divide the organizations. OMM makes the merging and splitting of organizations simple. Chapter 7 discusses modularization and decentralization, which are common techniques used in organization reengineering. In that discussion, we will present the details of how OMM supports merging and splitting of organizations and provide some examples.

At the design-implementation phase, we will consider the underlying database schema that will be used to physically represent the OMM organizations. An OMM organization is described by defining a set of attribute names, which represent the common

characteristics of the resources within the organization. It is common to use some relational tables within a database environment to capture the definition of an OMM organization. The OMM methodology does not dictate the physical layer of the underlying data model, although our current prototype implementation uses an RDBMS as its repository. When a relational database implementation is chosen, users define the attribute names as column names in relational tables. If an object-oriented database is used, the attribute definition maps directly to the attribute definition of a class. Indeed, an ideal implementation is to use XML as the underlying data model to capture the resource definition and information such that enterprises can achieve interoperability along the value chain easily through Web Services (Kleijnin and Raju 2003; Stal 2002).

4.3.2 Relationships between Organizations

Resource types relate to one another within the business context of the enterprise. OMM provides a many-to-many relationship to connect organizations. The technique of connecting organizations is common in the business world to formulate how a resource type relates to another resource type that may or may not inherit a common set of attributes (refer to Figure 4-3). For example, for many companies, a department always exists under a certain division and within one of many departmental trees of the company hierarchy. To capture this knowledge in the organization model, we define a relationship to connect the DEPARTMENT organization with the DIVISION organization, and another to connect the DEPARTMENT organization with itself. Similarly, since each employee must be working in a certain department, a relationship is defined to connect the EMPLOYEE organization and the DEPARTMENT organization. Figure 4-5 illustrates the application of the notation  in Figure 4-1 to represent these relationships between the organizations.

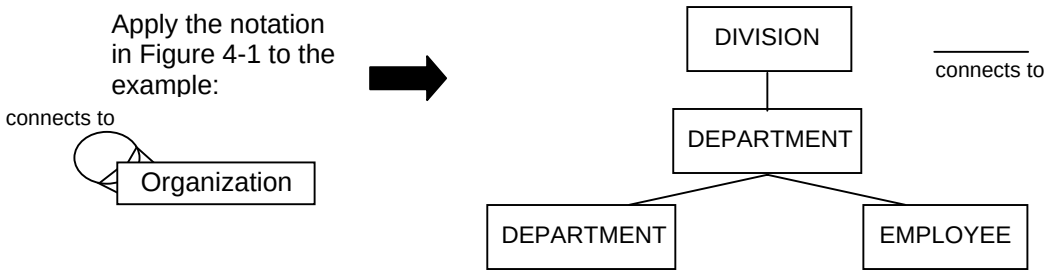


Figure 4-5 Relationships between OMM Organizations

Note that DIVISION only connects to one organization, namely DEPARTMENT, while DEPARTMENT connects to multiple organizations, namely DIVISION, EMPLOYEE

and itself. Finally, EMPLOYEE connects to DEPARTMENT. These *connects to* relationships help the person who is navigating through the model to easily find relevant and related resource types.

Unlike the virtual links between members that we will discuss in Section 4.5, the relationships between OMM organizations are static relationships that exist on the resource type level. They are represented simply by a relationship name and a pair of organization IDs. Given that the number of type-level relationships are few (usually $O(N)$, where N is the number of resource types in the enterprise) and the changes to type-level relationships are infrequent (once over several years or never after they are defined), it is sufficient to model them with a pair of static IDs. However, the number of instance-level relationships between members is larger than type-level relationships by an order of magnitude (usually $O(NM)$, where M is the number of instances within a resource type) and, moreover, instance-level relationships tend to have very rapid changes; it is therefore not practical to model them with a static approach. In Section 4.5, we discuss the challenge of modelling instance-level relationships and our dynamic approach to solve this issue.

4.4 Members and the Information Model

From the OO viewpoint, an OMM organization is a class, and each entity within the OMM organization is an object instance of that class. In OMM, we call this object instance the OMM *member*. These constitute a class of members for each OMM organization. In this section, we will discuss member objects, or simply members, in OMM, in the context of the information model that was introduced in Section 4.2.

Definition 4.2 (Member) A *member* in OMM is a conceptual entity within an OMM organization. It represents a resource instance within the enterprise and can be uniquely identified.

4.4.1 Object-Orientation of OMM

OMM employs an object-oriented information model to define the OMM members. The object-oriented approach assumes that the world is made of an organized collection of objects (Larman 2001). The fundamental construct in the OO approach is the object, which combines both data structure and functions in a single entity. Since we apply the

OO model to OMM members, we sometimes refer to a member instance as a member object, especially when discussing their OO nature.

Member objects in OMM are used to represent different enterprise object instances. Like objects in the OO model, OMM member objects may interact with one another via the exchange of messages. Messages are requests with or without data; messages interact with the methods, which are executable programs, associated with the member objects.

OMM uses the OO model to capture enterprise resource information. From the OO point of view, OMM organizations are the different classes representing the various resource types within the enterprise. Each OMM member, representing an instance of a resource type within the enterprise, is an object of a class. Figure 4-6 shows these two correspondences.

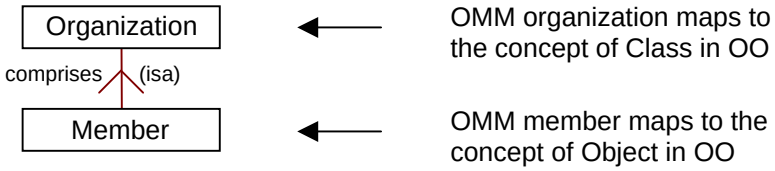


Figure 4-6 OMM Conceptual Entities Correspond to OO Concepts

OMM organizations and member objects observe the fundamental principles of OO, such as object identification, encapsulation, inheritance, and message passing. The concept of "class" in the OMM model is similar to the *Object Class* in the directory model of X.500 (CCITT 1988). OMM member objects are different from the X.500 members in that OMM supports class inheritance, method extension, and object life cycle.

All OMM member objects inherit a set of system attributes and methods from a system-defined superclass called `OmsMember` (see Appendix A for the definition of the `OmsMember` class). Figure 4-7 shows this simple hierarchy.

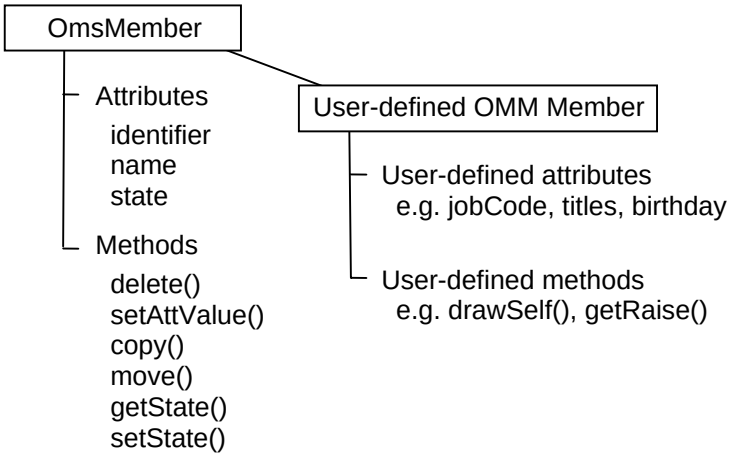


Figure 4-7 Hierarchy of the OMM Member

Note that although the attribute definition is associated with the `OmsOrganziation` (see Table 4-1) as discussed in Section 4.3, when a user invokes an API call to get a member object from OMM, OMM will retrieve the attribute values from the database, attach these values to the member object and return the member object to the user. The following code segment shows how this is done in Java by calling the OMM API (see Appendix A for the listing of OMM API):

```
OmsOrganization empOrg = new OmsOrganization("EMPLOYEE");
OmsMember memObj = new OmsMember(empOrg, "john_smith");
OmsObList valueList = memObj.getValList();
```

When the above call sequence is executed, `valueList` (a Java `Vector` object) will contain a list of attribute values for *john_smith*, who is a member object in the EMPLOYEE organization. Put in a different way, if EMPLOYEE was defined to have the attributes First_Name, Last_Name and Title, then `valueList` will have the values corresponding to these attributes (e.g. John, Smith and Engineer, respectively). In addition, the member object, `memObj`, will also contain an OMM system-generated object *identifier*, an object *name* ("john_smith"), and a *state* (initially set by default to *active* — the state transition of OMM members will be discussed in Section 4.4.2).

The attributes and methods listed under `OmsMember` in Figure 4-7 are extended from `OmsObject` (see Appendix A). The *identifier* attribute serves as the object ID, and is unique for each member object across the entire enterprise. This identifier is a 64-bit numeric string generated by the OMM system at the time the member object is created. In addition, each member object has an easily recognized *name* that is given by the user. This object name must be unique within the OMM organization to which the member

object belongs, but it does not need to be unique across multiple organizations. The *state* attribute represents the state within a life cycle that the member object currently holds (see Section 4.4.2).

Since our OMM Prototype System is implemented in Java and OMM organizations are implemented by Java classes, users can easily define Java methods to associate with OMM organizations. All user-defined organizations inherit from the system-defined superclass, `OmsOrganization` (see Appendix A 1.2 for its definition), a set of attributes and methods. We refer readers to Chapter 9 for the implementation details of the OMM Prototype System.

The full class diagrams of the OMM conceptual entities can be seen in Appendix B. The diagrams were constructed automatically from the Java code using BlueJ (BlueJ 2003).

When users define an OMM organization, they specify the attribute definition that is used to describe the member objects of that organization. Each user-defined attribute is used to describe a certain aspect of the member objects within the organization. An attribute has four properties: a name, a data type, a value constraint and a set of values. Note that an attribute may have zero or more values. In the OMM Prototype System, attributes are implemented as Java classes. The attribute definition associated with an OMM organization is dynamic; it can be augmented in its lifetime. There now follows the formal definition.

Definition 4.3 (Attribute) *Attributes* in OMM are properties of OMM organizations.

Member objects of the same OMM organization therefore share common attribute names although they can carry the same or different attribute values. For example, the Title attribute of a certain employee *John_Smith* may have the following properties:

Name:	Title
Data type:	Character string
Value constraint:	many-to-many
Value:	‘Engineer’, ‘Architect, ‘Technical Staff’

An attribute name can begin with one or more alpha characters followed by any number of alphanumeric characters:

Attribute Name ::= Alphabet^{+} Alphanumeric*
Alphabet ::= A-Z | a-z

Alphanumeric	::=	Alphabet | Numeric
Numeric	::=	0-9

Table 4-2 Syntax of OMM Member Attribute

Examples of attribute names include “Manager”, “Title”, “Price”, “Serial Number”, and “Last Service Date”.

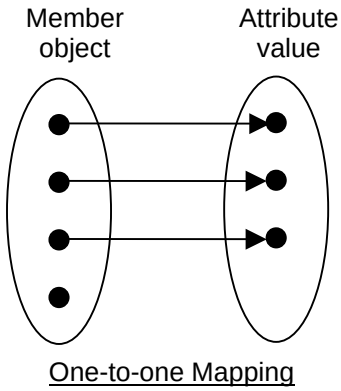
An attribute can have any one of five data types. The five supported data types in OMM are listed in the following table, with an example to capture the employee information of the running example in Chapter 3:

Data Type	Description	Example
Integer	Any whole number numeric value	Age in years
Float	Decimal number	Salary, Price, Cost
Character String	Alphanumeric	Title, Phone Number, Address
Date	Year, month, date or time format	Birthday, Deadline, Hiring date
Raw	Bitmap or bit stream	Picture, Document, Voice mail

Table 4-3 Supported Data Types in OMM

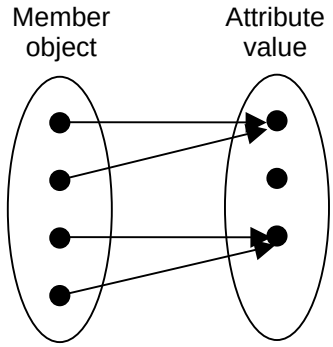
The attribute values of the OMM members are stored persistently either in the OMM repository or in existing corporate DBMS. The methodology of mapping OMM attributes to existing DBMS is discussed fully in Chapter 9.

Finally, the value constraints of OMM attributes define the mapping between member objects and their possible attribute values. Figure 4-8 depicts the possible mappings along with a description and example of each constraint.



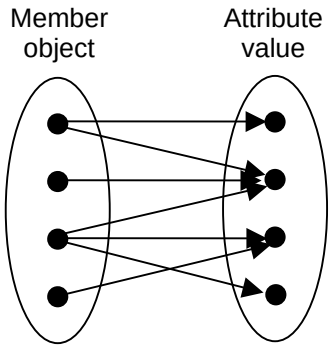
One-to-one Mapping

One-to-one mapping corresponds to unique attribute values where each member object will have at most one value on this attribute, and the attribute value is unique within the organization. E.g. Employee ID, Email address.



Many-to-one Mapping

Many-to-one mapping corresponds to single-valued attributes where each member object has at most one value on this attribute, and multiple member objects may share the same value. E.g. Office number, Birthday, Department, Manager.



Many-to-many Mapping

Many-to-many mapping corresponds to multiple-valued attributes where each member object may have any number of values on this attribute, and multiple member objects may share one or more of the values. E.g. Title, Ice cream flavour, Projects.

Figure 4-8 Supported Value Constraints in OMM

4.4.2 State Transition of OMM Members

Enterprise resources are going through a life cycle. They enter the company or get created at some point in time, stay active and provide their services to other resources, and eventually become consumed or are retired from the company. As OMM member objects are used to represent the entities in an enterprise, they also go through a life cycle, which is represented by the state transition diagram shown in Figure 4-9.

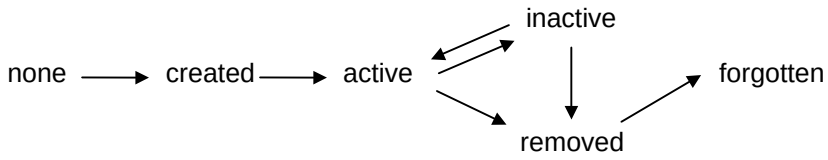


Figure 4-9 State Diagram of OMM Members

When an OMM member is created or instantiated, it immediately enters the *active* state. Thereafter the state changes are triggered by users through a system-defined method, `setState()`. An OMM member may cycle between the *active* and *inactive* states, thus simulating in reality some resources being suspended, on-leave, off-line, or in maintenance. Eventually an OMM member is *removed*. This means that although the resource will not provide any more services, or produce or consume other resources, its information is still retained in the system's repository and can be queried until it enters the *forgotten* state, which corresponds to the situation where the resource information is archived away or truly deleted.

The following table shows some examples of organizational resources going through different stages, and how they map to the life cycle of OMM members.

Resources	Created	Active	Inactive	Removed	Forgotten
Employee	employed	on duty	on leave	terminated	archived
Equipment	purchased	in service	maintenance	replaced	re-cycled
Support Centre	established	open	closed	shut down	destroyed
Facility	built	occupied	vacant	shut down	demolished

Table 4-4 Examples of Organizational Resources Going Through Life Cycle

4.4.3 Transfer of Member Objects between OMM Organizations

In OMM, member ownership can be transferred from one OMM organization to another. When a member is moved to another OMM organization, some of the user-defined attributes from the original OMM organization may be mapped to attributes of the new OMM organization; user-defined attributes that are not mapped are deemed irrelevant

information and are dropped. However, the system-defined attributes are always retained. This maintains the unique identity of the member object even though it may be moved between different OMM organizations in the enterprise. Since there is an `orgId` (organization ID) attribute associated to the `OmsMember` object (see Appendix A), once we changed the `orgId` value to represent the new ownership, the member object is transferred. The move operation of `OmsMember` is implemented in the Java method `move()` associated to `OmsMember` (see Appendix A).

Although membership transfer may happen between any two OMM organizations, it usually only happens when the user has applied vertical partitioning to divide the same type of resources into multiple OMM organizations; the transfer is usually between these OMM organizations. For example, OMM membership transfer can occur when a corporation has partitioned the employees into different business units. When an employee moves from one business unit to another, the OMM member representing this employee has to be transferred from the old OMM organization to the new.

Since member name is required to be unique only within an OMM organization, it is possible that a transfer will result in name collision. The user can then resolve the collision. Alternatively, an automatic process can be put in place so as to append a monotonically increasing number behind the name whenever a collision is found. For example, when employee *John_Smith* is transferred from the R&D Unit to the Support and Services Unit, which already has an employee by the name of *John_Smith*, the new member could be given the name *John_Smith_1*. If *John_Smith_1* already exists, then the name *John_Smith_2* will be used. The OMM System continues to increment the appended number until there is no name collision.

4.5 Virtual Links and the Relationship Model

In this section, we will discuss *virtual links*, the third conceptual entity in the OMM model, in the context of the relationship model that was introduced in Section 2.2. As collaborative efforts exist between company resources, it is necessary to model relationships between them (Roos and Bruss 1994; Scott-Morton 1990; Willcocks and Smith 1995). Many state-of-the-art collaborative groupware applications support the definition of object relationship — such features can be found in OVAL and ORM for example (refer to Section 3.6 for a discussion on OVAL and ORM). However, all of

them handle a relationship as a static, hardwired connection between two objects. This adds administrative overhead to information management — the user has to not only update the underlying resource information and profile, but also the constantly changing relationships between the resources.

OMM uses *virtual links* — a rule-based approach —to define dynamic relationships between member objects. The relationships are dynamic because they change when the underlying information of the related objects changes. Virtual links are rules expressed in computable expressions based on the member attributes and *contextual variables*. Contextual variables are variables understood by the OMM Prototype System. Basically these variables are similar to environment variables common to most operating systems. They are defined and maintained by users or programs through the OMM API. Examples of contextual variables include *$day_of_week, $initiator_of_flow, $system_load, $number_of_workers_on_duty*. Users are responsible for defining these variables and maintaining their up-to-date values by calling the OMM `set()` API. Once the contextual variables are defined in OMM, users can reference them when defining the virtual link rules. Virtual links are used to abstract roles and relationships among the company's resources. The OMM engine evaluates the virtual link rules at runtime to find out who plays what roles. By the same token, relationships between different OMM members are resolved at runtime by evaluating virtual links. Virtual links are implemented in the Java class called `OmsVirtualLink` (see Appendix A for the implementation details).

In OMM a relationship is established from one OMM member to another, and as such it can be represented as a directed edge. The OMM member from which the directed edge begins is termed the *owner* of the relationship edge. If a bi-directional relationship (such as the supervisor-subordinate relationship pair) is desired, it can be modelled as two relationships; one as a *reverse* relationship of the other (we will discuss bi-directional reverse relationships in Chapter 5). In this respect, company resource objects, or members, are like nodes in a digraph while virtual links are the arcs of the digraph. Figure 4-10 shows a digraph of how employee and department objects are connected to one another through virtual links.

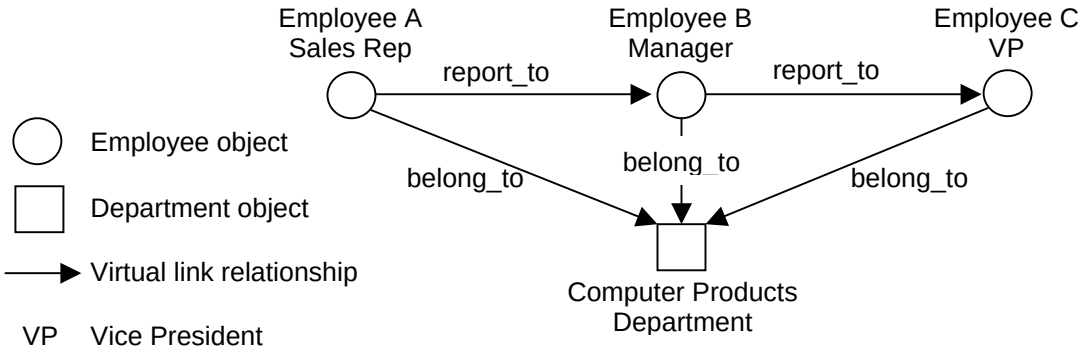


Figure 4-10 Digraph Showing Relationships Between Resources Using Virtual Links

For instance, regarding the Employee A member object, if we want to find out who this person reports to, we can simply call OMM to resolve the virtual link "report to" on Employee A. OMM will then return the Employee B member object.

The connections between resources are dynamic and virtual because relationships are defined via computable expressions over the attributes rather than via a pair of static resource identifiers. When a workflow management system queries OMM to resolve a relationship of a certain resource (e.g. the manager of John Smith), the relationship expression is evaluated over the current member attribute values and the referenced contextual variable values. Any number of resources may satisfy the said expression indicating that a relationship exists between the owner and these resources.

We will provide the syntax rules for defining OMM virtual links in Chapter 5.

4.6 An Example

We will use the running example in Chapter 3 to illustrate the conceptual entities of OMM in modelling a corporation. In this example, we model the departmental infrastructure and the reporting hierarchy by describing three resource types. Let U = {DIVISION, DEPARTMENT, EMPLOYEE} be the universal set of resource types in the corporation. The semantics of U are as follows: the corporation is divided into a number of DIVISIONs, under which there are a number of DEPARTMENTs. A DEPARTMENT may contain other DEPARTMENTs. Finally, there are EMPLOYEEs working in the various DEPARTMENTs. This enterprise can be represented by the OMM model. First, this enterprise is composed of three OMM organizations.

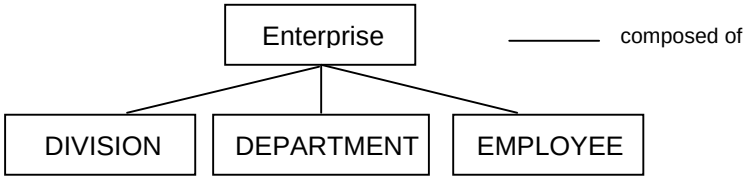


Figure 4-11 An Enterprise Composed of Three OMM Organizations

Note that the three OMM organizations are related to one another. We represent the connections between these three OMM organizations in Figure 4-12. (Please refer to Section 4.3.2 for a discussion of type-level relationships between OMM organizations.)

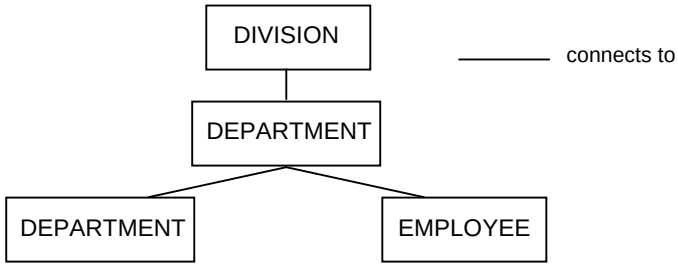


Figure 4-12 Relationships Between OMM Organizations

According to the OMM Model, members are instances within each OMM organization. For example, in the EMPLOYEE OMM organization, there may exist members such as *John_Smith, Mary_Ann* and *Tom_Hanks.* Each of these members shares common attributes such as *Title, Job_Code, Email, Department, Home_Address* and *Home_Phone.* Figure 4-13 shows that there are three members in the EMPLOYEE OMM organization.

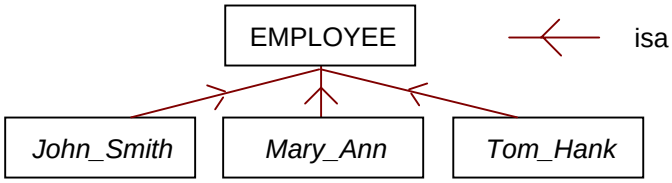


Figure 4-13 Three Members Exist in the EMPLOYEE OMM Organization

Note that EMPLOYEE is at the class or resource type level while John_Smith, Mary_Ann and Tom_Hank are at the object level.

The ensuing three tables show the class and the user-defined member attribute names of each of these OMM organizations. The "constraint" stands for the value constraints as described in Section 4.4.1.

Attribute Name	Data Type	Value Constraint
Manager	Character string	Many-to-one
Description	Character string	Many-to-one
Mission_Statement	Character string	Many-to-one
URL	Character string	Many-to-one

Table 4-5 Attribute Definition of the DIVISION OMM Organization

Attribute Name	Data Type	Value Constraint
Manager	Character string	Many-to-one
Description	Character string	Many-to-one
Parent_Division	Character string	Many-to-one
Parent_Department	Character string	Many-to-one

Table 4-6 Attribute Definition of the DEPARTMENT OMM Organization

Attribute Name	Data Type	Value Constraint
Employee_Number	Character string	Many-to-one
Manager	Character string	Many-to-one
Title	Character string	Many-to-many
Job_Code	Integer	Many-to-one
Email	Character string	Many-to-one
Department	Character string	Many-to-one
Location	Character string	Many-to-one
Office_Phone	Character string	Many-to-one
Fax	Character string	Many-to-one
Home_Address	Character string	Many-to-one
Home_Phone	Character string	Many-to-one

Table 4-7 Attribute Definition of the EMPLOYEE OMM Organization

Note that for each OMM organization, the *Name*, *UID*, and *State* are already included implicitly as system-defined attributes.

Finally, OMM members relate to one another through virtual links. For example, *Mary_Ann* may be *acting for John_Smith* during his absence. *Acting_for* therefore is a relationship between *Mary_Ann* and *John_Smith*. In OMM, in order to signify that *Mary_Ann* is *acting_for John_Smith*, we use the following virtual link:

Owner: 'mary_ann'
Relationship Name: acting_for
Expression: (name == 'john_smith')
Organization Scope: employee

We next present another example to illustrate the usage of virtual link. For instance, a company assigns different *Job_Code*s to employees to represent their responsibilities and to define the reporting hierarchy. For example, *Job_Code* 15 represents the

responsibilities of departmental staff, and *Job_Code* 120 represents the manager of a department. All people with *Job_Code* less than 100 report to the person with *Job_Code* 120 of the same department. The relationship of *departmental_manager_of* is then defined via the following virtual link:

Owner:	any employee
Relationship Name:	departmental_manager_of
Expression:	((Job_Code != $owner.Job_Code) AND (Job_Code == 120) AND (department == '$owner.department'))
Organization Scope:	employee

The concept of virtual link and its syntax are dealt with fully in Chapter 5; more examples of virtual link relationships will be given there also.

4.7 Summarized Features of OMM and Other OM Systems

In Chapter 3 we discussed existing approaches to organization modelling and pointed out their shortcomings with respect to satisfying the requirements of a dynamic collaborative application environment. We surveyed the organization-modelling components of state-of-the-art workflow management systems, directory services models and systems, stand-alone organization modelling systems and other office systems.

In the following tables, we compare and contrast these systems with the OMM in the areas of organization model, relationship model, process model and their ability to integrate with external systems. We first begin with OMM.

OMM	
Organization model	Generic reference model based on the OO and ER paradigms. Uses building block concepts which include organization, member and attributes to construct various types of enterprise resources. There is no restriction in modelling different types of enterprise and in describing different types of resources.
Relationship model	A rule-based model to abstract common relationships between resource objects into virtual links. Supports both static and dynamic relationships.
Process model	The process model is decoupled from the organization model. OMM does not have its own process model; it uses an open architecture to connect to other process management systems.
Integration with other systems	OMM provides a Java package as an open API to support integration with other OM systems and WFMS. Developers may also use this API to build applications and tools on top of OMM to define and manipulate resource objects.

Table 4-8 Organization Modelling Features of OMM

ARIS	
Organization model	Organization model focuses mainly on human resources. Users in ARIS are assigned to organization units which group together workers with similar job functions. The process model assigns workflow tasks to the organization units.
Relationship model	Supports the definition of inter-object relationships using a hardwired relationship model.
Process model	The process model is the focus of the ARIS system. It uses the concepts of data, function, and control to define a process-oriented information system.
Integration with other systems	Integration with other systems can be done through the back-end database. There is no open API to connect to external systems.

CIMOSA	
Organization model	Assumes the construct of the CIMOSA process model; uses fundamental entities, including resources, organization units and organization cells, to formulate the organization model.
Relationship model	Active resources are connected as agents to perform various functional operations (activities) within concurrent processes. There is no relationship model to connect resources outside of the process model.
Process model	Provides a strong framework to model business processes which focus on modelling CIM system life cycles.
Integration with other systems	Provides an integration infrastructure consisting of information services to support enterprise integration, application interoperability and model execution. However, this integration infrastructure aims to bring the various enterprise functional units together only under the CIMOSA model; there is no open interface mechanism to integrate with external systems.

EMS	
Organization model	Organizational structure is represented as a hierarchical tree of generic business units, which can be any functional units consisting of a set of resources performing business processes. The organization model is developed solely for the support of EMS's process model.
Relationship model	Users enter hardwired hierarchical relationships as resources in the business units. Users are associated to processes and activities for each business unit. There is no generic relationship model apart from the concept of the tree-structured business units.
Process model	Focuses on continuous process improvement and supporting decision-making alternatives. Emphasizes the needs of inter-departmental workflow.
Integration with other systems	No open API to connect to external systems.

M*OBJECT	
Organization model	An object-oriented model to describe the organization architecture on three levels: organization units, enterprise and enterprise environment. An organization unit is composed of work centres which connect to one another in a hierarchical structure.
Relationship model	Through the organization model, resources such as engineers and managers are defined. They are related to one another only through the work centres. There is no generic relationship model apart from the implicit relationships through association with work centres.
Process model	The process model is established around the organization model. At the work centre level, a user activity is defined as a work step within a business process. A process can be defined as a coordinated and partially ordered set of activities that fulfil an objective of an organization unit.
Integration with other systems	No open API to connect to external systems.

Objectflow	
Organization model	Able to define organization entities and relationships. Organization entities are mainly human resources and tangible resources such as printers and conference rooms that are consumed by human workers. A role model is tightly coupled to the organization model to support role resolution in workflow.
Relationship model	Uses a policy resolution model to define relationships between human resources. The policies specified in the relationships are computable expressions defined over the attributes of the resources. It does not take into account contextual information in resolving relationships.
Process model	Focuses on supporting long-lived and multi-user computations. A process is made up of concurrent steps, each of which is capable of associating with an agent application to perform certain business functions.
Integration with other systems	It has an open interface to allow external software applications to be associated to the workflow step. There is no open interface to connect to other organization or process modelling systems.

SAM*	
Organization model	An object-oriented model able to describe different types of enterprise objects. Enterprise entities are modelled by a network of inter-related concepts which can be physical objects, abstract objects, or events. A concept is further defined by a set of attributes.
Relationship model	Concepts (or objects) can be grouped together through different types of association. There are seven types of association in SAM* to represent different relationships between enterprise entities. Although set operations can be used to define associations, it does not support building relationships through flexible rule-based expressions.
Process model	SAM* does not have a process model; it also lacks an open interface or architecture to connect to other process modelling systems.
Integration with other systems	No open API to connect to external systems.

Table 4-9 Organization Modelling Features of WFMS

X.500, LDAP, NDS, Active Directory	
Organization model	A flexible object-oriented model to define different classes of enterprise objects which are in turn defined by attributes. A Directory Information Tree is used to layout the hierarchy of the object classes within a global enterprise.
Relationship model	Uses aliases to define associations between enterprise objects. Aliases are hardwired links represented as attributes of the objects.
Process model	No process model and no effort to interface with process modelling systems.
Integration with other systems	Open API is available for external systems to define and manipulate organizational entities. Requires additional programming effort to develop role resolution and other organization modelling-related interfaces on top of the open API.

Table 4-10 Organization Modelling Features of Directory Services Systems

ORM	
Organization model	Organization model decoupled from the process model allowing ORM to support various collaborative software applications. However, roles are defined simply as a type of resources within the enterprise; thus it fails to support dynamic roles.
Relationship model	Resource relationships are defined as hardwired connections using resource attributes. There is no dynamic relationship model.
Process model	The process model is decoupled from the organization model. ORM does not have its own process model; it uses an open architecture to connect to other process management systems.
Integration with other systems	Open API available for external systems to define and manipulate organizational entities, and to perform organization modelling functions.

OVAL	
Organization model	Semi-structured objects to define different classes of enterprise resources. Attributes and methods can be associated to different classes.
Relationship model	Uses hypertext links to define hardwired relationships between enterprise resources. Does not support dynamic links.
Process model	No process model and no facility to interface with process modelling systems.
Integration with other systems	Rule-based agents can be defined in OVAL to respond to certain events. Agents can further trigger other events which result in updating the objects in the database. This mechanism can be used indirectly to connect to other systems. No open API to connect to other systems.

Other Organizational and Office Systems	
Organization model	Either a rigid organization model or no well-defined organization model.
Relationship model	No well-defined relationship model other than some organization chart of graphical connections.
Process model	No process model and no effort to interface with process modelling systems.
Integration with other systems	No open API to connect to other systems.

Table 4-11 Organization Modelling Features of Stand-alone OM Systems

4.8 OMM Versus Other OM Systems

In contrast to existing OM systems, OMM separates the organization model from the role model of a BPM system (Bussler 1994; Cheng 1998; Mertins et al. 1997; Singh and Rein 1992). Figure 4-14 shows how the different models and components interface with one another. Interaction between each component in the OMM system is performed by using a set of published application programming interfaces. With the OMM approach, organizations are modelled separately from the business processes and applications. Role definition and resolution are done through the organizational modelling and management interface of OMM.

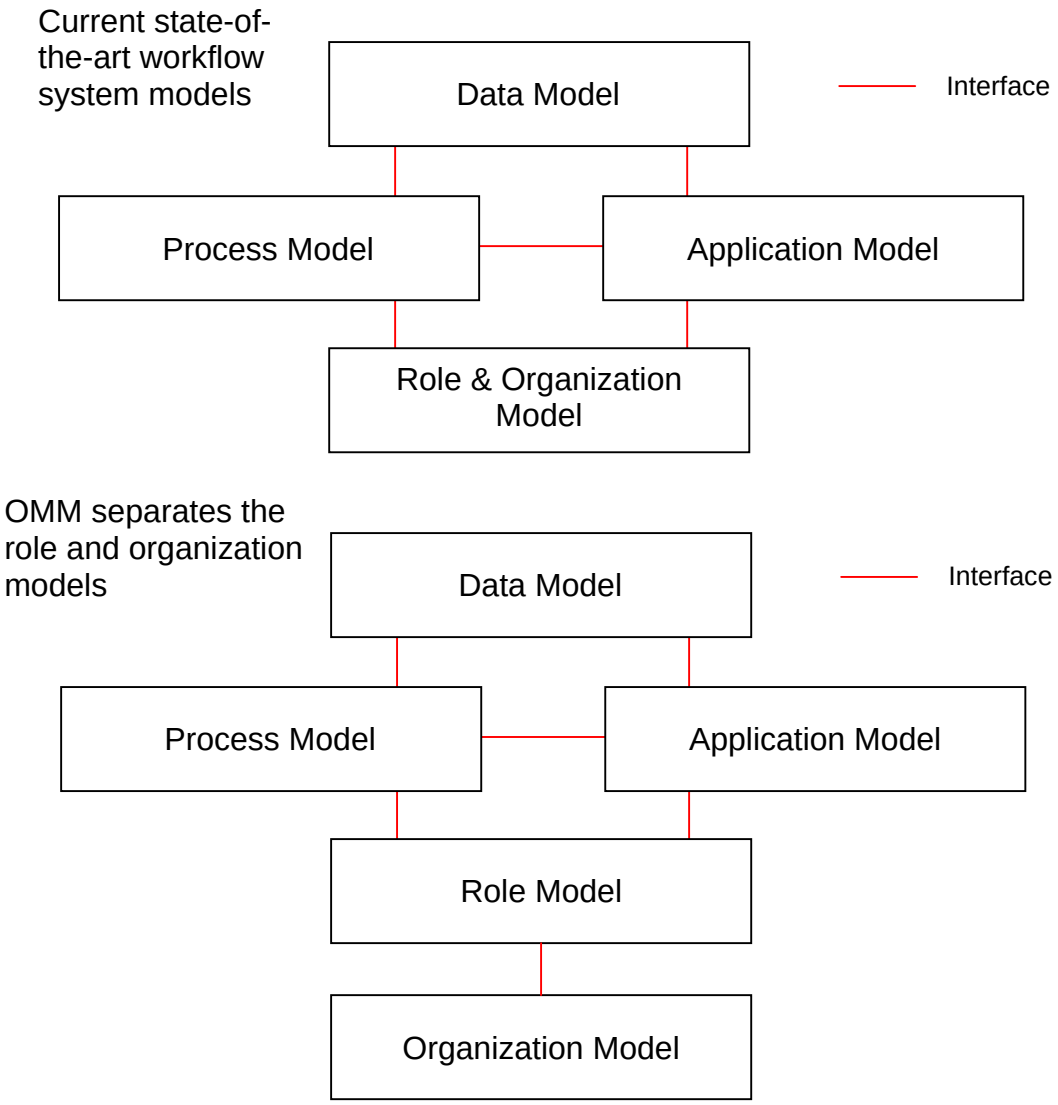


Figure 4-14 OMM Separates the Role and Organization Models

The separation is significant for the following two reasons:

1. The separation forces an open API to be published for the interface between the role model and the organization model. Any WFMS can be built on top of the organization model of OMM using the interface. In OMM, roles are defined on top of the generic organization model by using computable expressions. As a result, OMM supports context sensitive roles, which cannot be supported by existing OM systems, as evidenced by the exposition in the previous section.

2. The organization model of OMM is decoupled from the process model, which is used to represent operational sequences within the organization (refer to Section 2.2.2 for the definition of the process model). Thus the organization modeller can freely

design his/her conceptual enterprise without the limitation of viewing the enterprise solely from the process point of view. This gives flexibility in defining enterprise resources and their interactions in all three phases of organization modelling and reengineering. This contrasts sharply with the deficiencies, in this respect, of current organization models in WFMS identified in the previous section.

In addition, since OMM supports object life cycle that captures the dynamic real-world behaviour of a resource within a corporation, it allows the workflow system to properly perform *work-list* management. A work-list in WFMS can be viewed as a container of work tasks. WFMS uses a work-list to push tasks to different employees in the company. With OMM, based on the state of a resource object at runtime, the WFMS may choose to avoid pushing a task to an employee when s/he is not in the *active* state, consequently it reduces the possibility of assigning work to an employee who is currently unavailable.

4.9 Conclusion

In this chapter we have defined the conceptual model of OMM. OMM employs organizations, members, and virtual links to represent all types of resources and relationships in an enterprise. We have given definitions and provided examples to illustrate these OMM conceptual entities and how they are used to represent an enterprise.

Section 4.3 describes the conceptual entity of OMM organization and the partitioning technique of modelling to break down an enterprise into smaller manageable components. We discussed the horizontal and vertical partitioning approaches and outlined their benefits of flexibility and autonomy in enterprise modelling.

In Section 4.4 the object-orientated properties of OMM member objects are introduced. These include object identity, encapsulation, object class and inheritance properties. OMM members go through a life cycle which reflects the reality of enterprise resources. We also discuss the transfer of OMM members between two OMM organizations which happens quite often during company restructuring.

Section 4.5 is devoted to discussing the virtual link conceptual entity of the OMM model. Virtual links are used to represent dynamic roles and relationships in an

enterprise. They are a key innovation in the OMM model. When compared to previous efforts in organization modelling, the superiority of the OMM approach is clear; virtual links allow OMM to abstract hundreds of thousands of ever-changing relationships into a few dozens of rules. These are not the static relationship identifier pairs used by previous systems — these are dynamic virtual relationships that will adjust themselves as the underlying resource information changes. This reduces the overhead of maintaining static relationships between resource objects. We will present the syntax rules for defining virtual links and their applications in the next chapter.

Herein we have also concisely presented four salient features of organization modelling in order to compare and contrast OMM with other research work in organization modelling. These are organization model, relationship model, process model and the ability of the system to integrate with other external systems. Through this comparison, we demonstrated that OMM is more suited to support an open collaborative computing environment.

In the next chapter we present the relationship model of OMM, and in Chapter 9 we will present the OMM API, which allows applications and software tools to interact with the OMM Prototype System.

CHAPTER 5 The Policy-Based Relationship Model in OMM

The OMM organization model presented in Chapter 4 provides a flexible object-oriented approach with which to describe the organizational entities of an enterprise. However, even when we clearly lay out thousands, or hundreds of thousands, of resource objects, and how they are categorized into different resource types, it is unclear how these resource objects interact with one another. The connections between resource objects are critical to an enterprise; knowledge of how they function indicates how resources work together to accomplish a set of common goals. Connections can be used to describe the data flow between different entities (such as document links between the engineering department and the quality assurance department) and to give a functional representation of a decision-making hierarchy (such as links between sales and service regions, operating companies and headquarters). Modelling an enterprise without capturing the dynamic inter-connections within it would be like architecting a complex software system without specifying the interfaces between the different components.

In this chapter we describe the relationship model in OMM. In Section 5.2, we provide a formal definition of OMM's *virtual links*. In Sections 5.3 and 5.4, we show how to apply virtual links to model dynamic roles and relationships. Sections 5.5 and 5.6 discuss the bi-directional and transitivity properties of virtual links, which are important concepts for advanced applications. In Section 5.7, we briefly describe the Publish and Subscribe paradigm, and in Section 5.8, in order to demonstrate the adaptability and flexibility of the relationship model of OMM, we present very briefly a prototype, called OMM/P&S, designed to support the e-business of an American insurance company, called InsurePoint, in the context of Publish and Subscribe technology. We conclude our discussion of enterprise relationships and roles in Section 5.9.

Herein (resource) objects and resources are used interchangeably.

5.1 Introduction

The OMM relationship model defines how OMM members relate to one another. (Recall from Chapter 4 that members are used to represent resource objects in an enterprise.) Existing organization models either do not support the notion of

relationship, such as the X.500 directory information model, or simply store relationships between objects as "hardwired" records, such as in OVAL (Cheng 1998). The fact that such relationships are defined independently of the attributes of the resource objects incurs additional administrative overheads in order to maintain these relationships. In a hardwired relationship model, whenever the attributes of an object are changed in such a way that its relationships with other objects are affected, in addition to having to update the attribute information of the object, we have to update the relationship information as well. Moreover, static connections fail to support collaborative computing across the entire enterprise or in workflow management systems, because in business process automation a work task often needs to be assigned to an agent who is related to another agent that had executed a previous task in the business process. If the underlying organization modelling system is not able to dynamically define up-to-date connections between resources, once the relationships between steps in a business process change, the task assignment will fail to route the work task to the right resource for execution.

In OMM, instead of creating static links between two resource objects, a relationship is defined as a function of the attributes of resource objects. Computable expressions are used to abstract and group together complex but similar relationships within a large enterprise. Examples of relationships between resources include those of supervisor and subordinate, project team members, products and support personnel, customer accounts and sales representatives, and many others. Actual connections between any two resource objects are resolved at runtime by evaluating the computable expression over the object's attribute values. Since the connections are not simply hardwired between two object identifiers, but are defined by a computable expression and resolved at runtime, we call such connections *virtual links.*

There are several advantages to modelling relationships with virtual links rather than hardwired ones.

First, the connections between the resource objects are dynamically changed and self-maintained. As the attributes of a resource object change, the relationships between this object and other objects will also change. This correctly reflects the reality of the enterprise: relationships between resource objects are changing because the conditions, or the attributes, of the objects frequently change. For instance, when a person moves from one department to another, once his department assignment (the attribute value

reflecting his department name) is changed, his relationship with his old department (as a departmental member) no longer exists.

Second, representing a relationship as a function of underlying attributes obviates the need to maintain hardwired relationships. In X.500, relationships are defined as aliases, which record in a resource object the UUID of another object that they relate to. With this approach, when a person moves from one department to another, we not only have to update his personnel record to reflect the department change, but also inspect all his relationships and update those that are affected by his departmental transfer.

Third, the process of formally abstracting relationships into company policies provides a methodology and a model for the enterprise to strategically organize its workforce. Changes within the company can be more strategically designed, widely communicated and filtered down. In the absence of a virtual-link based system, individual groups or departments are called upon to define their own connections between resources. Indeed, due to the overhead of maintaining such rapidly changing information, in most cases companies do not even store formal records of relationships except through some unstructured administrative means such as printing organization charts or drawing data flow diagrams. With OMM, companies may define and enforce policies over the entire enterprise to describe the different roles that resources play and how resources relate to one another in different business contexts and operations. As a result, organization analysis can be performed to review how people are fulfilling their roles and how resources are related to one another. Analysts can also measure success and evaluate failure based on the actual interactions within these relationships.

Finally, the concept of dynamic relationships can be used to model dynamic roles. Since the abstraction model is a well-defined architecture with open interfaces, it allows other collaborative software such as WFMS, document management systems and e-commerce applications to run on top of it. Using OMM, groupware applications can flexibly perform dynamic role-based document routing, dynamic task assignment, and role-based access control (Cheng 1999a). A detailed discussion and examples of role-based routing, task assignment, and access control can be found in Chapter 6.

5.2 Virtual Link Definition

OMM uses *virtual links* to define dynamic relationships and dynamic roles.

Definition 5.1 (Virtual Link) A *virtual link* is a relationship type between a source OMM organization O and N target OMM organizations O_1, O_2, …, O_N, where $N \geq 1$.

A virtual link has the following syntax (Cheng 1997):

<Virtual Link>	::=	[Owner], <Relationship Type> , <Expression>, <Organization Scope>
<Owner>	::=	null | <Member ID>
<Member ID>	::=	<Character String Constant>
<Relationship Type>	::=	<Relationship Name> [REVERSE <Relationship Name>] [TRANSITIVE]
<Relationship Name>	::=	<Character String Constant>
<Expression>	::=	(<Expression>) <Log Op> (<Expression>) | <Attribute Name> <Comparison Op> <Value> | <Contextual Variable Name> <Comparison Op> <Value>
<Attribute Name>	::=	<Character String Constant>
<Comparison Op>	::=	== | != | >= | > | < | <=
<Value>	::=	<Constant> | <Attribute Name> | <Contextual Variable Name> | $owner.<Attribute Name>
<Contextual Variable Name>	::=	$<Character String Constant>
<Log Op>	::=	AND | OR | NOT
<Organization Scope>	::=	<Organization Name>$^+$
<Organization Name>	::=	<Character String Constant>

Table 5-1 The Syntax of an OMM Virtual Link

Note that the precedence of logical operators (AND, OR, NOT) is determined by parentheses ("(" and ")") from inside out. In other words, the Expression within the innermost parentheses is resolved first, before resolving Expressions enclosed by outer parentheses.

A virtual link is composed of an *owner*, a *relationship type*, a computable *expression* (or simply expression), and an *organization scope*. The owner of a relationship type can be

given as part of the definition. However, in most situations, the owner is selected at runtime when the underlying WFMS resolves a relationship of a certain resource. A relationship always has a name. It may also be a *reverse* or *transitive* relationship. These two properties are dealt with in Sections 5.5 and 5.6. The computable expression is the key element of a virtual link and specifies how the owner relates to other resources. The organization scope is a list of OMM organizations to include the types of resources that the owner is connected to through this relationship. For example, the *current_project* relationship may relate an employee to the PROJECT organization; similarly, a *project_resource* relationship may connect a project to the EMPLOYEE, MACHINE and ROBOT organizations. Finally, the computable expression specifies the conditions of the relationship. It is defined using OMM member attributes and contextual variables.

A contextual variable has a *context name*, which is a user-defined name representing some contextual information of the operating or application environment. *$day_of_week, $flow_initiator* and *$max_workload* are some examples of context names. Users define and set the value of context names through the OMM Application Programming Interface (API).

In OMM, a virtual link is defined by a computable expression. When the virtual link expression is instantiated by a certain OMM member of the source OMM organization (we call this member *owner*), we can resolve that expression and obtain a set of OMM members of the target OMM organizations.

Definition 5.2 (Link) A *link l* in OMM is an instance of a relationship type obtained by resolving a virtual link between an OMM member (called *owner*) in the source OMM organization O and one or more target OMM organizations O_1, O_2, …, O_N, where $N \geq 1$.

5.3 Applying Virtual Links to Model Dynamic Roles

The organizational role has been created as a way to assign responsibility and capability within organizational units. An organization provides the environment within which a role is meaningful. Earlier research has treated roles as a label associated to human resources (Sandhu and Munawer 1998), or has modelled roles as objects (Goh and

Baldwin 1998). Moffett (1998) in his work on defining role hierarchies discusses the need for having *virtual roles* in addition to ordinary roles. In Moffett's conception, virtual roles are defined to abstract the commonality of existing roles. For instance, Project Member is a virtual role, constructed to capture the commonality between Test Engineer and Programmer. A virtual role is a superclass over two or more underlying roles, which are primarily role labels with a set of responsibilities. All of these approaches assume the simplistic view that employees within the enterprise play a relatively small number of static roles. However, in reality, resources, especially human resources, usually play different roles within the enterprise, carrying out various corporate functions at different times and under different contexts. In many corporations, roles are defined as titles, each of which connotes certain privileges, authorities, responsibilities and functions. For example, an employee might be playing the roles of a *project manager* within a department and an *architect* on a corporate-wide architectural review board. In his first role as a manager, he may perform project reviews and sign off on project specifications as part of an ISO 9000 certification process. In his second role as an architect, he may be allowed to access classified documents only available to the architectural review board. The two roles carry very different responsibilities and authorities, but are played by the same person under two different contexts.

A *static role* is a label, which is usually implemented as an attribute of the resource object playing the role, although it is possible to implement a static role as a separate object rather than simply as an attribute. The latter approach allows the use of a set of attributes to describe a role's responsibilities and other properties, but does not change the nature of the association of the role to the person playing the role. This association will not change even when the person's attributes have changed. It is thus still static in nature.

Dynamic roles refer to role definitions that are linked or associated with organizational entities through some *dynamic conditions*. With a dynamic role, a person's role is not a static label, but is determined by resolving the criteria associated with the role over the current state of the person's personal attributes and/or some contextual variables.

OMM virtual links can be used to model dynamic roles. Rather than relating two or more OMM members through the relationship expression as we have discussed in

Section 5.2, the expression for dynamic roles will be defined based on one's attributes and/or contextual variables. In addition, a dynamic relationship is a directed edge connecting two nodes (Section 4.5) while a dynamic role only involves one node. As a result, there is no need to specify the *owner* in the dynamic role definition.

Referring back to the electronic parts ordering process example in Section 3.3, the *shipping* step can be processed by an employee in the shipping department. The step definition of the workflow can be represented by the expression (Cheng 1995; Hsu et al. 1991):

{… step definition …} Executed by *shipping_clerk*,

where shipping_clerk is a dynamic role which can be defined in OMM as a virtual link:

Role Name:	shipping_clerk
Expression:	(department == 'Shipping') AND (title == 'Clerk')
Organization Scope:	EMPLOYEE

With this role definition, whoever in the company whose department attribute equals 'Shipping' and title attribute equals 'Clerk' will be playing the role shipping_clerk. If someone who is currently playing the shipping_clerk role is transferred to another department, and thus changes the department attribute to a different value (such as Accounting), he will cease to play his role as a shipping_clerk.

Through this ability to abstract role definitions into computable expressions, dynamic roles give the company greater flexibility in managing role changes. For instance, to allow access to valuable resources in the shipping department only when an employee is on duty, shipping_clerk may be redefined as:

Role Name:	shipping_clerk
Expression:	(department == 'Shipping') AND (title == 'Clerk') AND (onDutyDays == $today)

where onDutyDays is an employee attribute indicating the days of the week that an employee is on duty. $today is a contextual variable defined and stored in OMM; its value is set by a method that is provided by users.

5.4 Applying Virtual Links to Model Dynamic Relationships

So far we have defined the syntax rules for defining virtual links. We have also shown how virtual links are used to model dynamic roles. This section will show how virtual links are used to model dynamic relationships.

Similar to dynamic roles, dynamic relationships are defined using a condition or computable expression over some attributes and contextual variables. As the underlying conditions change, the relationships between enterprise resources also change. An example of a dynamic relationship defined by an OMM virtual link follows:

Owner:	null
Relationship Name:	manager_of
Expression:	(deptNo == $owner.deptNo) AND (jobCode < 101) AND ($day_of_week != 'Sunday')
Organization Scope:	EMPLOYEE

Applying this dynamic relationship to find all employees that are managed by john_smith, we can resolve this virtual link over the EMPLOYEE organization. In this case, the owner is set to 'john_smith', its attribute values are retrieved and used to substitute corresponding fields ($owner.deptNo) in the virtual link expression. Each member within the EMPLOYEE organization is evaluated against the virtual link expression; the attribute names (deptNo and jobCode) are replaced by the corresponding attribute values of the OMM member object under consideration.

By means of virtual links a WFMS can flexibly assign and authorize steps in a business process, using available resources. As an example, we will return to the e-commerce ordering process introduced in Section 3.3. The roles in this example are:

- "sales rep for the company to which the flow initiator belongs"; and
- "vice president of the division to which the sales rep of the process instance belongs".

They can both be expressed as virtual links. The definition for the sales rep is:

Owner:	$flow_initiator
Relationship Name:	company_sales_rep
Expression:	(company == $owner.company) AND (title == 'Sales')

Organization Scope: EMPLOYEE

Analyzing this relationship for a given customer (the flow initiator) from the company $owner.company, OMM will return the corresponding sales rep from the EMPLOYEE organization when resolving the company_sales_rep virtual link for this customer. The definition for the division vice president (VP) is:

Owner: the sales rep who executes *process_order*
Relationship Name: division_VP
Expression: (division == $owner.division) AND (title == 'VP')
Organization Scope: EMPLOYEE

The flow engine of the WFMS has to retrieve the information of "the sales rep who executes process_order." This knowledge should be kept as part of the workflow system data (Cheng 1995; Hsu et al. 1991). Section 6.2 provides a detailed discussion of role resolution in workflow. Once the owner of this virtual link is identified, OMM can evaluate the expression and return the VP of the division to which the sales rep belongs.

Despite the dynamic characteristics of relationships in OMM, hardwired relationships between two specific entities can still be modelled with virtual links. For example, to define that Mary Ann is acting_for John Smith, we can use the following virtual link:

Owner: 'mary_ann'
Relationship Name: acting_for
Expression: (name == 'john_smith')
Organization Scope: EMPLOYEE

5.5 Bi-directional Relationships

When defining a relationship type, a *reverse* relationship can be specified (Cheng 1999a). For example, if relationship types R_1 and R_2 (between entity types E_1 and E_2) are defined as reverse of each other, and if OMM member object $m_1 \in E_1$ relates to another OMM member object $m_2 \in E_2$, then m_2 relates to m_1. (See Definitions 2.3 and 2.4.)

Figure 5-1 shows a relationship graph within an OMM organization; note that here relationships supervisor_of and subordinate_of are represented by reverse links of one another:

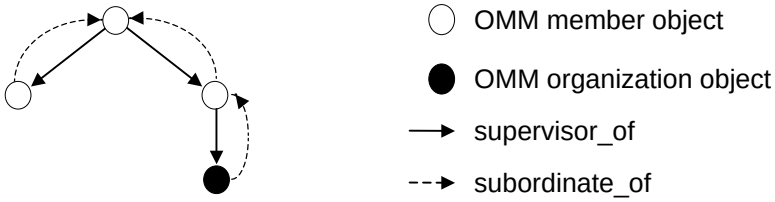


Figure 5-1 OMM Relationship Graph Showing Reverse Relationships

Although the example only covers relationships within a single OMM organization (EMPLOYEE), virtual links can actually be defined across multiple OMM organizations. In such a case, the organization scope will list all OMM organizations involved. For instance, a relationship digraph may be used to represent the connections between a project and its machine resources and the employees who are involved in the project. In this case, the owner is a particular *project* while the organization scope will include both MACHINE and EMPLOYEE.

5.6 Transitivity of Virtual Links

Note that a virtual link may be *transitive* in nature. An example of the transitive property in virtual links is the supervisor relationship. Figure 5-2 shows a digraph representing such a relationship:

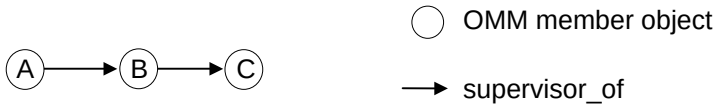


Figure 5-2 The Supervisor Relationship Showing the Transitivity Property

In this example, A is the supervisor of B, and B is the supervisor of C. If supervisor_of satisfies the transitivity property, then it follows that A is also the supervisor of C.

Obviously, because of the recursive operations involved in resolving transitive relationships, depending on the size of the digraph and the size of the organizational database, there can be a high computational cost associated with resolving transitive relationships; they should therefore be used with care.

5.7 Publish-and-Subscribe

Publish-and-Subscribe (P&S) is a form of information push technology (Cheng and Loizou 2000; Berghel 1998; Franklin and Zdonik 1998) that concerns mainly the flow of information from a set of resource producers to a set of resource consumers. P&S allows consumers to pre-declare their interests and situation. When some relevant information is published, the P&S system automatically pushes the resource to the subscribed members. In addition, a P&S system may also monitor the subscribers so that when their subscriptions change, readily published information may be pushed to them. As such, P&S is not only concerned with the push of large amounts of data over the network, but also specifically deals with the issue of matching publisher or the published resources with the subscriber.

OMM supports P&S by first allowing flexible modelling of both publishers and subscribers with an object-oriented organization model. Note that in an e-commerce environment, there could be many types of publishers and many types of subscribers, and that OMM is capable of including them all. Furthermore, OMM uses a policy-based relationship model to support the P&S system by enabling the definition of matching criteria between publishers and subscribers. Every time a resource is published, the P&S system will query OMM to identify the matching subscribers and push the resource to them.

5.8 An OMM/P&S Prototype

As part of our research effort to support P&S in e-commerce, we have built a research prototype P&S application using OMM to support an Internet-based commercial insurance company, InsurePoint, which was sponsored by Atlantic Mutual Insurance Co. with the aim of conducting e-commerce entirely on the Web. The purpose of the OMM/P&S prototype is to verify our claims that OMM is able to automate P&S when a variety of information is published and pushed over the network to selective customers. We also would like to observe customer responses to the P&S solution as well as issues that may arise in such an environment.

InsurePoint has about 150 customers all classified as hi-tech, start-up firms (nationwide, under 25 employees). InsurePoint sells to clients various insurance policies including

general liability and workers' compensation. It has always maintained the most current customer profiles. Indeed, customers can simply logon to their personal page on InsurePoint's Web site and update their own profile. In the past, account managers at InsurePoint periodically queried the profiles database to identify further business opportunities with existing clients. InsurePoint management would like to enhance customer relations and to maximize new business opportunities using P&S technology. Therefore, InsurePoint has two immediate objectives. First, they want to publish periodic news briefs which would help their clients better manage their risks in doing business; this would help build customer loyalty. Second, they would like to automatically push relevant new product information to clients who have updated their profiles. Notification should also be sent to the relevant account managers so that they can follow up with the customers.

With OMM/P&S, we modelled 4 OMM organizations: CUSTOMER, PROSPECT (people who asked for a quote over the Web), POLICY_DOC (documents describing insurance policies), and NEWS (articles about events that impact risks the customers or prospects are facing). Virtual links are defined to capture the relationships between these OMM organizations (classes). Figure 5-3 shows these organizations and their organizational relationships:

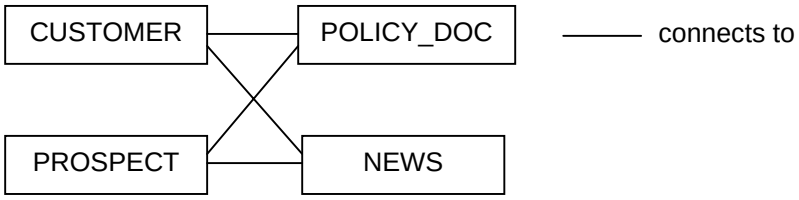


Figure 5-3 OMM/P&S Organizations for InsurePoint

About a dozen attributes are defined for each class. In the interests of brevity and simplicity, we list below only those attributes that are used to define the virtual links:

Organization	Attribute Name	Data Type	Value Constraint
CUSTOMER	asset	Integer	Many-to-one
	propertyCoverage	Integer	Many-to-one
	projectedSales	Integer	Many-to-one
	liabilityCoveredSales	Integer	Many-to-one
	region	Character string	Many-to-many
	regionCoverage	Integer	Many-to-one
	businessType	Character string	Many-to-many
PROSPECT	region	Character string	Many-to-many
	businessType	Character string	Many-to-many
POLICY_DOC	policyType	Character string	Many-to-one
	liabilityCoveredSales	Integer	Many-to-one
	region	Character string	Many-to-many
NEWS	region	Character string	Many-to-many
	businessType	Character string	Many-to-many

Table 5-2 Attribute Definitions of InsurePoint Organizations

Subscribers in this case can either be customers or prospects. Customer profiles are captured when the customer purchases a policy from InsurePoint. Prospect profiles are obtained when potential clients ask for quotations of insurance policies by filling out a form over the Web. Also customers using their user name and password can update their profiles to provide their latest situation anytime on the Web.

Whenever a customer profile changes, the P&S system evaluates the following virtual links over the policy document class to decide if certain documents should be pushed to the client. InsurePoint has identified several conditions (or business rules) that should trigger the push of information to customers. We list them below along with the policy (virtual link) that helps to discover such conditions:

1. When a customer's projected sales increase beyond the recommended liability coverage.

   Owner: a CUSTOMER member object
   Relationship: new_liability
   Expression: (liabilityCoveredSales >= $owner.projectedSales)
   AND (policyType == 'GENERAL LIABILITY')
   Organization Scope: POLICY_DOC

2. When a customer's assets grow beyond the current assets coverage.

   Owner: a CUSTOMER member object
   Relationship: new_asset_policy
   Expression: (propertyCoverage >= $owner.asset) AND
   (policyType == 'PROPERTY')

Organization Scope: POLICY_DOC

3. When a customer expands to a new region.

Owner:	a CUSTOMER member object
Relationship:	new_region
Expression:	(region == $owner.region) AND (region != $owner.regionCoverage) AND (businessType == $owner.businessType)
Organization Scope:	POLICY_DOC

In all three cases, the organization scope of the virtual link is the POLICY_DOC organization while owner is an OMM member of the CUSTOMER organization. When a customer profile is updated, each of these business rules will be evaluated by OMM/P&S, and documents found to satisfy these business rules will be sent to that customer electronically. Users can also add other business rules to include new conditions for pushing information when a customer profile is updated.

In addition to triggering push by customer update, InsurePoint can also initiate pushing of information by publishing a news item. When a news brief is published, the author will profile the news item through a Web interface. The *region* attribute signifies whether the item is location-sensitive; the *businessType* attribute indicates whether the content concerns a specific industry such as Consulting, Electronics, Manufacturing, Pharmaceuticals and others. Once the news item is published, OMM/P&S evaluates the following business rule to decide who among the customers and prospects should receive this news brief:

Owner:	a NEWS member object
Relationship:	matching_news
Expression:	(region == $owner.region) OR (businessType == $owner.businessType)
Organization Scope:	CUSTOMER; PROSPECT

5.9 Conclusion

In organization modelling, the modeller has to analyze the organizational resources and how they interact with one another. In this process, it helps to understand the decision-

making policies and processes at various levels of the enterprise. When the time arrives to conceptually design the new organization and to improve the performance of the organization's operations, the modeller has to refine the roles that each resource plays, and to enhance the communication, coordination and collaboration between the resources. Finally, to physically implement the new organizational design, it is important that the modeller be able to translate the conceptual roles and relationships of the model into formal specifications. Ideally, these specifications will be able to adjust themselves as the business conditions change. Otherwise the company will have to pay a high cost to maintain and manage dynamic aspects of the implementation, such as roles and relationships.

In this chapter, we have discussed how the dynamic relationship model of OMM can define dynamic roles and relationships, which are necessary to support organization modelling. This is realized using the OMM virtual link, which is the policy-based relationship model of OMM. With the concept of dynamic roles and relationships, we enable the organization modeller to formally abstract roles and relationships into computable expressions, thus reducing the tremendous administrative overhead involved with maintaining static links. The OMM approach also allows collaborative software to route data and workflow across an enterprise much more flexibly and realistically than is possible with static link approaches.

We have provided the syntax rules for defining OMM virtual links. We have also given some examples to show how virtual links are used to model dynamic roles and relationships. Finally, an OMM/P&S prototype with the InsurePoint Company is very briefly described. This exhibits the adaptability and flexibility of our approach.

Overall, virtual links simplify the definition and maintenance of dynamic roles and dynamic relationships in an enterprise.

CHAPTER 6 Role Resolution in Workflow Management Systems and Other Cooperative Applications

The OMM organization model was introduced in Chapters 4 and 5. We will now apply this model to address role resolution in WFMS and other cooperative computing systems.

In this chapter, we will describe the challenge of role resolution in an enterprise-wide, production-level workflow environment. In Section 6.2, we define the concept of role resolution, task assignment, task authorization, and routing control in the context of workflow management. In Section 6.3, we describe how dynamic roles, as defined in OMM, are applied to overcome the challenges of role resolution. In Section 6.4, we discuss the application of dynamic roles to role-based access control and show how OMM can more flexibly support RBAC than a hardwired role model.

6.1 Introduction

WFMS and other cooperative software, such as document management systems, voting software, group discussion applications and conferencing software, are forms of group computing that enable a number of parties to participate and interact with one another. Contrary to traditional personal computing, which involves only the individual, group computing involves multiple users who share a set of computing resources over a network. In certain applications, such as workflow, and document routing and management, an ordering is imposed on processing. Conversely, note-based discussion software, for example, does not impose such an ordering; all authorized users can participate at their convenience. When ordered processing is required, the group computing system may assign a certain task to a role, allowing a subset of users who play that role to perform the task. However, whether ordered processing needs to be enforced or not, the concept of role is important in controlling access to the common resources of group computing.

As discussed in Chapter 5, there are different approaches to defining roles for a group-computing environment. The simplest way is to associate role labels with different users. This static approach, although simple, lacks flexibility in allowing users to play

different roles under different contexts. Dynamic roles, as defined in OMM, enable a user to change roles when his/her situation, or the business context in which he/she is operating, changes. The open interfaces in OMM means that role resolution can be performed in WFMS and other groupware running on top of OMM. A detailed discussion of dynamic role versus static role systems was presented in Chapter 5.

6.2 Role Resolution in Workflow Management Systems

Concurrent engineering technology supports business process integration and automation (Cheng 1995; Medina-Mora et al. 1992), and provides a framework within which a business process consisting of multiple tasks and applications can be accomplished by integrating the various tasks in a network of steps (Vanderaalst and Vanhee 1996). A workflow process can be modelled as a digraph, where each node is a task (which we will also refer to as a workflow step or simply as a step) and the edges are condition arcs governing the route of the process (refer to Figure 3.2 for a Petri Net representation of a workflow process). We refer the reader to Section 3.3 for a detailed discussion of modelling a workflow process.

During the execution of a workflow process, different work tasks are created and assigned to various resources in the company. Sometimes a particular resource may be chosen to execute a step (the *push* model). Conversely, a group of employees may be identified as potential candidates to perform a task. In this case, one of the candidates will (attempt) to execute the task on their own initiative (the *pull* model). In both cases, authorization checking must be performed when someone attempts to open and work on a workflow step. To allow flexibility in workflow authorization, the WFMS often adopts a *role model*. A role model describes how roles are defined and how different resources are associated with one or more roles within the company. With the abstraction of a role model, the WFMS simply resolves roles in order to perform task assignments and task authorization, and to make routing decisions.

Role resolution refers to the process of identifying the resources within a system that are playing a certain role (Cheng 1998; Reim 1992; Roos and Bruss 1994; Singh and Rein 1992). Task assignment and task authorization, functions which are usually dependent on role resolution, are among the biggest challenges of a successful workflow solution. Because of the high mobility and turnover rate within an enterprise (see Chapter 1 for a

discussion on today's challenges), it would be impractical to deploy an enterprise-wide collaborative computing solution which performs task assignment and authorization using a static role model.

6.2.1 Task Assignment

A *workflow process* (or simply *process*) is composed of a number of workflow steps (or simply *steps*), where each step corresponds to a *task* to be completed by an agent. An agent may be a human resource, a robot, a machine resource, or a software program.

Each workflow step has an associated *state*. When the agent completes a task, the state of the corresponding step changes from Active to Committed. In this case, we will say that the step is *committed*. This state transition of the step triggers the WFMS to create the next step or steps according to the corresponding business process specification. When the END step of the process is committed, the entire process is said to have committed (Cheng 1995).

In an object-oriented system, representational objects are used to model steps in a process and the process itself. Each of these objects has a life cycle corresponding to the state transitions of the construct it represents. An object representing a step is called a *step object*. An object representing the entire process is called a *process object*. Both step and process objects are called workflow objects. Figure 6-1 shows the state transition diagrams for these objects.

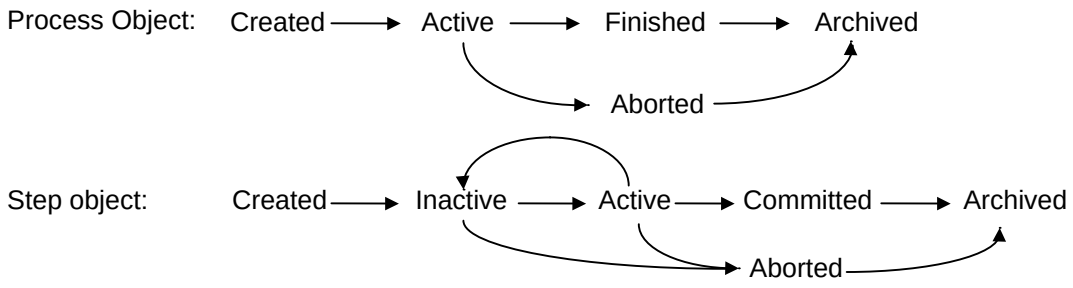


Figure 6-1 Life Cycle States and State Transitions of Workflow Objects

The WFMS is responsible for creating a step object, effecting the state transitions of that object and for assigning the step to one or more agents for execution. (This assignment is usually governed by the business policies of the organization that is being modelled.)

In essence, the WFMS is matching a pool of tasks (step objects) to a pool of available resources (agents).

With OMM, all enterprise resources can be modelled and accessed through the OMM API. Moreover, dynamic roles can be defined using resource attributes and contextual variables (Cheng 1998). As a result, task assignment specifications that are embedded within the step definition can directly reference these roles. For example, the *shipping* step in the electronic parts ordering process outlined in Chapter 3 may be assigned to any clerk in the shipping department of the company:

{… step definition …} execute by *shipping_clerk*;

In this example the execution of the shipping step is authorized to a role called *shipping_clerk,* which is defined in OMM using a virtual link:

(Department == 'Shipping') AND (Title == 'Clerk')

If we desire more specificity, we can define the *shipping_clerk* virtual link to include only those clerks who are currently on duty:

(Department == 'Shipping') AND (Title == 'Clerk') AND (Shift == *$current_shift*)

where *$current_shift* is a contextual variable that defines the shift based on the current time clock. All the attributes used in this example, *Department, Title* and *Shift* are associated with the EMPLOYEE organization.

In Section 6.3, we will provide the syntax of the step authorization statement along with a detailed discussion and further examples of role resolution.

At runtime, the WFMS queries OMM through the open API to resolve roles and identify the resources that are available to perform tasks. Since OMM is able to model different resource types and the roles they play under different business contexts, we are able to support dynamic task assignment to all available enterprise resources.

6.2.2 Task Authorization

In Section 6.2.1, we discussed the assignment of WFMS step objects to a pool of agents based on dynamic roles (as defined by OMM). The assigned agent is free to open and operate on a step object provided the agent has the appropriate authorization. Task authorization in WFMS concerns the control of accesses to process and step objects. In

this section, we will describe the various kinds of operations that an agent is allowed to perform on process and step objects.

Although the data of the WFMS may be managed by a DBMS, with access to this data protected by the authorization sub-component of the DBMS, WFMS task authorization is concerned with protecting both process and step objects at a higher level. In addition to the underlying DBMS protection, WFMS needs to protect shared workflow objects from unauthorized access (see Section 6.2.1). In general, WFMS supports four types of operation upon process and step objects, including *Read, Write, Execute* and *Manage* (Cheng 1998; Hsu et al. 1991). Table 6-1 shows the operation types that are available on workflow objects and the meaning of each operation:

Operation	Process Object	Step Object	Meaning
Read	No	Yes	Read data associated with the workflow object.
Write	No	Yes	Update data associated with the workflow object.
Execute	Yes	Yes	Open the workflow object for execution; includes the right to run the associated application.
Manage	Yes	Yes	Commit, abort, throw exception or retry of the workflow object.

Table 6-1 Operations Available on Workflow Objects

Read and *Write* privileges are needed to read and update data associated with a step or a process. Granting the *Execute* privilege implies at least the *Read* permission on the step object and sometimes the *Write* permission as well. If the agent does not need to update the workflow-related data but is simply making a decision that affects the routing of the workflow process, it will not need to have the *Write* permission in order to accomplish its work. *Manage* privilege is usually only given to administrators; this allows them to invoke state transitions for workflow objects through an external event, such as calling an API or throwing an exception.

OMM allows the WFMS to define task authorization on dynamic roles. Consider the previous example of a workflow step that involves a shipping clerk in Section 6.2.1, where a step is assigned to a role:

{… step definition …} execute by *shipping_clerk*;

When a user of the WFMS attempts to open the step object for execution, the WFMS calls the OMM system to resolve the role *shipping_clerk*, and identify whether or not the user is playing that role. The user would only be allowed to execute this step if the role

resolution result is positive, otherwise his request will be rejected. The mechanism of role resolution within OMM is discussed in detail in Section 6.3.

6.2.3 Routing Decision

A WFMS makes a routing decision every time a process under its control undergoes a transition from one step to another. Role resolution is an important factor in these routing decisions: the WFMS has to decide to whom a step should be assigned and who is authorized to open a step; it may also choose a route based upon the result of role resolution. Referring back to the e-commerce example in Section 3.3, the electronic parts ordering process will require managerial approval if the order amount is higher than the customer account's current credit limit. This is represented by route A in Figure 6-2; *approval_1* must be executed by the manager of the person who has executed *process_order*. Continuing our example, in order to reduce a performance bottleneck due to an overwhelming number of order processing requests within a department, a policy is set up such that if the current department workload exceeds a certain threshold, approval can be obtained from a different departmental manager within the same division. This is represented by route B in Figure 6-2; *approval_2* must be executed by a departmental manager who is within the same division of the employee, who has processed the *process_order* step, but who may belong to a different department. Figure 6-2 shows the partial flow diagram of this routing decision.

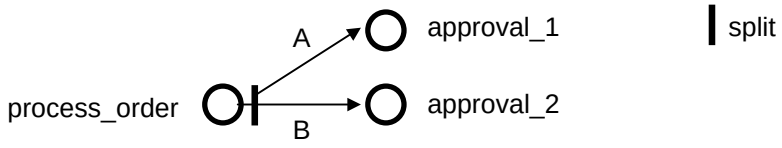


Figure 6-2 A Workflow Routing Decision Which Requires Role Resolution

The flow routes can be defined in the following manner:

```
If (the current step executor's department has less than N approval cases)
    Go route A;
Else
    Go route B;
```

Note that the routing condition here is not only referencing an attribute of the executor of the step, it is also making an indirect reference to the department object associated with

the executor of the current step. Modelling this workflow routing condition is a challenge to current state-of-the-art WFMS. Firstly, no existing organization modelling system supports the modelling of department objects with live links to employee objects. This is an important issue in the above example, in which the If condition requires the system to test the number of approval cases in the current step executor's department in order to make a decision. Secondly, no current organization modelling system provides an open interface to enable the WFMS to make calls upon the organization model to make this kind of routing decision. Consequently, all existing WFMS fail to implement the type of workflow routing decision that is required in this example. OMM is able to model this routing problem since it can model not only the employees of the company, but also the divisions and departments, as well as the dynamic connections between them. With the OMM API, WFMS are able to support this type of workflow routing specification.

Referring again to our example, the routing decision from step *process_order* to the next step can be described by the following pseudo-code program segment:

1. get the current step's (process_order) executor;
2. get the department object which is the department of the current executor;
3. lookup the number of approval cases pending (which is an attribute value) in this department;
4. if the result of (3) is less than N (say, N = 10), pick route (A);
5. else pick route (B).

Table 6-2 Pseudo-code Program Segment for Routing Decision of *Process Order*

The actual code segment that calls the OMM API is written in Java and is listed in the appendix section at the end of this chapter.

6.3 Role Resolution with OMM

Role resolution is the process of determining who plays a certain role in a given system. Two types of question are asked in role resolution. One is definitive in nature, and the other is relational. The following examples illustrate both types of question:

1. Is X an *engineer*? Or who are the *engineers*? (definitive)
2. Is Y the *manager* of X? Or who is X's *manager*? (relational)

OMM is adept at answering both of these types of question, and thus provides a strong foundation on which to model task assignment and authorization in a workflow management system (Cheng 1995; Vidgen et al. 1994).

Although the syntax of task assignment and task authorization specifications in workflow is implementation-dependent, most workflow management systems support the abstraction of *roles* to allow more flexibility than simply using user IDs (Gottlob et al. 1996; Hsu and Kleissner 1996). A control statement is usually included in the step definition for that purpose. The following syntax for a control statement illustrates the idea behind such specifications:

{… step definition …} <Control Statement>;

<Control Statement>	::=	<Operation> BY <Role>
<Operation>	::=	READ | WRITE | EXECUTE | MANAGE
<Role>	::=	<Role Label>
<Role Label>	::=	<Character String Constant>

Table 6-3 Syntax of Control Statement in Workflow Step Definition

The following is an example of a task authorization statement within the workflow script:

{… step definition …} READ BY *Manager*

In the current state-of-the-art, users are categorized by roles like *Manager,* which is a simple label. The use of role labels, although providing more flexibility than simply using a user name in the control statement, does not support resource relationships, which are required in most realistic business processes such as the e-commerce process discussed in Section 3.3. With OMM, role definition can be expanded to cover these relationships:

<Role>	::= <Role Label> | <Relationship Name> <Resource>
<Relationship Name>	::= <Character String Constant>
<Resource>	::= <Member ID> | $INITIATOR_OF_PROCESS

Table 6-4 Expanding Roles to Cover Relationships

where $INITIATOR_OF_PROCESS is a workflow system-defined data item which can be retrieved through the workflow callable interface. The workflow script now reads:

{… step definition …} EXECUTE BY *manager_of* $INITIATOR_OF_PROCESS

At runtime, when a member object m attempts to open this step for execution, the WFMS will query OMM to verify whether m is a *manager_of* the initiator of the workflow process. Here *manager_of* is a virtual link. The initiator of the process instance becomes the owner of the virtual link, and m is the member in question. The task authorization checking logic is reduced to the following question:

Is m the *manager_of* $INITIATOR_OF_PROCESS?

where *manager_of* is defined by a computable expression such as:

(deptNo == $owner.deptNo) AND (jobCode == 101)

or

(m.deptNo == $INITIATOR_OF_PROCESS.deptNo) AND
(m.jobCode == 101)

Evaluation of this expression will return a Boolean value indicating whether m is authorized to execute this step.

6.4 Role-based Access Control

The concept of dynamic roles can be applied to improve flexibility of role-based access control. Various RBAC models have been developed to control accesses to common and secured resources (Park et al. 2001; Sandhu et al. 1996). RBAC provides more flexible security management than the traditional approach of assigning roles by using user and group identifiers. In RBAC, access privileges are given to roles rather than to individual users. Users acquire the corresponding permissions when playing different roles. Roles can be defined simply as a label, but such an approach lacks the ability to allow users to automatically change roles under different contexts. This static method also adds administrative overhead in role assignment. In e-commerce and other cooperative computing environments, access to shared resources has to be controlled in the context of the entire business process. Therefore, it is necessary to model dynamic roles as a function of resource attributes and contextual information, and to use these roles to control access to common resources (Cheng 1999a; Cheng 2000a).

Role-based security has been applied in various areas of computer systems security (Youman et al. 1996). Researchers, such as Osborn (1997), Kuhn (1997) and Sandhu (1996), have proposed formal models for RBAC. Access privileges are granted to different roles. A user can play multiple roles by binding with a number of role names. Although this approach gives more flexibility to access control than the simple granting-to-user-identifier method (such as the scheme used in Microsoft Access), it is still a static approach and ignores entirely the overall organization model.

OMM provides a strong basis to support RBAC. The ability to model different types of enterprise resources facilitates the identification and categorization of the responsibilities of different resources in respect of one another. With OMM, both *active* and *inactive resources* can be modelled within the organization model. Active resources are agents that take on responsibilities to execute work tasks within certain business processes. Workers, managers and contractors are some examples of active resources. Inactive resources are produced and consumed during the business process cycle. Examples of inactive resources include products, materials and documents. Once the two types of resource are captured in an organization model, we can analyze the different responsibilities and roles the active resources adopt in executing various business processes within the company. Access to shared inactive resources can be controlled by assigning privileges to these roles.

Dynamic roles are modelled as a function of resource attributes and contextual variables using the OMM virtual link (Sections 6.2 and 6.3). This allows (active) enterprise resources to play various roles at different times and under different business contexts. Although we focused on the use of dynamic roles in workflow applications in the discussion of task authorization and role resolution in Section 6.3, the same idea will apply to and benefit RBAC in general. Applying OMM to RBAC allows any RM to have a higher degree of flexibility in controlling accesses to shared and privileged resources. OMM roles are defined as computable expressions and are resolved at runtime, rather than the traditional model of assigning access rights to a static role (which in turn may be assigned to a number of active resources).

In Section 6.3 we discussed an example of task authorization in workflow. Following the same methodology of granting permissions in executing workflow steps, one can define RBAC with a similar syntax as outlined in Table 6-3.

GRANT <Privilege> TO <Role>

We note that *privileges* are system implementation-dependent; roles are role labels which are in turn defined by using the virtual link syntax as described in Section 5.2.

6.5 Conclusion

In this chapter, we discussed the concept and challenges of role resolution in WFMS and how OMM addresses these issues using dynamic roles. In collaborative computing environments, role resolution is required when performing task assignment, task authorization, and process routing decisions. OMM supports these functions by providing a reference model to define various enterprise resources and their relationships flexibly. It also has an open API that allows the WFMS to call upon the full resources of the organization model when performing role resolution.

Role-based access control is discussed in the context of supporting e-commerce and collaborative computing applications. Again, OMM creates the potential for RMs to realize RBAC by using dynamic roles. It also provides a comprehensive model and an open interface for these systems to practically design fine-grained access control over a complex and rapidly changing enterprise.

In addition, the fact that OMM has separated the role and organization models from the workflow engine allows multiple WFMS to run on top of the same organizational conceptual design and design implementation. This creates integration opportunities over multiple groupware applications. OMM makes possible the routing of one business process to another by managing a common organizational information system.

Appendix 6A. Java Code Segment for Routing Control by Using Dynamic Roles

```
// Date:  02/07/03
// Author: ECC
// Method: chooseApprovalRoute()
// Parameter currentExecutor: the employee object who is the current
//   step executor;
// Parameter threshold: the maximum number of pending approval cases
//   for a department.
//
// Description:
// Apply OMM to workflow routing decision support.  In this case, not
//   only the employee object is involved, but also the department
//   object.
//

public int chooseApprovalRoute(OmsMember currentExecutor, int
threshold)
{
   // get the department name by retrieving the Department attribute
   String curDepartment = currentExecutor.getStringValue(“Department”);

   // get the department object
   OmsMember deptObj = new OmsMember (“department”, curDepartment);

   // get the department’s current total number of approval cases
   // pending
   int totalCases = deptObj.getIntValue(“NumOfCases”);
   if (totalCases < threshold)
      return 1;         // choose route A
   else
      return 2;         // choose route B
}
```

CHAPTER 7 Organization Modelling and Reengineering By Using OMM

To maintain a competitive edge in today's marketplace, companies have to rapidly adjust in order to adapt to the rapidly changing business environment. This requires improving flexibility in resource allocation and organizational structure, as well as streamlining the business processes. Organization reengineering has emerged as an important strategy in this endeavour; the term refers to the multi-step cycle of organization modelling, which involves careful organization analysis, strategic design and systematic changes in the organizational structure, management hierarchy, reporting infrastructure, and employee roles. The principles of organization modelling and reengineering were introduced in Chapter 2. In this chapter, we will cover the practical aspects of organization modelling and show how OMM can help to support the different approaches.

In Section 7.2 we discuss the integral relationship between Business Process Reengineering (BPR) and Organization Reengineering (OR). As a consequence of the close connection between BPR and OR, OMM's support for organization modelling contributes to improving both the structural aspect and the functional aspect of an enterprise. Section 7.3 introduces the various industrial approaches to organization modelling. We show how OMM is used to support the approaches relevant to the OMM model. In Section 7.4 we present some common techniques used in organization modelling. We then discuss how these techniques are enhanced by OMM.

7.1 Introduction

Business process reengineering is often a driving force and an integral part of organization reengineering. Changing the organizational structure often involves modifying the hierarchy of business units (or departments) within a company. These changes trigger further changes to operations, management responsibilities and reporting structure, which in turn trigger changes to the overall employee roles and functions. Although arbitrary changes may happen anytime within a small area of an enterprise, large scale organization reengineering has to be done systematically with significant fore-planning. Though the practice of organization reengineering may vary from

company to company, the different approaches usually fit into one of two categories, namely top-down and bottom-up (Hammer 1996).

7.2 Business Process Reengineering and Organization Reengineering

BPR refers to the cycle of streamlining and fine-tuning business processes. Since business processes are composed of steps executed by organizational resources, OM is a prerequisite to BPR. With the OR cycle, a corporation continually restructures and repositions its resources to accomplish its business objectives. The goal of OR is to create an efficient interactive workforce with the most optimal performance at the lowest cost. Unnecessary layers of management and overlapping resources are eliminated by consolidation of job functions and functional groups. Additional resources are allocated or acquired to take on new challenges and opportunities.

OR is only the first step in enhancing the performance and ROI of a corporation. A corporation carries out its function through the use and cooperation of its resources, which work in concert within various business processes to accomplish business goals. As a result, it is critical to reengineer the business processes to receive ultimate performance improvement. Once the OR cycle is in place, we are ready to focus on improving the business processes. A corporation adopts various business processes to accomplish its business objectives. Some business processes are mission critical — those involved in generating revenues — and impact directly and immediately the condition of the company. Others are administrative and supportive business processes. While these certainly impact the organization, their effect is usually indirect. The focus of improving supportive business processes is to reduce cost rather than bringing in higher revenue. Organization analysts may apply BPR techniques to streamline both types of business processes. The BPR cycle generally involves 4 stages; these are business process analysis, refinement, definition and management (Cheng 1995). Figure 7-1 shows the flow of the BPR cycle between the various stages.

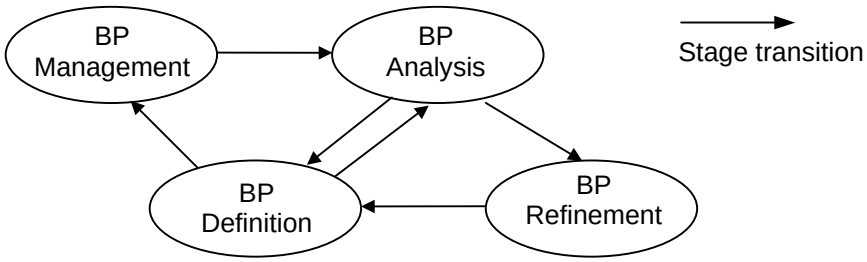


Figure 7-1 The Business Process Reengineering Cycle

7.3 Organization Modelling Approaches and OMM

Different collaborative computing systems take different approaches to address the issue of organization modelling. None of these approaches is more complete than any of the others; they each have their own design points and complement one another. However, all these approaches can be traced to three general and generic methodologies, which include the information system approach, the object-oriented approach, and the Petri Net approach. In this section, we discuss these organization modelling (OM) approaches and indicate how OMM supports and complements them.

7.3.1 Information System Approach

Information system methodologies have seriously influenced almost all organization modelling approaches and methodologies. There have been many attempts to use software engineering methodologies designed for information system analysis and design to model CIM systems. A typical example is SADT designed by Ross and Schoman, Jr. (Ross and Schoman 1977). Unfortunately, this starting point has largely contributed to the biases that limit organization modelling to data flows diagrams for modelling processes and entity-relationship diagrams for modelling data. However, as discussed in Chapter 2, true enterprise-level organization modelling deals with dimensions other than just functions and data. Two areas, resource management and organization structure especially, are poorly addressed by the co-opted methodologies of data flow and ER.

This is not to say that the diagrammatic techniques of data flow and ER have no place in OM. Information system design and analysis has an important role to play in such areas as:

1. support for modelling of data,

2. analyzing the information issues of the information view, and

3. assisting in the design of the database schemas for the databases that will be implemented in the integrated enterprise environment.

M*OBJECT discussed in Chapter 3 is an example of an enterprise modelling system taking the information system approach (Di Leva et al. 1997). OMM, using an object-oriented approach, has simplified the modelling and analysis of information and information views. With our implementation of mapping the OO layer to the underlying database schema, we eliminate altogether the complexity of defining the database schema for representing the enterprise model. We refer readers to Sections 4.4 and 9.4 to see how OMM can model the information system approach.

7.3.2 Object-Oriented Approach

The object-oriented approach is a generic and universal modelling tool, although based on only one modelling construct: the *object.* The paradigm has been suggested for enterprise modelling as well as for manufacturing system modelling by numerous works, including X.500 and LDAP (Radicati 1994).

According to Vernadat (Vernadat 1996), although the object-oriented approach has six fundamental principles (see Chapter 3 for a discussion of the fundamental principles of the OO methodology), the main characteristic of the object-oriented modelling approach in OM is the *encapsulation* property. This property has the effect of combining function modelling and information modelling into one unified paradigm (Vernadat 1996). With the OO approach, objects are uniquely identified, have a state, and possibly a set of behaviours (i.e. a set of callable operations, called methods, that represent functionality). They are able to depict both abstract and concrete aspects of the enterprise. With the OO approach, the whole model is defined as a set of communicating objects.

Two other OO properties, *inheritance* and *reusability,* also hold significant advantages for the OO approach in organization modelling. These features are not present in the other two generic modelling approaches mentioned earlier in this section, namely ER and data flow. Inheritance refers to the feature of allowing classes of objects to share common properties by inheriting from their common parent class, which is called the superclass. These common properties can then be reused in other objects, and objects can be reused from one model to another, thus saving development time. The OMM model we propose adopts the OO approach to model the enterprise resources. (We note that M*OBJECT also includes some OO features.)

Although the OO approach is widely used in systems modelling, an issue that has slowed its application to enterprise modelling is that traditionally business users are more process-oriented than object-oriented. It therefore seems unusual, on the face of it, to represent business processes by using objects. However, OO is ideal for modelling enterprise resources such as people, products, equipment and services. The Petri Nets approach, on the other hand, is ideal for modelling processes.

7.3.3 Petri Nets Approach

Petri Nets are a graphical and mathematical modelling tool used to represent and analyze the behaviour of concurrent and parallel systems (Peterson 1993). The approach is based on a simple, abstract representation of a system in terms of a multi-partite directed graph made of two types of node, respectively called places (represented by circle) and transitions (represented by vertical bars) connected by directed arcs. The behaviour of the system is modelled by tokens flowing from place to place via the firing of transitions. Figure 7-2 shows an example of a Petri Net.

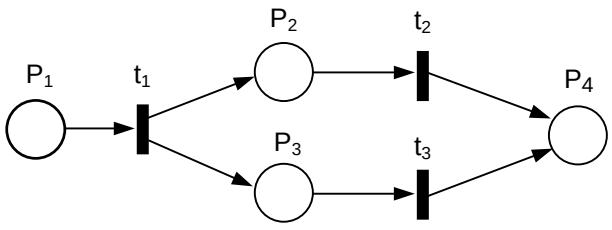


Figure 7-2 An Example of a Petri Net

The Petri Net is very popular in modelling enterprise processes because of its ability to describe the flow of a business process easily. It is also used widely for the analysis of

both enterprise functionality and enterprise behaviour (Vernadat 1996). However, it falls short of modelling enterprise resources. Objectflow is an example of a collaborative system — one which uses Petri Nets to model processes (Hsu and Kleissner 1996). Although OMM does not use the Petri Nets approach to model interactions within an enterprise, with the OMM organization model and role model, we can support and enhance any Petri Nets based modelling system.

7.4 Common Techniques in Organization Reengineering

An enterprise is a very complex entity composed of a large number of business objects interacting and relating to one another to accomplish a set of goals. To analyze and streamline an enterprise, it is necessary to break down this complex entity into smaller logical pieces, identify all its resources, functions and goals, and eventually place and connect them together in a logical way to represent the interactions and relationships among them. We will introduce four common techniques that organization modellers and analysts use to carry out this process of breaking down and building up enterprise components. We will also use some examples to illustrate how OMM is used to support these techniques.

7.4.1 OR Modularization

Modularization is a technique used in organization reengineering. Smaller sub-components of a company are grouped together based on some common attributes to create bigger units. Modularization can be applied recursively to create even larger units until ultimately an enterprise is formed. It can be realized via the following two fundamental concepts:

- *Function*: entities of similar function are put together for clarity and better manageability. For example, in certain companies, all marketing offices belong to a marketing organization.

- *Location*: units responsible for different aspects of the same location may be grouped together to support one another. This may mean that the different components are physically located in the same area and share a common set of opportunities and

issues. It may also mean that they are geographically distributed but responsible for the same market segment.

In OMM, modularization can occur in two ways. Firstly, two OMM organizational databases can be merged together to become one. OMM member attributes with the same semantics and data types can be merged with one another and form a new OMM member attribute of the consolidated organization. The name of the new attribute in the newly merged OMM organization can be the same as one of the two names in the former OMM organizations, or a new name given by the administrator. Attributes with different semantic meanings or types can either be dropped or added to the resulting OMM organization. In the latter case, member objects of the original OMM organization that do not have these attributes will contain the NULL value for such attributes. As discussed in Section 4.4, each member in the OMM organization has a unique identifier, the member ID. Since the member ID is globally unique within the enterprise, there will be no confusion in the identity of the members (even after merging two OMM organizations). Please refer to Appendix A for the Java implementation of the `OmsOrganization` and `OmsMember` classes.

As an example, assume that as a result of two companies going through merger or acquisition, it is required that the OMM employee organizations of the two companies be merged. Figure 7-3 shows the correspondence mapping of the attributes of the two employee organizations.

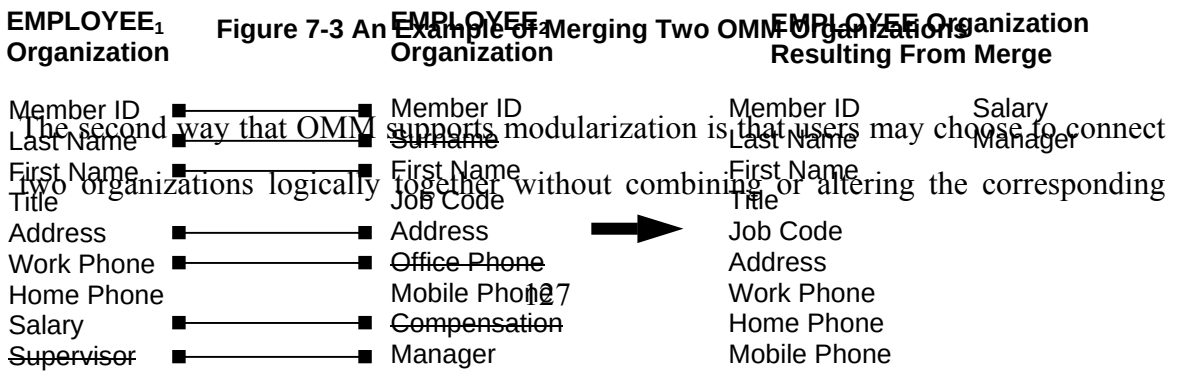


Figure 7-3 An Example of Merging Two OMM Organizations

The second way that OMM supports modularization is that users may choose to connect two organizations logically together without combining or altering the corresponding

database schemes. To do this, one may simply define a link between the two objects. This can be done using an OMM virtual link. Alternatively, a third organization can be created to have a link connected to each of the two OMM organizations. Thus, diagrammatically, a parent OMM organization is created on top of the original two OMM organizations.

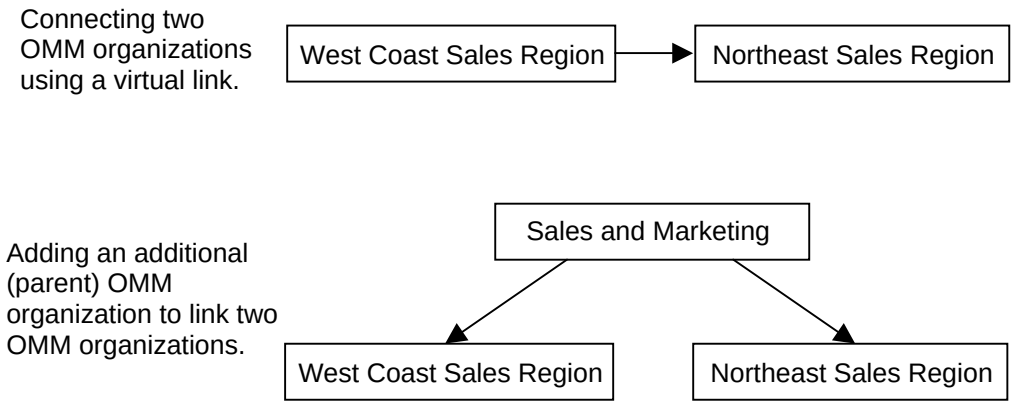


Figure 7-4 Different Ways to Logically Connect OMM Organizations

7.4.2 OR Decentralization

Decentralization is a technique used to logically divide an organization into smaller sub-components. Together with modularization, decentralization creates a balanced force, which allows a company structure to attain a state of equilibrium. This method is often used in restructuring when the company is growing in size and wants to give autonomy to certain organizations. Decentralization also comes naturally when a company wants to venture into a new business sector or market region. It may also happen when a certain organization has grown too big and has become difficult to manage and slow to respond to changes.

OMM supports decentralization by creating new OMM organizations within the enterprise. Member objects representing company resources are flexibly migrated from one OMM organization to another. Similar to the case of merging OMM organizations discussed in Section 7.4.1, when moving member objects from one OMM organization to another, OMM member attributes may be dropped from the old OMM organization, or mapped to the new one. This operation is implemented in the Java method `move()` in `OmsMember`. Please refer to Appendix A for the implementation of the `move()` method and the `OmsMember` class.

7.4.3 Bottom-Up Analysis in Organization Reengineering

One way to model an organization is by using a bottom-up technique, which starts the analysis from the "hands-on" level of the company, i.e. the projects, tasks, and people resources level. Bottom-up analysis is performed through interviews involving not only department managers but also supervisors, foremen, and workers (Di Leva et al. 1987). People are asked to describe the functions which they are involved in and the activities they perform. This style of analysis requires at least three meetings per enterprise function: one to do the interview and the function diagnosis with the director or supervisor, one to model the function (i.e. identify inputs, outputs, constraints and operations), and a last meeting to discuss the results obtained with function personnel. Documents that are processed in the business flow are identified and the routing of documents modelled. Verbal descriptions of the processing of documents and the execution of function activities are recorded on tapes and reported in written notes. Based on the result of these interviews and analysis, resources are grouped together based on commonalties such as functions, responsibilities, production focus, and geographic locations. Relationships such as reporting infrastructure and project teams are also defined. At this level, entities, as defined by the OR architect, can be perceived as the basic functional units for carrying out business objectives. The bottom layer entities are the building blocks of an organization. Units of similar focus and complementary functions can be categorized together logically on the next higher level to form a smaller number of strategic units. Once the first level of organizations is created, the modularization technique (described in Section 7.4.1) can be applied recursively to establish an enterprise.

The goals of the bottom-up analysis are:

1. To validate the top-down analysis which is performed by interacting with the top level of management in the enterprise. The top-down approach will be discussed in the next subsection.

2. To finely model each function identified in the general functional diagram. This includes identifying the resources involved in each function, their attributes and responsibilities, and their relationship and interaction with one another.

3. To make a diagnosis of each function in order to evaluate its operations.

Figure 7-5 shows how the bottom-up analysis is applied in organization reengineering to produce a functional pyramid diagram of the enterprise.

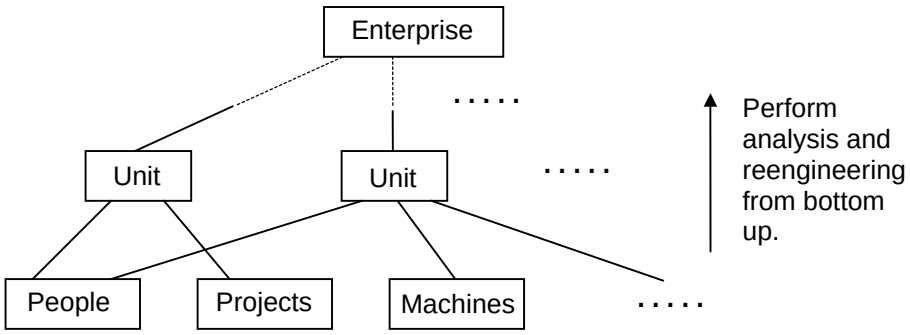


Figure 7-5 Bottom-Up Organization Reengineering Method

Because OMM is sufficiently flexible to support resource definition and organization modularization, users can use OMM to capture the lowest level resources. By logically (through virtual links) or physically (through merging of organizations) pulling the organizations together, a complete, well-partitioned organization structure will emerge.

7.4.4 Top-Down Analysis in Organization Reengineering

Another commonly used technique in OR is the Top-Down Analysis. This may start by identifying a company-wide mission statement from the top level of management. According to Di Leva et al., the top-down analysis usually requires two to three short meetings with enterprise directors (Di Leva et al. 1987). The discussion concerns the general organization of the enterprise, the size of the enterprise, the financial situation, the operational problems faced by the company's decision-makers, or the long-term vision of the enterprise. One meeting can be used to present to management the analysis approach and the tools used in the process. By identifying top-level areas, objectives, and goals which help to accomplish the company's mission, different divisions or operations of the company are defined. Each of these divisions or operational units has its own mission statement and a set of objectives. These mission statements and objectives should all work together to achieve the company's top-level mission.

The goal of top-down analysis is to understand the overall structure of the company. Through this process we discover the pyramidal structure of the organization's decision system as well as its business objective tree, identify the various levels of decision-

making and objectives, and the decision-making centres. This step is crucial when the OR analysts come from outside the enterprise to be analyzed.

With OMM, each business unit can be viewed as an organization. Descending the structure, more refined definitions are given to objectives, and the means of reaching specific goals are clarified. Ultimately, some organizations on the same level may be merged (modularized) to simplify the structure, and units that are too big may be decentralized by being split into multiple organizations to improve manageability. Such fine-tuning can be done continually as part of the life cycle of OR.

Figure 7-6 shows the top-down analysis applied in organization reengineering.

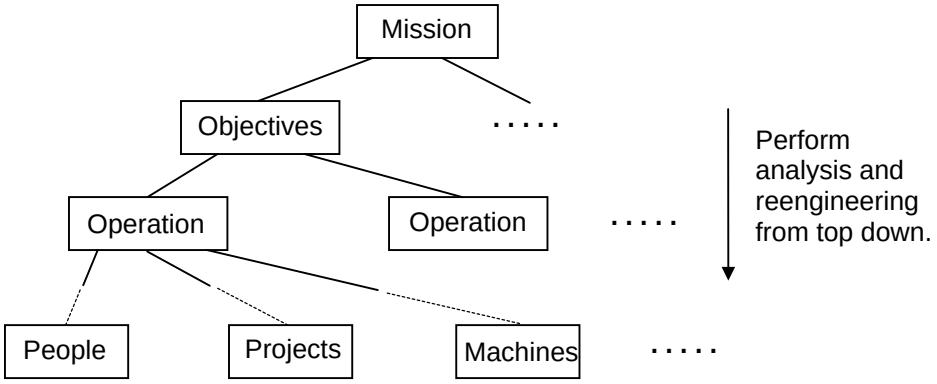


Figure 7-6 Top-Down Organization Reengineering Method

Starting from the existing structure, one plan of attack is to begin analysis with the less productive organizations, identifying their missions and defining the associated subgoals. Depending on other factors, such as the size of the organization, budget, type of business and work, and other heuristic factors, new business units may emerge. With OMM, people as well as other types of resources can be allocated to new units, and functional teams can be established by connecting the resources from one OMM organization to another via virtual links. For example, a functional team, say a production department, consists of raw materials, machinery and personnel (such as direct labour, supervisor and administration).

The actual sub-organization structure can become very complex depending on the intersection of these various factors. However, with OMM, organization modellers have a powerful and flexible model, capable of supporting top-down decentralization of even the most complex corporations. This is achieved via horizontal and vertical partitioning.

7.5 Conclusion

Organization modelling and reengineering methods and techniques have been developed for decades. In this chapter we have discussed these practices and shown how an organization modeller can apply the OMM organization model in combination with these techniques.

We discussed the three different approaches people take in organization modelling. The information system approach and the object-oriented approach aim to take care of the information model aspect in organization modelling, while the Petri Nets approach focuses on fulfilling the functional aspect. OMM takes the object-oriented approach in creating an information model. OMM does not contain a separate business process model to address the functional aspect of OR; it is assumed that this need can be fulfilled by the process modelling components in most standard WFMS. However, OMM also has a rule-based relationship model which enables the WFMS to model enterprise-level business processes by providing dynamic role resolution. This was discussed in Chapter 6 of the thesis.

We reviewed four common techniques used in organization modelling: modularization, decentralization, bottom-up analysis and top-down analysis. OMM supports all four of these. Using an OO model, OMM allows the organization modeller to flexibly modularize or decentralize company resources into various OMM organizations. Different OMM organizations can be brought together using OMM virtual links. These allow the organization modeller to perform organization re-design with great flexibility and effectiveness, enabling both top-down and bottom-up analysis methods.

CHAPTER 8 Concurrency Control

Concurrency control is the activity of coordinating the actions of processes that operate in parallel over shared data, and therefore potentially interfere with each other. Concurrency control problems arise in the design of hardware, operating systems, real time systems, communications systems, and database systems, among others. As OMM is not only a modelling tool but also a real-time system running on the Internet which allows concurrent access to enterprise information, we have to deal with the concurrency control problems that may arise.

This chapter discusses the problems of concurrency control and provides examples of how the problems arise in the context of OMM. In Section 8.2, we present the concept of transaction, which is fundamental to enable concurrency control. In Section 8.3, we discuss our technique of allowing users to pre-claim resources in order to solve the concurrency issue. Sections 8.4 and 8.5 are devoted to discussing deadlock issues and how OMM deals with them.

8.1 Introduction

We have shown that an organization modeller can apply OMM to reengineer an enterprise. However, some means of maintaining the organization database thus created must be established; otherwise the information collected during the organization analysis process will remain static and will quickly be outdated as the organization undergoes constant changes. Hence it is important for the OMM system to be able to continually receive input, and to have the ability to adjust itself automatically as the underlying information of the organization changes. The object-oriented approach and the rule-based relationship model in OMM provide the primary technology to make this automatic adjustment possible. However, in order for the resulting enterprise model to be fully functional and dynamic, we must also provide ways for employees on all levels of the enterprise to continually and easily maintain up-to-date information for those enterprise objects that they manage. The OMM system accomplishes this by providing an open Java API to support the development of Internet and Intranet applications to accomplish these tasks. Users cannot only access and review organizational resource

information, organization structure and resource connections, but they can also update this resource information anywhere, anytime through the World Wide Web. As the underlying information is updated, the specific organization models created to capture the network of enterprise resources and their corresponding inter-object relationships will automatically adjust themselves to represent the most up-to-date picture of the enterprise. These are all very possible outcomes of exposing the organizational information to multi-user updating. However, once we allow multiple users to modify the organizational information concurrently, it is possible for them to run into conflicts that may lead to data inconsistency and deadlock.

Since we are primarily dealing with organizational information management, we will discuss these problems and solutions from the view of a database system. Database system concurrency control mechanisms apply to many types of data handling systems (such as database management systems for data processing applications, transaction processing systems for banking or airline reservations, and file systems for a general purpose computing environment). These mechanisms will also apply to the OMM system, which handles organizational information stored in a DBMS.

The main component of concurrency control in a database system is the *transaction.* A transaction is an execution of a program that accesses a shared database. The goal of concurrency control is to ensure the *atomic* properties of transactions during execution, meaning that:

1. each transaction that accesses shared data is transparent to all other transactions, and

2. if a transaction terminates normally (commits), then all of its effects are made permanently and reliably in the database; otherwise (if it aborts) it has no effect at all.

8.2 Background

8.2.1 Database Systems

A *database* consists of a set of named *data items*. Each data item has a *value*. The values of the data items at any one time constitute the *state* of the database. A data item

could be a word in the computer memory, a page on a hard disk, or a record in a file. The size of a data item is called the *granularity* of the data item.

A database system, also known as database management system, is a collection of hardware and software elements that support access to the database. Access events are called database operations. In general, operations can be categorized into either *Read* or *Write* operations. Read operations will not alter the state of the database, while write operations will. DBMS execute each operation atomically and consider each operation to be a single undivided unit of work. This means the database system ensures that either all or none of the effects of an operation remain in the database. As discussed above, this is known as the *atomic* property of the DBMS, and it is essential for ensuring data integrity. Note that read operations will not change the state of the database while write operations will move the database from one state to another.

8.2.2 Transactions

The concept of a transaction has been used to ensure atomicity, consistency, isolation, and durability of a unit of work in a computing environment. A transaction may be composed of multiple database operations. The ACID properties are critical and sufficient to maintain data integrity:

- *Atomicity:* a database system guarantees its operations possess atomicity if it ensures that each operation must either be completed or not started at all; there is no partial operation under any possible circumstances.

- *Consistency:* a database system is said to be consistent if it does what it claims it does repeatedly.

- *Isolation:* the isolation property ensures that no two concurrent transactions will see the effect of one another. A database system that possesses the isolation property will give the appearance that all of its transactions are serialized.

- *Durability:* some database transactions bring about state transitions to the database. A database system has the durability property if all the state transitions brought about by its committed transactions are guaranteed to persist.

A number of transaction models have been proposed and defined over the years in an effort to achieve the ACID properties in the context of complex distributed computations. A distributed complex computation usually involves a number of sub-computations spread over a heterogeneous computer network. Each of these sub-computations or computation components may employ the idea of a transaction to ensure its own atomicity and consistency in a multi-user environment. In order to guarantee transactional behaviour over the whole distributed computation, the transaction components are linked together to form a global transaction or a transaction tree. A defined set of behaviours is maintained between any two sub-transactions by a transaction manager, which may be embedded in the DBMS. Each distinctive set of transactional behaviours is termed a transaction model (Cheng et al. 1991). These concepts can be depicted by the following:

Transaction Component, t_i	=	a unit of local computation or non-undoable action.
Transaction Model, M_{ij}	=	$\{b_1, b_2, \ldots b_n\}$, where b_k, k = 1, 2, …, n, is a transactional behaviour between transaction components t_i and t_j.
Transaction, T	=	$\{t_i, t_j, t_k, \ldots, M_{ij}, M_{jk}, \ldots\}$.

Once a transaction is started, it will end at a later point in time, either by committing its computations and the effect of its operations will be externalized to all other transactions, or by aborting its computations such that all its modifications to the database will be undone.

8.2.3 Commit and Abort

A DBMS supports transaction operations, which are *Start, Commit*, and *Abort*. These operations are specified in the database programs, which constitute one or more transactions. *Start* is simply to begin a transaction. *Commit* is to request that all the state transitions of the current transaction be persistently reflected in the database. *Abort* on the other hand requests that all the state transitions of the current transaction be cancelled or rolled back. The syntax of a transactional database program looks like this:

```
boolean withdraw(String fromAccount; Float amount)
{
   Begin Transaction;
   float temp = readAccountBalance(fromAccount);
```

```
   if (temp < amount)
   {
      output("Withdrawal failed: insufficient funds.");
      Abort;
      return false;
   }
   else
   {
      updateAccount(fromAccount, temp – amount);
      Commit;
      Output("Withdrawal completed");
      Return true;
   }
}
```

Figure 8-1 A Database Program to Withdraw Money from a Bank Account

After the DBMS executes a transaction's *Commit* (or *Abort*) operation, the transaction is said to be committed (or aborted). A transaction that has issued the *Begin Transaction* but is not yet committed or aborted is said to be *active.*

A transaction issues Abort if it cannot complete correctly. The database program may issue the Abort because it has detected an error situation, such as in the example outlined in Figure 8-1 that the account has insufficient funds. Or the Abort may be incurred on a transaction by circumstances beyond its control, such as in the case of power outage or other types of system failures.

When a transaction aborts, the DBMS removes all of its effects. The fact that a transaction may be aborted requires that the DBMS has to determine a point in time after which the DBMS guarantees the users that the transaction will not be aborted and its effects will be permanent (Bernstein et al. 1987). For example, in processing a withdrawal from a bank account, the system cannot abort the effect once the cash is dispensed to the customer.

The Commit operation accomplishes the guarantee of a point of no return. Its invocation signifies that a transaction terminated "normally" and that its effects are made permanent to the database. Executing a transaction's Commit constitutes a certainty given by the DBMS that it will not abort the transaction and that the transaction's effects will survive subsequent failure of the system.

In order to attain the ACID properties of transactions, a transaction manager works with a lock manager to serialize accesses to shared resources. A lock manager grants different types of lock to transactions. In general, lock types can be categorized into

Read or Write locks. Read locks are also known as shared locks since multiple users can be locking (read only) the same database resource without conflicts. Write locks are also known as exclusive locks since once a user locks a resource with a Write lock, no other transactions may acquire locks on that resource until the user releases the exclusive lock. A transaction requests for locks when it attempts to access a database resource. It releases all the locks it acquired throughout its life when it commits or aborts. When two transactions attempt to issue a request for a lock type that conflicts on a common resource, only one of them will get the lock and the other transaction will have to wait until the first transaction releases the lock.

8.3 Concurrent Access to Shared OMM Objects

The transaction manager of the RDBMS controls and coordinates database accesses through locks. Although this is sufficient to maintain the ACID properties of the data, it could have poor usability when it comes to OMM, which is an interactive, object-based system. For instance, when multiple users are accessing OMM resources (objects) through the Web to maintain enterprise resource information, it is possible that two users are updating different attributes of an object at the same time. These attributes might be located in different tables. In the case of distributed OMM objects, they may even be managed by different DBMS. Although we can hold locks on all attributes of the object and serialize the concurrent operations, this will significantly reduce the parallelism and performance of the system as we are dealing with human interactive accesses. If we lock only those attributes that are interesting to the specific operation, then from the view of the DBMS there is no conflict in locks between the two operations, but from the users' point of view, the two transactions are conflicting on the object level. In this case, even though data integrity is not sacrificed from the view of DBMS, on the object level of OMM, the two transactions will appear to have lost the isolation property. Therefore, in order to improve performance and retain integrity at the object level, it is necessary to have the flexibility to support optional object-level locking in OMM. Note that locking the entire object can be optional rather than mandatory because the described conflict does not impact the data's ACID properties, which is critical to data integrity. Strictly speaking, the issue is only a usability issue. Indeed in some cases, such as when two users are assigned to manage two unrelated aspects of the same enterprise resource, the

above scenario might not have any conceptual conflict at all. The following example illustrates the case in which object level locking is desirable:

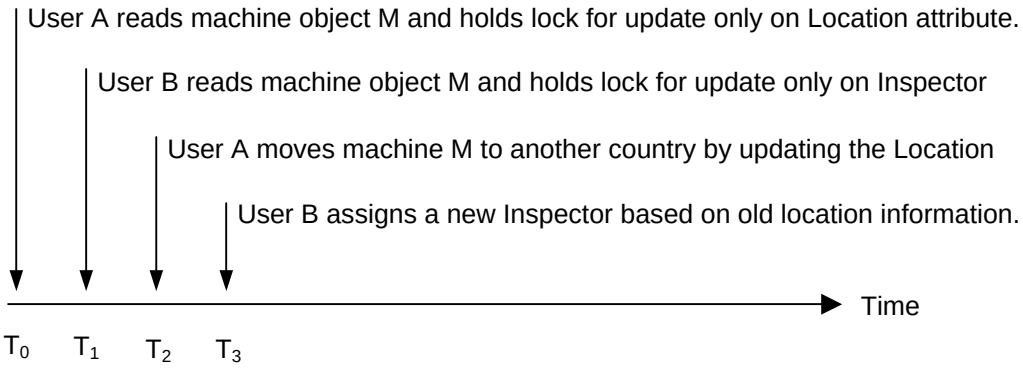


Figure 8-2 An Example of Conflict on the Object Level but not on Data Level

Should object lock be applied when user A reads the machine object M, user B will be blocked or get an exception when attempting to access the same machine object. The exception will arise either when user B reads the machine object or when he requests for an update lock on the *Inspector* attribute, depending on whether the object-level lock held by user A on M is a *read* lock or a *write* lock. This seems to point to the suggestion that object-level lock is always desirable; however, the following example illustrates a case in which object-level locking is unnecessary:

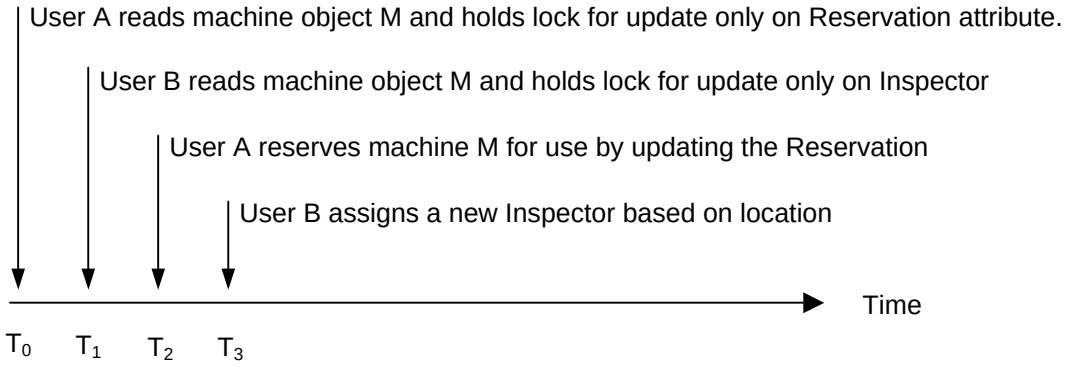


Figure 8-3 An Example of Concurrent Accesses without Conflict on Object Level

Two observations are important here. Firstly, an object-level conflict occurs only when there are two or more applications with write operations involved. It will not occur even with one *write* and many *reads*. In the case of read-only operations, there should not be any object-level conflict. Secondly, an object-level conflict occurs only when two or

more operations are referencing, via either *read* or *write*, one or more common attributes. In the example depicted in Figure 8-2, conflict exists because both user A and B are referencing to the *Location* attribute.

As discussed, one way to solve this issue is to allow the application or the interactive users to pre-claim object-level locks on update and delete operations. If the application programmer does not foresee a potential conflict on the object level, then it is not necessary to serialize accesses by pre-claiming locks.

An alternative to having object-level locks is to have attribute locks. Programmers still have to declare pre-claim locks, but instead of requesting locks on an object level, pre-claim locks are requested on the attribute level. Attribute locks have finer granularity and therefore reduce lock collision and improve concurrency. However, it is more complicated to program because application programmers will have to decide which attributes to lock rather than simply locking the entire object. In the example shown in Figure 8-2, user A may pre-claim an exclusive lock on the attribute *Location* of machine object M, rather than locking the entire object. When user B tries to read *Location* by requesting a read lock on *Location*, the request will be blocked. If user B gets in first and obtains the read lock on *Location,* user A will have to wait when he attempts to pre-claim an exclusive lock on *Location.* In either case, the operations will be serialized.

8.4 Deadlock Detection

Resource allocation and scheduling is a generic computer science topic. In resource management and transaction processing systems, scheduling consists of controlling the transactions to minimize response time or maximize throughput. Very primitive scheduling techniques have sufficed in the past. One simple approach is to limit the degree of multiprogramming, which in turn impacts the number of concurrent transactions. Another common technique is time-slicing the execution of each transaction, so that each makes forward advance one small portion at a time.

The lock manager is a scheduler. If many processes are waiting for a lock, a scheduling decision is made when the lock becomes available; this decision determines which transactions should continue to wait and which one should be granted access to the

resource. Traditionally, simple first-come, first-served scheduling has been used in lock managers.

Whenever the system allows multiple transactions to request for locks, there is a possibility of waiting for locks, which may give rise to *deadlock*. Deadlocks will only occur when someone is requesting for a lock and results in the wait queue. In a deadlock situation, each member of the deadlock is waiting for another member of the set. In other words, no one in the set is making any progress, and no one will until someone in the set completes (or gives up). An easy solution is never to wait, but instead to cancel any request that might wait, do a partial rollback, and then restart the program. In our examples in Section 8.3, we have indicated that upon lock conflict, the latter transaction may either be blocked (i.e. wait), or receive an exception (i.e. cancel its operations). Cancelling a transaction is easier to implement than detecting and handling deadlocks. Such an approach also definitely avoids deadlock, but it may create a *livelock* situation in which each member of the livelock set may soon want to wait for another member of the set, resulting in another rollback and restart. Livelock is actually worse than deadlock, because it is harder to detect and because it wastes resources (Gray and Reuter 1993).

At any time, the transactions of the database system define a directed *wait-for* graph. The transactions are the graph nodes, and there is an edge from transaction T to transaction T' if:

T is waiting for a resource held by T', or

T will eventually wait for a resource to be granted to T'. That is, they are both waiting for the same resource, and T is behind T' in the waiting list, and their requests are incompatible.

A cycle in the wait-for graph indicates a deadlock, and the transactions in the cycle are said to be deadlocked. When resources are taken from the shared pool of resources, the wait-for graph must be labelled with the kind of resource being demanded. The simplest deadlock detection is to treat this as a standard transitive closure algorithm, which could be inefficient especially when dealing with a large and distributed system.

An intermediate approach is to cancel only the lock request without rolling back the entire transaction. With our proposal of pre-claim object-level locks, when the user (or the program) attempts to acquire an object-level lock and runs into a conflict, OMM

simply throws an exception and refuses the lock request; the user may then decide to try again or to abort the whole update operation. This approach gives much more flexibility to users and is particularly useful in an interactive environment, as is the case of OMM in maintaining enterprise resources information.

8.5 Deadlock Avoidance and Resolution

It is possible to avoid or prevent deadlock entirely. The standard deadlock avoidance technique is to linearly order the resources and acquire them only in that order. Such technique is observed and implemented on the application level. A deadlock requires a waits-for cycle, and the linear order avoids such cycles. However, in our case where resource objects are distributed over the Internet and having accesses coming from anywhere in the world, it is not practical to apply such simplistic solution to our concurrency issue.

For situations of sharing pools of resources, such as a pool of tape drives, the system can ask jobs to pre-declare their maximum needs and schedule the jobs accordingly. Such a deadlock avoidance scheme is used at low levels within systems, but they break down as the layers of abstraction build. Any static declaration is likely to be wildly pessimistic, pre-claiming the maximum possible resources. Hence, there has been a strong trend toward dynamic resource allocation on demand. In any case, according to Gray and Reuter (Gray and Reuter 1993), deadlock is assumed to be a rare event that can be resolved by the transaction rollback mechanism as discussed in Section 8.4.

In the case when a system that uses OMM happens to wait on an object-level lock, i.e. a conflict has arisen, then the system has to resolve potential deadlocks. Given that deadlocks are allowed to occur, one solution is *timeout*. Whenever a transaction waits for more than a certain time, declare that something is wrong and rollback that transaction. This is a technique used in many commercial transaction processing systems such as IBM's CICS, Tandem's (now part of Compaq Computer) Encompass and NonStop SQL Systems. According to Gray and Reuter (Gray and Reuter 1993), all systems must ultimately depend on timeout to detect some deadlocks; for example, an application waiting for terminal input or for a lost tape to be mounted may have to be rolled back if it holds resources needed by other applications. They also observed that lock waits are very rare events. Waits cause unacceptably high variance in service times.

Systems are designed so that most locks are free most of the time; this is also true when a system has a large pool of commonly shared resources such as in the case of an organizational database. If lock waits are very rare, then deadlocks are even less likely to occur. It is therefore an acceptable solution to deal with deadlocks by using the simple timeout mechanism.

Timeout is a very pessimistic deadlock detector. Any long wait will be declared a deadlock. It seems more elegant to detect deadlocks and distinguish between deadlocks and long waits. A compromise solution is to run a deadlock detection algorithm after a transaction timeout, just to see if there is a deadlock that can be resolved. If one is found, then a victim can be picked, and the other transactions can continue to wait until they are granted access. This complicates the deadlock resolution algorithm, but reduces the possibility of killing non-deadlocked transactions.

8.6 Crash and Recovery

Through the use of transactions, resource managers are able to ensure the ACID properties in a multi-user computing environment. Two distinguishing properties are observed among the various transaction models: namely, concurrency and recovery. Recovery is necessary whenever we have a failure. There are three types of failure that are most important in database management systems. They are *transaction failures, system failures* and *media failures.* A transaction failure occurs when a transaction aborts. A system failure refers to the loss or corruption of the contents of volatile storage. This can happen in a power failure or a fault in the operating system. A media failure occurs when any part of the stable storage is destroyed, such as in a hard disk crash. DBMS use the technique of journalling to ensure that upon recovery from a crash, transactions are correctly undone or re-done such that the ACID properties of the database are preserved (Cheng 1992; Bernstein et al. 1987).

In OMM, the backend failure and the corresponding recovery are taken care of by the backend database management system. When the OMM system crashes, whether it is due to software or hardware failures, the recovery logic of the DBMS will ensure that the transactions be undone or re-done based on the database journal records. The pre-claim lock information that is kept in OMM is on volatile storage; as a result, all locks will be removed once the system is restarted. However, the use of pre-claim object level locks,

as proposed in this thesis, requires that the OMM system be able to perform garbage collection by removing all outstanding locks that belong to a certain session when the system detects that the session has crashed.

When a client connects to OMM through the Internet, a session block is created to represent the session between the client and the server. Garbage collection on locks can be done by checking on the condition of sessions on a periodic basis. Once the OMM system detects a broken session, it will release all the locks that are associated with the session. Other transactions that are waiting for the locks will therefore be granted access to move forward.

Another area closely related to recovery is transaction logging or journalling. The logging mechanism records the database changes involved in transactions to ensure that in the case when a transaction crashes, the database management system will be able to recover the database state such that the ACID properties will be preserved. Since the logging mechanism journals reliably the actions taken over the database, with slight modifications, it can also be used for the support of auditing.

The challenge of applying audit logs to an e-commerce environment is on the high expectation of having the audit logs to also serve as a History Manager. In an e-commerce environment, sellers analyze the buying patterns of the buyers to make just-in-time product manufacturing decisions or to adjust pricing strategy based on the demand in the market as indicated by the activities recorded in the journal. The current implementation of the database system does not support indexing on history tables or the recovery logs. Standard disk-based index structures such as the B-tree will effectively double the I/O cost of transactions to maintain indexes on history tables in real time, increasing the total system cost by up to fifty percent. A technique known as the log-structured merge-tree, or LSM-tree, has been developed to counter this problem (O'Neil et al. 1996). An LSM-tree breaks the index on the history table into two portions. First is the memory-based small B-tree (SB-tree) which allows insertion of index records into the memory buffer in high speed and in real time. When the small B-tree has reached a threshold size determined by the buffer size, the LSM-tree technique will flush this out to disk storage section-by-section, and merge the memory SB-tree with the disk-based large B-tree (LB-tree). The LB-tree can be further broken down into multiple segments, thus reducing even further the amount of I/O required to merge the trees. Since the

flushing of the SB-tree is done on a section-by-section basis, we are able to manage the LSM-tree without impeding normal operations.

NOTE: The discussion of concurrency control in this chapter is significant to OMM because the traditional approach of maintaining data integrity in databases is not sufficient for the OMM applications to ensure object level data integrity; the information of an enterprise resource may be located in different databases and be in different data formats. This chapter provides the fundamental concepts and approaches to maintain a multi-user organization management environment using OMM.

8.7 Conclusion

In this chapter, we have reviewed the basic theory behind concurrency control in information management systems. The concept of transaction, along with its commit and abort operations, is defined and discussed through some examples. Although the OMM Prototype System is an information management system aimed at managing specifically enterprise resource information, it employs the object-oriented approach in order to achieve this. As a result, simply relying on the underlying DBMS to handle concurrency issues will sacrifice usability by not being able to resolve conflicts at the object level.

In OMM, we have provided an optional mechanism to allow programmers to pre-claim object-level locks over OMM member objects. OMM only supports *read* and *write* locks. Whenever a conflict arises in a lock request which specifies the no wait option, OMM will throw an exception to the lock requester, which in turn may choose to rollback the transaction, or to resubmit the lock request. This algorithm, although simple, is sufficient to resolve all deadlock situations.

We also discuss the case in which the user would rather wait for a lock when a conflict arises. In this case, we have to deal with potential deadlocks. Deadlock detection by running a transitive closure algorithm can be very expensive especially if the enterprise resources information is distributed over multiple domains or locations. We adopt a simple timeout algorithm to resolve deadlocks. When a transaction waits for locks beyond a timeout period, the transaction is assumed to be deadlocked with some other transactions and is rolled back. In OMM, when a timeout event occurs to an object-level

lock wait, OMM will simply throw a deadlock exception and the user program can be restarted again.

We want to emphasize that our current OMM Prototype System is built on top of Oracle 9i, a reliable commercial RDBMS. As such, even if we turn off object-level locking, there will not arise fundamental data integrity issues. However, the feature of object-level locks in OMM improves user-friendliness by allowing users to explicitly lock an entire OMM member object, thus ensuring a consistent view over enterprise resources throughout a transaction.

CHAPTER 9 Implementation Experiments and Applications of the OMM Prototype System

We have implemented a prototype to demonstrate the OMM model and services thereof. This chapter is devoted to describing the software architecture of the OMM Prototype System. We will also discuss the detailed implementation and deployment experience of our Prototype System. This includes the integration with existing databases, and the population of the organization databases. Note that this prototype was implemented in 1998 in Java; since then a number of technological advances have taken place, most notably in the area of object representation, transformation and storage, such as those that involve XML, Enterprise Java Beans and Web Services.

Section 9.1 describes the OMM software prototype architecture, which includes the run-time system architecture and the breakdown of software components in the OMM Server. Section 9.2 discusses the callable user interface to allow other software programs, such as WFMS, or end-user interactive applications to interact with OMM. To enable collaborative software such as workflow and e-commerce applications to coordinate computations over an entire enterprise on top of OMM, we need to define an open and callable API for defining and manipulating the OMM conceptual entities. In Section 9.2.1, the OMM API will be discussed; the definitions are given in Appendix A. In Section 9.2.2, we will describe a simple graphical user interface that we used in the OMM Prototype System to allow users to maintain the organizational information anywhere in the world through the Internet. Section 9.3 covers the naming convention in OMM. This is important especially when we have enterprise information distributed in more than one OMM domain. In Section 9.4, the generic database schema of OMM is presented. This database schema design allows us to easily map an existing database schema to our implementation and thus simplifies the task of downloading existing data into the OMM Prototype System. We will also describe the customizable agent applications we built to automatically pull data from some existing data stores. Section 9.5 describes the practical steps to map an RDBMS database schema to the OMM database schema. This mapping allows us to retrieve organizational data from existing databases in order to populate the OMM data store. Section 9.6 provides a guided tour for users to implement an X.500 directory with OMM. Conversely, it also shows how a

user may employ the X.500 directory objects to implement the OMM conceptual entities. Section 9.7 briefly refers to the application of the OMM Prototype System in industry. In Section 9.8 we give some conclusions.

In the subsequent development of this chapter the terms database(s) and data store(s) are used interchangeably.

9.1 The OMM Prototype System Architecture

The OMM Prototype System has a backend database to store the enterprise resource information. In most cases, however, when integrating with an existing enterprise WFMS, such information about the enterprise resources is already stored in various existing databases. It is, therefore, important for OMM to be able to pull information from existing data stores instead of requiring users to re-enter the information.

At the design implementation phase of the OR cycle, the existing databases of an organization, such as the human-resource database and the corporate directory, are analyzed and mapped to an OMM organization design. Based on this mapping, the OMM *agent programs,* which make up a part of the OMM Server architecture, can populate the backend data store of OMM by accessing the existing databases. In some cases, due to the continuing usage of legacy ERP and HR applications over the existing databases of an organization, it is necessary to periodically refresh some part of the OMM data store by rerunning the OMM agent programs.

OMM is designed to work in a dynamic, real-time fashion with existing resource management applications. This integration with existing systems can take place in both moderated and direct fashion. For example, at runtime, the RM accesses the OMM organizational information and performs role resolution by calling the OMM API. The OMM Server evaluates the input from the RM, together with the rules representing the role to be resolved, and returns the result to the RM. As an example of direct manipulation, OMM provides a graphical administrative tool that users can use to call the OMM API to manage the organizational objects through a GUI interface.

Figure 9-1 shows the OMM run-time system architecture encompassing the RMs and the existing organizational databases.

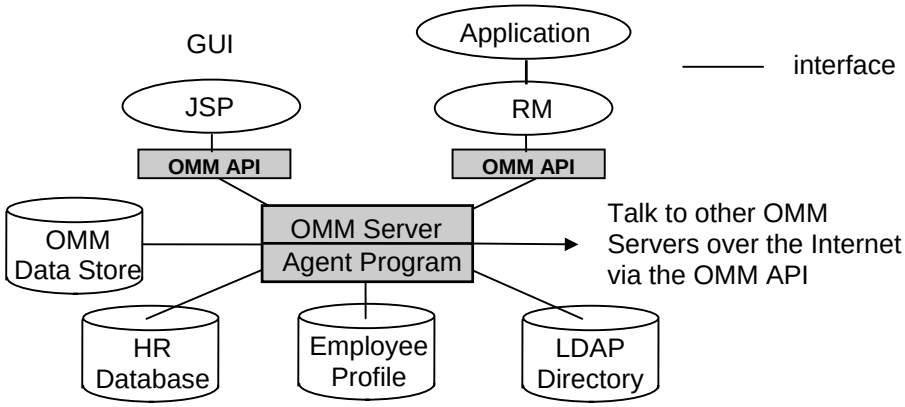


Figure 9-1 The OMM Run-time System Architecture

The OMM Server consists of three software components, which include the OMM Object Manager, the Database Connection Module, and the Service Module. The OMM Object Manager is responsible for the management of the OMM conceptual entities (or simply OMM objects), which are *organizations, members* and *virtual links*. These OMM objects are implemented in the Java classes, `OmsOrganization`, `OmsMember` and `OmsVirtualLink`, respectively. Their definitions are shown in Appendix A. The Database Connection Module connects external existing databases to the OMM data store through the JDBC standard interface. The Service Module includes the Session Manager, the Command Dispatcher and the Lock Manager.

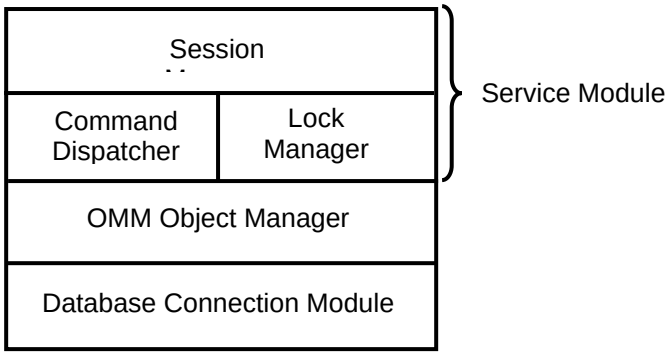


Figure 9-2 The OMM Prototype System Software Architecture

The Session Manager controls the connection between the client and the server. It performs password authentication and employs a timeout mechanism to allow garbage collection on broken connections. When a client submits a request to the server, the Command Dispatcher identifies the type of the command and triggers the corresponding modules in the OMM Object Manager. The only types of request that are not handled by

the OMM Object Manager are the `lock` and `unlock` requests; the only commands not handled by the OMM Object Manager are the `connect` and `disconnect` commands. In the case of locking interfaces, the Command Dispatcher will forward the request to the Lock Manager. The `connect` and `disconnect` commands are handled by the Session Manager.

The OMM Lock Manager grants and revokes locks on the object level. It grants available resource objects to requestors but rejects conflict lock requests. Our current implementation does not support blocking and lock waits. When conflicts occur, the Lock Manager simply throws an exception. Calling applications catching the lock exception may choose to resubmit the request or abort the operation. With this simple "no wait" mechanism, we do not have to be concerned about deadlock situations. The Lock Manager also has a timeout mechanism to clean up locks in the case where the application holding locks has crashed.

The OMM Object Manager receives requests for defining or manipulating the OMM objects. It subsequently maps these requests to the underlying database access requests, which are handled by the Database Connection Module. The Database Connection Module connects to the backend database through the JDBC standard protocol.

The core service, or function, of the OMM Server is therefore to provide services for enterprise object management. These services are categorized and described in the next section, which describes the OMM API.

9.2 User Interface for Defining and Manipulating Organizational Objects

9.2.1 Application Programming Interface

The OMM API is implemented in the OMM Prototype System as Java classes and methods to support organizational resource definition and manipulation. These classes and methods are listed in Appendix A. They can be categorized according to the object types with which they interact:

Object Type	Object Definition	Object Manipulation
Organization	create, delete	get
Member	create, delete,	get, set, move, lock, unlock
Attribute	create, delete, associate, disassociate	get
Attribute Value	none	get, set
Virtual Link	create, delete	resolve, isLink

Table 9-1 OMM Application Programming Interface Categorized by Class

The `create` and `delete` operations are responsible for creating and removing the respective objects: organizations, members, attributes, and virtual links. The `copy` command allows us to create a duplicate member object with a new identifier. The `move` action puts the member object into another OMM organization. The `get` operation allows retrieving the corresponding objects. OMM organization definitions are changed through the `associate` and `disassociate` interfaces; these are used to add and drop attributes from organizations. As a feature of this design, attribute definitions can be defined once and reused by different OMM organizations. The attribute-value `get` and `set` interfaces support manipulation of values of different data types. Attribute-value `set` operations affect only values in memory; users use the member object `set` operation to write the new attribute values back to the OMM data store. The `lock` and `unlock` interfaces allow users to grant and revoke locks on member objects. Finally, users may resolve a virtual link using the `resolve` interface, or query whether a connection exists between two objects by calling the `isLink` method. Detailed syntax and semantic definitions of these APIs can be found in Appendix A.

9.2.2 Graphical User Interface

The current OMM Prototype System also provides a tool with a simple charting interface for displaying and manipulating OMM member objects. This tool is built on top of the OMM API. Figure 9-3 shows a sample relationship graph as displayed in the OMM front end tool.

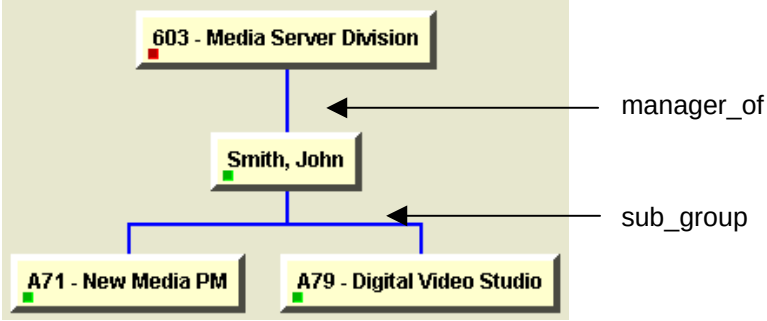


Figure 9-3 A Relationship Graph Displayed in Organization Chart Format

Two different relationships are shown here. The *manager_of* '603 - Media Server Division' is shown first on top. The *sub_group* relationship is shown at the bottom. By double clicking on a box displayed in Figure 9-3, detailed information for the corresponding resource will be displayed through a pop-up window (Figure 9-4). Users can view and update the member information through this window.

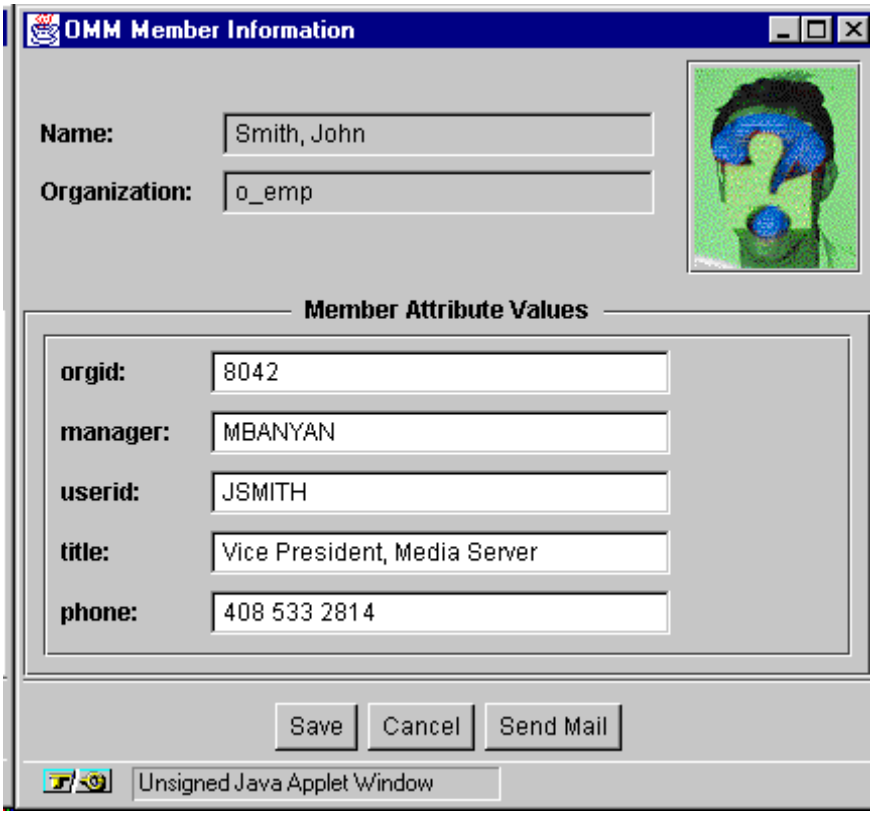


Figure 9-4 Pop-up Window Showing Member Attributes with the Web Interface

9.3 Domain UUID and Naming Convention

Each OMM Server attends clients within an OMM *domain*. An OMM domain corresponds to a physical implementation of an OMM data store. OMM Servers can exchange information across OMM domains through the OMM API; the current implementation of the OMM Prototype System only focuses on a single domain. Multiple OMM organizations may reside in a domain but an OMM organization does not span domains. A domain has a globally unique identifier while OMM organization names are unique only within a domain. However, the unique name of an OMM organization within a domain, combined with the unique domain name, must be a universally unique identifier. For instance, domain *london* and domain *seattle* may both contain an OMM organization called EMPLOYEE. The corresponding unique OMM organization names will look like this:

EMPLOYEE.*london*
EMPLOYEE.*seattle*

Similarly, although an OMM member name is only unique within an OMM organization, by concatenating the member name with the UUID of the OMM organization, we can obtain a UUID for the member as well. For instance, the OMM member names

*john_smith.*EMPLOYEE.*london*
*john_smith.*EMPLOYEE.*seattle*

are both globally unique.

In a distributed computing environment, multiple OMM Servers residing at different locations need to exchange information with one another. For all users to be able to access the same global organizational information, updates to domain information and OMM organization definitions ought to be propagated to all OMM Servers on a regular basis (such as once every hour). It is not necessary to replicate updates of OMM members, virtual links, or attributes outside of a domain — the OMM organization UUID will indicate which server manages the underlying information. Based on this UUID, the local server may retrieve data from any remote OMM Server.

9.4 Information Exchange with Existing Databases

Most companies have accumulated huge amounts of enterprise resource information over the years, some in legacy databases, and others in ERP applications, data warehouses, corporate infrastructure directories, and historical databases. In this section we describe the generic database schema used in the OMM Prototype System. The OMM database has been designed with the aim of facilitating the job of mapping OMM to each of the above potentially existing databases. In Sections 9.5 and 9.6, we will discuss our practical experience of mapping OMM to some existing data sources.

9.4.1 Generic Database Schema for Mapping Existing Databases to OMM

In our current OMM Prototype System implementation, Oracle 9i on Windows 2000 is used to store the OMM data. In the database schema for this prototype, each OMM organization is represented by 5 tables, each of which holds different data types. These data types include character string, integer, float, date and raw (bitmaps and bit streams). Table 9-2 describes the definitions for all of these tables. Note that the *id* value at the end of the table name is the ID of the corresponding OMM organization:

OMM_Char_*id*

Column Name	Data Type	Column Description
Member ID	Integer	The unique member identifier.
Attribute ID	Integer	The attribute identifier.
Attribute Value	Varchar2	The attribute value as character string.

OMM_Int_*id*

Column Name	Data Type	Column Description
Member ID	Integer	The unique member identifier.
Attribute ID	Integer	The attribute identifier.
Attribute Value	Integer	The attribute value as an integer.

OMM_Float_*id*

Column Name	Data Type	Column Description
Member ID	Integer	The unique member identifier.
Attribute ID	Integer	The attribute identifier.
Attribute Value	Decimal(2)	The attribute value as a decimal.

OMM_Date_*id*

Column Name	Data Type	Column Description
Member ID	Integer	The unique member identifier.
Attribute ID	Integer	The attribute identifier.
Attribute Value	Date	The attribute value as an Oracle date data type.

OMM_Raw_*id*

Column Name	Data Type	Column Description
Member ID	Integer	The unique member identifier.
Attribute ID	Integer	The attribute identifier.
Attribute Value	Long Raw	The attribute value in long raw type (support up to 2GB of data). This is used to store images, documents and other bit streams.

Table 9-2 The Database Schema Used in OMM to Store an OMM Organization

We note that `Varchar2`, `Decimal(2)` and `Long Raw` are data types encountered in Oracle 9i. With this approach, all attribute values that are of the same data type will be stored in the same table. For instance, the *Title* and *Home_Address* attributes (see Table 4-7 in Chapter 4) of an employee are both of type character, and they are both stored in the same OMM_Char_*id* table.

This database design is adopted because it simplifies the task of mapping the above schema to any existing relational database schema. The primary job of the database designer is reduced simply to identifying the mapping between the OMM organizational attributes and the legacy database table column names. Once this mapping has been completed, the *mapping agent applications* are run to pull legacy data from existing databases into sets of 5 tables that form the OMM database. These mapping agent applications split up the incoming data according to the data types of the attributes. They are described in the next section.

9.4.2 Mapping Agent Applications

The mapping agent applications are Java programs written to pull legacy data from some existing data stores into the OMM repository. Our current Prototype System uses Oracle DBMS as the OMM repository, but the implementation can indeed support any JDBC-compliant RDBMS. On the other hand, existing enterprise data may be stored in any RDBMS, such as Oracle, Sybase, Informix or SQL Server. The existing enterprise data may also be corporate infrastructure data stored in other types of data management systems such as the LDAP Server. Furthermore, there might be enterprise resource information that exists as unstructured data, such as spreadsheets, Web pages, PDF files, and other types of non-RDBMS data. In our OMM implementation experiments, we have written template applications to pull data from RDBMS through a JDBC interface, as well as from an LDAP directory server through the LDAP API. In the case of LDAP

directories, the mapping agent application is written in Java and calls the Java LDAP API to accomplish the job.

Figure 9-5 shows a mapping agent application and its input and output:

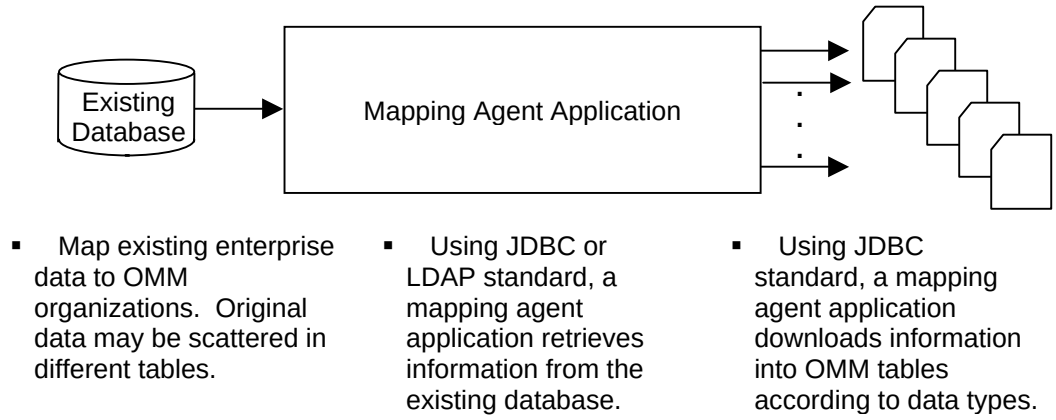


Figure 9-5 An OMM Mapping Agent Application

Note that when using the mapping agent application programs to download data from a legacy database to the OMM data store, we do not have a simple approach to migrate also the legacy database applications to some OMM methods. Rather, new OMM methods will have to be developed to assume the functions that the legacy database applications perform on the legacy database.

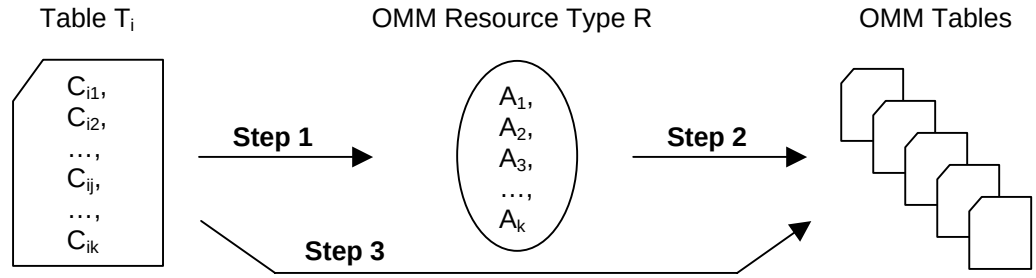
9.5 Mapping the OMM Database Schema to Other Relational Database Schemas

The OMM information model encompasses aspects of the object-oriented data model as discussed in Chapter 4. Enterprise resources in OMM are stored as objects, each of which maps into 5 tables in the backend database (see Section 9-4). Once the organization designer has identified and modelled a type of enterprise resource, such as employee, department, project, or product, s/he must determine whether the data for that resource type is currently held in existing data stores. If so, it is desirable to download such data into OMM, rather than requiring users to re-enter such data. As a general rule, regardless of how disconnected the enterprise resources may be, most of the underlying resource information will exist in one form or another. In fact, most of the desired data will probably exist in one or more RDBMS. For this reason, it is important to have a

methodology to map an existing RDBMS schema to the OMM generic database schema (as presented in Section 9.4.1).

An example will help in describing the methodology we use in performing this mapping. Assume the data of a resource type, say R, is stored in N relational tables, T_1, T_2, …, T_N, each of which contains some or all of the column names, C_{i1}, C_{i2}, …, C_{ik}, that are of interest to OMM in terms of constructing the attributes of R, namely A_1, A_2, …, A_k. Each table column name, C_{ij}, corresponds to one and only one attribute definition, A_i, in the context of OMM. In other words, there is a one-to-one mapping between certain columns in T_i, i = 1, 2, …, N, and the defined attributes of R. Finally, as each of the attributes of R is stored in one of the 5 OMM tables based on data type, the mapping of C_{ij} to A_i results in downloading the data of C_{ij} into the OMM table that stores the values of attribute A_i.

Figure 9-6 shows the mapping of RDBMS table columns into OMM attributes:



Step 1: Identify column names in RDBMS tables, each of which corresponds to one attribute in R.

Step 2: Identify the OMM table which stores the values of attribute A_k.

Step 3: Map the column, C_{ij}, corresponding to A_k into the OMM table. Run the mapping agent program to download data.

Figure 9-6 Mapping RDBMS Column Names to OMM Attributes

9.6 Mapping OMM Objects to X.500 Directory Objects

The directory service is one of the most fundamental components in a distributed networking environment. Directories are specialized databases designed to hold network and organization information for quick lookups. Through such services, human users and applications are able to efficiently locate desired information.

X.500, as defined by the ISO and the CCITT in the 1988 document (CCITT 1988) and the 1993 extensions (Radicati 1994), specifies a directory service on the logical and physical levels. The specification includes the Directory Information Tree (DIT) hierarchy, the directory access and service protocols, and the requirements for distribution and replication of the directory information and services. Although the deployment of X.500 is still in progress, many companies have accepted it as a standard and the preferred model for naming services.

OMM is a generic and abstract model to describe organization resources and their hierarchical or other arbitrary relationships. It also specifies a protocol for manipulating and exchanging directory information between services and user agents. OMM is extensible and dynamic in nature. It is particularly strong in defining flexible relationships to support collaborative computing environments such as workflow, publish-and-subscribe, and Internet-type applications.

Although OMM is richer in functional features than X.500 and is designed to support today's dynamic business environment, for practical reasons, we cannot ignore the existing X.500 directories. Rather, we need to consider how to integrate OMM with the existing installed base of X.500 data stores and leverage the large number of corporate directories that have been implemented using it as a standard.

This section deals with the mapping between the X.500 directory objects in the DIT and the conceptual entities in OMM. In Section 9.6.1, we will present the information model of X.500. In Sections 9.6.2, 9.6.3, and 9.6.4, we will describe the mapping methodologies from OMM to X.500 and vice versa.

9.6.1 The X.500 Directory Model

The DIT is the logical representation of the information database in X.500. A list of classes corresponding to different levels of resources in a company hierarchy is defined in the DIT. (The later X.521 recommendation adds more object classes and attributes for naming purposes (CCITT 1998).) In this section, we will use an example which deals with some very basic X.500 objects to illustrate how they are mapped to OMM objects. The algorithm given, however, is applicable to more complex tree structures. Refer to Section 9.6.5 for a discussion of the reverse mapping, which goes from OMM objects to X.500 directory objects. Figure 9-7 shows a sample DIT.

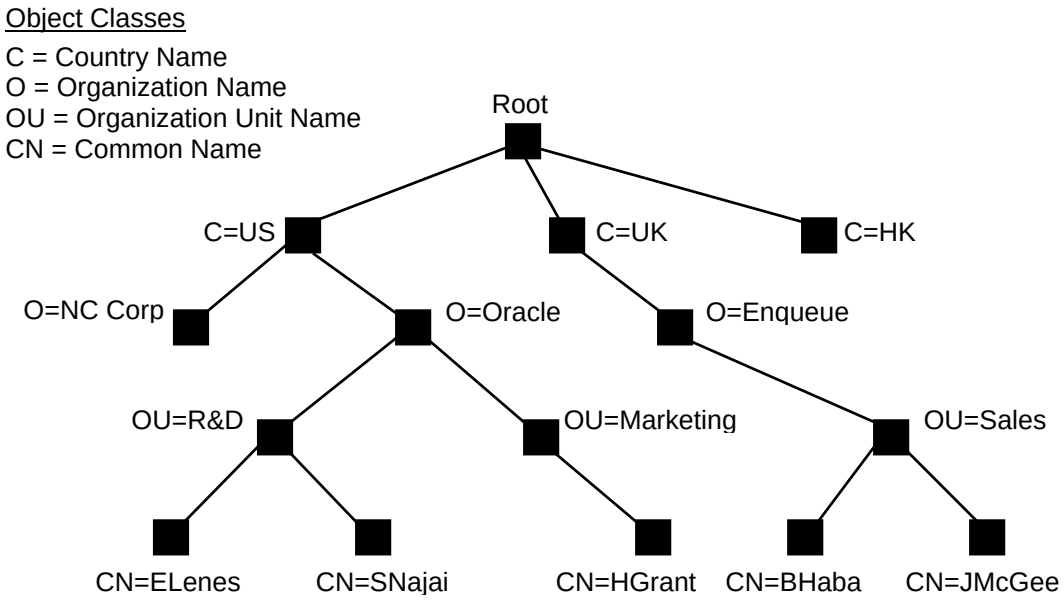


Figure 9-7 A Sample DIT

The sample DIT has 4 levels under the root, namely *countries (C)*, *organizations (O)*, *organization units (OU)*, and *common names (CN)*. Each of these is an object class in X.500.

9.6.1.1 Object Classes

In DIT a hierarchy of object classes is defined. In the example shown in Figure 9-7, the hierarchy is simply: C — O — OU — CN. Expressed in another way, each level represents an object class. In X.500, each directory entry (a node in the tree) must belong to an object class. An object class has a name (e.g. Country) and a set of attribute definitions. A class can be a subclass of another class, in which case it will inherit the attributes of its superclass.

9.6.1.2 Directory Entries

Each node in the DIT is a directory entry, typically representing a country, a company, a department, a user, a machine, a network resource, or a group of users. The lower the tree, the finer the granularity of the entity. Each entry in the DIT has a Relative Distinguishing Name (RDN). The sequence of RDNs from the root to the directory entry

under consideration represents the object's unique name within the directory. For example, *US.Oracle.Marketing.HGrant* is the unique name for a person.

X.500 incorporates the concept of *aliases*. Using an alias, which is a directory entry itself, users can define a short cut to a directory object. A one-way logical pointer is stored in the alias to quickly find another directory entry. This shortens the potentially long name given by the complete sequence of RDNs. We note that in X.500 an alias is only a static link and is in one direction; there is no back pointer from the entry to the alias. Furthermore, the alias has to be updated manually every time the location of the entity that it references changes.

9.6.1.3 Directory Entry Attributes

Each directory entry has a number of attributes. These attributes are defined in the object class to which the directory entry belongs. For example, the object class *organization unit* may have 5 attributes: manager, address, phone, fax, and URL. An attribute has a type and a set of values. The attribute type is used as an identifier of that attribute within the global name service. X.500 supports attributes with single or multiple values.

9.6.2 Using OMM Organizations to Model DIT

As described in Chapter 4, OMM has three basic conceptual entities to represent a corporation; they are the *organizations, members*, and *virtual links*. The OMM members are related to one another through *virtual links*. OMM members are contained in OMM organizations and are used to store attribute values.

Using the OMM conceptual entities, we can model a DIT by simply defining an OMM organization for each level of the DIT *except the root*. Figure 9-8 shows a DIT modelled by OMM organizations.

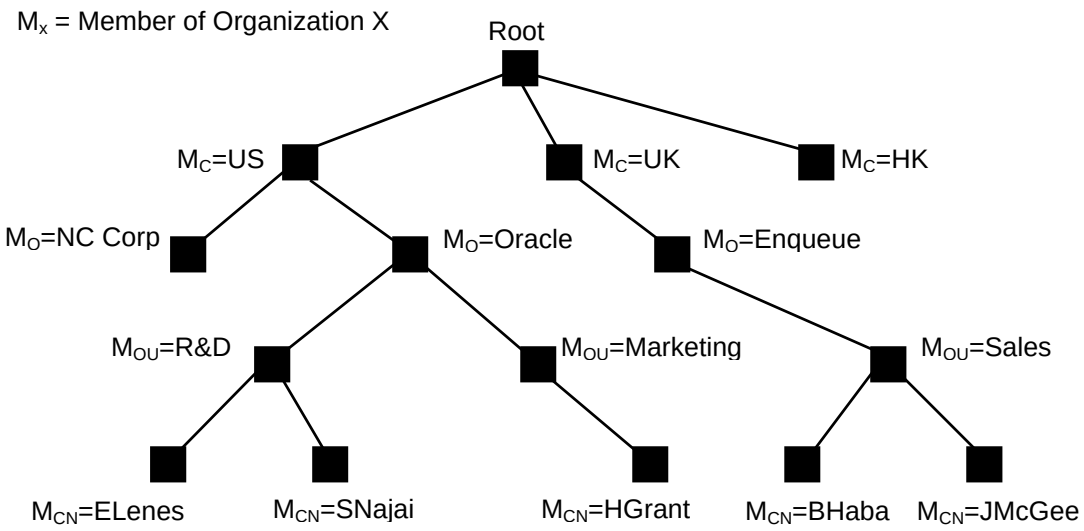


Figure 9-8 Sample Corporation Tree Modelled by OMM

9.6.3 Using OMM Objects to Implement X.500 Objects

Once the OMM organizations are created to represent the DIT, we can start converting each of the X.500 objects into an OMM object. To map X.500 objects into OMM objects, we have to take care of the object classes, the attribute definitions, the directory entries, and the naming hierarchy. The following rules describe the mapping procedure:

Rule 1: Replace each object class in the X.500 DIT by an OMM organization as described in Section 9.6.2.

Rule 2: For each attribute defined in the attribute set of the X.500 object class, add a corresponding attribute definition to the OMM member class.

Rule 3: Replace each directory entry (node) in the X.500 DIT with a member object in OMM. Note that the directory entries must belong to an object class in X.500; similarly, the corresponding OMM member object must also belong to an OMM organization that is created according to rule 1.

Rule 4: For every connection between two levels in the DIT hierarchy, define a virtual link between the *parent* and *child* objects in the corresponding OMM structure. This may require adding attributes to the OMM organization in order to construct the virtual link expression.

Rule 5: For every OMM member object except the root, add a *parent* attribute to store the name of the parent of the directory entry that this member object represents.

Table 9-3 The Rules for Mapping X.500 Objects to OMM Objects

Using the example in Figure 9-8, let us examine how to apply these rules:

Rule	Apply the Rules to the Scenario in Figure 9-8
Rule 1	Create 4 OMM organizations to represent each of the 4 levels in the DIT: Country, Organization, Organization Unit, and Common Name, respectively.
Rule 2	Add attributes to each of the 4 OMM organizations based on the class definition of each level in DIT. For instance, the OU class contains attributes: manager, address, phone, fax, and URL. Define these same attributes in "Organization Unit" in OMM.
Rule 3	Now, for each X.500 object in the DIT, replace it by defining an OMM member object. Since the attributes representing the X.500 object are the same as those for the OMM member object, there is a one-to-one mapping of the attribute values.
Rule 4	Define virtual links to connect member objects in each OMM organization to the member objects in the next OMM organization in order to reproduce the hierarchy defined by the DIT. For example, between the Organization and Organization Unit, a virtual link is defined to represent how member objects at the Organization Unit level are connected to member objects in the Organization superclass. To define this virtual link, we may need to define a new attribute in one or more OMM organizations. For instance, we will need to add an attribute "organization_name" in the Organization Unit class such that a rule can be defined for the virtual link between this class and its superclass.
Rule 5	To each OMM organization, add an attribute called "parent". The value of this attribute in each of the OMM member objects will record the name of its immediate parent in the DIT. Note that in the simplest scenarios, Rule 4 can be satisfied by defining a virtual link using the attribute "parent".

Table 9-4 Examples for Applying Rules to Map X.500 to OMM

Applying these five rules to an X.500 DIT and directory information database will result in a corresponding OMM organizational database. These rules represent a simple mapping design. Users can apply a more complex mapping design whereby a single X.500 object class may be mapped to multiple OMM organizations. This adds flexibility for directory migration and increases autonomy in cases where a branch of a company has decided to change its own information structure.

Since different subtrees within a DIT can be partitioned into different Directory Service Agents (DSAs) located in different physical locations, when mapping an X.500 object to a corresponding OMM object, one may want to take this physical partitioning into consideration. In such cases, even if directory entries belong to the same object class, if they are partitioned into separate DSAs, we can create separate OMM organizations to hold the directory entries. This provides a flexible way to not only map the X.500 objects on a logical level, but it also enables mapping based on physical partitioning. Figure 9-9 shows an example of DSA partitioning and the corresponding OMM organizations mapping.

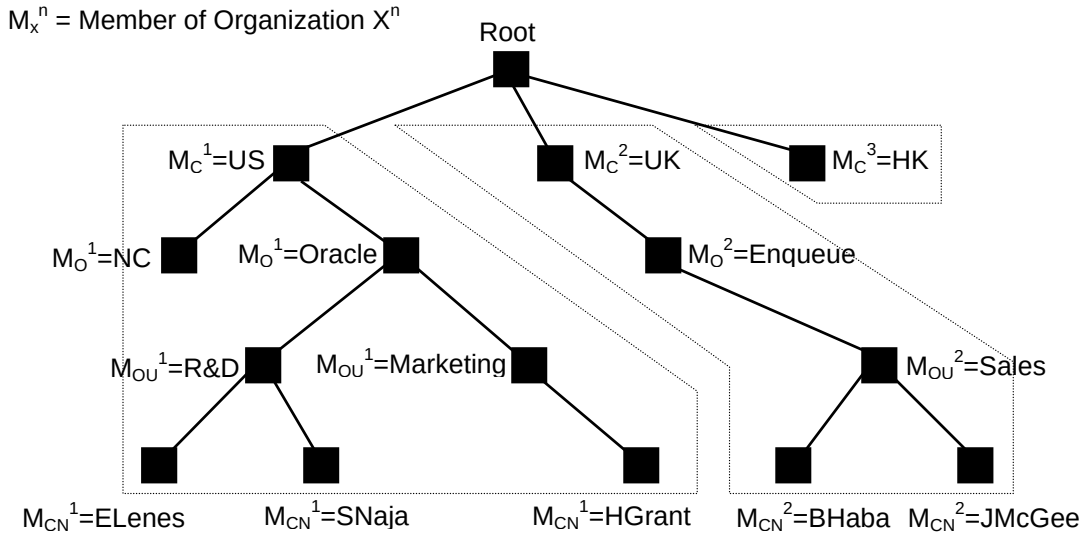


Figure 9-9 Partitioned OMM Organizations

Aliases in X.500 can be represented by virtual links in OMM. Depending on the design of the X.500 DIT structure rules, an alias may point to the same object class to which it belongs, or it may point to another object class. In either case we can map the object class to an OMM organization by following the method indicated in Section 9.6.3. After that, an alias entry is replaced by an OMM member object. A virtual link is then defined between this OMM member object and the directory entry that it is referencing. Since X.500 aliases are static, this virtual link will also be static and has to be cleaned up manually when the referenced entry is deleted or moved, just as it is in the case of X.500.

9.6.4 Using X.500 to Implement OMM Conceptual Entities

The alias is the only method that X.500 uses to create relationships. As such, X.500 is quite weak in describing relationships between resource objects. Dynamic relationships implemented by virtual links in OMM cannot be represented by using X.500 objects. As a result, virtual links will not be considered when mapping from OMM to X.500.

Although virtual links in OMM cannot be mapped to X.500, OMM organizations and OMM member objects, as well as the attributes thereof, can be mapped simply and efficiently. Since OMM organizations and OMM members are able to represent all resource types, we can define OMM organizations to represent *groups* within a company. In the following example, a group is a virtual entity which holds together

multiple employees from across the company. Although the group concept is very commonly used for constructing project teams in many companies, X.500 does not have a way to implement groups. In OMM, a group is represented handily by an OMM organization, with virtual links connecting the group to all its members. However, because it is unable to represent virtual links, as discussed above, X.500 is not able to deal with OMM member objects that are used to represent groups.

Figure 9-10 shows how groups and relationships are used in OMM to model the previous example.

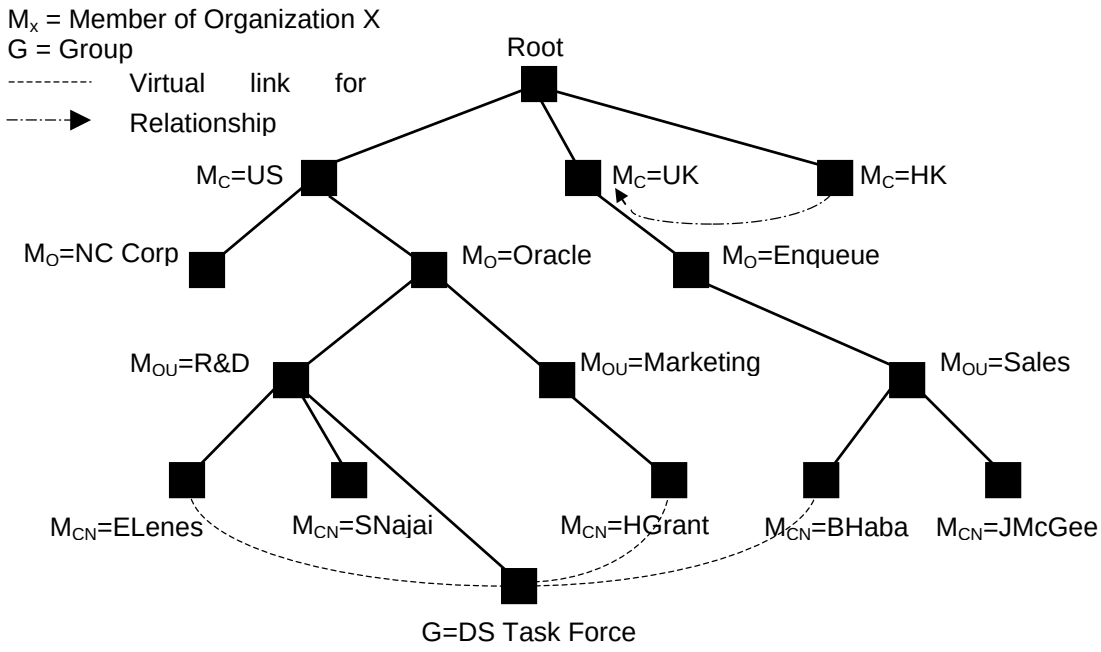


Figure 9-10 A Sample Corporation with Groups and Relationships Modelled by OMM

The mapping methodology to bring an OMM system into X.500 format can be defined almost as a reverse of the rules outlined in Section 9.6.3.

Rule 1:	Replace each OMM organization and its OMM member class with an object class in X.500.
Rule 2:	For each attribute defined in an OMM member class, add a corresponding attribute definition to the X.500 object class.
Rule 3:	Replace each OMM member object with a directory entry in X.500.
Rule 4:	For each virtual link defined in OMM, construct multiple static hierarchical relationships in X.500, or simply drop it.

Table 9-5 The Rules for Mapping OMM Conceptual Entities to X.500 Objects

As discussed earlier, when organizational information changes, relationships between organization objects may change also. OMM, with its dynamic virtual links, is designed to reflect such a changing environment. Under the X.500 implementation, on the other hand, information managers must manually update the system with these changes so as to reflect the current state of the organization.

Although groups cannot be easily modelled by X.500, one can always create an X.500 directory entry with a fixed number of positions (attributes) in it. When an organization object (e.g. a person) is assigned to take a position, we simply fill the corresponding attribute with the distinguishing name. Furthermore, since the attribute definition cannot be changed dynamically in X.500, once the group is created, users can no longer add or delete positions.

9.6.5 Other Considerations

The X.521 extension recommends the *locality* object class be added to the DIT. It also adds a number of common names to capture devices, residential persons, organization persons, and roles. All of these are represented as directory entries in the DIT. As a result of this, the mapping rules described in Section 9.6.3 will still apply when mapping an X.521 DIT to an OMM structure.

As shown in Sections 9.6.3 and 9.6.4, the mapping from X.500 to OMM can go both ways. However, from a functional point of view, OMM is a superset of X.500. Mapping from OMM to X.500 will mean losing some of the OMM functional features, particularly in the areas of capturing dynamic relationships and link management. One

example of the type of functionality that will be lost is the seemingly simple task of determining how many groups a member is associated with. In an OMM system, this is a simple task. Under X.500, finding the answer to this question is very difficult, or even at times impossible.

Another important difference between the two systems lies in the area of role resolution. Roles in X.500 are simply represented by attribute values, while OMM supports dynamic role definition and resolution, which is required in workflow and other groupware.

It should be noted that both OMM and X.500 support single and multiple value attributes. This is an important feature that allows organizational resources to have a number of titles or to allow the capture of any information that may consist of multiple values.

The mapping methodology described in this section has no performance degradation implications. Performance of the directory services is implementation-dependent. From the architecture point of view, there is no clear difference in performance between X.500 and OMM.

9.7 Application of the OMM Prototype System in Industry

We have applied OMM to support two separate corporations in their businesses. In Chapter 5, we described the OMM/P&S prototype, which is being used by InsurePoint to support their e-commerce applications. In addition, we have applied OMM to perform enterprise modelling for Hitachi America. In this section, we will discuss details of applying OMM to Hitachi America, in order to model its organizations and the relationships between its employees and the various business units.

9.7.1 An OMM Prototype System to Support Enterprise Modelling in Hitachi, America

This prototype application focuses on modelling and capturing information regarding the divisions, departments and employees of the company. By defining some business policies over these classes, hundreds of graphical models are generated dynamically, representing the connections between divisions, departments and employees. Here is the breakdown of the estimated number of objects in the pilot database.

	COMPANY	REGION	DIVISION	DEPARTMENT	EMPLOYEE
Total No. of objects	3	5	10	250	6000

Table 9-6 Estimated Number of Resource Objects in Hitachi America

This modelling effort may eventually expand to cover other types of objects such as services, machines, products, subsidiaries, applications, projects, and many others.

9.7.2 OMM Organization Definitions

We define five OMM organizations to represent the five resource types shown in Table 9-6. Based on the OMM model, a type-level relationship can be defined between OMM organizations. Applying the notation of Chapter 4, the relationships between the OMM organizations are shown in Figure 9-11 (cf. Figures 4-1 and 4-5).

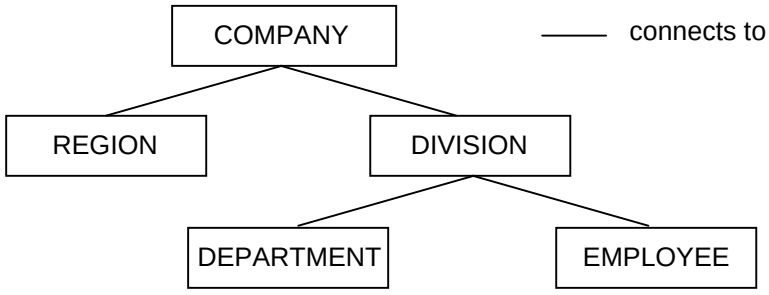


Figure 9-11 OMM Organizations Defined for Hitachi America

OMM organizations are defined to map to the various resource types. The organizational relationships provide users with the context of how a resource type is related to other resource types in the enterprise. Please refer to Section 4.3.2 for a detailed discussion of type-level relationships between OMM organizations.

Tables 9-7 to 9-10 show all the OMM organizations and the associated attribute definitions. Note that all organizations implicitly contain the *name* and the *identifier* attributes, which are OMM system-defined attributes (see Figure 4-7). The asterisk (*) indicates that the attribute is a required field. Again, "constraint" stands for the attribute's value constraint as discussed in Section 4.4.1.

Attribute Name	Data Type	Constraint	Description
Region	String	many-to-one	The region name.
Manager	String	many-to-one	The manager of this company.
Address	String	many-to-one	Street address.
City	String	many-to-one	City.
State	String	many-to-one	State.
Zip	String	many-to-one	Postal zip code.
Country	String	many-to-one	Country.

Table 9-7 Attribute Definition of the COMPANY Organization

Attribute Name	Data Type	Constraint	Description
Manager	String	many-to-one	Manager of the region.

Table 9-8 Attribute Definition of the REGION Organization

Attribute Name	Data Type	Constraint	Description
Company*	String	many-to-one	Company name that this division belongs to.
Manager*	String	many-to-one	The manager of this division.
URL	String	many-to-one	The URL of the division.

Table 9-9 Attribute Definition of the DIVISION Organization

Attribute Name	Data Type	Constraint	Description
DivisionName*	String	many-to-one	Representing the division this department belongs to.
ParentDepartment	String	many-to-one	Support for nested departmental structure.
Manager	String	many-to-one	The manager of this department.

Table 9-10 Attribute Definition of the DEPARTMENT Organization

Attribute Name	Data Type	Constraint	Description
EmpolyeeNo*	String	one-to-one	Employee number.
LastName*	String	many-to-one	Last name.
FirstName*	String	many-to-one	First name.
MiddleInitial	String	many-to-one	Middle initial.
EmpType	String	many-to-one	Employee type.
Title	String	many-to-many	Title.
DivisionName	String	many-to-one	The division this person belongs to.
DepartmentName	String	many-to-one	The department this person belongs to.
OtherDepts	String	many-to-many	Matrix reporting.
HireDate	Date	many-to-one	The date this person is hired.
Type	String	many-to-one	Representing the employment type.
WorkPhone	String	many-to-one	Office telephone number.
HomePhone	String	many-to-one	Home telephone number.
Fax	String	many-to-one	Fax number.
Email*	String	one-to-one	The electronic mail address.
Address	String	many-to-one	Home street address.
City	String	many-to-one	Home city.
State	String	many-to-one	Home state.
Zip	String	many-to-one	Home zip.
Picture	Bitmap	one-to-one	The picture of this person.
Manager	String	many-to-one	The immediate manager of this person.
Supervisor	String	many-to-many	Matrix reporting.
Skill	String	many-to-many	Job-related skills.

Table 9-11 Attribute Definition of the EMPLOYEE Organization

9.7.3 Organizational Relationship Modelling

Organizational rules are defined to relate the different resources together. Consequently, dynamic graphical models can be created on top of these organizational rules.

Six organizational rules are defined in this project, representing the relationships between the different types of resources.

Virtual Link Name	Virtual Link Expression *(Organization Scope)*	Description
com_reg	Region == $owner.om_name *(COMPANY)*	Relationship between the region and the companies.
com_div	Company == $owner.om_name *(DIVISION)*	Relationship between the company and the divisions.
div_subdept	DivisionName == $owner.om_name *(DEPARTMENT)*	Relationship between the division and the departments.
dept_mem	DepartmentName == $owner.om_name *(EMPLOYEE)*	Define who (people) are working in a department.
dept_subdept	ParentDepartment == $owner.om_name *(DEPARTMENT)*	Define the nested departmental structure.
emp_mgr	Manager == $owner.om_name *(EMPLOYEE)*	Define the supervisor-subordinate relationship between employees.

Table 9-12 OMM Virtual Links Representing Relationships in Hitachi America

We refer readers to Chapter 5 for a detailed discussion of the Virtual Link syntax. These six organizational rules represent thousands of relationship instances which exist between the companies, regions, divisions, departments and employees within Hitachi America. Using our graphical tool, we are able to generate thousands of organizational charts to represent the relationships between the various resources. These charts change dynamically when the underlying organizational information changes.

In this case study, OMM is applied to model five resource types in the enterprise. Using the OMM organizations, we define attributes to characterize each of these resource types, and the relationships that exist between them. OMM members are used to represent instances of resources in each type. Finally, virtual links are used to define the instance-level relationships between the resources.

9.7.4 Using Virtual Links to Support Workflow

A simple workflow application is developed in this OM environment to allow employees to request for vacations and seek approval from their corresponding managers. Using

Petri-Net like notation as in Figure 3-2, this simple workflow is represented in the following figure.

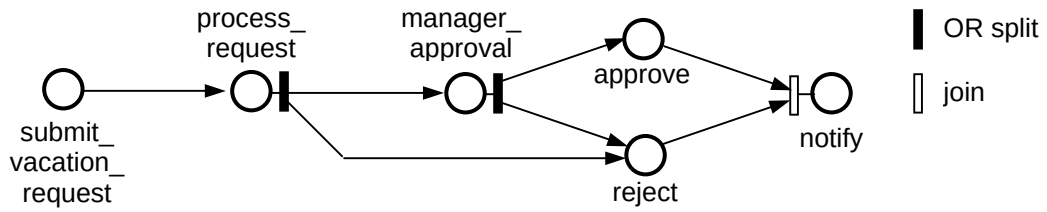


Figure 9-12 Vacation Request Process

In this workflow, the *process_request, approve, reject* and *notify* steps are automated steps that can be executed by software programs without any human intervention. The initial step, *submit_vacation_request*, can be executed by any resource in the EMPLOYEE organization. Once an employee initiated a particular *vacation request process* instance, only his or her manager is authorized to execute the *manager_approval* step. This task authorization is accomplished by resolving the virtual link *emp_mgr*, which is defined in Table 9-12.

9.8 Conclusion

In this chapter, we discussed our design and implementation experience gained while implementing the OMM Prototype System. This prototype is a Java software application, which incorporates all the salient design features of the OMM model. It allows users to define and manipulate organizational resources representing the real entities of their enterprise. We discussed the software architecture of the OMM Prototype System. Applications, such as WFMS, can be layered on top of OMM to perform workflow-related functions amongst the enterprise resources. These applications interact with OMM by calling the OMM API, a collection of more than 50 Java methods; these are listed in Appendix A.

We also implemented a simple graphical tool to allow us to view and manipulate the resource object information. This tool provides users with a graphical means to review the dynamic relationships between objects.

We also discussed the methodology of integrating existing data stores into the OMM data store.

In Section 9.5, we presented the steps for mapping RDBMS records into OMM. OMM has a database design that is particularly easy to map into from any relational database design. Each OMM organization is represented by 5 database tables, each of which holds one of the different data types that OMM currently supports. These data types are character string, integer, float, date and bit streams (for holding such things as voice and images). Column names in the relational tables of existing databases can be mapped directly into the OMM tables based on their data types.

We discussed the mappings between X.500 and OMM in Section 9.6. From the functional point of view, OMM is a superset of X.500. OMM can be used to represent all information contained in an X.500 system. However, when we try to do the reverse, that is when we implement the OMM conceptual entities with X.500, the dynamic relationships in the OMM environment are lost. In addition, the dynamic groups or project teams concept, which can be implemented easily in OMM, are difficult or even impossible to replicate in X.500.

In the course of our research, we have used the OMM Prototype System to perform organization modelling for Hitachi America. Hitachi America is an engineering company with 5 regions, 10 divisions containing about 6,000 employees and over 250 departments. We defined 6 organizational rules, or virtual links, to represent the instance-level dynamic relationships between the operating companies, regions, divisions, departments and employees. This attests to the versatility and flexibility of OMM.

CHAPTER 10 Concluding Remarks and Further Research

The work in this thesis has been motivated by the need for a formal model to describe enterprise resources and the way that they are shared and used across the enterprise. This need has become more apparent as system integrators have attempted to provide computing solutions that allow users to work collaboratively over the Internet. This thesis has proposed a modelling methodology and an organization reference model, OMM, to be used for modelling different types of enterprise resources. It also provides a rule-based relationship model to define the interactions and relationships between these resources.

We have discussed the application of our work to role resolution problems in WFMS. The dynamic role resolution features of OMM allow a WFMS to flexibly assign work tasks to authorized users based on the roles they play under different business contexts, even in the case where the company's personnel hierarchy and business conditions are constantly changing.

The thesis has described the OMM organization modelling methodology and shown how to apply it to support the various approaches and techniques used in the different phases of organization reengineering.

One feature of OMM is that it allows employees of all levels to maintain the organizational resource information for which they are responsible (such as their corporate directory information). This feature allows OMM-based data to be used in the continuous cycle of organization reengineering. OMM accomplishes this by providing an API and a GUI application that allows users to view and to update the information in OMM over the Web.

Since OMM supports multi-user access, it must address concerns of concurrency control. The thesis discusses the issues of concurrency and deadlock in the context of OMM, and details our solutions to these problems.

We have discussed our implementation experience with the OMM Prototype System. This prototype allows us to create and manage the various objects in the OMM environment, including organizations, members, attributes, roles and virtual links. Using

the OMM software, we have modelled Hitachi America. In addition, we have developed a prototype, OMM/P&S, a publish-and-subscribe software based on the OMM model. This prototype is used by InsurePoint Company to model and facilitate marketing interactions between the company and its customers.

A practical challenge that must be addressed when deploying OMM is the downloading of existing organizational information into the OMM data store. To meet this challenge, we have defined a relational database schema to store the OMM objects representing the various enterprise resources. In the thesis, we have shown how our methodology maps an RDBMS schema and X.500 directory into an OMM schema. This allows us to quickly migrate existing databases into the OMM data store.

10.1 Organization Modelling Principles

In Chapter 2, we discussed four general modelling principles suggested by Ross and Schoman (Ross and Schoman 1977), and four additional principles suggested by Vernadat (Vernadat 1996) that are particular to organization modelling. In this section, we review the OMM model with respect to these principles.

According to Ross and Schoman (Ross and Schoman 1977), any modelling technique is characterized by four principles:

- *The purpose of the model:* The OMM model has a clear purpose in mind, and that is to support organization modellers in the performance of organization modelling, focusing specifically in the areas of defining enterprise resources and their relationships.

- *The scope of the model:* In Chapter 2, we point out that the scope of organization modelling covers the *what, how, when,* and *who*. The OMM model concerns mainly the *what* and *who*. OMM has a generic reference model that allows the inclusion of all enterprise resources into its organization model. The OMM model includes strong role resolution capabilities, applicable mainly to human resources, that allow the system to dynamically define the roles that resources play in different business contexts. With our approach, we decouple the process model, which focuses on the *how* and *when* aspects of the enterprise, from the organization model, which focuses

on the *what* and *who*. We assume that the process model, which is concerned with the functional details of the various business processes, as well as the business policies governing the conditions under which process steps are initiated, is adequately handled by any of a number of existing WFMS.

- *The viewpoint of the model:* OMM focuses on modelling organizations from the perspective of management and decision-makers. The hands-on aspect of the organization, which relates to the operational details of the organization, is left out in our model. We rely on other workflow models to describe the execution details of the enterprise. As a result of this, OMM has an open architecture which allows it to integrate with other workflow system models. Once the organization model is completed, workflow designers can define business processes on top of OMM.

- *The detailing level of the model:* This defines the level of precision or granularity of the model regarding the reality being modelled. With the object-oriented approach and the horizontal and vertical partitioning techniques, OMM allows the organization modeller to define the different aspects of the enterprise to any desired granularity.

Furthermore, Vernadat suggests four additional principles to be considered particularly for organization modelling (Vernadat 1996):

- *Principle of modularity:* OMM is modular in nature. It is made up of some basic building block entities, namely *organizations, members,* and *virtual links*. An organization is a container which holds enterprise resources of the same type. Each resource object in an organization is termed a member. Members are related to one another through virtual links. With this generic model, OMM can be applied to the modelling of complex enterprise structures.

- *Principle of model genericity:* With the object-oriented approach of OMM, it is relatively easy to use the model to represent different aspects of the enterprise. We can define superclasses to group together similar resource types, and use subclasses to define particular classes of resources.

- *Principle of reusability:* Because it is an object-oriented system, OMM member classes can be reused and easily modified for different types of resources. Our approach of abstracting individual relationships into relationship types allows us to

define similar relationships across the enterprise without hardwiring them one by one. Reusability in OMM actually extends down to the member attribute level — member attribute definitions can be copied from one organization and associated with another, thus eliminating the need to define every attribute for every organization.

- *Principle of process and resource decoupling:* In order to preserve operational flexibility it is important to separately consider the actions that are being performed (the business processes) and the agents performing them (the resources). OMM accomplishes this by decoupling the organization model from the process model.

In this thesis, we have shown that all these principles, which are critical for supporting a flexible and robust OR cycle, are fulfilled in OMM.

10.2 Review of Aims and Accomplishments

The thesis has presented a conceptual model and a methodology for organization modelling. From the standpoint of organization reengineering, our goal is that the proposed model will aid designers to achieve successful organization modelling. The creation of the organization model is, of course, a critical step that must be completed before it can be used as the foundation on which the WFMS is built. In Chapter 2, we present the four attributes, which constitute the criteria of success in OR, namely scalability, extensibility, flexibility, and performance.

10.2.1 Scalability

In the thesis, we discussed how we designed OMM on top of an RDBMS. Current relational database technology can easily manage up to multi-tera bytes of data. Assuming, for the purposes of discussion, that we are managing 2 million objects, each of which has 200 KB of associated data, we will then be dealing with 2 GB of data. With today's storage and database technology, this is not a challenge at all. Therefore, scalability in this dimension is very manageable. In addition, our database schema is unique in that if an object does not have a value in a certain attribute, it will not take up any space in the database file. This approach has significant space saving advantages as

compared to the standard relational database approach of defining tables with NULL values for empty fields.

More importantly, OMM is able to abstract relationships between millions of objects into a handful of rules. In OMM, each object may have more than one relationship with another object. However, if we assume only one relationship between any two objects, there can be up to $O(N^2)$ relationships for N objects. Instead of recording the huge number of static relationships required by other competitor systems, OMM only records the rules representing the different types of relationships. By defining one relationship type, such as the supervisor-and-subordinate relationship type, OMM is able to represent thousands of relationship instances of this type between resources.

10.2.2 Extensibility

The object-oriented approach of OMM makes it a very extensible system. To add a new type of enterprise resource to the organization model, we simply create a new OMM organization and define the corresponding attributes to reflect the resource type.

Moreover, the rule-based relationship model also contributes to OMM's extensibility. To model a new type of relationship within the enterprise, we do not have to hardwire every link between all the resources; rather, we simply define an OMM virtual link which is a computable expression over the OMM member attributes.

10.2.3 Flexibility

With OMM, the organization modeller can partition the enterprise horizontally and vertically. This partitioning gives rise to a number of OMM organizations representing different resource types. Partitioning also allows users to group similar resources based on their functions, locations, or other management purposes. OMM organizations can also be merged together easily. This allows the organization modeller to easily bring multiple business units together, which is a common activity in company restructuring and in mergers-and-acquisitions.

10.2.4 Performance

Once we have used OMM to create a specific organization model for an enterprise, employees may frequently access the model and the resource information that is stored

in OMM. The performance of the OMM API is critical if we expect users and the WFMS to be relying on it. The Read and Write interfaces are particularly important since they will be accessed most frequently. Table 10-1 shows the performance of various operations in milliseconds:

Object Type	Operation Type	Elapse Time (ms)
Session	connect	853.09
	disconnect	2.98
Organization	create	3700.00
	delete	2086.88
	read	23.76
Member	create	85.09
	delete	34.50
	read	62.00
	write	149.53
Virtual Link	resolve role	40.95
	resolve relationship	41.56

Table 10-1 Elapsed Time of the OMM API Categorized by Object and Operation Type

The performance test from which these figures were obtained was run on an Intel PC with two Pentium III 400 MHz processors and 256 MB RAM, running Windows NT 4.0.

A "proof of concept" of OMM is provided by the Prototype System which was applied in the real environments of Hitachi America and Insure Point. Further testing as to how OMM scales up would be desirable before its overall potential is critically assessed and fully ascertained.

10.3 Further Work

There are several future areas of research that stem from this thesis. In defining our conceptual model and software architecture, we did not apply any modelling tools, such as the UML, to model our objects and components. UML is a standardized language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for modelling business processes and other non-software systems. The UML represents a collection of some of the best engineering practices in the modelling of large and complex systems (Booch 1999). UML could potentially be used to model the conceptional entities and software components of OMM.

In the Prototype System created for this thesis, we assumed an enterprise with only one domain, and with all the enterprise resources managed by one OMM server. In reality, a

large enterprise would have its organizational database distributed in different regions. Each region would have the autonomy to define and manage its own resources. It is therefore important for us to develop a distributed architecture to allow multiple OMM servers to exchange information with one another. The fact that this extended organization model and the underlying information are distributed should be transparent to users accessing information in the OMM environment.

The concept of distribution can be further expanded to support distributed OMM organizations. The goal here is to create an OMM organization that can be distributed over two or more domains and managed by multiple OMM servers. This would give users even more flexibility in global management. It would allow, for instance, all the machines of a company to be captured in one single distributed OMM organization, even though they were indeed located in different regions of the world and managed by different branches of the company. Likewise, an OMM member object can also be distributed, meaning that its attributes can be resident in different OMM domains.

Several directions for further work stem from the discussion of concurrency control in Chapter 8. The first area relates to garbage collection in pre-claimed locks. The use of a timeout algorithm on pre-claimed locks is necessary to ensure garbage collection in the case of application crashes. Timeout mechanisms also reduce performance bottlenecks due to convoys. However, since we are dealing with interactive users accessing OMM, once we start using timeouts on locks, there is the possibility that an active user's work will be timed out and rolled back. This may give rise to usability issues. Another future area of research also concerns pre-claimed locks: our current implementation only performs authorization checking when users attempt to read or update data. However, authorization checks should also be performed when users explicitly request locks.

Thus far in our modelling of dynamic relationships, we have defined relationships between OMM member objects. The model would be more powerful if we were to expand virtual links to cover relationships between a member and an organization, or even between two OMM organizations. When an OMM organization is part of a relationship, all member objects within that organization are involved in the relationship. For instance, if Tom Moore is a supervisor of an OMM organization, then it is assumed that he supervises all resources within that organization.

In the thesis we have compared OMM with the X.500 directory. One noticeable weakness of X.500 is in its inability to deal with dynamic relationships. In principle, it is possible to implement the virtual link concept of OMM as an extension to the X.500 directory. This would greatly enhance the X.500 directory service and would immediately benefit the large X.500 installed base in today's world.

Appendix A: JavaDoc Listing of the OMM API

The OMM API is categorized by the OMM object classes, namely the organization, member and virtual link. The corresponding code name for these classes are `OmsOrganization`, `OmsMember` and `OmsVirtualLink`. `OmsObject` is the superclass of these three classes. The prefix "Oms" in the class name stands for OMM Services.

A. 1 OmsObject

The `OmsObject` class is the superclass of the three basic entities in OMM, namely `OmsOrganziation`, `OmsMember` and `OmsVirtualLink`.

Constructors

OmsObject

```
public OmsObject()
```

Construct an empty `OmsObject` object

OmsObject

```
public OmsObject(int id, String name) throws OmsException
```

Construct an `OmsObject` with the given id and name

Parameters:

id - identification number of the object

name - name of the object

Throws: OmsException

thrown if any error is encountered

Methods

- **getId**

```
public int getId()
```

Return the id of the object

Returns:

identification number of the object

- **setId**

```
public int setId(int id)
```

Set the id of the object

Parameters:

id - identification number of the object to be set

- **getName**

```
public String getName()
```

Return the name of the object

Returns:

String - name of the object

- **setName**

```
public void setName(String name)
```

Set the name of the object

Parameters:

name - name of the object to be set

- **getOrgId**

```
public int getOrgId()
```

Return the organization id of the object

Returns:

organization Id of the object

- **setOrgId**

```
public void setOrgId(int id)
```

Set the organization id of the object

Parameters:

id - identification number of the organization to be set

- **getState**

```
public int getState()
```

Return the life cycle state of the object

Returns:

Life cycle state of the object

- **setState**

```
public void setState(int state)
```

Set the life cycle state of the object

Parameters:

state – life cycle state of the object to be set

- **copy**

```
public void copy(OmsObject toObject) throws OmsException
```

Copy the member attributes of this `OmsObject` to the target `OmsObject`

Parameters:

toObject - the target `OmsObject`

Throws: OmsException

thrown if any error is encountered

A. 2 OmsOrganization

The `OmsOrganization` class implements the Organization conceptual entity of OMM. It allows the user to define and manipulate an organization in OMM

Constructors

- **OmsOrganization**

```
public OmsOrganization()
```

Construct an empty `OmsOrganization` object

OmsOrganization

```
public OmsOrganization(int orgId) throws OmsNotFoundException,
OmsException
```

Retrieve an existing `OmsOrganization` object by the object ID

Parameters:

orgId - the id of the organization to be opened

Throws: OmsNotFoundException

thrown if the organization does not exist

Throws: OmsException

thrown if other types of error are encountered

OmsOrganization

```
public OmsOrganization(String orgName) throws OmsNotFoundException,
OmsException
```

Retrieve an existing `OmsOrganization` object by name

Parameters:

orgName - the name of the organization to be opened

thrown if other types of error are encountered

Throws: OmsNotFoundException

thrown if the organization does not exist

Throws: OmsException

Methods

getType

```
public int getType()
```

Get the organization type. The type can be one of the following values:
`OmsObject.OMS_NATIVE_USER_ORG`
`OmsObject.OMS_NATIVE_NONUSER_ORG`
`OmsObject.OMS_NON_NATIVE_USER_ORG`

Returns:

organization identification number

setType

```
public void setType(int orgType)
```

Set the type attribute of the `OmsOrganization` object

Parameters:

orgType - the type of the organization

getAttribDefList

```
public OmsObList getAttribDefList()
```

Get the attribute definition list of the organization

Returns:

list of attribute definitions that is associated to this organization

create

```
public int create(String orgName,
                  int orgType) throws OmsDupException,
OmsParamException, OmsException
```

Create a new organization in the domain

Parameters:

orgName - the name of the organization

orgType - the type of the organization

Returns:

organization identification number

Throws: OmsDupException

thrown if the organization name already exists

Throws: OmsParamException

thrown if invalid parameters are found

Throws: OmsException

thrown if other types of error are encountered

delete

```
public void delete() throws OmsNotFoundException, OmsException
```

Delete an existing organization

thrown if the organization object to be deleted is NULL

Throws: OmsNotFoundException

thrown if the organization does not exist

Throws: OmsException

addAttribute

```
public void addAttribute(OmsAttribDef attrib,
                         int extern,
                         String extnName,
                         boolean bRequire) throws
OmsNotFoundException, OmsDupException, OmsParamException, OmsException
```

Add an attribute definition to the organization

Parameters:

attrib - the object containing the definition of the attribute

extern - 0 = default, 1 = support LDAP

extnName – the name of the LDAP attribute name if extern = 1; else = 0

bRequire – true if this is a required attribute, otherwise false

Throws: OmsNotFoundException

thrown if the attribute does not exist

Throws: OmsDupException

thrown if the attribute was already associated with this organization

Throws: OmsParamException

thrown if the attribute parameter is NULL

Throws: OmsException

thrown if other types of error are encountered

dropAttribute

```
public void dropAttribute(OmsAttribDef attrib) throws
OmsNotFoundException, OmsParamException, OmsException
```

Drop an attribute definition from the organization

Parameters:

attrib - the object containing the definition of the attribute

Throws: OmsNotFoundException

thrown if the attribute is not associated with this organization

Throws: OmsParamException

thrown if the attribute parameter is NULL

Throws: OmsException

thrown if other types of error are encountered

getAttributeList

```
public OmsObList getAttributeList() throws OmsNotFoundException,
OmsException
```

Retrieve a list of attribute definitions associated with this organization

Returns:

a list of attribute definition objects

Throws: OmsNotFoundException

thrown if the attribute does not exist

Throws: OmsException

thrown if other types of error are encountered

getMembers

```
public OmsObList getMembers(OmsObject memObj,
                            int numOfMem,
                            boolean inclusive,
                            boolean forward) throws OmsParamException,
OmsException
```

Retrieve the list of member objects of the organization.

Parameters:

memObj - the object containing the member ID. If the object is NULL, the method will return the name list starting with the smallest member id

numOfMem - the total number of members to get. Note that only name and ID are returned

inclusive – `true` if it includes the specified member in the return list; else `false`

forward - retrieve a list of members with greater or smaller id

Returns:

a list of member objects

Throws: OmsParamException

thrown if invalid parameters are found

Throws: OmsException

thrown if other types of error are encountered

getMembersByAttrib

```
public OmsObList getMembersByAttrib(String attName,
                                    String relOp,
                                    int attValue) throws
OmsParamException, OmsException
```

Retrieve the list of member names and id's of the organization by the integer attribute name and value

Parameters:

attName - the name of the attribute

relOp - the relational operator: `==, <>, >=, <=, >, <`

attValue - the value of the attribute

Returns:

a vector of `OmsObject` containing the member names and ID's

Throws: OmsParamException

thrown if invalid parameters are found

Throws: OmsException

thrown if other types of error are encountered

getMembersByAttrib

```
public OmsObList getMembersByAttrib(String attName,
                                    String relOp,
                                    float attValue) throws
OmsParamException, OmsException
```

Retrieve the list of member names and id's of the organization by the float attribute name and value

Parameters:

attName - the name of the attribute

relOp - the relational operator: `==, <>, >=, <=, >, <`

attValue - the value of the attribute

Returns:

a vector of `OmsObject` containing the member names and ID's

Throws: OmsParamException

thrown if invalid parameters are found

Throws: OmsException

thrown if other types of error are encountered

getMembersByDateAttrib

```
public OmsObList getMembersByDateAttrib(String attName,
                                        String attValue,
                                        String relOp) throws
OmsParamException, OmsException
```

Retrieve the list of member names and ID's by specifying the date value

Parameters:

attName - the name of the attribute

attValue - the value of the attribute

relOp - the relational operator: `==, <>, >=, <=, >, <`

Returns:

a vector of `OmsObject` containing the member names and ID's

Throws: OmsParamException

thrown if invalid parameters are found

Throws: OmsException

thrown if other types of error are encountered

resolveExpression

```
public OmsObList resolveExpression(String expr) throws OmsException
```

Resolve all the members who are playing the role that is expressed by the expression

Parameters:

expr - the computable expression, e.g. `CountryOfOrigin == 'Belgium'`

Returns:

the `OmsObList` list of `OmsMember` objects that satisfy the expression

Throws: OmsException

thrown if other types of error are encountered

A. 3 OmsMember

The `OmsMember` class implements the Member conceptual entity of OMM. It provides methods for users to define and manipulate the member objects in OMM

Constructors

OmsMember

```
public OmsMember()
```

Construct an empty member object

OmsMember

```
public OmsMember(OmsOrganization org,
                 String memName) throws OmsParamException,
OmsException
```

Given an organization object and member name, this constructor retrieves member information from the database

Parameters:

org - organization object that the member belongs to

memName - the name of the member to be retrieved

Throws: OmsParamException

if the given member name contains illegal characters or name exceeds the maximum length. The maximum name length is 50. The following characters are considered illegal characters in name: ~`!@ # $%^&*()+=[ ] { } |?><, ."

Throws: OmsException

if there is problem retrieving member

OmsMember

```
public OmsMember(OmsOrganization org,
                 String memName,
                 String passwd) throws OmsParamException,
OmsAuthException, OmsException
```

This constructor checks to see if the name and password match within a particular organization. If so, it retrieves that member's information from the database

Parameters:

org - organization object that the member belongs to

memName - the name of the member to be retrieved

passwd - the password of the member

Throws: OmsParamException

if the given member name contains illegal characters or name exceeds the maximum length. The maximum name length is 50. The following characters are considered illegal characters in name: ~`!@ # $%^&*()+=[ ] { } |?><, ."

Throws: OmsAuthException

if the password does not match

Throws: OmsException

if there is problem retrieving the member

Methods

create

```
public int create(OmsOrganization org,
                  String memName,
                  String passwd) throws OmsParamException,
OmsDupException, OmsException
```

Create a new member in the given organization

Parameters:

org - organization object that the member will belong to

memName - the name of the member to be created

passwd - password of the new member

Returns:

the unique member identification number

Throws: OmsParamException

if the given member name contains illegal characters or name exceeds the maximum length. The maximum name length is 50. The following characters are considered illegal characters in name: ~`!@ # $%^&*()+=[ ] { } |?><, ."

Throws: OmsDupException

if a member with the same name already exists in the organization

Throws: OmsException

if there is a problem creating the member

delete

```
public void delete() throws OmsException
```

Delete the member from the organization

Parameters:

Throws: OmsException

if there is a problem deleting the member

setPassword

```
public void setPassword(String oldPass,
                        String newPass) throws OmsAuthException,
OmsException
```

Set new password for the member

Parameters:

oldPass - the old password

newPass - the new password

Throws: OmsAuthException

if the old password does not match

Throws: OmsException

if there is a problem in setting the new password

setNewName

```
public void setNewName(String newName) throws OmsAuthException,
OmsException
```

Set new name for the member

Parameters:

newName - the new member name

Throws: OmsAuthException

if the user does not have authority to update his or her name

Throws: OmsException

if there is a problem in setting the new name

setAttValue

```
public void setAttValue() throws OmsAttribValException,
OmsDupException, OmsException
```

Save the member attribute values to the database. The attribute value list of this member should now contain the values to be saved. Each element in the list represents one value (for multiple-valued attributes, multiple elements represent all the values to be saved). Note that the old values will be deleted from the database and be replaced by the new values in the list. Each value element must have the attribute id and value set

Throws: OmsAttribValException

if there is incorrect attribute value such as mismatch of attribute data type

Throws: OmsDupException

if the value of a particular attribute already exists and the particular attribute is not multi-valued

Throws: OmsException

if there is a problem saving the member attribute values

getOrgId

```
public int getOrgId()
```

Return the organization identification number of the organization that this member belongs to

Returns:

organization identification number

Overrides:

`getOrgId` in class `OmsObject`

getValList

```
public OmsObList getValList()
```

Return a list of attribute values of this member object

Returns:

an instance of `OmsObList` that contains the attribute values

getValListByAttrib

```
public OmsObList getValListByAttrib(String name) throws
OmsNotFoundException
```

Return a list containing the attribute values of the given attribute name

Parameters:

name – the name of the attribute

Returns:

an instance of `OmsObList` that contains the attribute values

Throws: OmsNotFoundException

if the given attribute name is not found in the member

setValListByAttrib

```
public void setValListByAttrib(String attName,
                               OmsObList valList,
                               boolean bSave) throws
OmsNotFoundException, OmsException
```

Replace the member value of a specific attribute

Parameters:

attName - attribute name

valList - value list

bSave – `true` to save the attribute values to the OMM data store; or `false` to only keep the change in memory

Throws: OmsNotFoundException

if the given attribute name is not found in the member

Throws: OmsException

OMS error. The detail error will be returned in the message string

getOrg

```
public OmsOrganization getOrg()
```

Return the `OmsOrganization` that this member belongs to

Returns:

`OmsOrganization` object

getStringValue

```
public String getStringValue(String attName) throws
OmsNotFoundException, OmsException
```

Get one attribute value. The attribute must be of type `String`

Parameters:

attName - attribute name

Returns:

string value

Throws: OmsNotFoundException

if the given attribute name is not found in the member

Throws: OmsException

if the given attribute name is not found in the member

getIntValue

```
public int getIntValue(String attName) throws OmsNotFoundException,
OmsException
```

Get one attribute value. The attribute must be of type `integer`

Parameters:

attName - attribute name

Returns:

`int` value

Throws: OmsNotFoundException

if the given attribute name is not found in the member

Throws: OmsException

if the given attribute name is not found in the member

getDateValue

```
public Date getDateValue(String attName) throws OmsNotFoundException,
OmsException
```

Get one attribute value. The attribute must be of type `Date`

Parameters:

attName - attribute name

Returns:

`int` value

Throws: OmsNotFoundException

if the given attribute name is not found in the member

Throws: OmsException

if the given attribute name is not found in the member

setRaw

```
public void setRaw(String attName,
                   byte array[]) throws OmsAttribValException,
OmsException
```

Set a raw attribute with a byte stream of data

Parameters:

attName - attribute name

byte - a byte array

Throws: OmsAttribValException

if member does not have this attribute

Throws: OmsException

if an OMS internal error is encountered

getRaw

```
public byte[] getRaw(String attName) throws OmsException
```

Return a raw attribute as a byte stream of the file

Parameters:

attName - attribute name

Returns:

a byte array

Throws: OmsException

if an OMS internal error is encountered

copy

```
public void copy(OmsObject toObject) throws OmsException
```

Copy the member attributes of this `OmsMember` to the target `OmsMember`

Parameters:

toObject - the target `OmsMember` object

Throws: OmsException

thrown if any error is encountered

Overrides:

`copy` in class `OmsObject`

move

```
public void move(int orgId) throws OmsNotFoundException, OmsException
```

Move this `OmsMember` from the current organization to another organization. Those attributes that are associated to both organizations will be retained. Attributes that are in the source organization but not in the target organization will be dropped. Attributes that are in the target organization but not in the source organization will be set to NULL.

Parameters:

orgId – the organization Id of the target organization value

Throws: OmsNotFoundException

if the given organization Id is not found in the system

Throws: OmsException

thrown if any error is encountered

lock

```
public int lock(int mode) throws OmsException
```

Claim Read or Write lock on the member

Parameters:

mode – this can either be OMS_READ or OMS_WRITE lock. OMS_READ lock can be shared among multiple readers while OMS_WRITE lock is exclusive.

Returns:

the identification number

Throws: OmsLockConflictException

if there is already a lock on the member that is conflicting with the requesting lock mode. For instance, requesting an OMS_READ lock on a member that is currently locked by an OMS_WRITE lock.

Throws: OmsException

if there is a problem in claiming the requested lock on the member

unlock

```
public void unlock(int lockId) throws OmsNotFoundException,
OmsException
```

Release the lock with lock identification equal to lockId.

Parameters:

lockId – the Id of the previously claimed lock

Throws: OmsNotFoundException

if the given lock Id is not found in the system

Throws: OmsException

thrown if any error is encountered in releasing the lock

A. 4 OmsVirtualLink

The `OmsVirtualLink` class implements the Virtual Link concept in OMM. It provides methods for users to define and manipulate the Virtual Links in OMM

Constructors

OmsVirtualLink

```
public OmsVirtualLink()
```

Constructs an empty link object

OmsVirtualLink

```
public OmsVirtualLink(String vlinkName) throws OmsParamException,
OmsException
```

Get an existing `OmsVirtualLink` object by name

Parameters:

vlinkName - the name of the virtual link

Throws: OmsParamException

if invalid parameters are found

Throws: OmsException

if other types of error are encountered

Methods

isLink

```
public boolean isLink(OmsMember owner,
                      OmsMember member) throws OmsNotFoundException,
OmsException
```

Query if a virtual link exists from an owner to a member

Parameters:

owner - the instance of `OmsMember` that is the owner

member - the instance of `OmsMember` that is the member

Returns:

`true` or `false` for the link query

Throws: OmsNotFoundException

if the owner or member is not found

Throws: OmsException

if other types of error are encountered

resolveLink

```
public OmsObList resolveLink(OmsMember owner) throws
OmsParamException, OmsException
```

Resolve the virtual link connections from an owner to the member objects within the scope list

Parameters:

owner - the `OmsMember` object of the owner

Returns:

`OmsObList` list of members that satisfy the link

Throws: OmsParamException

if invalid parameters are found

Throws: OmsException

if other types of error are encountered

resolveLink

```
public OmsObList resolveLink(OmsMember owner,
                             String attributeName) throws
OmsParamException, OmsException
```

Resolve the virtual link connections from an owner to the member objects within the scope list, and retrieve the values of the given attribute for each of these members

Parameters:

owner - the member object of the owner

attributeName - name of the attribute whose values should be retrieved

Returns:

an instance of `OmsObList`, a list of `OmsMember` objects that satisfy link

Throws: OmsParamException

if invalid parameters are found

Throws: OmsException

if other types of error are encountered

resolveLinkAll

```
public static OmsObList resolveLinkAll(OmsMember root,
                                       OmsObList linkList) throws
OmsParamException, OmsException
```

Resolve all virtual link connections from a root to all the member objects beneath the root that are within the scope list

Parameters:

root - the member object sitting at the root of a tree

linkList – a list of virtual links with the first element as the virtual link between level 0 and level 1 of the tree, the second element as the virtual link between level 1 and level 2, and so on

Returns:

an instance of `OmsObList`, a list of `OmsMember` objects that satisfy virtual link and the parent ID

Throws: OmsParamException

if invalid parameters are found

Throws: OmsException

if other types of error are encountered

Appendix B: Class Diagrams of OMM Conceptual Entities

The OMM Prototype System is implemented in Java. The Java files are organized into five packages arranged in a directory tree.

- Client: class implementation of the OMM API.
- Common: type and utility classes that are shared by the other OMM classes.
- Db: database classes that call JDBC and LDAP to access the OMM data store.
    - DbRel: classes that parse and translate the OmsVirtualLink expressions into JDBC queries.
- Server: top level class implementation of the OMM Prototype System.

The class diagrams of the OMM classes in each of these packages are shown in Figure B-1 through Figure B-5. These class diagrams were generated by using the BlueJ interactive Java environment tool, version 1.3.5 (BlueJ 2003).

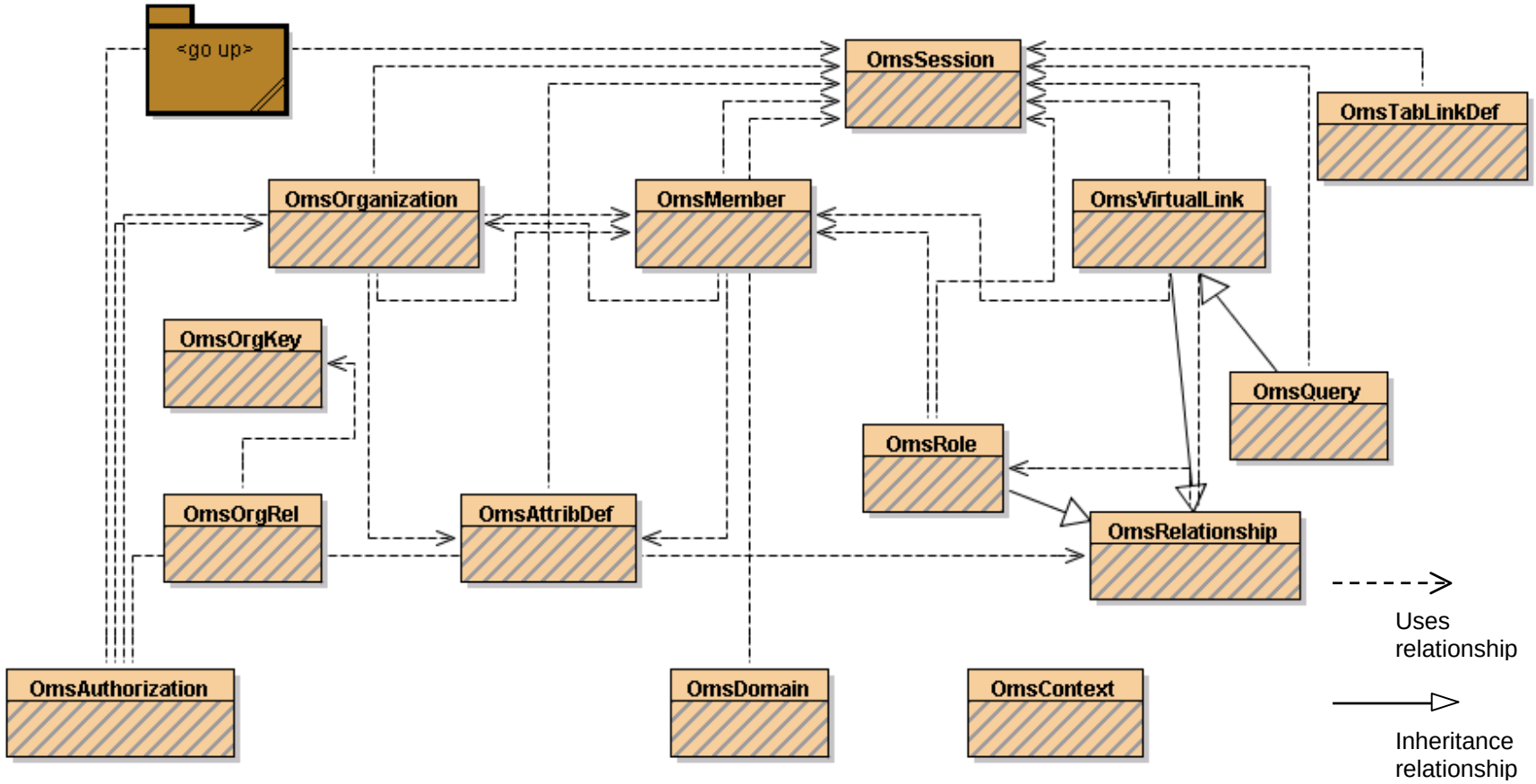


Figure B-1 Class Diagram of OMM Classes in the Client Package

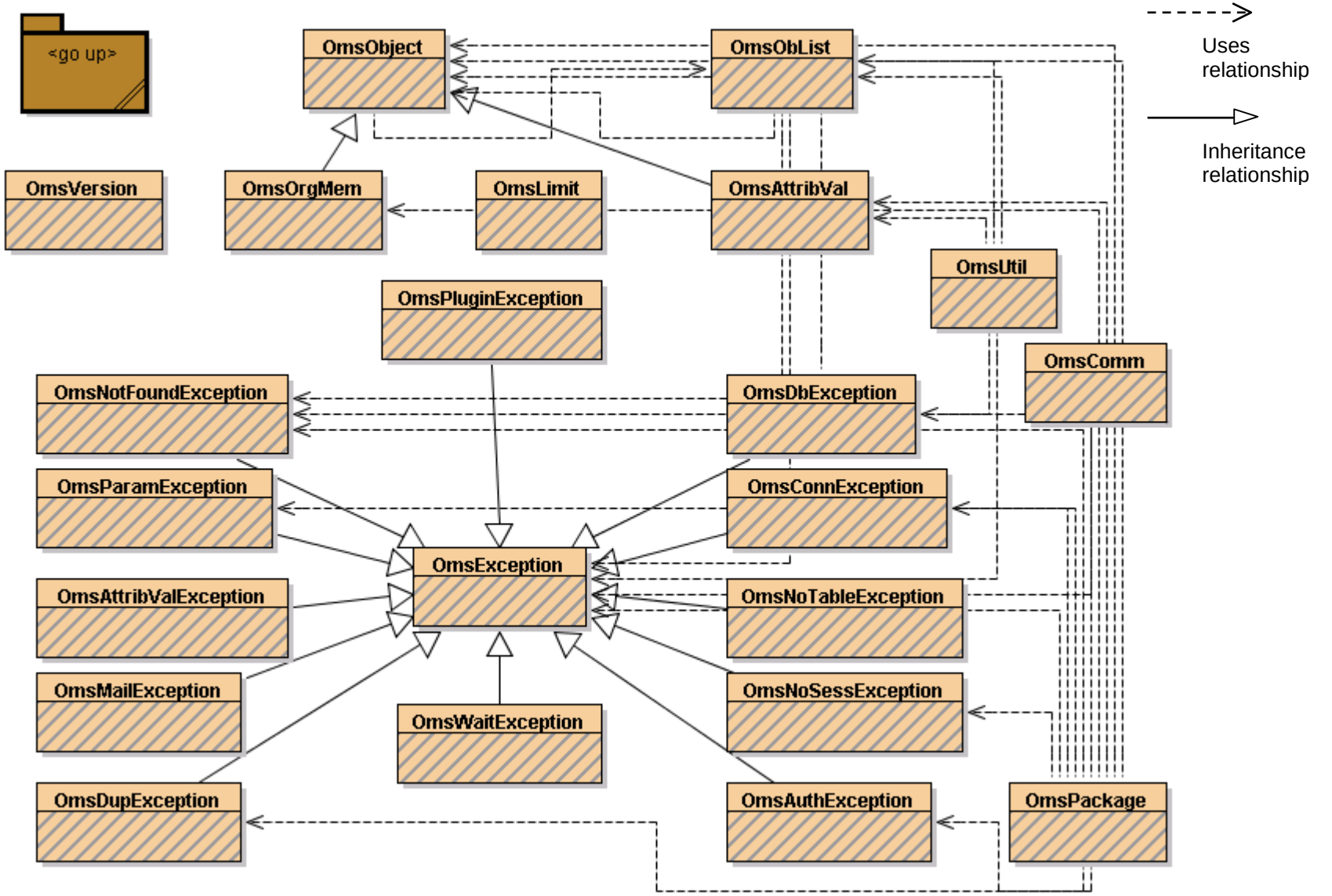


Figure B-2 Class Diagram of OMM Classes in the Common Package

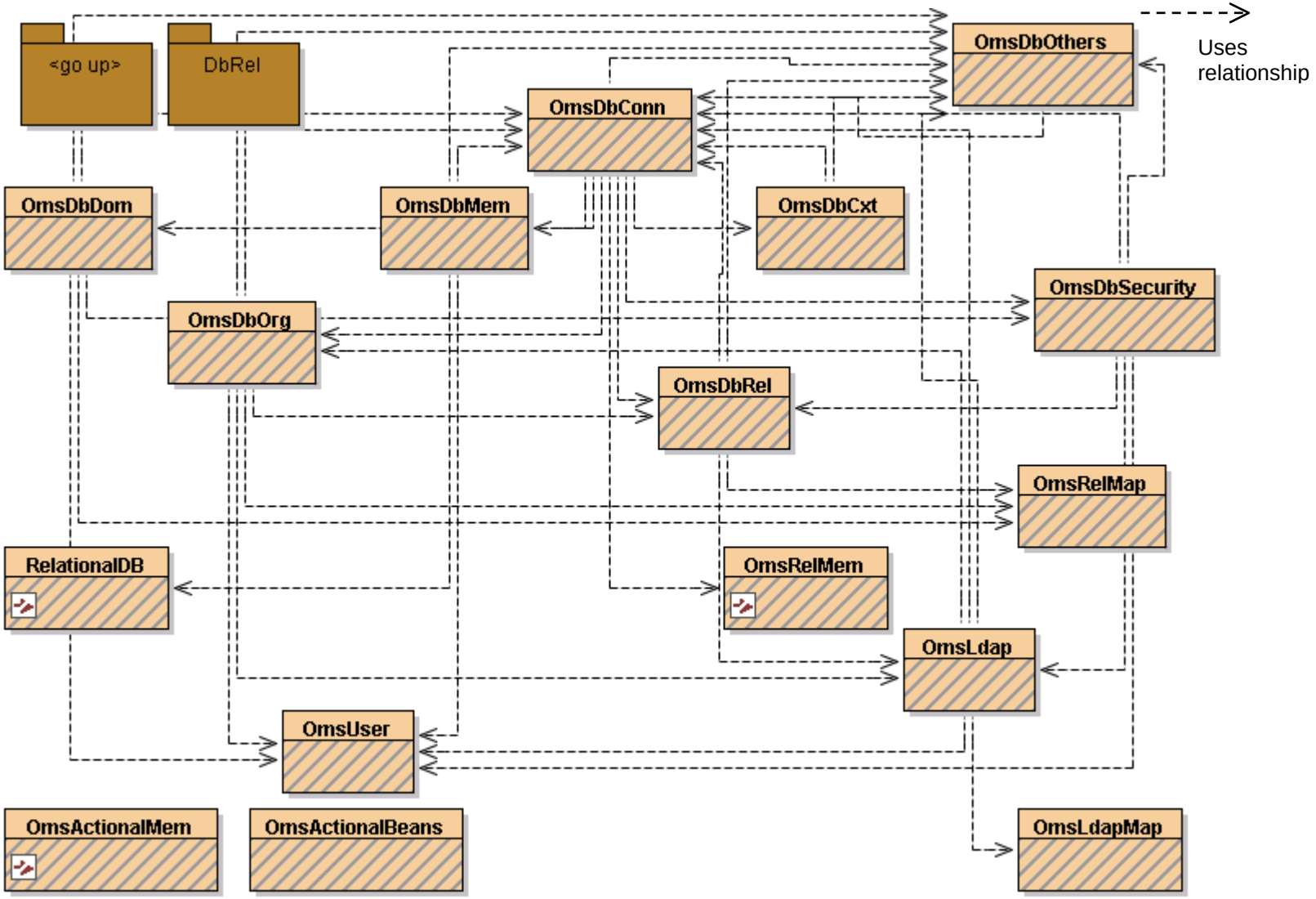


Figure B-3 Class Diagram of OMM Classes in the Db Package

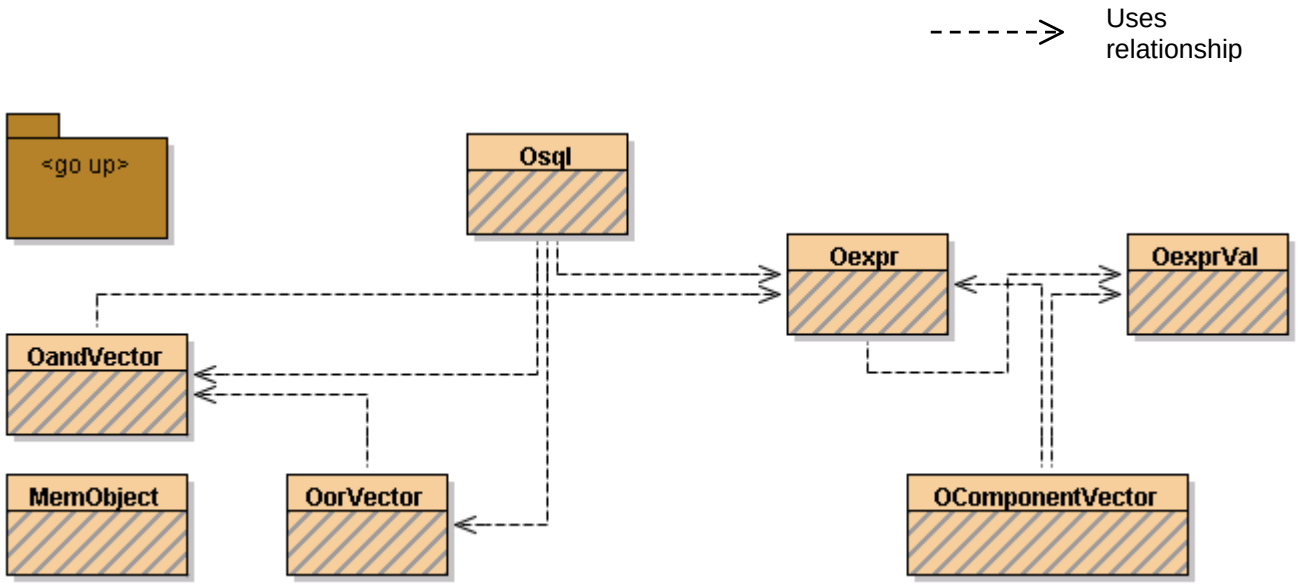


Figure B-4 Class Diagram of OMM Classes in the DbRel Package

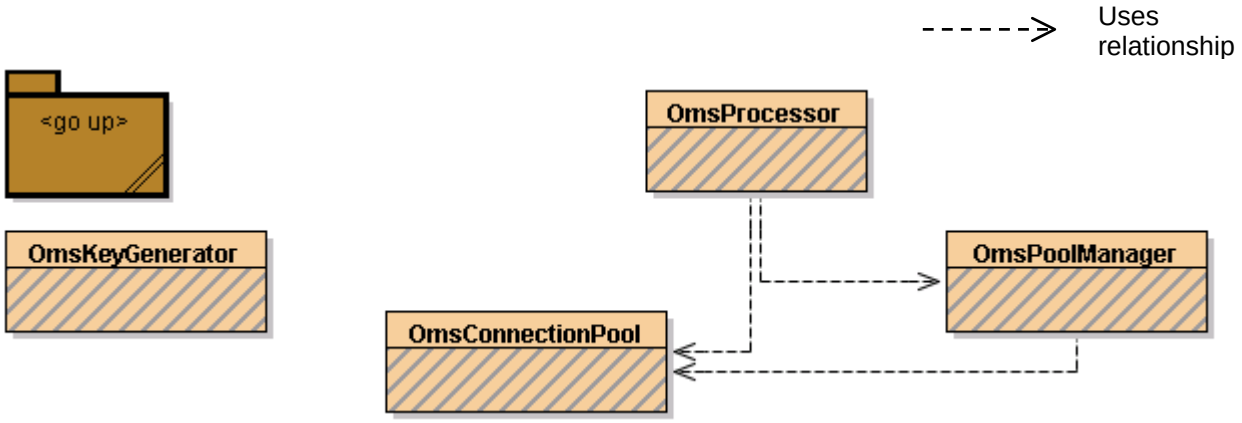


Figure B-5 Class Diagram of OMM Classes in the Server Package

References

Anderl, R., Wasmer, A. (1997) Integration of product life cycle views on the basis of a shared conceptual information model. *IFIP/JSPE Transactions. Information Infrastructure Systems for Manufacturing,* 47-58. Chapman & Hall, London UK.

Andrew, C., Shropshire, E. (1998) *Novell's NDS Developer's Guide.* Novell Press, Hartford, CT.

Belli, F., Dreyer, J. (1994) Systems modelling and simulation by means of predicate/transition nets and logic programming. *Proceedings of the 7th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems,* Austin, TX, 465-474.

Berghel, H. (1998) The new push for push technology. *NetWorkers,* 2(3):28-36.

Berio, G., Di Leva, A., Giolito, P., Vernadat, F. (1995) The M*-OBJECT methodology for information system design in CIM environments. *IEEE Transactions on Systems, Man, and Cybernetics,* 25(1):68-85.

Bernstein, P.A., Hadzilacos, V., Goodman, N. (1987) *Concurrency Control and Recovery in Database Systems.* Addison-Wesley, Reading, MA.

BlueJ – The Interactive Java Environment. (2003) http://www.BlueJ.org/

Blum, D., Litwack, D. (1994) *The E-Mail Frontier: Emerging Markets and Evolving Technologies.* Addison-Wesley, Menlo Park, CA.

Booch, G. (1999) UML in action. *Communications of the ACM*, 42(10):26-28.

Burns, M. (1998) *Active Directory: A Scalable Directory for the Enterprise?* Network Solutions, February. http://www.nwsolutions.com

Bussler, C. (1994) Enterprise process modeling and enactment in GERAM. *Proceedings of the 3rd International Conference on Automation, Robotics and Computer Vision (ICARCV '94),* Singapore, 1-8.

Bussler, C., Jablonski, S. (1995) Policy resolution for workflow management systems. *Proceedings of the Hawaii International Conference on System Sciences (HICSS-28),* Maui, Hawaii, 45-54.

Bussler, C. (1996) Analysis of the organization modeling capability of workflow management systems. *Proceedings of the PRIISM '96 Conference,* Maui, Hawaii.

CCITT Recommendation X.500 to X.521. (1988) *Data Communication Networks, Directory.* Blue Book of ISO/IEC Standards ISO 9594-1 to ISO 9594-7.

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

Cheng, E. (2000a) An object-oriented organizational model to support dynamic role-based access control in electronic commerce. *The International Journal of Decision Support Systems*, 29:357-369.

Cheng, E., Loizou, G. (2000b) A publish/subscribe framework: push technology in e-commerce. *Proceedings of the 17th British National Conference on Databases,* Exeter, U.K., 153-170.

Cheng, E. (1999a) An object-oriented organizational model to support dynamic role-based access control in electronic commerce applications. *Proceedings of the 32nd Hawaii International Conference on System Sciences*, Maui, Hawaii.

Cheng, E. (1999b) Publish-and-Subscribe: a framework of push technology for electronic commerce. *Proceedings of the International Computer Science Conference,* Hong Kong, 33-40.

Cheng, E. (1998) A rule-based organization modeling system to support dynamic role resolution in workflow. *Proceedings of the ISCA 11th International Conference,* Chicago, IL, 67-74.

Cheng, E. (1997) *The OMM Model.* Technical Report of OCT Research Laboratory and The College of Notre Dame. Belmont, CA.

Cheng, E. (1995) Re-engineering and automating enterprise-wide business processes. *Proceedings of International Working Conference on Information Industry,* Bangkok, Thailand, 98-106.

Cheng, E. (1992) A high-speed open journal system in a distributed computing environment. *Proceedings of the International Computer Science Conference,* Hong Kong, 141-147.

Cheng, E., Chang, E., Klein, J., Lu, E., Lutgardo, A., Obermarck, R. (1991) An open and extensible event-based transaction manager. *Proceedings of USENIX Conference*, Nashville, TN, 49-58.

Cluet, S., Kapitskaia, O., Srivastava, D. (1999) Using LDAP directory caches. *Proceedings of the 18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems,* Philadelphia, PA, 273-284.

Cole, J. (2003) *The UCLA Internet report: surveying the digital future, year three.* UCLA Center for Communication Policy. Los Angeles, CA.

Curtis, B., Kellner, M., Over, J. (1992) Process modeling. *Communications of the ACM,* 35(9): 75-90.

Di Cesare, F., Harhalakis, G., Proth, J.M., Silva, M., Vernadat, F.B. (1993) *Practice of Petri Nets in Manufacturing.* Chapman & Hall, London, UK.

Di Felice, P., Clementini, E. (1991) Officeaid VPE: a visual programming with examples system for specifying routine office tasks. *Journal of Visual Languages and Computing,* 2(3):275-296.

Di Leva, A., Giolito, P., Vernadat, F. (1997) The M*-OBJECT organisation model for enterprise modelling of integrated engineering environments. *Concurrent Engineering - Research and Applications,* 5(2):183-194.

Di Leva, A., Vernadat, F., Bizier, D. (1987) Information system analysis and conceptual database design in production environments with M*. *Computers in Industry,* 9:183-217.

Doumeingts, G., Chen, D., Vallespir, P. (1993) GIM (GRAI Integrated Methodology) and its evolutions. A methodology to design and specify advanced manufacturing systems. *IFIP/JSPE Transactions. Information Infrastructure Systems for Manufacturing,* 101-117. Elsevier Science B.V., North-Holland.

Franklin, M., Zdonik, S. (1998) Data in your face: push technology in perspective. *Proceedings of SIGMOD '98,* Seattle, WA, 516-519.

Frenkel, K. A. (1991) The human genome project and informatics. *Communications of the ACM*, 34(11):40-51.

Gaudin, S. (1999) Battle of the directories. Computer World magazine, April 19. http://www.computerworld.com

Global Internet Statistics by Language. (2003) http://www.glreach.com/globstats/

Goh, C., Baldwin, A. (1998) Towards a more complete model of role. *Proceedings of the 3rd ACM Workshop on Role-Based Access Control,* Fairfax, VA, 55-62.

Gottlob, G., Schrefl, M., Röck, B. (1996) Extending object-oriented systems with roles. *ACM Transactions on Information Systems,* 14(3):268-296

Graefe, U., Chan, A.W. (1993) An enterprise model as a design tool for information infrastructure. *IFIP/JSPE Transactions. Information Infrastructure Systems for Manufacturing,* 65-79. Elsevier Science B.V., North-Holland.

Gray, J., Reuter, A. (1993) *Transaction Processing: Concepts and Techniques.* Morgan Kaufmann Publishers Inc., San Mateo, CA.

Hammer, M. (1993) *Reengineering the Corporation*. Harper Business, New York, NY.

Hirakawa, M., Tanaka, M., Ichikawa, T. (1990) An iconic programming system, HI-VISUAL. *IEEE Transactions on Software Engineering,* 16(10):1178-1184.

Ho, A., Cheng, E. (2000) Transforming a legacy relational database system to an object-oriented e-commerce environment. *Proceedings of World Multiconference on Systemics, Cybernetics and Informatics,* Orlando, FL, Vol. 9:623-628.

Howard, M. (1991) Work Flow: the coordination of business processes. *Gartner Group Presentation Highlights,* August.

Howes, T. (1995) The Lightweight Directory Access Protocol: X.500 lite. *CITI Technical Report 95-8,* Center for Information Technology Integration, University of Michigan. Ann Arbor, MI.

Hsu, C., Rattner, L. (1990) Information modeling for computerized manufacturing. *IEEE Transactions on Systems, Man, and Cybernetics,* 20(4):758-776.

Hsu, M., Ghoneimy, A., Kleissner, C. (1991) An execution model for an activity management system. *Digital Technical Report*, Digital Equipment Press, Boston, MA.

Hsu, M., Kleissner, C. (1996) Objectflow - towards a process management infrastructure. *Distributed and Parallel Databases,* 4(2):169-194.

Internet economy indicators. (2003) http://www.InternetIndicators.com/

Internet Trak, 2nd Quarter. (1998) Ziff-Davis Publishing Company. http://www.zd.com/marketresearch/IT2Q.htm.

ISO 9000 Survey -- Comprehensive Data and Analysis of U.S. Registered Companies. (1996). Irwin Professional Publishing, Burr Ridge, USA.

Jia, X., Maekawa, M. (1999) Naming and directory services. *Encyclopedia of Distributed Computing,* Urban, J. and Dasgupta, P. (eds), Kluwer Academic Publisher.

Kleijnin, S., Raju, S. (2003) An open Web services architecture**.** *Queue*, 1(1):38-46.

Kosanke, K., Vernadat, F., Zelm, M. (1997) CIMOSA process model for enterprise modelling. *IFIP/JSPE Transactions. Information Infrastructure Systems for Manufacturing*, 59-67. Chapman & Hall, London UK.

Krovi, R. (1997) Organizational intelligence: AI in organizational design, modeling, and control. *Interfaces,* 27(3)114-115.

Kuhn, D.R. (1997) Mutual exclusion of roles as a means of implementing separation of duty in role-based access control systems. *Proceedings of the 2nd ACM Workshop on Role-Based Access Control*, Fairfax, VA, 23-30.

Larman, C. (2001) *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process.* Prentice Hall, NJ.

Levene, M., Loizou, G. (1999) *A Guided Tour of Relational Databases and Beyond.* Springer-Verlag, London, UK.

Malone, T.W., Crowston, K., Lee, J., Pentland, B. (1993) Tools for inventing organizations: Toward a handbook of organizational processes. *Proceedings of the 2nd IEEE Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises,* Morgantown, WV, 1-20.

Malone, T.W., Lai, K., Fry, C. (1995) Experiments with Oval: A radically tailorable tool for cooperative work. *ACM Transactions on Information Systems,* 13(2):177-205.

McFadden, M. (1999) *Directory Services.* Enterprise Magazine, March. http://www.entmag.com

Medina-Mora, P., Winograd, T., Flores, R., Flores F. (1992) The action workflow approach to workflow management technology. *Proceedings of the Communications of the ACM* CSCW, Toronto, Canada, 281-288.

Mehandjiev, N., Bottaci, L. (1996) User-enhanceability for organizational information systems through visual programming. *Lecture Notes in Computer Science,* 1080:432-456.

Mertins, K., Heisig, P., Krause, O. (1997) Integrating business-process re-engineering with human-resource development for continuous improvement. *International Journal of Technology Management,* 14(1):39-49.

Moffett, J. (1998) Control principles and access right inheritance through role hierarchies. *Proceedings of the 3rd ACM Workshop on Role-Based Access Control,* Fairfax, VA, 63-69.

Murata, T. (1989) Petri Nets: properties, analysis and applications. *Proceedings of the IEEE,* 77(4):541-580.

OMM Application Programming Interface (1997) The OCT Research Laboratory.

O'Neil, P., Cheng, E., Gawlick, D., O'Neil, E. (1996) The log-structured merge-tree (LSM-tree). *Acta Informatica,* 33:351-385.

Osborn, S. (1997) Mandatory access control and role-based access control revisited. *Proceedings of 2nd ACM Workshop on Role-Based Access Control,* Fairfax, VA, 31-40.

Oval Version 1.1 User's Guide. (1992) Center for Coordination Science, MIT, Cambridge, Massachusetts.

Park, J., Sandhu, R., Ahn, G. (2001) Role-based access control on the Web. *ACM Transactions on Information and System Security*, 4(1):37-71.

Pallot, M., Sandoval, V. (1998) *Concurrent Enterprising: Toward the Concurrent Enterprise in the Era of the Internet and Electronic Commerce.* Kluwer Academic Publishing, Norwell, MA.

Peterson, G.E. (1987) *Tutorial: Object-oriented Computing.* IEEE Computing Society Press, Washington, DC.

Peterson, J.L. (1993) *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, Inc. NJ.

Radicati, S. (1994) *X.500 Directory Services, Technology and Deployment.* Van Nostrand Reinhold, New York, NY.

Reim, F. (1992) Organizational integration of the information-system design process. *Lecture Notes in Computer Science,* 693:410-424.

Roos, H., Bruss, L. (1994) Human and organization issues. *The Workflow Paradigm*, pp. 35-49, Future Strategies Publishing, Ann Arbor, MI.

Ross, D., Schoman, K. Jr. (1977) Structured analysis for requirements definition. *IEEE Transactions on Software Engineering,* 3(1): 6-15.

Rupietta, W. (1994) Organization models for cooperative office applications. *Proceedings of the 5th International Conference, DEXA '94,* Athens, Greece, 114-124.

Sandhu, R., Coyne, E., Feinstein, H. (1996) Role-based access control models. *IEEE Computer*, 29(2):38 – 47.

Sandhu, R., Munawer, Q. (1998) How to do discretionary access control using roles. *Proceedings of the 3rd ACM Workshop on Role-Based Access Control,* Fairfax, VA, 47-54.

Scheer, A.W. (1993) Architecture of integrated information systems (ARIS). *IFIP/JSPE Transactions. Information Infrastructure Systems for Manufacturing,* 85-99. Elsevier Science B.V., North-Holland.

Scott-Morton, M. (1990) *The Corporations of the 1990s: Information Technology and Transformation of Organizations.* Oxford University Press, NY.

Singh, B., Rein, G. L. (1992) Role Interaction Nets (RIN): A process description formalism. Technical Report No. CT-083-92, Austin, TX: MCC.

Stal, M. (2002) Web services: beyond component-based computing. *Communications of the ACM, special issue: Developing and integrating enterprise components and services*, 45(10):71-76.

Su, S., Lo D. H. (1980) A semantic association model for conceptual database design. *Proceedings of International Conference on Entity-Relationship Approach to System Analysis and Design,* Los Angeles, CA, 169-192.

Su, S. (1986) Modeling integrated manufacturing data with SAM-*. *IEEE Computer,* 19(1):34-49.

Swenson, K. (1994) A business process environment supporting collaborative planning. *Journal of Collaborative Computing,* 1(1):15-34.

Vanderaalst, W., Vanhee, K. (1996) Business process redesign - a Petri-net based approach. *Computers in Industry,* 29(1-2):15-26.

Vernadat, F. (1996) *Enterprise Modeling and Integration: Principles and Applications.* Chapman and Hall, London, UK.

Vernadat, F. (1993) CIMOSA: Enterprise modelling and enterprise integration using a process-based approach. *IFIP/JSPE Transactions. Information Infrastructure Systems for Manufacturing,* 65-79. Elsevier Science B.V., North-Holland.

Vidgen, R., Rose, J., Woodharper, T. (1994) BPR - the need for a methodology to revision the organization. *IFIP Transactions A - Computer Science and Technology,* 54:603-612.

Voss, G. (2002) *Java Beans, Part 1 – Basic Beans Concepts*. Javasoft, Sun Microsystems, Santa Clara, CA. http://developer.java.sun.com/developer/onlineTraining/Beans/Beans1/bean-definition.html

Whinston, A., Barua, A., Shutter, J., Wilson, B., Pinnell, J. (2001) *Measuring the Internet Economy.* Center for Research in Electronic Commerce, University of Texas, Austin, TX. http://www.InternetIndicators.com/jan_2001.pdf

Willcocks, L., Smith, G. (1995) IT-enabled BPR - organizational and human-resource dimensions. *Journal of Strategic Information Systems,* 4(3):279-301.

Yialelis, N., Sloman, M. (1996) A security framework supporting domain-based access control in distributed systems. *Proceedings of the ISOC Symposium on Network and Distributed Systems Security,* San Diego, CA, 26-39.

Youman, C.E., Sandhu, R., Coyne, E.J., Editors (1996) *Proceedings of the 1st ACM Workshop on Role Based-Access Control,* Gaithersburg, MD.