An Object-Oriented Workflow Management System

by

Samson Hui

B.Comm., University of British Columbia, 1995

A THESIS SUMBITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

(Faculty of Commerce and Business Administration)

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

October 1997

Department of Commerce and Business Administration

The University of British Columbia
Vancouver, Canada

Date Oct 19, 97

Abstract

Since many organizations have been facing pressure to reduce costs, to increase quality, and to provide rapid delivery of new services and products, they often resort to optimizing the way they do businesses. The use of workflow systems may improve the efficiency of an organizational process, thereby reducing costs and increasing workload capacity. It can also allow people to concentrate on value-added activities by freeing them from worrying about paper flows, filing, information tracing, and whether or not certain actions have been taken. Many workflow products, however, are fundamentally driven by vendor specifications without the support of a well-developed theoretical foundation. This thesis begins with an introduction of an Object-Oriented Workflow Model (OOWM). The OOWM extends an ontologically developed modelling method, Object-Oriented Enterprise Modeling (OOEM), by including workflow constructs with the purpose of describing the task structure of an organizational process. It also presents the architecture of an Object-Oriented Workflow Management System (OOWMS) which enacts the contents of the OOWM. Finally, based on the proposed architectural blueprint, a prototype of the workflow system was implemented, by using existing technologies, for a purchase requisition process.

Table of Contents

Lists of Figures

Lists of Tables

Acknowledgments

I owe a debt of gratitude to Prof. Carson Woo and Prof. Yair Wand. They have given me constructive criticisms and concrete advice, which have been very helpful in shaping the content and style of this thesis. I am also indebted to Prof. Shelby Brumelle who raised interesting questions which triggered a meaningful discussion during my thesis defense. While credit is due to many people, a few stand out: Daniel Chan, Jimmy Hui, Michael Han, and Scott Dalton. I would also like to thank Victor Ng, who has given me valuable insights into the use of entity-relationship (ER) diagrams and the development of a command-line interpreter. I am very grateful to Eve Shamash who provided excellent editorial assistance. Finally, I want to especially thank my family and Ada Chui. They have given me invaluable encouragement which allowed me to overcome the difficulties that I encountered while working on this project.

1. Introduction

The idea of structuring and managing processes has been in use since industrialization. This idea was originally concerned with the movement of physical entities in manufacturing plants. However, the idea of process management was extended to organizational-administrative processes where information flow is more emphasized than it is in the flow of physical products. Since these processes are usually well-structured and repetitive, the use of information technology to automate them becomes possible . Workflow management is concerned with the analysis, design, implementation, execution, and monitoring of organizational processes with the use of information technology (IT). According to Stark [1997], "workflow systems offer a new model for the division of labor between people and computers" [p.5]. They provide a "process control backbone" for business processes by mediating "the flow of responsibility in a process from person to person and from task to task" [p. 6].

Since many organizations have been facing pressure to reduce costs, to increase quality, and to provide rapid delivery of new services and products, they often resort to optimizing the way they do businesses. The use of workflow systems may improve the efficiency of an organizational process, thereby reducing costs and increasing workload capacity [Stark, 1997]. It can also allow people to concentrate on value-added activities by freeing them from worrying about paper flows, filing, information tracing, and whether or not certain actions have been taken.

1.1. Motivation

Many workflow products are fundamentally driven by vendor specifications without the support of a well-developed theoretical foundation. These products may demonstrate how "synergy is obtained by combining different technologies on client/server networks" [Orfali, Harkey, and Edwards, 1996, p.13] but fail to address the challenges an organization may actually face when it implements the workflow systems in a dynamic environment. These challenges include inconsistency of business objectives within an enterprise and demand for local autonomy [Ruiz, 1997]. Individual divisions typically overlook the objectives of the enterprise when business processes are reengineered at the local level. Consequently, conflicting business objectives, as well as political and cultural boundaries, are created, which discourage enterprise-wide workflow automation. Indeed, when a corporation deploys workflow applications that span an enterprise, the physical and the political boundaries of independent business units should be considered [Ruiz, 1997]. To address the problem with conflicting business objectives when a process is locally automated, analysts, as Ruiz [1997] suggests, must adopt a company-wide perspective to prevent themselves from developing locally optimized workflows that cannot inter-operate with other applications in the enterprise. To meet the demand for autonomy by decentralized units, workflow applications should allow for locally operational autonomy while enforcing policies at the corporation level.

1.2. Thesis Objectives

The Object Oriented Enterprise Modelling (OOEM) method, based on Wand and Woo [1993] and proposed by Zhao [1995], provides a framework to address these

challenges. First, it captures an organizational process from a company-wide perspective. It focuses on how independent objects work together in order to achieve company objectives. Second, OOEM adheres to the concept of object orientation. In other words, the objects in an OOEM model are autonomous.

Despite its merits, OOEM does not provide workflow specifications. For instance, it does not capture how organizational policies govern the activities of a process. To address this shortcoming, we will introduce a workflow model by adding workflow constructs to OOEM. We will also use the model to develop an implementation architecture of an object-oriented workflow system.

The objectives of this thesis are summarized as follow:

1. to develop a workflow model based on OOEM so that the model for an organizational process can provide workflow specifications and allow analysts to understand the process under study from an enterprise perspective;
2. to develop the architecture of a workflow management system which not only enacts the model but also allows for operational autonomy at the local level; and
3. to build a prototype by following the proposed architecture of the workflow system.

1.3. Thesis Outline

This thesis consists of five chapters.

Chapter 2 provides an overview of the concepts of workflow management. It also briefly reviews some groupware products. Different process modelling techniques are presented with an emphasis on OOEM.

Chapter 3 presents an Object-Oriented Workflow Model (OOWM) which is an extension of OOEM. It also discusses how an OOWM should be constructed. The chapter concludes with the introduction of an implementation model which enacts the OOWM.

Chapter 4 introduces the architecture of an Object-Oriented Workflow Management System (OOWMS). The objective of this chapter is to identify the functionality of components of an OOWMS, independently of specific implementation platforms.

Chapter 5 delves into the details of the design and implementation of an OOWMS, including a discussion of an implementation platform and the limitations of the implementation. The objective of this chapter is to demonstrate how the implementation architecture can be implemented using existing technologies.

Chapter 6 concludes the thesis by reviewing the contents and the contributions of the thesis. It also suggests a framework for future research efforts.

2. Introduction to Workflow Management

2.1. Introduction

Before we present our object-oriented workflow model, we would like to provide a broad overview of the theoretical and practical aspects of workflow management. We will first examine the basic concepts of workflow management, including commonly used terminology. We will then delve into the Workflow Reference Model proposed by the Workflow Management Coalition. The model generalizes the functionality of different workflow products in the market, and it can help us understand the critical components of a workflow system. We will also briefly review some groupware products which can be used to develop workflow systems. Several business process modeling techniques will be presented in this chapter. These techniques include the Integrated Definition Language Approach (IDEF0), the ActionWorkflow™ technique, and state transition diagrams. Moreover, we will discuss what role object-oriented analysis (OOA) plays in business process modeling and briefly look at some object-oriented analysis methods. These methods include Coad and Yourdon's OOA method, Jacobson's Use Case-Driven Approach, and Rumbaugh's OMT method; these three methods are among the best known OOA methods. In addition to these methods, we will provide a brief overview of the Object-Oriented Enterprise Modeling (OOEM) method proposed by Hao Zhao [1995]. As a continuation of the research efforts undertaken by Wand and Woo [1993], the OOEM method serves as a building block of our object-oriented workflow model presented in the next chapter.

2.2. *What Is Workflow Management?*

While some literature defines workflow management as a technology to automate the routing of documentation and tasks [Kobielus, 1997], to co-ordinate user and system participants, with the appropriate data resources, and to achieve defined objectives [Hales and Lavery, 1991], others define workflow management in a broader sense. Georgakopoulos, Hornick, and Sheth [1995] consider that workflow management involves "everything from modeling processes up to synchronizing the activities of information systems and humans that perform the processes" [p. 130]. Jablonski and Bussler [1996], and Joosten [1994] offer similar definitions of workflow management; they perceive workflow management as being a discipline which involves not only business modeling but also the execution of workflows. Broader definitions are parallel to our view of workflow management. Indeed, we believe that workflow management is a process which focuses on analyzing, designing, controlling, and executing business processes through the use of information technologies. Figure 2-1 illustrates the activities involved in workflow management and the requirements for these activities.

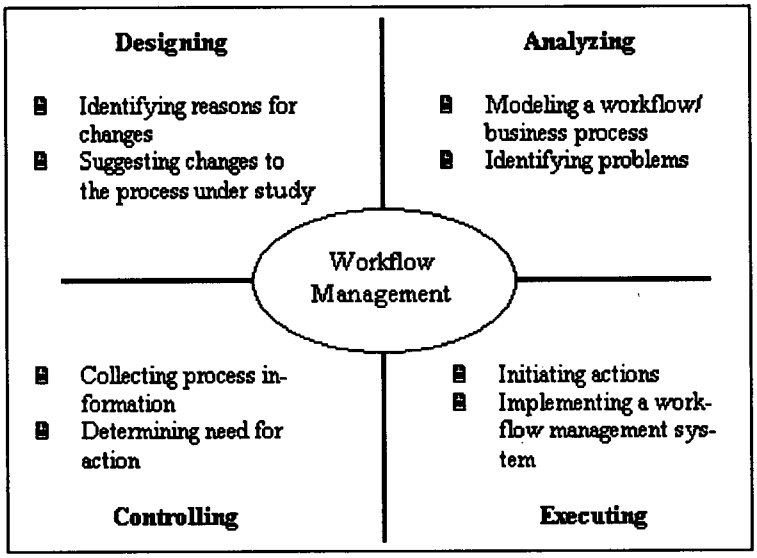


Figure 2-1 Activities Involved in Workflow Management

Not only does workflow management require the analysis of an existing business or workflow process, but it also involves the re-design and the implementation of the process. Table 1 shows some of the changes which result from work flow redesign.

Redesigning work flow:	**Resulting In:**
1. Eliminates tasks	1. Improved productivity
2. Eliminate bottlenecks and delays between the steps	2. Reduced cycle times to complete work
3. Enables work to be processed in parallel rather than serially.	3. Reduced costs
4. Provides simultaneous access to documents by multiple departments/people	4. Improved customer service
5. Allows for quick, simple access to information	5. Improved quality and consistency of results
6. Eliminates rework/retyping	6. Increased revenues (receive revenues sooner)
7. Provides broader responsibilities for workers	
8. Decreases defects	

SOURCE: T. May, "The First Steps to Imaging," Modern Office Technology (April 1991), p. 64.

Table 2-1 Effects of Workflow Redesign

2.3. Workflow Terminology

It is necessary to look at the terminology that we may encounter when we manage a work process and build workflow management systems (WFMS). Please note that this thesis does not offer an exhaustive description of this terminology; instead, it only highlights the most popular definitions used when managing workflow processes.

Many workflow product vendors provide their own definitions as building blocks to develop workflow management systems; these building blocks can affect the capabilities of the WFMS. But no matter how much these definitions may vary, many vendors follow a general specification of a workflow management system proposed by the Workflow Management Coalition (WfMC). The WfMC is a non-profit organization whose purpose is to advance opportunities for exploiting workflow technology through the development of common terminology and standards. By 1996, the WfMC had more than 170 members; nearly all the well-known vendors of WFMS were founding members [WfMC, 1997].

Figure 2-2 outlines the relationships underlying the basic workflow terminology proposed by the WfMC; the figure is directly taken from the WfMC's *Workflow Handbook 1997* [1997, p. 386].

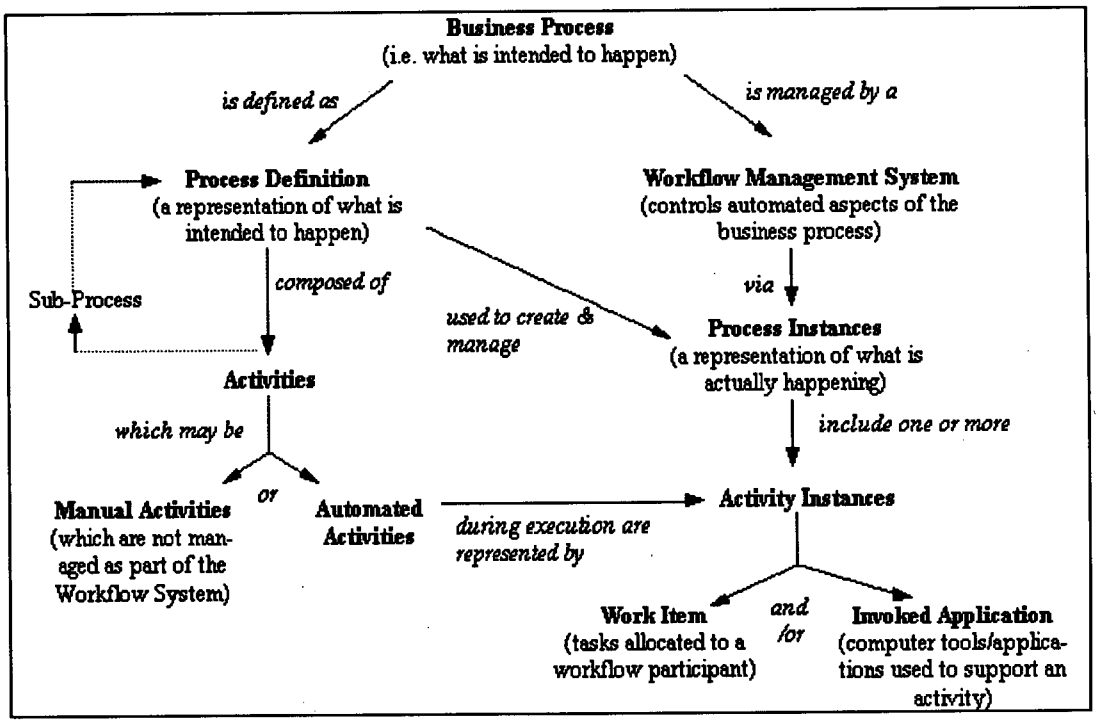


Figure 2-2 Workflow Terminology Relationships

A business process, according to its corresponding process definition, consists of a network of procedures or activities which "collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships" [WfMC, 1997, p. 387]. A reimbursement process is an example of a business process which contains different activities performed by different people. For instance, a division manager may approve a reimbursement form for an amount less than $450, and a corporate controller may approve a form for an amount which exceeds $450. A business process may not be confined to a single organizational unit, and it can span several different functional units and organizations. For instance, Bell Atlantic organized a case team to establish high-speed digital circuits for business customers. The team consisted of members from different departments in several geographic locations. From the organization's point of view, the team is a unit

that "naturally falls together to complete the whole piece of work – a process" [Hammer and Champy, 1993, p. 66].

According to the WfMC, a workflow process is an automated component of a business process which contains both automated and manual activities. An activity is the smallest unit of the business process. The workflow process involves a network of automated activities which are managed and coordinated by the workflow management system. The system initiates a particular activity instance based on one or more pre-conditions, decides if the activity instance is completed according to post-conditions, and moves data between activities based on navigational rules. This system can also monitor the state of all activities and report process status performance to human agents.

2.4. The Workflow Reference Model

According to the WfMC [1997], the workflow management system should help define, create, and manage the execution of workflow. Theoretically, the workflow management system should support the following functions:

- Process modeling

  The WFMS should include a tool to support the analysis and design of a business process. The system should be able to interpret the definition of the process and simulate the workflow under study.

- Process control

  The system should provide a mechanism which can monitor process status, measure process data, identify pre-determined process conditions, and report

process status, performance, and special conditions.

- Process execution

  The system should coordinate the interaction with workflow agents and applications. It should determine necessary actions or guide human agents when decisions should be made.

With these objectives in mind, the WfMC proposed the Workflow Reference Model in 1994. The model not only provides the general architectural representation of a workflow management system, but it also helps MIS practitioners understand the design and the functionality of many commercial workflow products. Figure 2-3 depicts the five major components of the model [WfMC, 1997, p. 260].

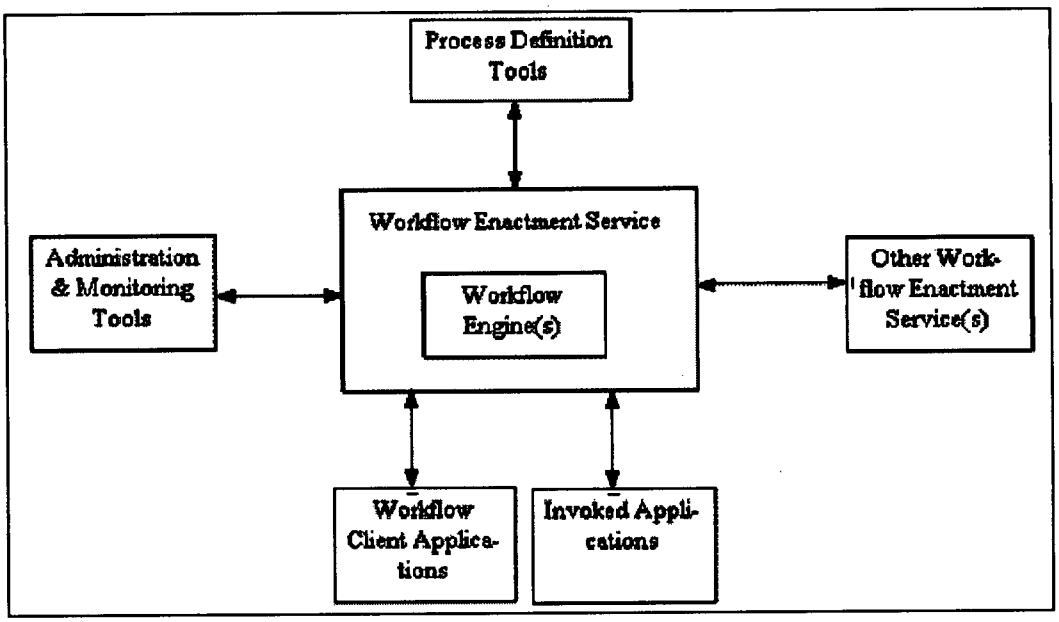


Figure 2-3 The WfMC's Workflow Reference Model

Since an understanding of the model serves as a basis for evaluating different workflow products and may help us determine the implementation platform of our proposed architecture, we will briefly examine each component of the model. The

examination of these components is largely based on the WfMC's *Workflow Handbook 1997* and on Kobielus' *Workflow Strategies* [1997].

2.4.1. Process Definition Tools

Process definition tools allow users to specify automated and manual activities, workflow conditions, and information about individual workflow participants. Theoretically, different workflow products should be able to interpret a logical process representation generated by one vendor's process definition tool; however, users are always limited to using the process definition features that come with their workflow products or to manually convert a form which can be understood by another *workflow engine* [Kobielus, 1997]. A workflow engine is a software program that provides functions to support the execution of business processes including the interpretation of a process definition, the creation of process instances, and the management of their execution [WfMC, 1997].

A process can be graphically described in many different ways which depend upon modeling techniques. We will review some of these techniques later in this chapter.

2.4.2. Workflow Enactment Service

A workflow enactment service creates a process execution environment which contains one or more workflow engines in order to create, manage, and execute particular workflow instances. It plays an administrative role in managing and coordinating workflow applications. It maintains information about process definitions and workflow data. It also invokes external applications which support the

processing of activity instances. The interoperability between different workflow enactment services is made possible by a functional interface which addresses the exchange of process definitions and controls information between the services.

2.4.3. Workflow Client Applications

Workflow client applications provide users with a front-end interface to a workflow enactment service. According to the Workflow Reference Model, the applications should perform the following functions:

- Access workflow relevant, application, and control data;
- Allow users to access a *worklist handler* which enables work items to be passed from the workflow management system to users and allows the status of a work process to be passed between the users and the system;
- Invoke external applications from the worklist handler; and
- Retrieve and manipulate process definition data.

The communication between the workflow client applications and a workflow engine is established via a workflow application interface (WAPI). We do not intend to review the specifications of the interface in this paper. They can be found in *Workflow Handbook 1997* published by the WfMC [1997].

2.4.4. Invoked Applications

Invoked applications allow users to work with workflow relevant information routed to them by the enactment service [Kobielus, 1997]. Application invocation can be undertaken by the enactment service via either direct invocation calls or an *application agent* which "provides a general mechanism for application invocation

independently from any native workflow management system facilities" [WfMC, 1997, p. 409]. External applications can also be invoked by workflow client applications if the applications are under user control or run on local workstations. The flexibility to invoke applications in different ways is very important to the object-oriented paradigm, because each object can independently and autonomously process a work item.

2.4.5. Administration and Monitoring Tools

A workflow system should allow process administrators to perform supervisory operations, including resource control; system configuration; audit management; and initiation, termination and restoration of a process instance. These functions ensure that a process runs smoothly and provide a basis for recovering from system failure.

2.5. Groupware Products

Since there are more than 100 vendors of workflow products, it is beyond the scope of this paper to review every product. However, the architecture of these products is aligned closely to that of the Workflow Reference Model presented in the previous section. These products may address different aspects of workflow functionality; they have been divided into different areas by the trade press and academic literature [WfMC, 1997; Kobielus, 1997; Georgakopoulos et al., 1995]. These areas include imaging processing, document management, electronic messaging, database management, and form management products. We do not make a clear distinction between the product types in this paper. These products provide

development platforms which allow developers to build workflow systems, since many workflow products share the objectives of assisting users in communicating, collaborating, and coordinating.

Groupware is a term for the development platform of the workflow systems. According to the Lotus Corporation, groupware should integrate business logic into the integrated push and pull model to support structured business activities. The integrated model addresses the coordination aspect of the activities [Lotus Corporation, 1995]. When people coordinate, they communicate and collaborate with each other. The push model focuses on the communication dimension: senders simply transmit information to recipients. The pull model addresses information sharing by allowing users to retrieve information from shared databases. To illustrate the application of the concept, a company stipulates specific policies about how a reimbursement form should be processed throughout the organization so that it is properly approved. These policies govern how people should coordinate with each other. The routing of the form is implemented by the push model using a messaging system. The tracking of the form can be achieved by the pull model using a shared database.

Even though many groupware products provide development environments for workflow applications to coordinate work activities, very few products offer an integrated package for process modeling, control, and execution. The process modeling feature is always separate from the other two features. For instance, one of the most popular products in the groupware market, Lotus Notes, which includes

Domino, does not come with a modeling component [Lotus Corporation, 1997]; instead, it offers a flexible development environment to build a workflow application. Oracle's Web Developer Suite 1.5 [Oracle Corporation, 1997] is one of the very few packages that contain all the components of a workflow management system. Its CASE tool is powerful and versatile enough to allow users to model business processes and to automatically translate the models into workflow applications.

As we mentioned earlier, a work process can involve entities external to organizations. Indeed, a workflow application should not be limited to the intra-organizational units by proprietary technological standards. For instance, customers can enter order information directly into a corporate database; the workflow system should then automatically notify the employees to handle such orders. The advent of the Internet has re-shaped the technical architecture of groupware products and offers new opportunities for extending the boundaries of work processes. Many groupware products, such as Lotus Notes and Novell GroupWise 5.2 [Novell Inc., 1997], which once depended upon their own proprietary technology, now support open Internet-based standards. The users do not need the proprietary client software to access information stored in Lotus Notes servers and Groupwise servers. Domino, for instance, turns a Notes server into a Web server and seamlessly integrates the Notes components and information into the Internet [Edwards, 1997]. Novell GroupWise allows users to access information via the Web by providing Java-based client software [Novell Inc., 1997]. Many software developers have even developed products which are solely based on the open standards. These products use the popular Web browsers

as standard interfaces to their systems. Products such as the Netscape SuiteSpot [Netscape Communications Corporation, 1997] and Cold Fusion 3.0 [Allaire Corporation, 1997] are pure web-based groupware products which allow users to collaborate via the Internet. It should be noted that these web-based groupware products offer little support for coordinating work activities, even though they help break down the walls between organizations. These products were developed based on push and pull models, and they provide few form-routing capabilities, such as the ones offered by Lotus Notes.

2.6. *Business Modeling Techniques*

Before we build a workflow management system, we need to understand the process under study. Business modeling graphically represents a business process; it can depict the functional relationships, the information flows, and the roles of workflow participants in the process. Specifically, a process model is an "abstract description of an actual or proposed process" that represents the selected components of the process [Wang, 1994, p. 37]. We divide business process modeling techniques into two approaches - the traditional approach and the object-oriented (OO) approach. The traditional approach usually addresses the functional and informational aspects of a process, whereas the OO approach captures the organizational aspect of a process. The organizational aspect usually represents "where, and by whom in the organization", the components of a process are performed [Wang, 1994]. We will first look into three different traditional process modeling methods in the following sections: the Integrated Definition Language 0 (IDEF0) Approach, the

ActionWorkflow™ technique, and state transition diagrams. However, these sections do not offer a critical review of these techniques.

2.6.1. Traditional Modeling Approach

As we mentioned earlier, this approach focuses on the functional and informational aspects of a process. It requires system analysts to decompose a process into functional areas or to model how information within a process is processed. The problem is that functional representations always change in a dynamic business environment. In turn, changes in these representations may cause inefficiency in system development and maintenance [Coad & Yourdon, 1991, Wang, 1994].

2.6.1.1.Integrated Definition Language 0 (IDEF0) Approach

IDEF0, based on the Structured Analysis and Design Technique™ (SADT™), was developed for the U.S. Air Force Program for Integrated Computer Aided Manufacturing (ICAM) in the 1970s [Laamanen, 1994]. Its original objective was to depict manufacturing processes, but this objective was later extended to include business process modeling application. An IDEF0 model is composed of a hierarchical series of diagrams which gradually display increasing levels of detail, describing functions and their interfaces within the context of a system. Each diagram contains boxes, arrows, and text. The boxes describe activities, processes, or transformations within the context of the system; the arrows represent data or objects associated with a function from which the arrows originate. The syntax and semantic rules of labeling the graphical constructs are beyond the scope of this paper. *Integration Definition for*

Function Modeling (IDEF0) can be consulted for further information [FIPS, 1993].

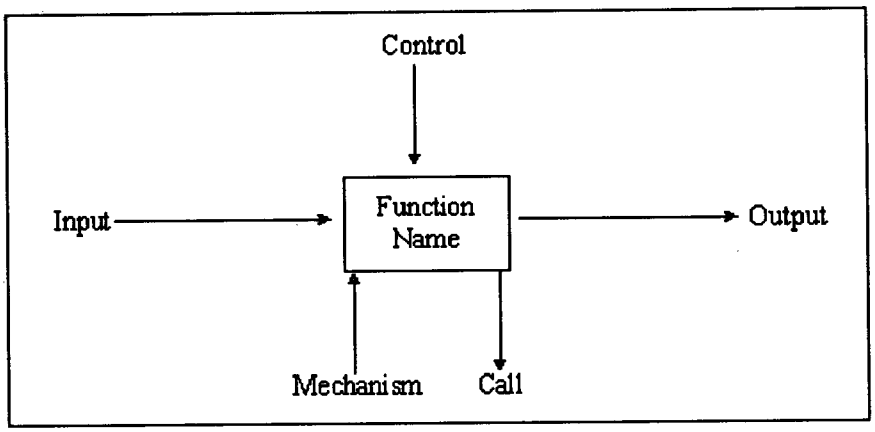


Figure 2-4 IDEF0 Graphical Constructs

Figure 2-4 illustrates the graphical constructs used in IDEF0. The input and output arrows are self-explanatory; however, control, mechanism, and call arrows deserve some explanation here. A control arrow specifies the conditions required for a function to produce outputs. A mechanism arrow represents some of the means that support the execution of the function. A call arrow simply refers to another box which captures the details of the caller box which does not have its own descendent diagram. The called box can be in the same or another model, and it can be shared by multiple caller boxes. To understand how the approach can be applied to business modeling, consider the following example which will also be used to illustrate the next two modeling techniques.

> *In order to have his/her expenses reimbursed, an employee of the ABC Company must submit a reimbursement form to the division manager or the corporate accountant for approval. Reimbursement amounts greater than $200 require a division manager's approval before they are approved by the corporate accountant. All other reimbursements are submitted directly to the corporate accountant. After his/her approval, the*

division manager submits the reimbursement form to the corporate accountant who then cuts the cheques and completes the process.

Figure 2-5 depicts the reimbursement process describe above. The mechanism arrow pointing toward box A-11 represents the division manager who approves a reimbursement request whose value is greater than $200.

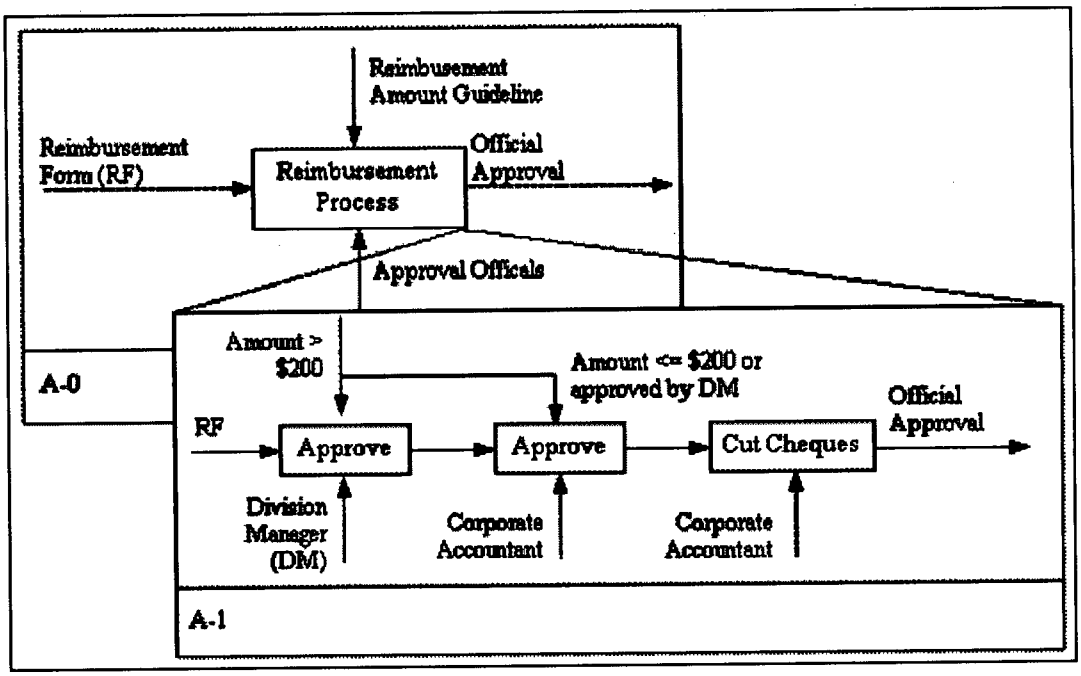


Figure 2-5 A Reimbursement Process in IDEF0

IDEF0 can be used to analyze complex information systems and to describe derivation and relationships among the documents used and produced during process performance [Laamanen, 1994]. However, this method may not be intuitive to first-time learners. Also, "time and cost, the usual business process reengineering objectives, can be derived but are not easily portrayed" [Lakin, Capon, and Botten, 1996, p. 18].

2.6.1.2.ActionWorkflow™ Approach

The ActionWorkflow™ approach focuses on the communication and

coordination aspects of a business process. Unlike IDEF0, the approach focuses on the domain of business processes in which people enter into language based transactions that have consequences for their future activities [Medina, Winograd, Flores, and Flores, 1992]. The approach also captures the negotiation based aspect of business processes. Such an approach is necessary because it combines "structured work with opportunity-based initiative and individual responsibility for quality and customer satisfaction" [p. 283]. Figure 2-6 shows an action workflow loop which consists of four phases.

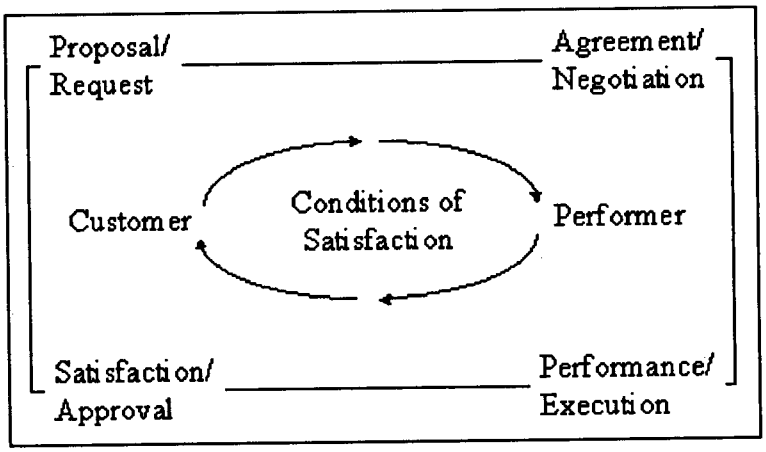


Figure 2-6 An ActionWorkflow™ Loop

The loop begins with a customer who requests that a particular action be completed according to conditions of satisfaction. In the agreement/negotiation phase, the customer and the performer have to mutually agree on the conditions of satisfaction. This agreement may not necessarily be based on negotiations, but sometimes on a shared background of assumptions and standard practices. The performer will then inform the customer of the completion of the action in the performance phase; the customer will lastly declare to the performer that the

completion is satisfactory.

We use the previously mentioned reimbursement process to demonstrate the approach.

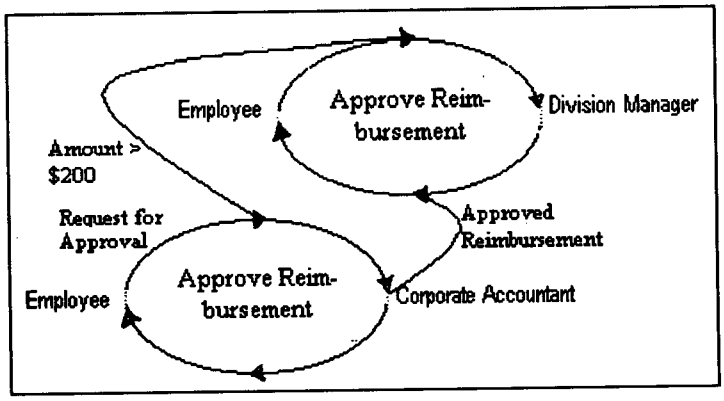


Figure 2-7 A Reimbursement Process in the ActionWorkflow™ Approach

In Figure 2-7, an employee first requests that either the division manager or the corporate accountant reimburse his/her expenses based on the reimbursement amounts. It does not matter to whom the request is first addressed; the corporate accountant completes the workflow loop. It is important to note that the ActionWorkflow™ approach does not address the information flow, but it focuses on the negotiation aspect of a work process. In this example, if the division manager has questions regarding the requested reimbursement, he/she will address his/her questions to the requester. This clarification process will continue in the agreement/negotiation phase until the manager agrees to approve the request.

The ActionWorkflow™ approach depicts the coordination structure of business processes instead of the task structure. The approach has been developed in a series of systems for coordination among users of networked computers. It defines tasks as the requests and commitments of the workflow participants, whereas IDEF0

considers actions of coordination one kind of task or as a flow of information between tasks.

2.6.1.3.State Transition Diagrams

State Transition Diagrams (STDs) capture the time-dependent behavior of systems. STDs can be used to identify a bottleneck in work processes by highlighting the states of the processes. Such systems range from telephone switching systems, high-speed data acquisition systems, to military and command systems. Even though customers do not often demand real-time response from business-oriented systems, a delayed response certainly causes customer dissatisfaction and frustration. Figure 2-8 illustrates the essential components of a STD.

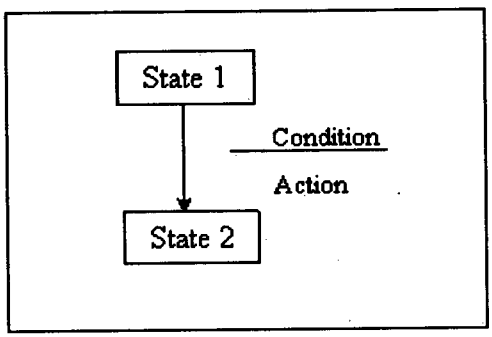


Figure 2-8 Major Components of a State Transition Diagram

A rectangle represents a state of a system; a state describes a characteristic of the system. For instance, in the example of the reimbursement process, waiting for the manager's approval is one state of the process. Waiting for the corporate accountant to cut cheques is another state. A state should represent some observable behavior of the system and last for some finite period of time [Yourdon, 1989]. A transition from one state to another is symbolized by an arrow. The arrow contains two major

components which specify the condition and the action of the transition.

Figure 2-9 demonstrates the application of STDs in the context of the reimbursement process.

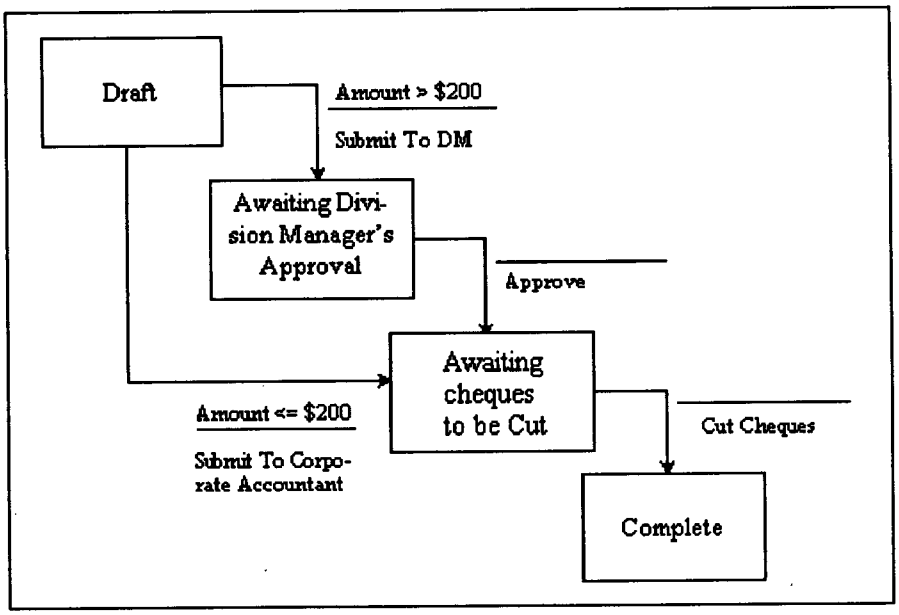


Figure 2-9 A Reimbursement Process in a State Transition Diagram

This diagram is directly taken from the *Lotus Notes Advisor*. The process contains three transitions, with actions being taken by either the division manager or the corporate accountant. The difference between STDs and the other two modeling techniques presented earlier is evident. STDs do not explicitly depict the functional activities of a process. Instead, they only describe the behavior of the process. Even though the functional activities may be illustrated by transition arrows, the objective of STDs is to help users examine the time-dependent behavior of a system and to identify bottlenecks in the system.

2.6.2. Object-Oriented (OO) Approach

The OO approach, according to Jacobson [1995, p. 72], is "very close to the

way in which human beings themselves view the world". It addresses the limitations of the traditional approach. Not only does it capture the organizational aspect of a process, but it also highlights the interactions between objects. Jacobson [1995] further argues the need for building a process model based on the concept of object-orientation. He says that the concept of object-orientation makes the process model become "comprehensive, understandable, changeable, adaptable, and reusable" [p. 69]. Changeability refers to a change in a class of objects in the model which does not affect other classes. Adaptability concerns the specializations of abstract classes based on the concept of inheritance. Reusability means that the classes of objects can be developed in such a way that their properties can be reused in different problem domains. Despite the advantages of the OO approach, objects in a problem domain may be interpreted in many different ways by different OO methodologies. Also, even though the OO methods may help define workflow specifications and derive implementations, they lack workflow model-specific constructs (i.e. pre-conditions and post-conditions to an activity) and provide no explicit support for business process modeling [Georgakopoulos, et al., 1994]. The following sections provide an overview of four OO modeling methods. These methods include Coad and Yourdon's OOA/OOD, Jacobson's use case driven approach, Rumbaugh's OMT, and the OOEM approach.

2.6.2.1.Coad and Yourdon's OOA/OOD

Coad and Yourdon [1991] propose the Object-Oriented Analysis (OOA) method in their book *Object-Oriented Analysis*. The method consists of five major

activities:

1. Finding class and object.
2. Identifying structures which capture the relationships between objects.
3. Identifying subjects which are used to partition large complex models.
4. Defining attributes.
5. Defining services.

Please note that the sequence of these activities does not affect how a model is built. These activities may result in a OOA model which consists of five layers:

1. Subject layer, which serves as a partitioning mechanism;
2. Class & Object layer, which captures classes and objects;
3. Structure layer, which captures inheritance and whole part structures;
4. Attribute layer, which captures attributes and instance connections between classes and objects; and
5. Service layer, which captures methods and message connections between classes and objects.

Coad and Yourdon [1991] also extend the OOA method to address the design of a system in their book *Object-Oriented Design*. The Object-Oriented Design (OOD) method introduces four additional components to the OOA model. These components include:

1. Human interaction component, which studies how users interact with a system by means of prototyping;
2. Data management component, which provides the basis for storage and retrieval of

objects from a database management system;

3. Problem domain component, which carries the OOA results into the OOD model, thereby improving the results by means of this component;
4. Task management component, which determines a need for tasks in the system and defines the tasks.

2.6.2.2.Jacobson's Use Case-Driven Approach

While Rumbaugh's OMT and Coad and Yourdon's OOA/OOD methods are based on software design and implementation, Jacobson's business process modeling approach emphasizes the modeling of organizational activities [Jung, 1997]. Jacobson's use case-driven approach was originally developed for system design and analysis [Jacobson et al., 1992]. The approach was later extended to include business process modeling [Jacobson et al., 1995]. Jacobson's modeling technique consists of two phases: the use case model which describes "what the business is meant to accomplish" [p. 146], and the object model which focuses on "how the business is to work" [p. 146].

The construction of the use case model begins with the identification of a problem domain (a business system) and then an environment (actors) which interacts with the domain. In the case of the reimbursement process presented in Section 2.5.1.1, the employees are the actors in the problem domain. A sequence of transactions is also identified and presented as use cases, which may be grouped into as a use-case class based on their similar characteristics. For instance, processing a PR form whose amounts exceed $100 and processing a form whose amounts are less than

$100 are different use cases but may belong to the same use-case class. The graphical constructs of the use case model are illustrated in Appendix A. The use case model provides a top-level view of a business process; the details are captured in the object model.

The object model contains three different types of objects: *control objects*, *entity objects*, and *interface objects*. Control objects represent a set of operations which may not have direct responsibility for contacts with the business environment. Interface objects are responsible for handling communication between the system and the external environment. For instance, these objects can be sales representatives who have direct contact with customers. Entity objects represent "occurrences such as products and things that are handled in the business" [p. 116]. Examples of entity objects are a sales order and a reimbursement form. To construct an object model, the following steps can be followed:

- Find subsystems that reflect the structure of an organization.
- Describe the use cases in relation to subsystems since the use cases may span different subsystems in the organization.
- Identify objects which work together to realize a use case.

The constructs of the object model are shown in Appendix B.

2.6.2.3.Rumbaugh's OMT

Rumbaugh's Object Modeling Technique (OMT) [1991] is composed of three models: the *Object Model*, the *Dynamic Model*, and the *Functional Model*. Although these three models of a system are constructed independently, Rumbaugh, et al.

believe that they are essential to derive a complete representation of a system. The object model describes the static structure of objects in a system through identity, relationships, and operations. The dynamic model, represented in state diagrams, portrays a sequence of operations over time within a system by modeling events, states, and state transitions. The functional model, represented in a data flow diagram (DFD), shows how data are transformed by the system's processes.

Rumbaugh considers the OMT "an enhanced form of the Entity-Relationship (ER) approach" [p. 217]. He further claims that OMT "synthesizes different camps of thought from databases, object-oriented concepts, and software engineering" [p. 273]. Guidelines for constructing different models are presented in Appendix C.

2.6.2.4. The OOEM Approach

The Object-Oriented Enterprise Modeling (OOEM) methodology, presented in [Zhao, 1995], is based on Wand and Woo's modeling rules [Wand and Woo, 1993] which are derived from Bunge's ontological[1] principles [Bunge, 1977]. With the notion that objects should reflect a "natural" view of the world [Wand, 1989], Wand and Woo [1993] adopt Bunge's ontological approach to develop a theoretical foundation for object-oriented modeling. OOEM, built on this foundation, provides a set of object-oriented analysis rules, a request propagation algorithm, and a model representation technique [Jung, 1997].

---

[1] Ontology, according to Angeles [1981], is defined as "That branch of philosophy which deals with the order and structure of reality in the broadest sense possible".

2.6.2.4.1.OOEM Constructs

Since OOEM is based on ontological principles proposed by Bunge [1997], it is useful to briefly summarize them below:

- The world is composed of things that possess properties.
- Attributes are characteristics humans assign to things.
- Every property can be modeled as an attribute.
- Everything abides by laws which are invariant relations among properties of things. These laws limit possible states and state transitions.
- Interacting things form systems or aggregates.
- Everything changes, and every change is a change of states of things.

These principles "provide concepts for how we can reason about the world" and serve as "the basis to model and talk about organizational activities" [Jung 1997, p. 15]. Based on these principles, Wand and Weber [1990] extend Bunge's ontology [1977] to information system (IS). Wand [1989] also categorizes the ontological constructs for IS into four categories as summarized by Zhao [1995, p. 11].

- Static model of a thing, which describes thing, property, state, transformation, and history;
- Dynamic model of an individual, which refers to event, transformation, and history;
- Static model of a system, which captures coupling, system, composition, environment, structure, subsystem; and
- Dynamic model of a system, which describes stable and unstable state, external

event, internal event, well-defined event, and poorly defined event.

Detailed information about these constructs can be found in Appendix F.

The fundamental constructs of OOEM are : *objects*, *services*, *attributes*, and *requests*. They are derived from the mapping of ontological constructs to the Object-Oriented constructs. Table 2-2 briefly outlines these constructs, and the details are presented below:

Construct	Meaning
Object	A model of a substantial thing in the problem domain that interacts with other objects. An object can be a client or an internal object. A client object is not considered a part of the system directly under study whereas an internal object is an object within the system. An object can represent an organizational unit, a division, a department, or a role.
Interface Attribute	A mutual property of things. It serves as a mechanism by which objects communicate with each other.
Internal Attribute	An intrinsic property of a thing. It can represent knowledge internal to an object and inaccessible to other objects
Service	A well-defined series of actions which satisfy a request. A service may access or modify the objects.
Request	A representation of an interaction between objects. It changes the interface attributes of a recipient object, and it may trigger a service.

Table 2-2 Summary of the OOEM Constructs

a) **Object**: Some object-oriented literature loosely defines the concept of an object. For instance, Jacobson et al. [1995] believe that an object is an occurrence that contains information and offers behavior within a problem domain. He considers, for example, a division reconciliation record an object in a company and a manager

another object in the same company. A broad definition of object does not give analysts effective guidelines to identify object types. OOEM asserts that the world is made of objects based on the ontological principle that states that the world is composed of things [Zhao, 1995]. An object is "a model of a substantial thing in the problem domain that interacts with other objects" [p. 12]. To be qualified as an object in the problem domain, the candidate for an object, as illustrated in Figure 2-10, should interact with other objects by either generating or responding to a *request*, or providing *services* (See Wand and Woo's modeling rule #2 in Appendix E). In other words, we do not consider the reconciliation record to be an object since it does not interactively participate in a process. However, depending upon the problem domain, we *may* consider the manager an object. An object and its dynamics are described by its *attributes*, *services*, and *requests* for other services.

b) **Attributes**: According to Bunge's ontological principles, attributes model properties of things. They represent the state of an object and its knowledge of the problem domain [Jung 1997]. In other words, as indicated in Figure 2-10, an attribute must belong to an object. There are two types of attributes: *internal* and *interface* attributes. Internal attributes model the intrinsic properties of a thing; they are not known to other objects and can only be accessed or modified through the services of the object. For example, the division reconciliation records represent the manager's knowledge of divisional financial status. These records should be kept inside the manager object which manipulates these records via a

service. Interface attributes model the mutual properties of things; they provide a mechanism by which objects communicate with each other. Zhao [1995] makes an interesting analogy between object communication and a procedure call in computer programming. Interface attributes function like procedure call parameters which enable one program to pass arguments to another. It should be noted that the change in interface attributes as a result of incoming requests *may* trigger a *service*.

c) **Service**: Ontologically, a service models the state transformation of an object. It comprises a series of actions performed by an object with the purpose of satisfying a *request*. These actions are encapsulated into an object. When a request is sent to an object, it invokes a service in the object (See Figure 2-10). A service, in its course of action, may generate or spawn one or more requests to objects.

d) **Request**: An interaction between two objects can be modeled by a request. When an object wants to communicate with another object, it sends a request to the latter. Ontologically speaking, the interaction is the change in the history of one thing as a result of the existence of another thing [Bunge, 1977]. Accordingly, sending requests changes the state of the responding objects by modifying the interface attributes of the recipients. The change in interface attributes may trigger services of the responding objects which may undergo state transformation [Zhao, 1995]. The consequence of a request may affect the state of either a requesting object, responding objects, or both. For instance, when Object A sends a request to Object B, the state of each object may be affected in the following situations:

1. Object A can be in an unstable state if it waits for the response from Object B, but Object B is doing nothing about it. For instance, a job seeker sends an unsolicited job application to a company which does not reply to him/her.
2. Object A is not concerned about the response to its request. In other words, it simply delivers information to Object B whose state becomes unstable since Object B needs the information to perform a service. This point can be illustrated by the situation of a purchase requisition process whereby a division manager approves a requisition form and forwards it to a corporate accountant for further approval. The manager does not expect the accountant to respond to him/her; instead, the accountant should inform a requester of the approval status.
3. Objects A and B are in an unstable state. This is, in fact, a combination of situations (1) and (2). When Object A sends a request to Object B, Object A expects Object B to act upon the request and to provide a response to the request. To illustrate this point, we can consider a room-booking inquiry process whereby a requester phones an administrative clerk to inquire about a room-booking schedule.

A request usually carries with it information which is required by the receiving object to process the request. The object obtains this information from its interface attribute which has been modified by the request.

It is important to note that a request is a communication protocol between two or more objects. There can be two kinds of protocols. One protocol involves two

objects where the first sends a request to the second, and the second directly responds back to the first. For instance, if an employee submits a reimbursement request whose value is less than $100 to the accountant, he/she will expect a response back from the accountant. Another protocol involves more than two objects when the sender of the request receives a response back from a different object to which it does not send the request in the first place. The response, in this case, is in the form of a different request. To illustrate our point, a customer orders an item from a sales representative object via phone. The sales representative, in turn, sends the order information to the accounting clerk who generates an invoice and mails it to the customer. It is the clerk who sends the invoice to the customer as a response to the original.

Since OOEM does not enforce the constraint that each request must have an immediate reply, the distinction between these two protocols is essential for understanding how an external request is processed by internal objects and which internal objects an external object interacts with. This understanding will be formalized in the concept of request propagation.

Figure 2-10, adapted from Tan [1997], describes the relationships of the constructs in OOEM using the entity-relationship diagram.

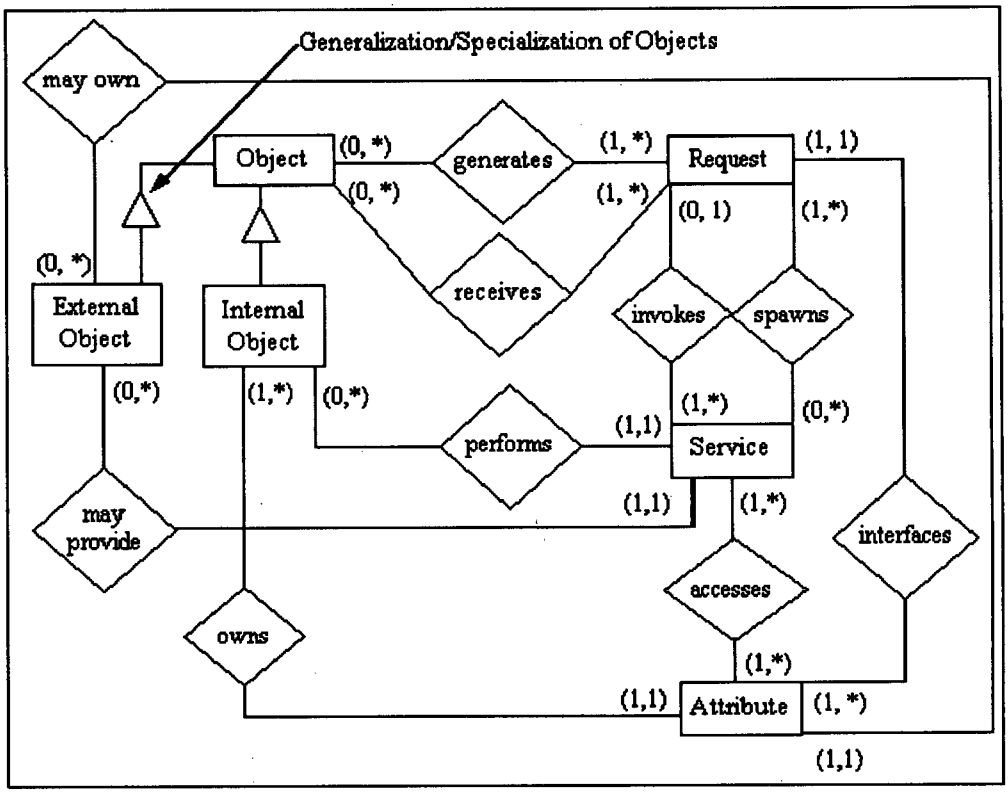


Figure 2-10 Meta-Model of OOEM

The meta-model depicts the constructs of OOEM as boxes and the relationships between the constructs as diamond-shaped symbols. The cardinality constraints, expressed by (m,n), mean that an entity is associated with at least m and at most n occurrences of the related entity. For example, in Figure 2-10, an object can generate no requests or any number of requests, but a request must be generated by at least one object. A triangle symbol indicates the specialized roles that external and internal objects play in the model. For example, an external object which belongs to a general object class may own attributes and perform services, since the information about the internal structure of an external object may not be readily available for analysts. However, an internal object whose internal representation should be made known to the analysts must own at least one attribute and perform at least one service.

2.6.2.4.2.Request Propagation

The central theme of OOEM is the concept of request propagation which defines an organizational process in terms of the behavioral characteristics of the participating objects. These characteristics include the interaction (requests) between objects and the operations (services) resulting from the interactions. The concept states that an organizational process is triggered by an external request [Wand and Woo, 1993]. As a result of the request, the internal object which receives the request may generate requests to other internal objects, which in turn may further generate more requests. This sequence of request generation, known as request propagation, may end with an external object receiving the result of the request or with internal object which do not generate any further requests to other objects.

2.6.2.4.3.OOEM Representation Technique

Figure 2.11 shows the graphical constructs of OOEM.

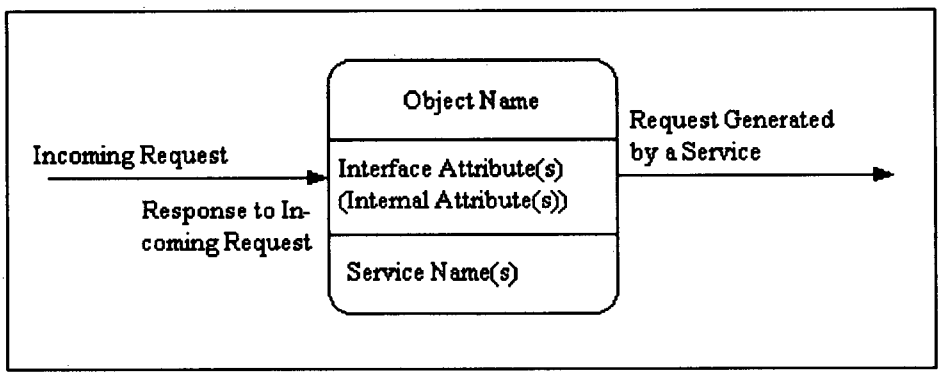


Figure 2-11 OOEM Graphical Constructs

We mentioned that an incoming request can imply a response. This response should be placed at the head of an arrow, whereas the request should be placed at the end of the arrow. Figure 2-12 illustrates the reimbursement process in the OOEM

model.

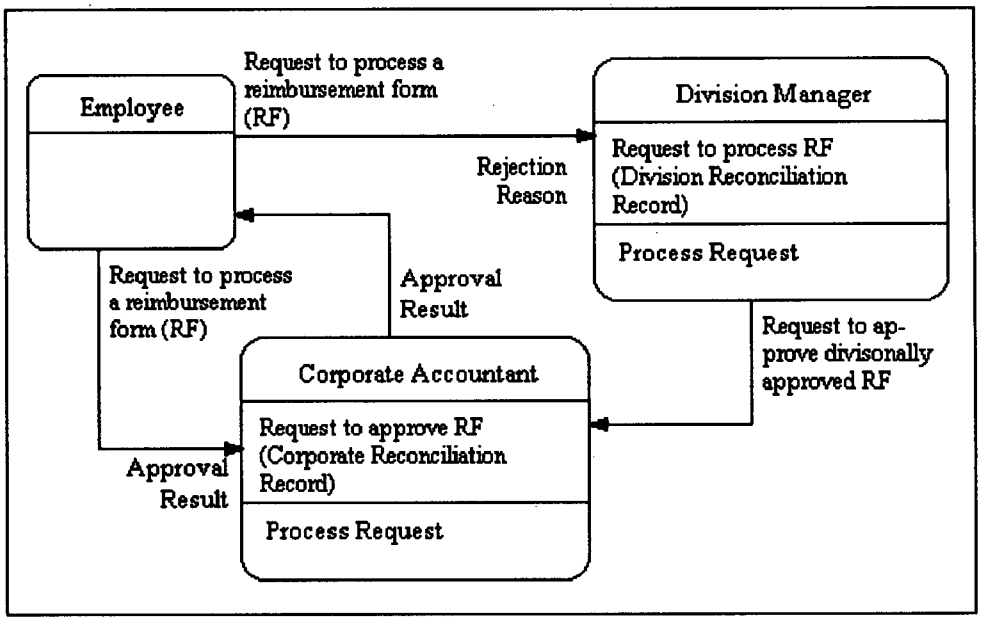


Figure 2-12 The Reimbursement Process in the OOEM Model

Several assumptions were made to construct the model. For instance, we assumed that the division manager will notify an employee if his/her form is rejected and that both the division manager and the corporate accountant maintain records for reconciliation. A rejection reason is modeled as a response to an employee's request for reimbursement whose amount exceeds $200. It should be noted that an approval result is sent to the employee object as a request. The result is a response to the employee's request for reimbursement whose amount exceeds $200.

Zhao also introduces an *object template* (See Appendix D) to capture the internal structure of an object. The object template may be used not only for describing an internal object but also for a client object if more information about the latter object is available (Tan, 1997). The template specifies which interface and internal attributes are accessed and used by a service, and which requests are

generated by the service.

2.6.2.4.4.A Shortcoming of OOEM

Even though OOEM provides a bird's eye view of organizational activities within a problem domain by focusing on the interactions among objects, it, in fact, does not capture all the behavioral aspects of an organizational process. According to Curtis et al. [1992], the behavior of a process is determined by the flow of control among its functional units. OOEM describes workflow participants, their responsibilities, and their interactions in a process; it does not capture the execution order of work [Zhao, 1995]. Amber [1997] suggests that one of the basic concepts for workflow modeling is the flow of work which determines "the control and data flow between activities" [p. 63]. To address this limitation, Zhao [1995] suggests that pre-conditions and post-conditions may be specified for services in an Internal Object Template (IOT). However, he does not formalize his suggestion in the context of workflow modelling.

2.7. Summary

This chapter provided an overview of workflow management. We introduced the basic workflow terminology proposed by the Workflow Management Coalition (WfMC) whose objective is to develop common terminology and standards for workflow technology. We also presented the WfMC Reference Model which identifies common characteristics of many workflow products in the market. Current trends in the groupware market were discussed as well. Finally, we reviewed two major categories of business process modeling techniques: the traditional approach and

the object-oriented approach. While the object-oriented approach seems to overcome some of the traditional approach's limitations, the OO approach also has its own limitations; these later limitations can be addressed by the object-oriented workflow model proposed in the next chapter.

3. The Object-Oriented Workflow Model

3.1. Introduction

The previous chapter presented the fundamental concepts of workflow management. This chapter introduces the Object-Oriented Workflow Model (OOWM) which represents our view of a business process in an object-oriented context. Based on our observation of the limitations of OOEM in association with the concepts of workflow management, we will present the OOWM which extends OOEM by including additional workflow constructs. The reasons that we build on OOEM are presented as follow:

1. Compared to other OO approaches, OOEM reflects how human beings perceive an organizational process. While other OO approaches are geared toward software development, OOEM is designed to provide a high level of abstraction to describe essential business activities. Accordingly, OOEM is more understandable to analysts and management who are more concerned with business processes rather than information on processing details.
2. Because of (1), OOEM provides analysts with a framework to design information systems without overlooking a company's objectives.
3. Because OOEM deliberately excludes certain low level details, such as the details of how objects process requests, and it concentrates how the objects communicate with each other, it offers analysts a basis for developing a system in a decentralized environment. Such a system gives decentralized units the flexibility to operate autonomously.

Since the example of the reimbursement process presented in the previous chapter will be used to demonstrate our concepts, we present it again in the following paragraph:

> *In order to have his/her expense reimbursed, an employee of the ABC Company must submit a reimbursement form to the division manager or the corporate accountant for approval. Reimbursement amounts greater than $200 require a division manager's approval before they are approved by the corporate accountant. All other reimbursements are submitted directly to the corporate accountant. After his/her approval, the division manager submits the reimbursement form to the corporate accountant who then cuts the cheques and completes the process.*

We will also put the OOWM into practice by presenting the OOWM method in order to provide analysts with a systematic approach to building the OOWM for an organizational process. At the end of the chapter, we will present the implementation model of an Object-Oriented Workflow Management System (OOWMS) which automates the organizational process based on the OOWM. The implementation model serves as a building block of the architecture of the OOWMS which will be formally presented in Chapter 4.

3.2. The Object-Oriented Workflow Model (OOWM)

Figure 3-1 illustrates how OOWM constructs are developed. The OOWM constructs, as represented by the first block in the diagram, are based on the OOEM constructs and the concepts of workflow management. Similarly, the OOEM constructs are derived from a combination of the ontological constructs for

information systems (IS) and the Object-Oriented constructs. The ontological constructs are based on Bunge's ontology [Bunge, 1977].

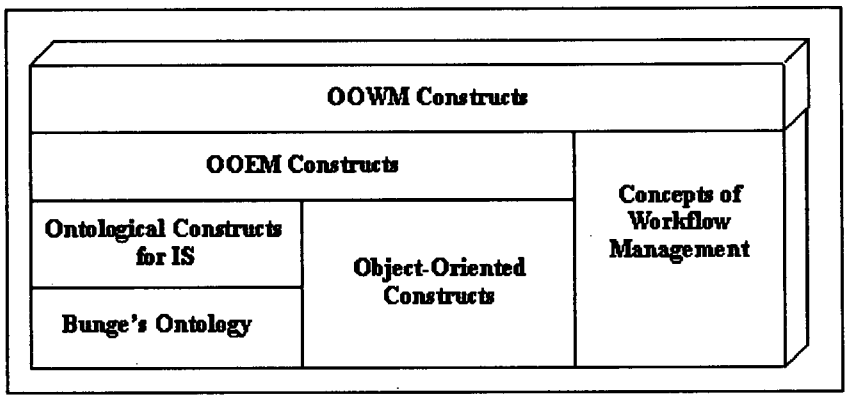


Figure 3-1 The Building Blocks of the OOWM

To address the shortcoming of OOEM, we extend OOEM by introducing additional constructs to support workflow modeling. These constructs include: *activity* and *business rule*. An activity refers to a unit of work that forms part of a business process [WfMC, 1997]; the activity can be a manual or automated activity. A business rule means an organizational policy that governs activities within a process. Figure 3-2 depicts all the constructs and their relationships in our object-oriented workflow model.

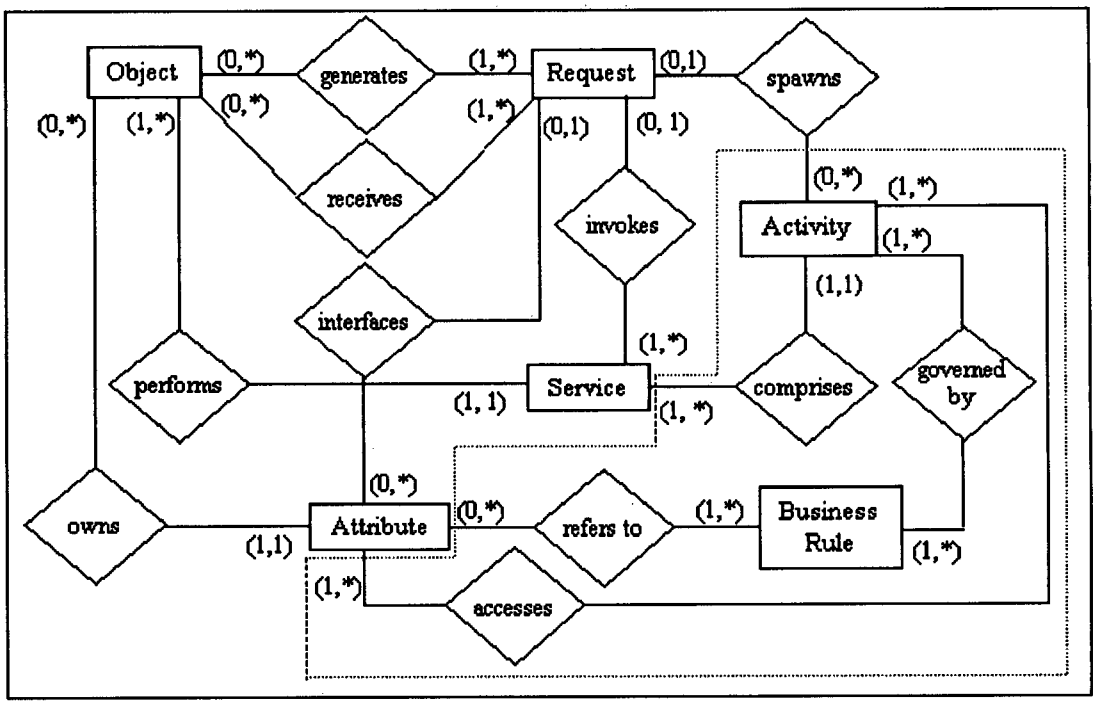


Figure 3-2 The Object-Oriented Workflow Model (OOWM)

The added constructs are enclosed by a dotted line in Figure 3-2. It is important to note that the addition of the new constructs also introduces changes in the relationships between the constructs originally defined in OOEM. In OOEM (See Figure 2-10), a *service* accesses at least one attribute and spawns any number of requests. But Figure 3-2 shows that it is an *activity* which accesses at least one attribute and spawns any number of requests. The service in the OOWM, performed by at least one object, comprises at least one activity, but OOEM does not define the relationship between the service and the activity. A *business rule* entity, which is not included in OOEM, is introduced in the OOWM to determine when activities should begin and end if certain conditions are true. These conditions always refer to the information included in the interface or internal attributes. We will examine the constructs in the proposed model.

3.2.1. Constructs in the Object-Oriented Workflow Model (OOWM)

In Chapter 2, we presented the modeling constructs of OOEM. In this section, we will formally define the additional constructs that we introduced earlier. While the information about the modeling constructs originally defined by OOEM is taken from existing OOEM literature [Wand and Woo, 1993; Zhao, 1995; Tan, 1997; Jung, 1997], we may extend the definitions of these constructs in order to support the semantics of our workflow model.

3.2.1.1.Activity

Activities are the basic units of operations taken by an object; they form services. A service contains an ordered set of activities $\{A_1 \ldots A_n\}$. The mechanism of activities is encapsulated within an object. According to Figure 3-2, they can access interface and internal attributes and generate requests to other objects. For instance, when the manager approves a reimbursement form, he/she needs to access the information about the request and the division reconciliation records. Figure 3-2 also shows that the activities are initiated by an object in accordance with business rules specified by an organization. The division manager, for example, cannot approve a reimbursement request unless the amount of the request exceeds $200.

By examining the relationships between a traditional activity-based model and OOEM, we may be able to determine how a service can be broken into activities in the context of our workflow model. Figure 3-3 shows the activity diagram and the OOEM model. We compare incoming and outgoing requests, and the returning result of the requests in OOEM to the information flows going into or out of activity blocks.

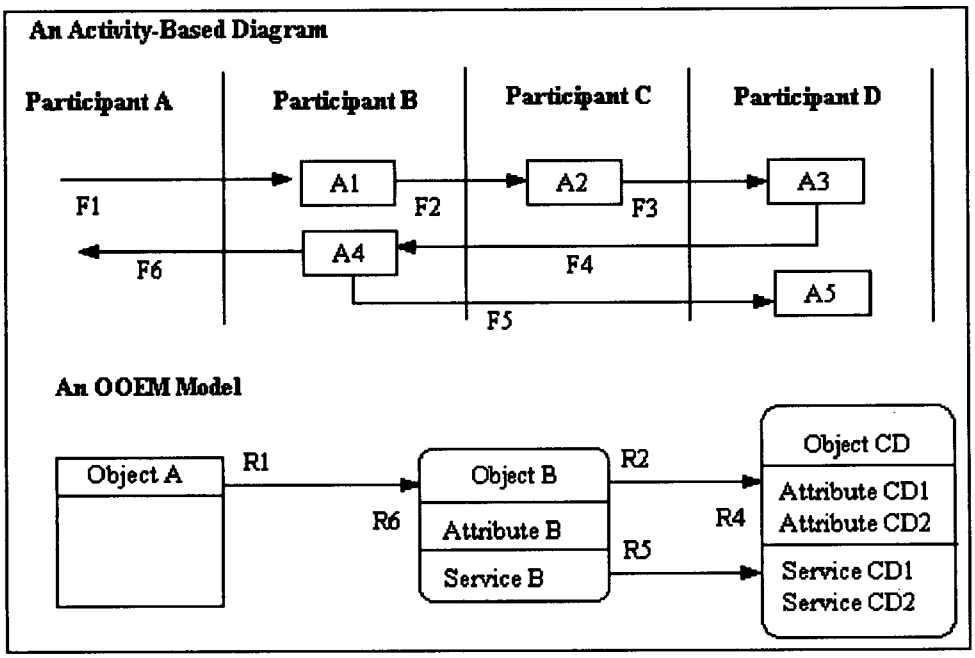


Figure 3-3 An Activity Diagram and an OOEM Model

The activity diagram is divided into columns. Each column corresponds to a participant in the process. The boxes in the column represent the activities associated with the participant, and an arrow indicates an information flow. All information and request flows in both diagrams are labeled. It should be noted that F1 in the activity diagram corresponds to R1 in the OOEM model, F2 to R2, F4 to R4, F5 to R5, and F6 to R6. Service B contains activities A1 and A4; service CD1 has A2 and A3; and service CD2 contains A5. The diagram helps explain the characteristics of an activity in the context of our OOWM. The granularity of an activity is related to an interaction between objects. In our OOWM, the activity begins with an incoming request or the response to a request from another object. It terminates when one of the following conditions is met:

1. The activity generates a request to another object.
2. The activity has completed all it needs to do.

In the activity diagram, there is no indication of showing which activities belong to a service in the OOEM model. For instance, we cannot decide if A2 and A3 form a service and if A5 belongs to another service. In OOEM, every service is responding to at least one request; therefore, a request defines the granularity of a service. Also, the activity diagram does not indicate which participants should form an object in OOEM. But when we refer to the OOEM model, we cannot tell what really happens in a service; we cannot identify the sequence of generating R2 and R5. In Section 3.2.3.1, we will show how an OOWM, an extension of OOEM, can be converted to an activity diagram.

3.2.1.2.Business Rules

Even though objects are autonomous and independent, their responsibilities within an organization are defined in organizational policies. It is the organization's policies that determine when and what tasks are processed and, by whom [Rupietta, 1997]. Business rules basically achieve the same objectives as organizational policies. As pointed out by Rupietta [1997, p. 165], "the cooperation and communication between members of an enterprise in workflow management systems requires that organizational rules be closely followed" [p. 165].

A business rule can be interpreted into pre-conditions and termination conditions for an activity. The pre-conditions can be defined as entry criteria to an activity, and the termination conditions as completion criteria for a particular activity [WfMC, 1997]. These conditions may refer to information accompanying requests or to state information about a process instance. For instance, the pre-condition for the

division manager to approve a reimbursement request is that the value of the request must exceed $200. The termination condition for the approval activity is when the manager approves or rejects the request. The pre- and termination conditions also affect the generation of requests. For instance, if the reimbursement request is approved by the division manager, another request is generated for the corporate accountant. It is important to note that the introduction of business rules does not violate the concept of object independence and encapsulation. These rules do not restrict how the object should perform the tasks; instead, they only control the interactions among the objects (i.e. incoming and outgoing requests).

3.2.1.3.Object Activity Template (OAT)

An Object Activity Template (OAT), shown in Table 3-1, is used to specify the behavior of objects. It is an extension of Zhao's Internal Object Template (IOT) and is intended to capture workflow information. This information includes activities and business rules which govern the activities. The name of an activity is expressed in the Activity column. Since business rules provide organizational control, we need to include them to understand their implications with respect to an activity. However, in actual implementation, the rules are stored separately from the objects to preserve object autonomy. The business rules can be represented by pre- and termination conditions for an activity; these conditions are represented by the Pre-Conditions and Termination Conditions columns.

Object Name - Object Code								
Interface Attributes	**Internal Attributes**		**Services**					
			Service 1					
incoming interface attributes	Internal Attribute to support Service 1	Access Mode	Pre-Conditions	Activity	Termination Conditions	Request Generated	Receiver	
activity code - R returning interface attributes			Pre-condition 1	activity code Activity 1	Termination condition 1	Request Generated from Activity 1	Object receiving a request generated from Activity 1	
activity code - R returning interface attributes			Pre-condition 2	activity code Activity 2	Termination condition 2	Request Generated from Activity 2	Object receiving a request generated from Activity 2	
			Service 2					
incoming interface attributes	Internal Attribute to support Service 2	Access Mode	Pre-Conditions	Activity	Termination Conditions	Request Generated	Receiver	
activity code - R returning interface attributes			Pre-condition 1	activity code Activity 1	Termination condition 1	Request Generated from Activity 1	Object receiving a request generated from Activity 1	

Table 3-1 An Object Activity Template (OAT)

Because each service consists of several activities, all of its associated activities and conditions are shown in a sub-table of a service. Each row of this table represents an activity which is attached to pre- and termination conditions. If a precondition of one activity holds, then that activity will be performed by an object. Similarly, if a termination condition is true while an activity is being performed, the activity will stop. The *returning interface attribute* captures the response to an incoming request. This response is labeled with an activity code and "R" in order to distinguish which activity

generates the response. The access mode indicates how activities of a service use internal attributes. "U" indicates read access, and "M" means read and write access.

To illustrate the application of the template, let us consider the object activity template for the division manager object in the purchase reimbursement process presented at the beginning of this chapter. The template is shown in Table 3-2.

Division Manager - DM							
Interface Attributes	**Internal Attributes**		**Service**				
Request to process RF: Items, Amounts, Requested Date, Requester DM-3-R Rejection Reason ↓ Employee Object	Division Reconciliation Record: Items, Amounts, Requested Date, Requester, Approval Decision, Approver	M	**Process Request for Reimbursement**				
			Pre-Conditions	Activity	Termination Conditions	Request Generated	Receiver
			Amounts > $200	DM - 1 Approve a request	Request Approved or Rejected		
			Request Approved	DM - 2 Generate a request	Request Generated	Request to approve divisionally approved RF	Corporate Accountant
			Request Rejected	DM - 3 Generate a reject reason	Reason Generage		
			DM-2 or DM-3 Completed	DM - 4 Update record			

Table 3-2 The Object Activity Template for the Division Manager

In the reimbursement process, an employee submits a request to the division manager. Such a request should be accompanied by information such as the name of the requester, the reimbursed amounts, the requested date, and the purchased item. They are listed under the Interface Attribute column as *incoming interface attributes*. The internal attributes that support the activities include the requester name, the requested date, the requested amounts, the purchased item, the approval decision, and

the approver name. The manager can approve or reject the request only if the request value exceeds $200. This condition is reflected by the 'Amounts > $200' statement in the Pre-Conditions column. If the condition is true, the activity 'Approve a request" labeled with the code 'DM' will be executed. This activity terminates when the request is either approved or rejected. The division manager will generate a request to the corporate accountant if the reimbursement request is approved; this information is captured in the second row of the Process Request for Reimbursement sub-table. The receiving object of the generated request, as specified in the Receiver column, is the corporate accountant. A rejection reason will be returned to the requester as an immediate response to the reimbursement request if the manager rejects the request.

3.2.1.3.1.From an OOWM to an Activity Diagram

Since activities and their associated conditions are represented in an Object Activity Template (OAT), it is always possible to convert an OOWM into an activity diagram. We can treat both requests and responses (i.e. the returning results of requests) in the OAT as the information going into or out of activities blocks in different participant columns in the activity diagram. In Figure 3-4, R1, R2 and R3 in the OAT correspond to F1, F2 and F3 respectively in the activity diagram. If a response to R1 is expected from object 2, how can a returning result of R1 be represented in the activity diagram?

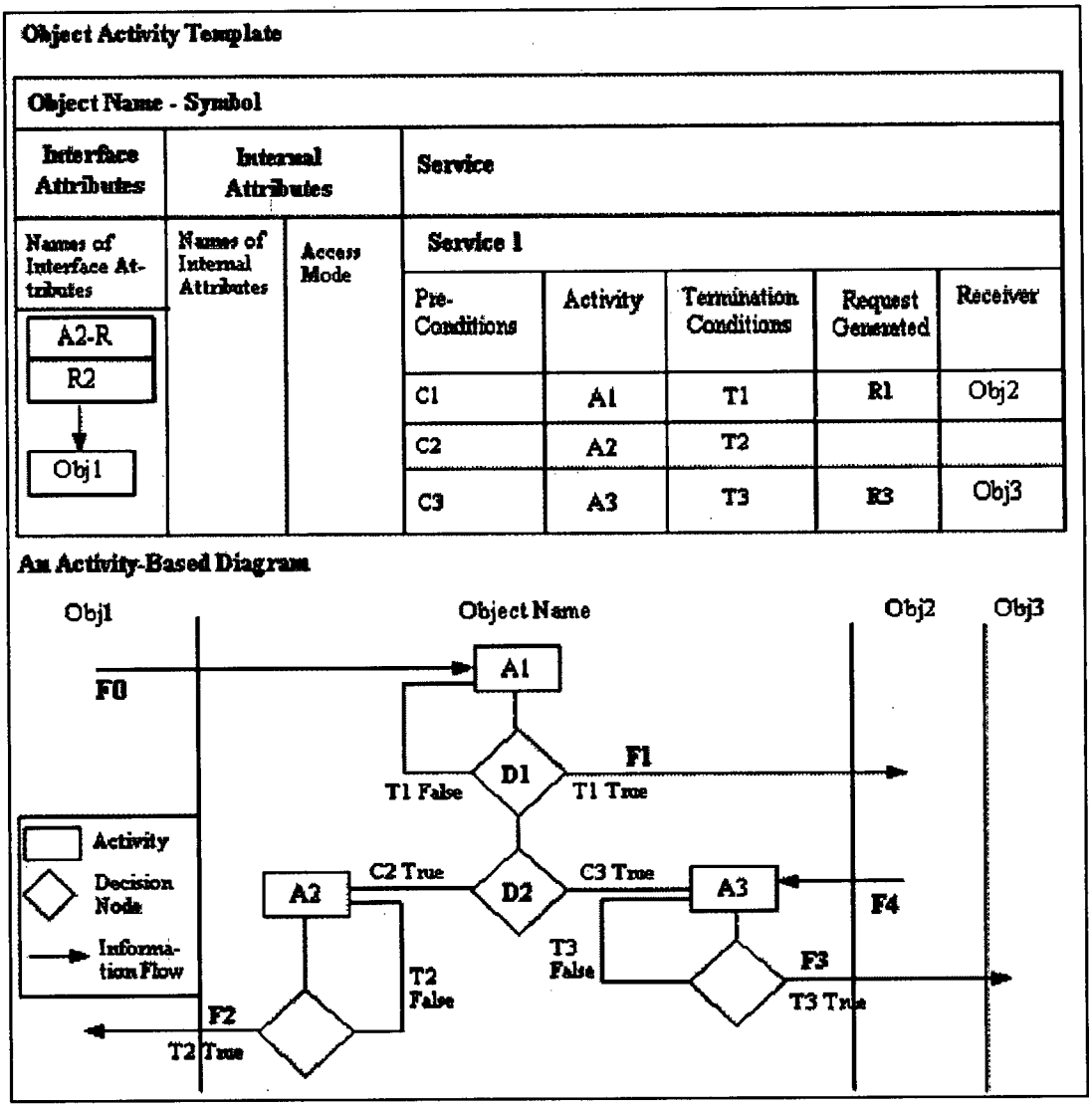


Figure 3-4 An OAT and An Activity-Based Diagram

In an OOWM, a response to a request is required by an object to continue its service. This requirement is usually captured in the pre-condition for an activity performed by the object. Whether such a response is received and triggers another activity is represented by a decision node. Assume that C3 specifies the need for the response to R1 from object 2 in order to trigger activity A3. If C3 is satisfied (i.e. C3 is evaluated to be true at the decision node D2), A3 will be triggered along with incoming

information represented by F4 in the activity diagram. The flows of activities blocks can be determined by pre- and termination conditions in the OAT. T1, for example, is represented by a diamond-shape symbol, D1, in Figure 3-4. If T1 is false, then A1 will continues until T1 is evaluated to be true. An activity following A1 is decided by whether C2 or C3 is true (i.e. the decision node is marked D2). By converting all the OATs in an OOWM into activity diagrams, a complete activity diagram to describe an organizational process can be developed.

3.3. The OOWM Method

The previous section presented the Object-Oriented Workflow Model (OOWM) which reflects our view of an organizational process in an object-oriented context. We also examined the theoretical foundations of the model. In this section, we will introduce the OOWM method to describe a given organizational process in our OOWM framework. The steps to building an OOWM for the process are similar to those of the OOEM method, except that we need to consider additional workflow constructs presented in the previous sections.

The quality of an OOWM is related to analysts' ability to apply the method to capture an organizational process under study. The analysts should have sufficient information to determine such a process. The sufficiency of information also affects the quality of the OOWM.

3.3.1. Steps to Building an OOWM for an Organizational Process

The procedure of creating an OOWM can be divided into two main steps. First, analysts should construct an OOEM model for an process under study. Second,

the analysts should look into the internal objects of the process and model their characteristics by using the Object Activity Template (OAT) presented in the previous section.

3.3.1.1.Constructing an OOEM Model

Zhao [1995] proposes an algorithm to identify objects, their services, interface and internal attributes, and requests of a process under study. The algorithm which is summarized in the following steps (See Table 3-3) provides an effective guideline for applying Wand and Woo's modeling rules [1993] presented in Appendix E. Please refer to Zhao's [1995] *Object-Oriented Enterprise Modeling* for the details of the algorithm.

Steps to constructing an OOEM Model	**Corresponding Wand and Woo's Modeling Rules**
1. Determine the scope of the process.	Rule #1: The scope identification rule is applied since all external objects and their requests being submitted to the process are identified.
2. Identify external clients of the process.	
3. Identify the requests generated by the external clients.	
4. Trace an individual external request and determine how the request is processed by other internal objects which may propagate other requests in the process. During the tracing process, identify internal objects, their interface and internal attributes, and their services.	Rules #2 - #5: The rules for identifying objects, services, internal and interface attributes, and the ownership of the attributes are satisfied since this step ensures that each object in the model provide at least one service which requires interface attributes and which may access internal attributes.

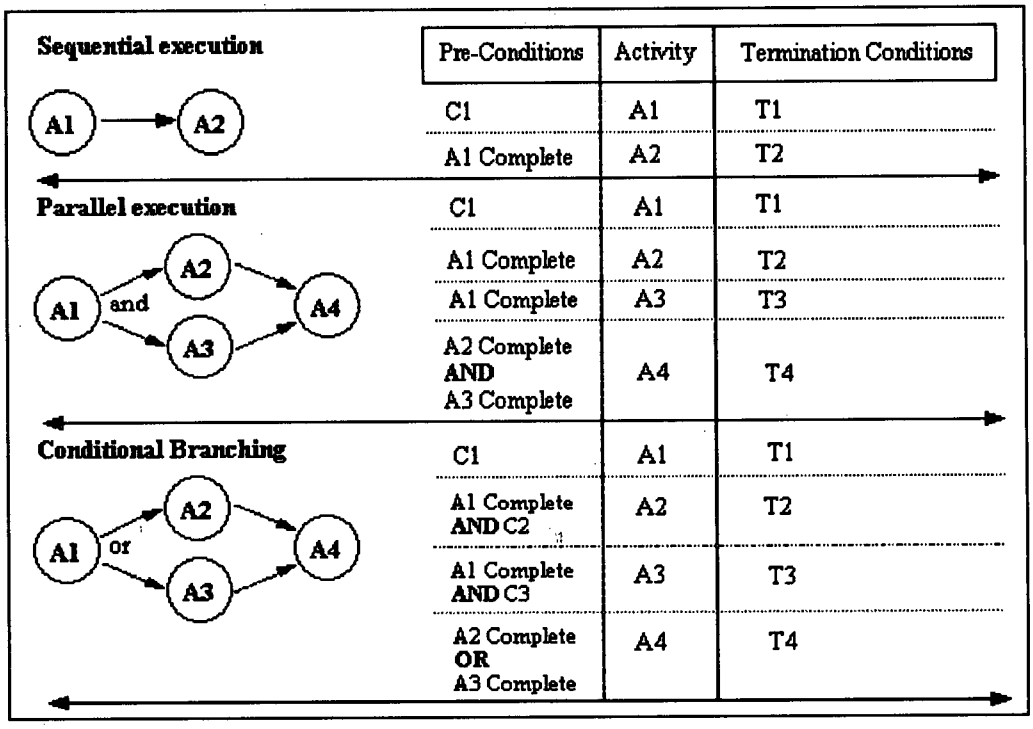
Table 3-3 Steps to Constructing an OOEM Model

Once all objects are identified, they can be organized into whole-part and generalization-specialization structures by applying Rules #6 and 7.

3.3.1.2.Creating an OAT for an Internal Object

This step requires information about company policies which stipulate how a particular process should be carried out by different workflow participants. Services identified in the previous step should be broken down into activities which are associated with pre- and termination conditions. What constitutes an activity was discussed in Section 3.2.1.1.

Appendix G shows the complete OOWM for the purchase reimbursement process. Three objects, their interactions, their interface and internal attributes, and their services are first identified in the OOEM model. Then, we take a microscopic view of the division manager and corporate accountant objects and include their internal characteristics in the OATs. The execution order of activities within an object can be specified by the use of pre- and termination conditions. As an example, we demonstrate how to model three common types of the order in Figure 3-5 [WfMC 1997, Grasso, Meunier, Pagani, and Pareschi, 1997].



Pre-Conditions	Activity	Termination Conditions
C1	A1	T1
A1 Complete	A2	T2
C1	A1	T1
A1 Complete	A2	T2
A1 Complete	A3	T3
A2 Complete **AND** A3 Complete	A4	T4
C1	A1	T1
A1 Complete **AND** C2	A2	T2
A1 Complete **AND** C3	A3	T3
A2 Complete **OR** A3 Complete	A4	T4

Figure 3-5 Types of Execution Order in OAT

Different types of execution order are separated by double-arrow-head solid lines. Sequential execution is captured by specifying the completion of the first activity as criteria for starting the second activity in OAT. Parallel execution is represented by two activities, namely A2 and A3, associated with the same pre-condition. The completion of A2 and A3 becomes a prerequisite to starting the convergent activity A4. Finally, an alternative activity can be determined based on the completion of A1 and its pre-condition in conditional branching.

3.4. An Implementation Model of an Object-Oriented Workflow Management System

In previous sections, we introduced the Object-Oriented Workflow Model (OOWM) methodology to study an organizational process from an ontologically object-oriented perspective. In this section, we will propose the implementation model

of an Object-Oriented Workflow Management System (OOWMS) which enacts an OOWM. The model serves as a basis for the architecture of the OOWMS which will be presented in the next chapter. The model identifies the important components of the architecture and specifies the general functions of the components.

Figure 3-6 illustrates the implementation model. The objective of our OOWMS is to automate interactions among objects in accordance with workflow business rules. Our system, however, does not control how objects perform their services because of object autonomy and independence. Specifically, our system controls and monitors *when* and *what* objects should react to requests. To achieve such an objective, we introduce the *Controller object* which monitors all controlled requests from and to the objects in a process. The specifications of the Controller object are presented in the following sections.

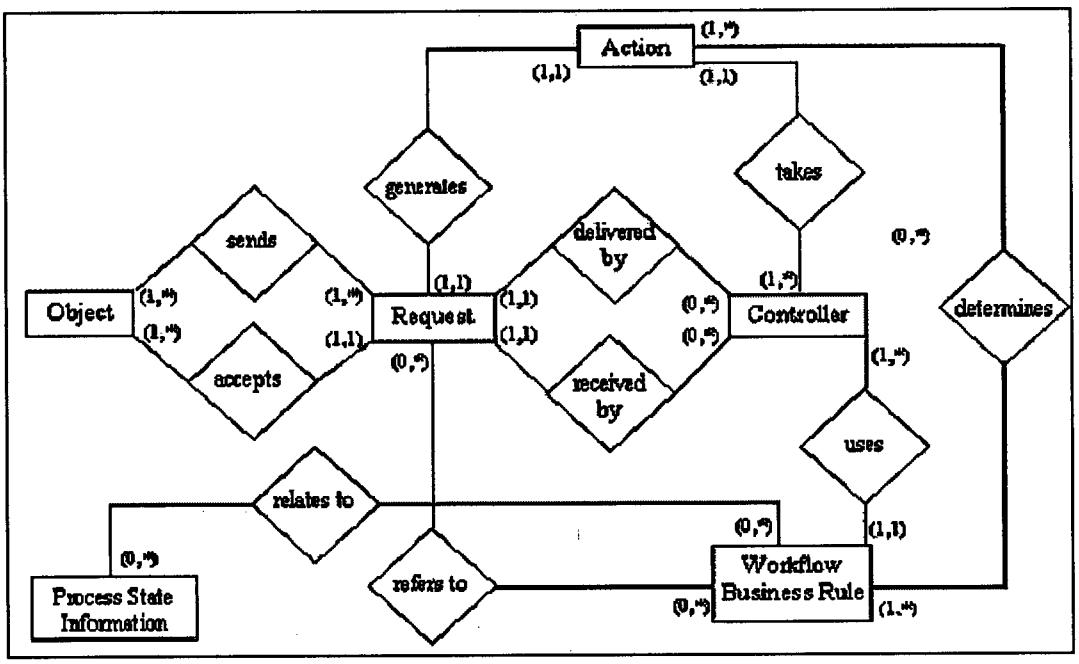


Figure 3-6 The Implementation Model of an OOWMS

3.5. Controller

Figure 3-6 depicts the general function of the Controller object. The object is introduced to ensure that business rules (i.e. organization policies) be followed in an automated workflow environment. It not only keeps track of the rules but it also evaluates the rules based on the information carried by a request and the state of a process. It generally takes two types of actions:

1. It can issue requests to other objects to obtain additional information for evaluating business rules.
2. It performs actions specified by the business rules after evaluating the rules.

The Controller monitors or controls the flow of requests based on business logic and the state of requests to ensure that interactions among objects satisfy organizational process. All external requests are sent to the Controller object, and the Controller takes care of all responses to external requests. If business rules are violated, the Controller may follow instructions specified in violation clauses. This procedure is equivalent to error handling in computer programming; the actions are defined to handle specific errors.

Not every request nor interaction in a process needs to be monitored nor controlled by the Controller object. To illustrate our point, let us return to the purchase reimbursement process.

Before the division manager approves a reimbursement form, he/she may consult the division accountant regarding the cash situation.

Figure 3-7 shows the OOEM for the extended reimbursement process, and all requests

flows are labeled. Since the division manager object may have its own policies and resource constraints, all requests that the manager sends are related to those internal policies and constraints (e.g., request R5 in the figure) are not monitored nor controlled by the Controller.

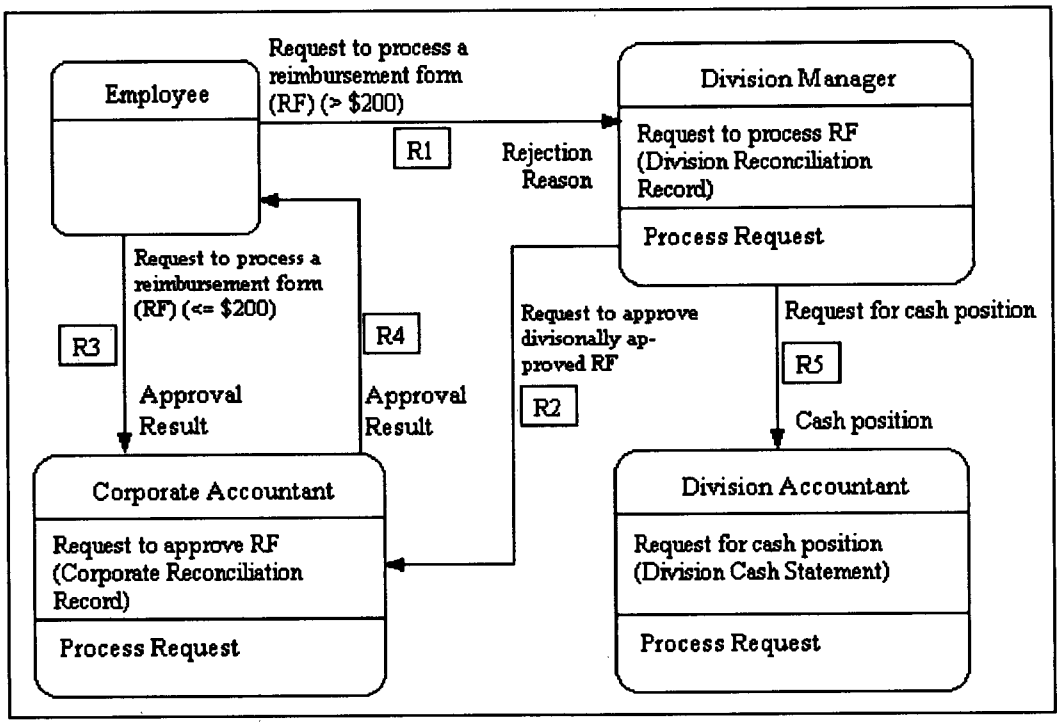


Figure 3-7 The OOEM for The Extended Reimbursement Process

3.5.1. Control Schema

A *control schema* specifies which requests are controlled by the Controller object, what workflow business rules are applied to the requests, and what actions should be taken by the Controller object if the rules are evaluated to be true. The organization decides what requests should be controlled or monitored and how these requests should be controlled.

As an example, consider the requests in the reimbursement process. The

workflow business rules can be obtained from the pre- and termination conditions as well as the Request Generated and the Receiver columns of the Object Activity Template (OAT) of each internal object. The result of this is the control schema presented in Table 3-4. Using the information in the table, a Controller object can then be formed, as is shown in Figure 3-8.

Request	Workflow Business Rule(s)	Action(s)
R1	Amounts > $200	Send a form to the Division Manager
R1' (immediate response to R1)	Amounts < $200 **AND** the form is rejected by the Division Manager	Inform the requester of the rejection reason.
R2	The form is approved by the Division Manager	Send the form to the Corporate Accountant
R3	Amounts <= $200	Send the form to the Corporate Accountant
R3' (immediate response to R3)	The form is approved **OR** rejected by the Corporate Accountant	Inform the requester of the approval result.
R4	The form is approved **OR** rejected by the Corporate Accountant	Inform the requester of the approval result.

Table 3-4 The Control Schema for the Purchase Reimbursement Process

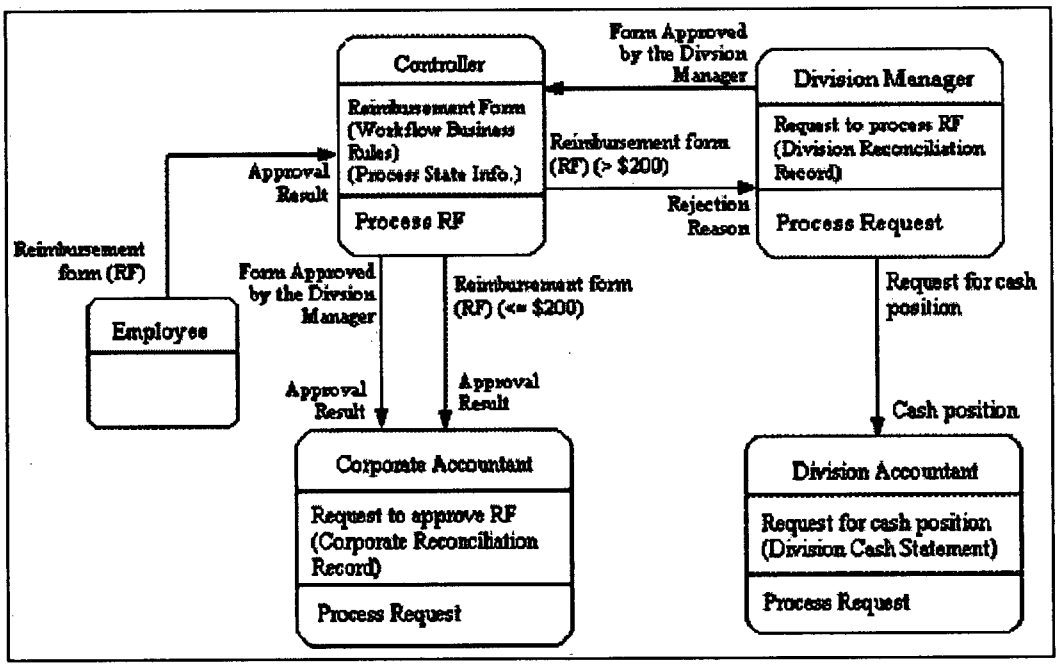


Figure 3-8 The Controller in the Purchase Reimbursement Process

As shown in Figure 3-8, the Controller does not control how the manager and the corporate accountant actually approve the form (i.e., the manager communicates directly with the division accountant). The figure also shows that the workflow business rules and the process state information are represented as internal attributes of the Controller object. Finally, it should be noted that the interaction between the employee object and the Controller object should be transparent in practice because when the employee object submits a request to the object with which he/she wants to communicate, the employee object does not know that his/her request first passes through the Controller object for evaluation. Similarly, the internal objects do not know the redirection of requests by the Controller object.

To evaluate the business rules, the Controller object requires not only information carried by a request but also the state information about the current process instance. However, the Controller may require information about other processes. In this case, the Controller needs to communicate with other Controllers in other processes to obtain such information. The details of accessing information about other processes will be discussed in the next section.

3.5.2. Access to Workflow Data in Other Processes

The Controller object must access the workflow relevant data to evaluate conditions throughout a process and to determine how to route, process, and otherwise handle a work item [Kobielus, 1997]. Even though access to the data is usually confined to the current process instance which, according to WfMC, "represents a separate thread of execution of the process" [1997, p.391], it is

sometimes possible for the Controller to require information about other process instances for evaluating conditions. For instance, process A cannot continue until process B is completed. In this case, the Controller in process A can request from the Controller in process B the state information about process B. We allow communication between Controller objects in different process instances. The way in which these Controller objects interact is similar to the way the internal objects interact. A request sent by one Controller modifies the interface state variables of a receiving Controller object; however, it is the receiving object which decides to invoke a service and to act upon the request. The communication between Controller objects also triggers an important question: If the Controller object determines a target object, how can the Controller object locate the target object?

Business objects always abide by business logic when they communicate with each other. Based on this logic, the business objects always know what and whom they should approach to solve their problems. For instance, if a division manager needs to know the cash position of his/her division before he/she can approve a reimbursement form, he/she will contact a division accountant for information because he/she knows that the accountant is responsible for keeping track of the financial health of his/her division. The same principle can be applied to the Controller objects because these objects are business objects, and they logically represent owners of organizational processes. For example, if the Controller object instance A which monitors an instance of a purchase reimbursement process requires information about the approval status of the budget for the sales department, how can the object instance

know which Controller object instance of a budget approval process it should contact since there may be many ongoing budget approval process instances? Instance A may first identify Controller instances in all active budget approval process instances and then locate the target instance based on the budget type, the submitted date, the submitted person, and so on. The search for the target instance can be achieved by referencing a directory maintained in a central repository or by querying each individual Controller object about all active budget approval process instances.

3.5.3. Time Control

Speed is an important concern for most business processes [Kobielus, 1997]. Thus, time control plays an important role in assuring the efficiency of an organizational process. For instance, a deadline is a time-based scheduling constraint which requires that a certain activity (or work item) be completed by a certain time [WfMC, 1997]. The Controller object, as we mentioned earlier, is introduced to enforce business rules. These rules may include scheduling conditions which describe the maximum and minimum time allotted for each activity, including in-queue time, process time, and out-queue time [Kobielus, 1997]. Conceptually speaking, the Controller object does not have an internal clock to keep track of time. Accordingly, it must obtain the information about time in order to evaluate the scheduling conditions. A clock object is proposed to provide the Controller object with the information. The clock object functions like an alarm clock. The Controller not only retrieves time information from the clock object, but it can also request the clock object to notify it about a specified time occurrence.

We expand the example of the purchase reimbursement process to illustrate what role the clock object can play in our implementation model.

After the division manager receives a reimbursement request from an employee, he/she has to approve the request within five calendar days; otherwise, the request will be assumed to have been rejected.

First, the control schema for the reimbursement process needs to be revised to reflect the time control over the process.

Request	Workflow Business Rule(s)	Action(s)
R1	Amounts > $200	Send a form to the Division Manager
R1' (immediate response to R1)	(Amounts < $200 **AND** the form is disapproved by the Division Manager) Or the current date > the submitted date + 5 calendar days	Inform the requester of the rejection reason.
R2	The form is approved by the Division Manager AND the current date <= the submitted date + 5 calendar days	Send the form to the Corporate Accountant
R3	Amounts <= $200	Send the form to the Corporate Accountant
R3' (immediate response to R3)	The form is approved **OR** rejected by the Corporate Accountant	Inform the requester of the approval result.
R4	The form is approved or rejected by the Corporate Accountant	Inform the requester of the approval result.

Table 3-5 The Revised Control Schema to Include the Approval Deadline

Please note that request R2 has to satisfy an additional condition which ensures that the request sent to the corporate accountant be approved by the division manager within five calendar days after the manager receives the request. To enforce such a condition, we include the clock object in the OOEM for the process (Figure 3-9).

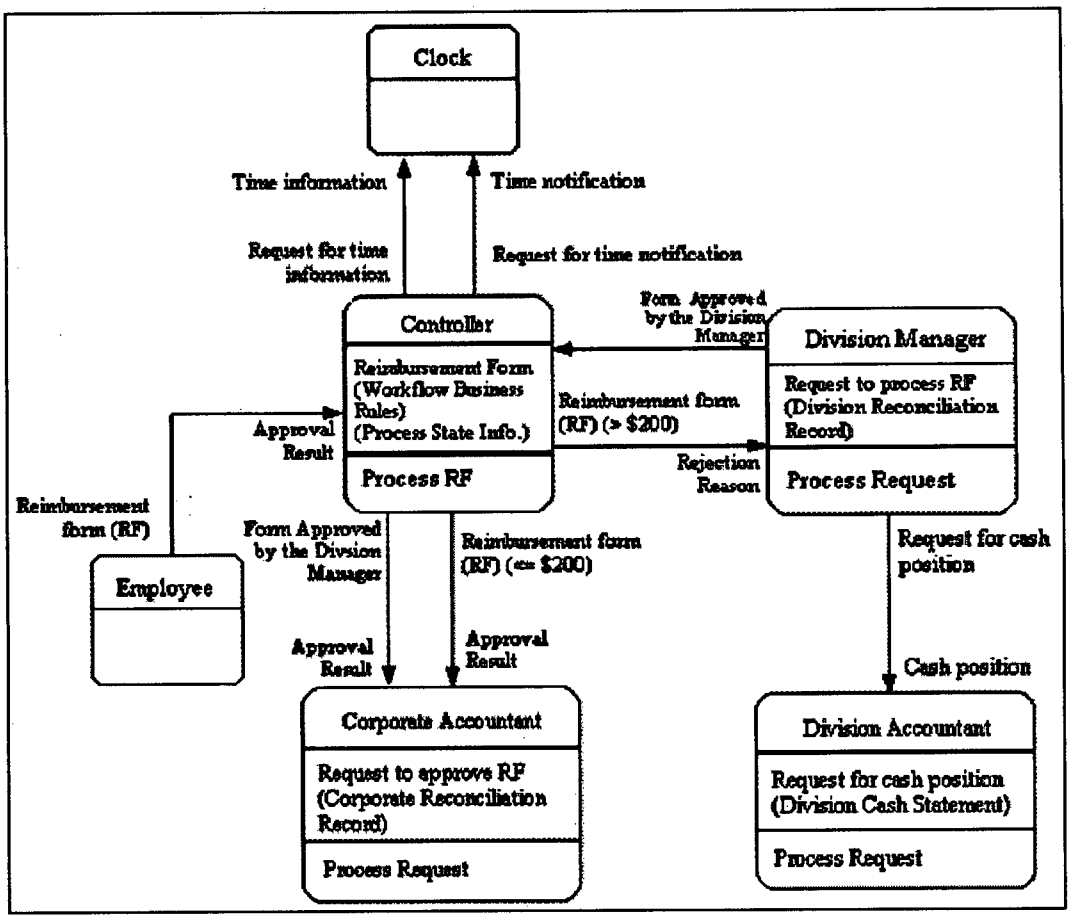


Figure 3-9 The Clock Object in the Purchase Reimbursement Process

When an employee submits a reimbursement form on September 12, 1997, the Controller object obtains time information from the clock object to time-stamp the form. If the form is routed to the division manager, the Controller object calculates the deadline, September 16, 1997, and requests the clock to remind it of the September 16, 1997. If the Controller object does not receive any approved form from the division manager by September 16, 1997, and it is reminded by the clock of the deadline, the Controller will reject the form and notify the requester.

3.6. Summary

In this chapter, we introduced the Object-Oriented Workflow Model (OOWM)

which extends Object-Oriented Enterprise Modeling (OOEM) by incorporating the concepts of workflow management into OOEM. We also presented the OOWM method so that analysts have guidelines for constructing the OOWM for an organizational process. In order to capture the internal characteristics of objects, we proposed the Object Activity Template (OAT) which enables Zhao's Internal Object Template (IOT) to describe the internal behavior of the objects in terms of activities which are governed by business rules. Finally, we presented the implementation model of an Object-Oriented Workflow Management System (OOWMS). The model identifies the major components of our OOWMS and provides the first step to developing the architecture of the OOWMS presented in the next chapter.

4. An Implementation Architecture of the OOWMS

This chapter presents the implementation architecture of the Object-Oriented Workflow Management System (OOWMS) which enacts the contents of the model presented in the previous chapter. In our model, the Controller object is introduced to monitor and control the flow of requests, and the services provided by the internal objects are encapsulated in the objects themselves. The Controller object is responsible for enforcing business rules which govern how internal objects should interact with each other. We therefore focus on the architectural blueprint for the Controller object. Specifically, we need to address the following questions:

1. What is the algorithm used for processing incoming requests by the Controller?
2. How is the information required by the Controller logically represented in the architecture?
3. What are the logical components of the Controller to implement the algorithm in (1)?

4.1. Request Processing Cycle

Before we answer the second and third questions, we must understand the algorithm used for processing the incoming requests by the Controller. The process of handling the requests is achieved by a *request processing cycle* which is similar to a machine cycle performed by a control unit and an arithmetic-logic unit (ALU) in a central processing unit (CPU). The control unit fetches an instruction from the program stored in primary storage, decodes the instruction, places it in a special instruction register, and directs the arithmetic-logic unit (ALU) to perform the

required tasks [Mano, 1993].

The purchase reimbursement process presented in the previous chapter will be used to facilitate our discussion of the request processing cycle. Since we revised the reimbursement process example in different places in Chapter 3, we will restate it to avoid any confusion:

In order to have his/her expense reimbursed, an employee of the ABC Company must submit a reimbursement form to the division manager or the corporate accountant for approval. Reimbursement amounts greater than $200 require a division manager's approval before they are approved by the corporate accountant. After the division manager receives a reimbursement request from an employee, he/she has to approve the request within five calendar days; otherwise, the request will be assumed to have been rejected. The division manager may consult the division accountant regarding cash situation when he/she approves the request. All other reimbursements are submitted directly to the corporate accountant. After his/her approval, the division manager submits the reimbursement form to the corporate accountant who then cuts the cheques and completes the process.

The OOEM model with the Controller object is presented in Figure 4-1.

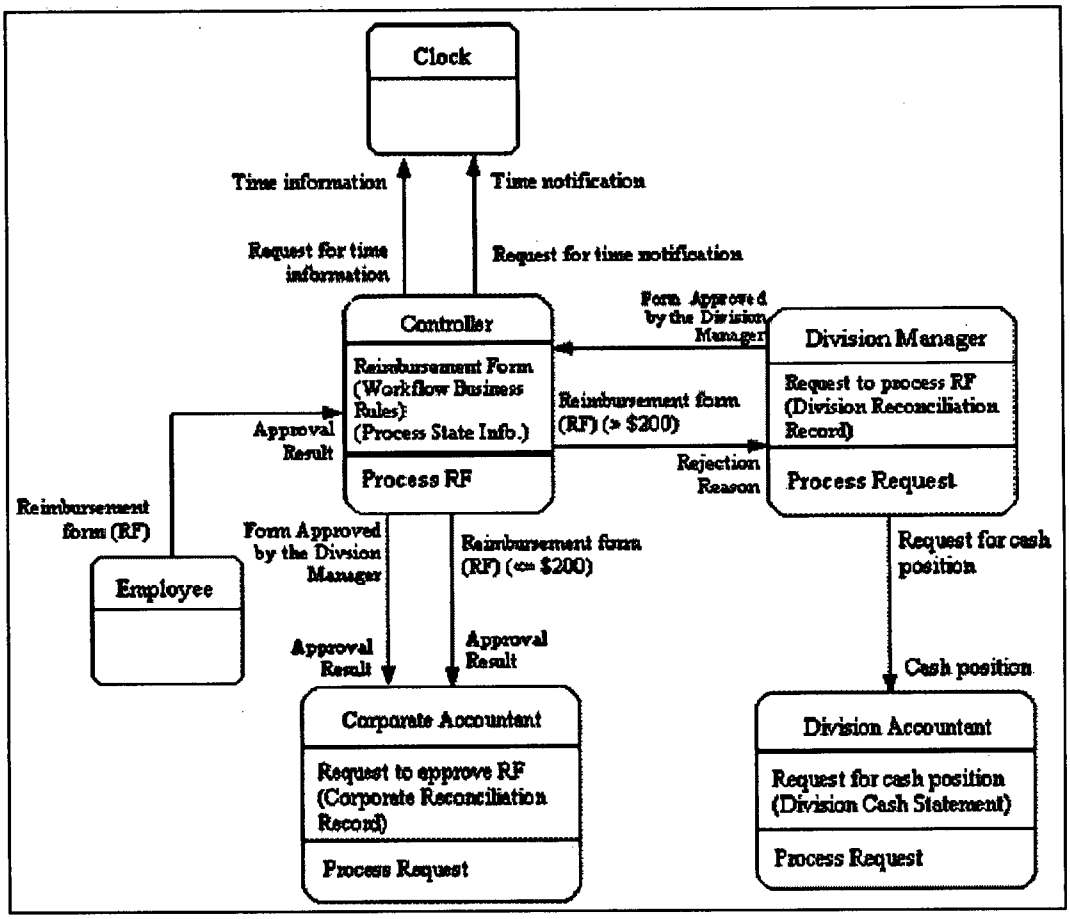


Figure 4-1 The OOEM Model with the Controller for the Reimbursement Process

In the Controller object, the business rules attribute contains the knowledge of how an organization wishes to control a work process, and the process state attribute captures the information about the state and the state transition of a process. According to the Workflow Management Coalition (WfMC) [1996], a process state is a "representation of the internal conditions, defining the status of a process instance at a particular point in time" [p. 411], and a state transition reflects a "change in the status of the workflow" [p. 413]. We will examine how this information can be logically represented in our architecture.

To process a request, the Controller fetches the first request in the request

list, decodes it, and retrieves information carried by the requests. For instance, the Controller in the reimbursement process receives a request from an employee (See Figure 4-1). It needs to decode the request and decide what type of the request it will process. In this case, the Controller should identify it as a reimbursement request. Then, the Controller retrieves information carried by the request. This information may include the name of a sender, the value of the reimbursement form, and so on. The Controller must first update the process state attribute by recording when and by whom the request is sent to the Controller. It then evaluates business rules according to the values of the request and the state of the process to determine what actions it should take. For instance, in the requisition process, the Controller has to determine the receiver of a PR form based on a requested item. Once it sends out the form, it updates the state of the process. The following list summarizes the request processing cycle:

1. Fetch the next request
2. Decode the request
3. Retrieve the information accompanying the request
4. Update a process state for the incoming request
5. Invoke and evaluate business rules
6. Determine actions based on the evaluation of the business rule
7. Take actions, including sending request for future time events (as mentioned in Chapter 3).
8. Update a process state for the outgoing request(s).

4.1.1. Request Instance Identification

We allow the Controller to be able to process more than one incoming request instance of a same request type. When the Controller receives a response from an operating object, it has to identify the original request instance to which the response addresses. Handling multiple instances becomes challenging when dealing with autonomous objects, because the Controller has no way of enforcing what an object can respond to in a request. An example may help to explain this point. An employee submits three reimbursement forms to the division manager at three different time points; an organizational policy specifies only *when* the manager should approve the form, but it does not specify *how*. The manager may send a message to the requester such as "Approve All", "Approve the first request and Reject the last two", or "Reject the first two request and Approve the last one". In this case, there is no indication of which requests the response addresses. The Controller, on the other hand, will be able to keep track of the origin of a request if one of the following conditions holds:

1. The responses include sufficient information to identify the original requests they are responding to.
2. The responses are sent back in the same sequence as the original requests were sent to the Controller.

These conditions reasonably reflect the way in which people work in reality. The first condition implies that a request can be uniquely identified by a set of information. There are many real-life examples to support the first condition. For instance, a monthly bank statement on a chequing account displays a list of cheques

issued by a client for reconciliation. The cheques are referred to by their numbers. If the statement simply printed out the total credit and debit amounts, then the client could not trace his/her spending. The second condition reflects the temporal sequence of processing a request. An example may illustrate our point. When we line up at a ticket booth for concert tickets, a ticket booth attendant basically processes individual requests one by one. We believe that these conditions do not restrict how people should work in an organizational process but that they are essential for the process to run efficiently and effectively.

4.2. *Information Representation*

The request processing cycle suggests answers to question (2) presented in the beginning of this chapter: How is the information required by the Controller logically represented in our architecture? The information required for processing the requests includes: request type definition, business rules, state information about a process, and detailed information about the requests.

4.2.1. Request Type Definition

From the Controller's perspective, there is no difference between a request and a response. The Controller is only concerned about what kind of data it is passing around in a process. For instance, in Figure 4-2, the Controller object treats the approval status sent from either the division manager object or the computer manager object as an instance of a request type, even though it appears as an immediate response to a request for purchase requisition. The Controller considers each incoming and outgoing request to be unique types of information. Each request type

should contain information about a sender and a receiver of a request:

Request-Type-Definition (request-type-id, sender, receiver)

4.2.2. Business Rules

Business rules specify the criteria for certain actions to be taken by the Controller. Business rules can be represented in the following conditional statement:

IF <*conditions*> THEN <*actions*>

The IF clause specifies the conditions under which the Controller should take specific actions stipulated in the THEN clause. The conditions refer to the workflow business rules specified in the control schema introduced in Chapter 3. For instance, according to the control schema for the requisition process (See Appendix H), the computer manager only approves a request for computer items. This condition can be encoded into the following clause:

IF <requested_item = computer>

We allow all operations associated with a business rule to be represented as a block of execution in the THEN clause. For example, we can instruct the Controller not only to decide the recipient of a PR form but also to calculate the deadline:

```
IF <request_item = computer>
THEN
{
        Send (Request_To_Computer_Manager);
        Calculate_Deadline();
}
```

The business rule information can be organized and represented in a database table format:

Business-Rules(rule-id, IF_clause, Then_clause)

where rule-id is the primary key of the table. In principle, the Controller scans all the business rules in the repository to determine which rules will be "fired".

4.2.3. State Information about a Process and Information about Requests

The Controller not only keeps track of all the states of a process in the process state attribute but also of the history of information about the values of requests. The history of information serves two purposes in our architecture:

1. It provides the Controller with references to determine its course of actions specified by the business rules.
2. It builds up an audit trail of workflow execution [Jablonski & Bussler, 1996].

An example can illustrate the need for the history of information. In the purchase requisition process, all managers must approve PR forms within three calendar days; otherwise, the forms will be automatically rejected. The Controller is responsible for keeping track of when the PR forms were first sent to, for example, the division manager. The Controller may refer to the time when the forms were sent and decide if the manager has passed the deadline. If the manager has passed the deadline, the Controller can reject the forms and notify the requesters on the manager's behalf. Since the manager object is autonomous, the Controller cannot stop the manager from submitting the approved forms to it. In this case, because all state information is in long-term storage, the Controller is able to once again to refer to the time when the

original PR forms were submitted to the manager, and, according to a business rule, it may simply inform the manager that he/she has passed the deadline and that the forms have been rejected on his/her behalf.

The Controller stores the state information in the following manner:

Workflow-State(request-id, sent-time, request-reference-id, sender, receiver)

The sent-time field stores the data about when a request is sent to the Controller and out of the Controller. The request id field is used to uniquely identify individual request instances. For instance, the reimbursement request submitted by employee A can be distinguished from employee B's by the reimbursement ids. The request reference id field refers to the id of the original request so that we will be able to know what other requests are generated as a result of the original request. Finally, the sender and receiver fields record the sender and the recipient of the request. The state information is stored in the *Workflow State Repository* represented in Figure 4-2.

Apart from the state information about a process, the Controller should also maintain a track record for the values of the requests throughout a process:

Request-Information(request-id, parameter, request-type-id, value)

where the parameter field records the names of all the variables carried by a request instance and the value field stores the values of the variables. Since each request is assigned to a unique id, we can trace back how a request instance was processed throughout a work process. Also, the request type to which the instance belongs can be identified by the request-type-id field. The request information is maintained in the *Request Information Repository* (See Figure 4-2).

The specifications of the Workflow State and Request Information Repositories are domain-independent; that is, they can be generally used for different organizational processes.

4.3. *Logical Components of the Controller Object*

In previous sections, we identified the information required by the Controller in the request processing cycle. This information can be stored and represented by using database technology. But we have not identified the logical components of the Controller to process this information. In this section, we will introduce two major processing units which fetch and evaluate data which reside in different repositories, and perform actions based on business logic. These two units are the *Business Rule Evaluator* and the *Workflow Executor*. The reason that we separate the evaluation of business rules from the execution of workflow operations is that while the structure of a business rule is defined independently of business processes, the workflow operations executed by the Controller vary from process to process. We will be able to customize a unit without changing another. The design of the processing units is analogous to the design of a silicon chip, which allows "a supplier to deliver tightly encapsulated unit of functionality to be specialized for its intended function, yet independent of any particular application" [Sprague & McNurlin, 1993, p. 280].

4.3.1. Business Rule Evaluator

The business rule evaluator functions like the control unit in the CPU does. The control directs the other components of the computer by reading stored program instructions one at a time [Mano, 1993]. Similarly, the business rule evaluator

instructs the Workflow Executor what to do based on the result of its evaluation of the business rules (See Figure 4-2). The evaluator has access to all information previously determined. In terms of the request processing cycle, the evaluator fetches a request, decodes it, determines the type of a request from the type definition, and retrieves its parametric values. The evaluator also evaluates the business rules that correspond to a process state and the parametric values of the request. The design of the business rule evaluator is independent of different process definitions. In other words, the same evaluator can be used for the purchase requisition process as for other organizational processes.

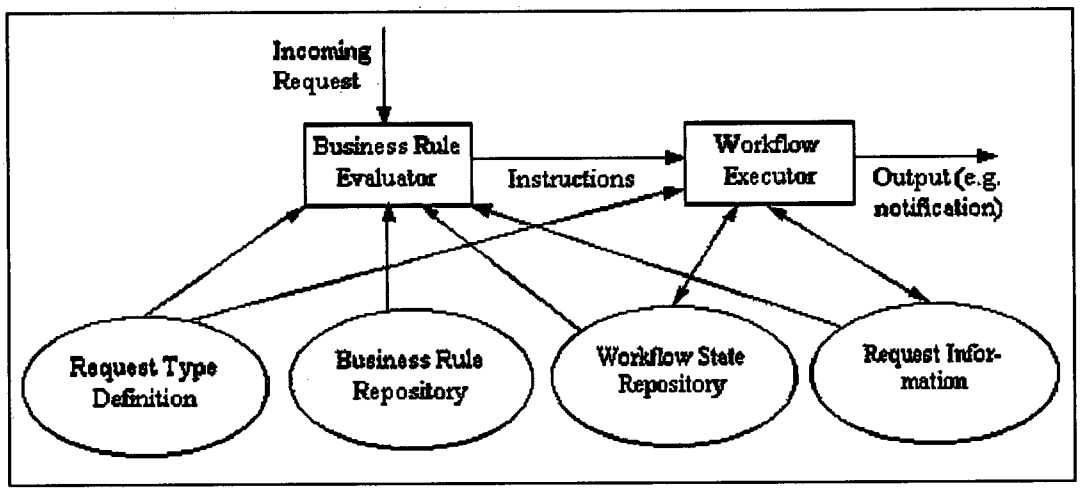


Figure 4-2 The Architecture of the Controller Object

4.3.2. Workflow Executor

The Workflow Executor, as suggested by its name, performs workflow operations specified by the business rules. It can access and write to the request information and the Workflow State Repository; it can also read information stored in the Request Type Definition. A read access to the Business Rule Repository by the Workflow Executor is not necessary because this component only carries out

the instructions sent by the Business Rule Evaluator which determines these instructions based on the business rules. However, the Workflow Executor cannot modify information residing in the Request Type Definition and the Business Rule Repository. If this information was altered at run-time, the execution of other process instance might also be affected. Since the workflow operations may vary from process to process, there is no general design framework specified for the Workflow Executor.

4.4. Another Look at the Architecture

We have examined different components of our architecture in association with the request processing cycle. We would like to demonstrate how these components work together by using the purchase reimbursement process presented at the beginning of this chapter.

Let us assume that an employee A submits a reimbursement form whose value exceeds $200 to the division manager on August 4, 1997. The form passes through the Controller object in which the business rule evaluator determines the type of a request to which the reimbursement form belongs and stores the parametric values of the form in the Request Information Repository. Then, the evaluator examines business rules which correspond to the submitted request. Based on the parametric values of the form and a current process state, the evaluator finds that this is a form whose amount exceeds $200. It instructs the Workflow Executor to generate a request to the division manager for approving the form within five calendar days starting August 4, 1997. The Executor sets a time event which triggers the evaluator to examine the approval status of the form on August 8, 1997. The Executor also

updates the Process State Repository to record when the form was sent to the division manager from the request and the recipient of the form. It also writes to the Request Information Repository the request that it generated for the manager. After the manager has approved the form, his/her approval prompts the evaluator to look up other business rules which determine the next action of the Executor based on the approval status of the form. The evaluator directs the Executor to notify the requester of the approval status if the form is rejected. If the form is approved, the evaluator, according to the business rules, will generate another request to corporate accountant for approving the reimbursement request. Similar tasks will be performed by the Executor following the instructions of the evaluator.

4.5. Summary

The architecture of our workflow management system is quite straightforward. Since all controlled requests must pass through the Controller object which acts on the requests in accordance with business logic, we are only concerned with the architecture of the Controller object. Our approach to developing the architecture of the Controller is to first understand how the Controller processes an incoming request in the request processing cycle. Then, we explore what basic information the Controller needs in the cycle. This information includes business rules, information about incoming request instances, process state information, and request type definition. Finally, we introduce the Business Rule Evaluator and the Workflow Executor in our architecture. These two units play different roles in processing information in the request processing cycle. The Business Rule Evaluator is used for

accepting and retrieving information necessary for evaluating business logic stored in the business rule repository. The Evaluator also instructs the Workflow Executor to perform actions in accordance with the business rules. While different process domains may require different designs and implementations of the Workflow Executor, the design of the Evaluator should remain independent of the process domains.

5. The Implementation of the Object-Oriented Workflow Management System (OOWMS)

5.1. Introduction

The objective of this chapter is to show how the implementation architecture presented in Chapter 4 can be implemented using existing technologies. We will explore topics which include the choice of development platform, the mapping of the architectural components to the facilities of the recommended development tool, and a sample workflow application. During the course of our discussion of the above topics, we will also identify the gap between the architecture and the actual implementation.

5.2. Development Platform

To determine the development platform for the architecture, we must understand what the current technologies can offer. In chapter 2, we briefly looked at some groupware products ranging from Lotus Notes, which implements a proprietary client-server protocol, to Web-based solutions such as Netscape's SuiteSpot which relies on the World Wide Web's open specifications. According to Ginsburg and Duliba [1997], the Web offers a variety of toolkits for application development. Users of Web applications only need the Web browsers, also known as "thin clients", to run the applications. These browsers are freely available on the Internet and support various operating systems such as Microsoft Windows, OS/2, MacOS, and UNIX. However, the Web technology does have weaknesses. For instance, the Hypertext Transfer Protocol (HTTP) which allows users to serve and browse distributed hypermedia documents on the Internet is "inherently stateless" [Ginsburg, et. al.,

1997]. Web servers keep "no memory of the clients' activities in prior sessions" [p. 207]; however, the state of client users is "crucial for security and collaborative work across sessions" [p. 207]. Another weakness is that there is a lack of agreement on security standards for the Web [Ginsburg, et. al., 1997]. Despite its proprietary design philosophy, we have selected Lotus Notes 4.1 to be our development platform because it offers an integrated development environment with a strong built-in security model. Notes provides agent facilities which facilitate the tasks of automating a process. It also offers a messaging system which allows users to communicate with others via electronic mail. To address the need for supporting open Internet standards, Notes moves toward compatibility with the HTTP and mail protocols by introducing Domino, which is a web server that integrates the Notes databases into the Web.

5.3. *Mappings of the Architectural Components to Notes Facilities*

The following table summarizes the mappings of the implementation architecture proposed in Chapter 4 to the Notes development environment.

Architectural Components	**Notes Facilities**
Process Domain	Shared Database
Request Type	Document Class
Request Instance	Document
Controller Object • Business Rule Evaluator • Workflow Executor	Business Controller Object Agent • Eval Module written in LotusScript • Executor Module written in LotusScript
Clock Object	Clock Object Agent
Business Rule Repository	Business Rules Documents
Process State Repository	Workflow State Documents
Request Information Repository	Request Information Documents

Table 5-1 Mappings of the Architecture to Notes Environment

A process domain corresponds to a shared database which contains its own definitions

of document classes and other corresponding parts of the architectural components. A request type resembles a document class which specifies the information requirement of a request; a request instance is equivalent to a document. The following sections explain the details of the other components presented in Table 5-1.

5.3.1. The Controller Object and the Business Controller Object Agent

The Controller object is implemented as a Notes agent, namely the Business Controller Object. Figure 5-1 illustrates the Business Controller Object Agent which can be triggered manually by users.

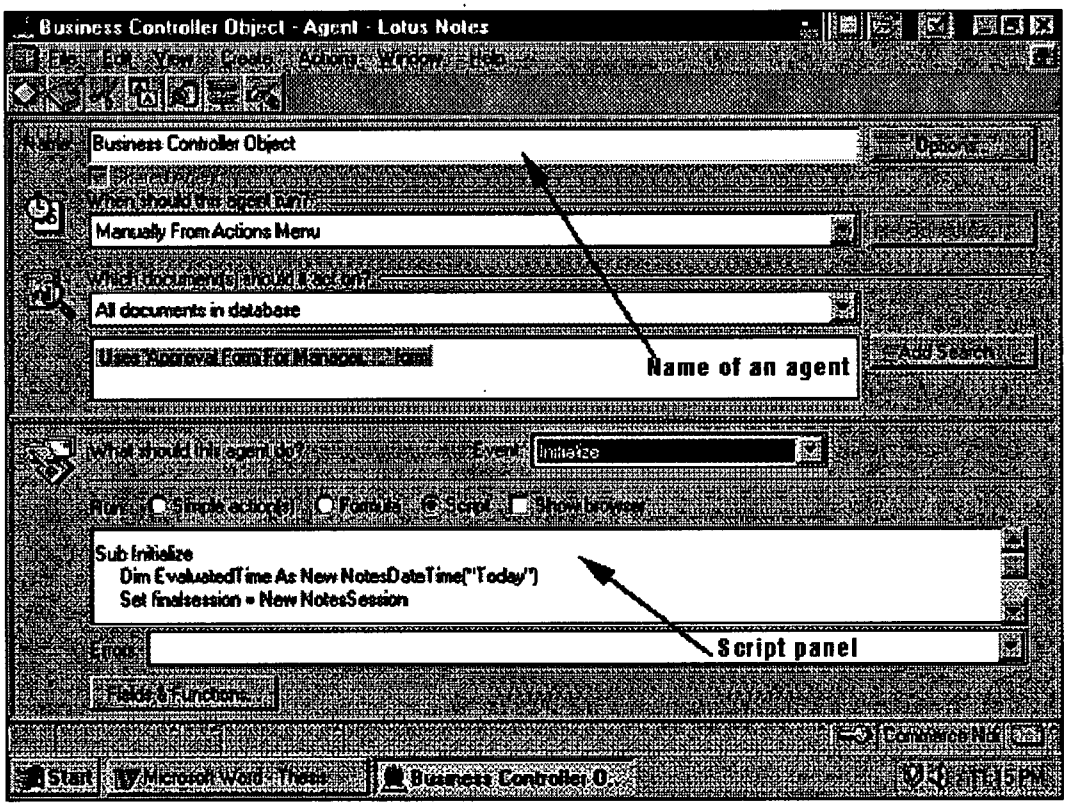


Figure 5-1 The Business Controller Object Agent

The agent is written in LotusScript, a Visual Basic-like scripting language. It contains two main modules: the Eval and Executor modules; the modules were entered in the

script panel. These modules serve the functions of the Business Rule Evaluator and the Workflow Executor presented in Chapter 4.

5.3.1.1.The Business Rule Evaluator and the Eval Module

When the Controller Agent is triggered, its Eval module looks at a submitted document (request) and evaluates all the business rules stored in a Notes database. The module invokes the rules by comparing the parameters of the rules to the field names of the document. If not all the parameters of a rule exist in the document, then the rule will be ignored. This rule selection mechanism requires unique field names for all document classes. The module not only retrieves the values from the document based on the parameters of a rule, but it also accepts complex conditional expressions. The flexibility to evaluate complicated expressions gives our system the potential to be used in automating complex processes.

5.3.1.2.The Workflow Executor and the Executor Module

The Executor Module, like the Workflow Executor, obtains an instruction from the Eval module which passes the THEN clause as a string value to the Executor module if a rule is evaluated to be true. The actions specified in the THEN clause are defined as subroutines in the Business Controller Object Agent at design time. The Executor module parses the passed string value and calls the subroutines that match the names of the actions.

It is important to note that the Eval module is reusable; in other words, it can be used in different process domains. The action subroutines in the Executor module,

however, may vary from process to process.

5.3.2. The Clock Object and the Clock Object Agent

In the architecture, the Clock object accepts requests from the Controller object; however, in the Notes environment, the Clock Object Agent runs itself periodically (See Figure 5-2).

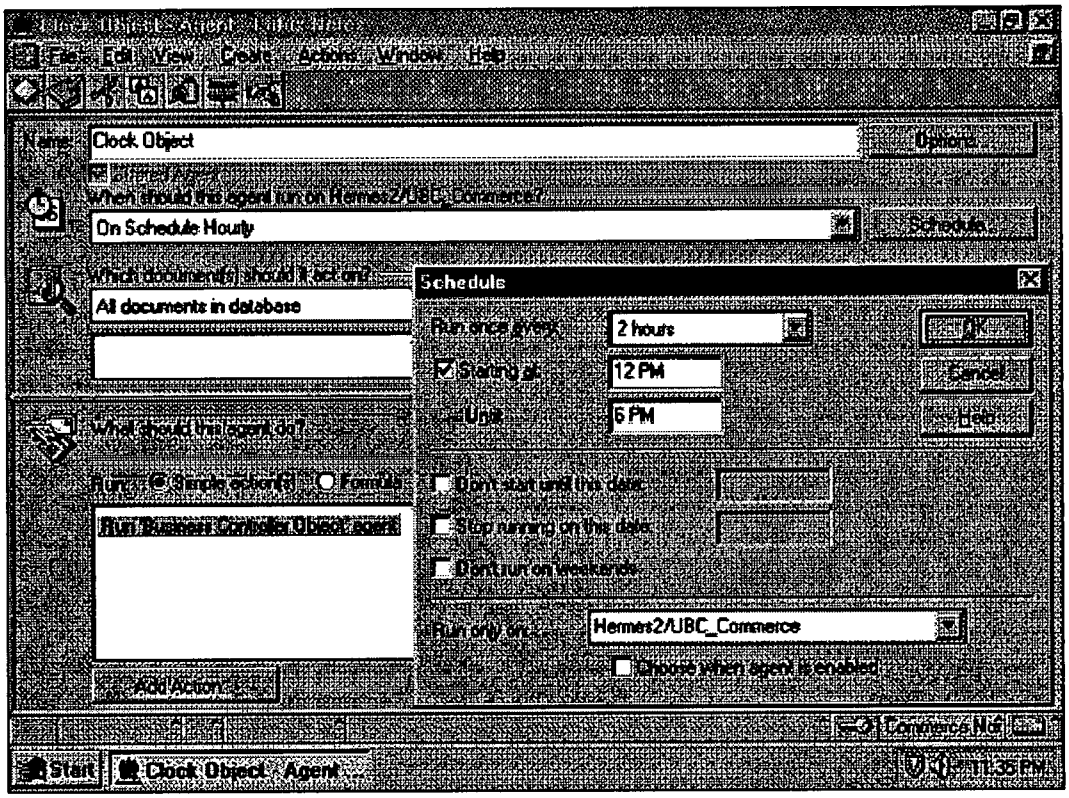


Figure 5-2 The Clock Object Agent

The interval at which the agent is triggered is specified by application developers at design time. Once the agent is triggered, it initiates the Business Controller Object Agent which may take actions depending upon the business rules and the state of a process.

5.3.3. The Business Rule Repository and the Business Rule Documents

A business rule document includes the If and Then fields which correspond to the data definitions of the Business Rule Repository introduced in the previous chapter. The If field accepts any comparison expressions which can be a set of conjunctions, disjunctions, or both. The conjuncts of a conjunction are separated by a keyword "AND" and the disjuncts of a disjunction by "OR" (See Figure 5-3).

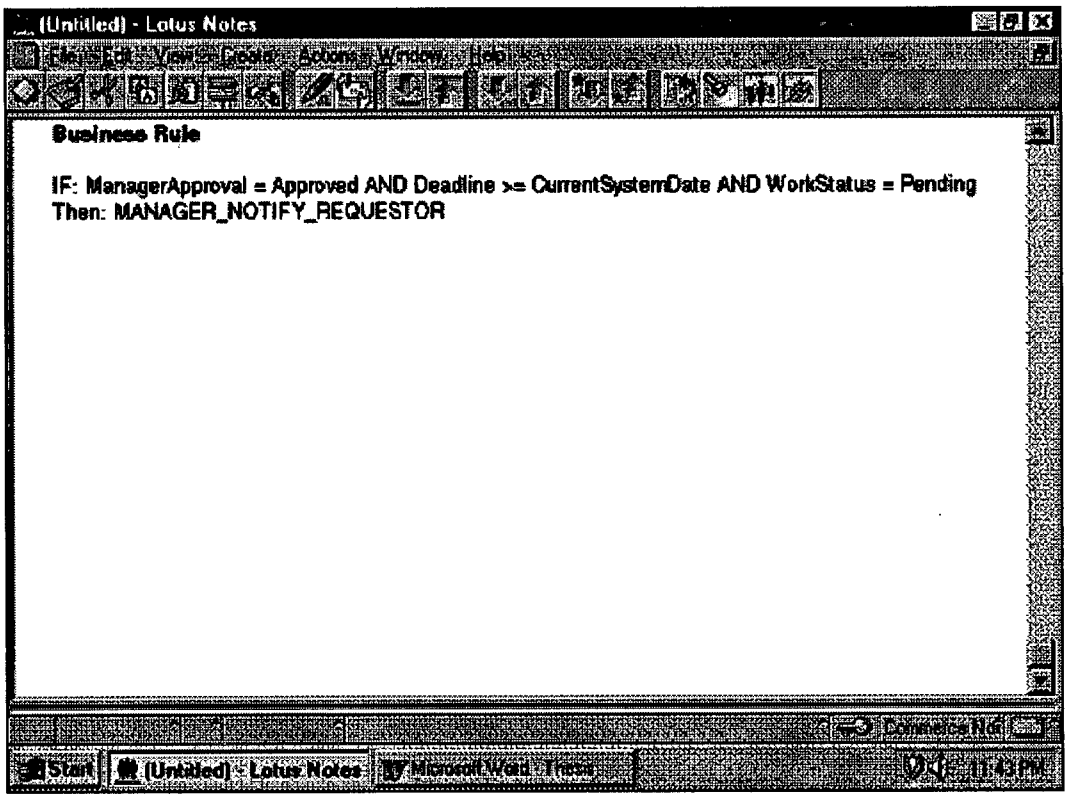


Figure 5-3 If and Then Fields in a Business Rules Document

When a rule is evaluated, it is parsed into substring values by the Eval module. Then, the substring values will be examined if they refer to the field names of the forms; the literal values such as a number, a string constant, or a date; logical comparison operators; or the conjunction or the disjunction keyword. If the substring value refers to the field name of a form, the Eval module will retrieve the value of that field for evaluation. The Then field allows a list of actions which are defined in the Business

Controller Object Agent as subroutines.

5.3.4. The Process State Repository and the Workflow State Documents

Whenever the Business Controller Object Agent acts on a request, a workflow state document is created and filled with information about a process. Figure 5-4 shows the workflow state document which contains the information specified by the Process State Repository in Chapter 4.

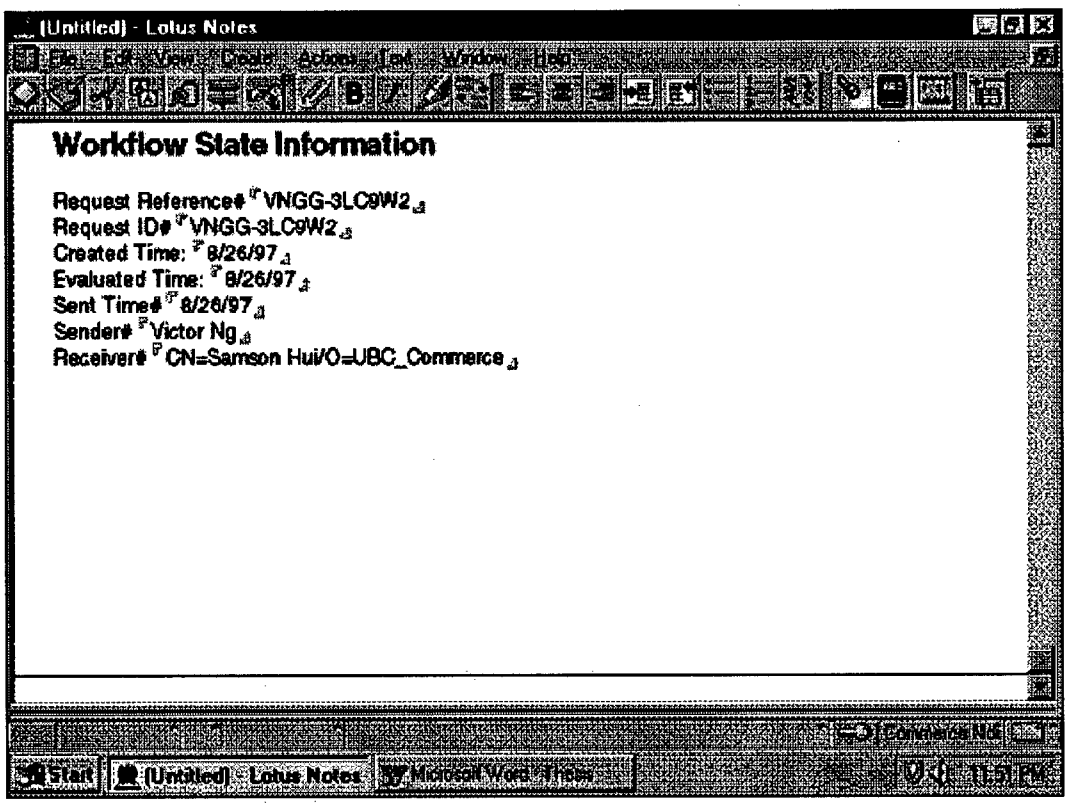


Figure 5-4 A Workflow State Document

The Request Reference field indicates the identification number of the original request submitted by an external object. The Request ID field simply refers to the request the Business Controller Object Agent has acted upon. The Sent Time field stores the information about when a request is sent from one party to another. The

Sender and Receiver fields are self-explanatory. Even though the Created Time and the Evaluated Time field are not specified in the architecture in Chapter 4, they are used here to keep track of when a request is generated and evaluated.

5.3.5. The Request Information Repository and the Request Information Documents

We standardize the interface between the users and the system by using forms, namely Request Information documents. The documents created in these forms correspond to requests in our workflow model. For instance, in the purchase requisition process, the requests sent by the requesters and by the internal objects can be implemented as different Request Information documents. The purchase requisition form can be one class of the Request Information document; the approval status can be another. In order for the users and the system to trace an original request document which triggers a process, an id is assigned to the request document and copied to other request documents as a result of the original request. In the next section, we will discuss how the requisition process can be automated in our system.

5.4. Workflow Application: A Purchase Requisition Process

We implemented our system to apply to a hypothetical purchase requisition process.

In order to purchase an item, an employee must submit a purchase requisition (PR) form to a division manager for approval. If the requested items are computer equipment, the requester must first obtain approval from the computer equipment manager and then the division manager. The person who approves the form must inform the requester of the approval status. All forms must be

approved by the recipients of the forms within three calendar days; otherwise, the forms will be assumed to have been rejected.

The OOEM model for the process, the model with the Controller object, the Object Activity Templates for the internal objects, and the control schema for the process are presented in Appendix H.

Figure 5-5 shows that all Business Rules documents which contain information transferred from the control schema. This information was translated in such a way that it can be interpreted by the Business Controller Object.

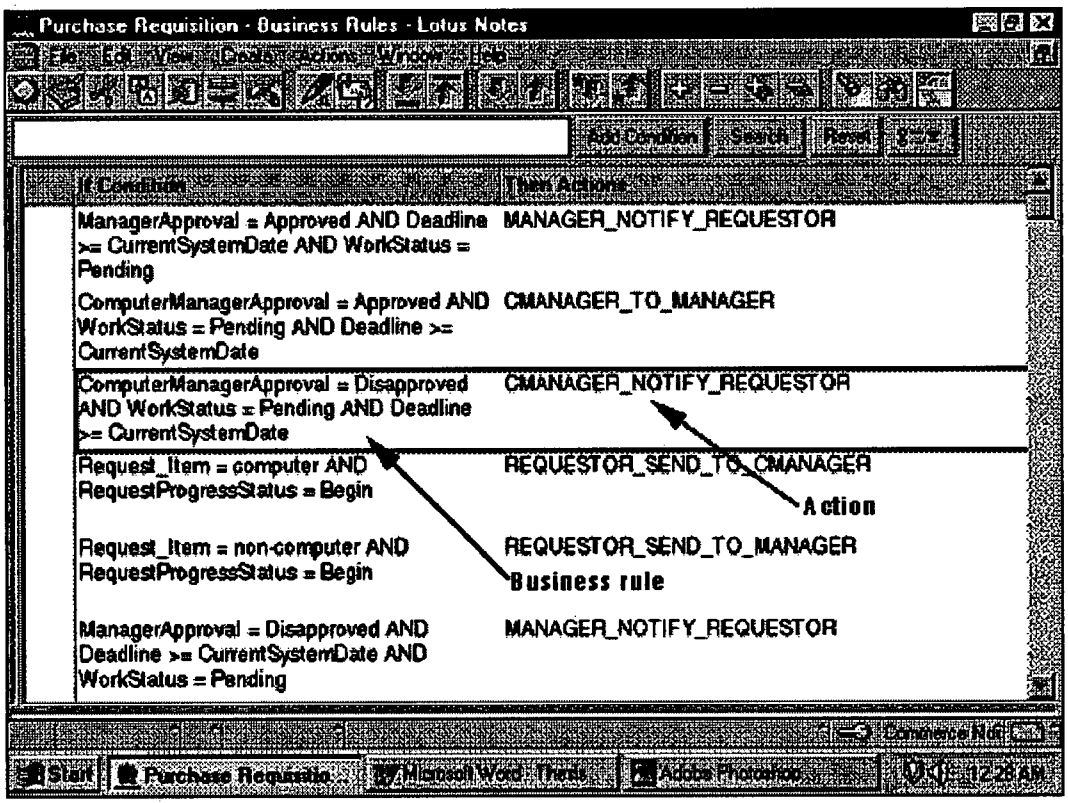


Figure 5-5 The Business Rules Documents for the Requisition Process

Three document classes were created as request types of the process. These classes are the requisition form (See Figure 5-6), the approval form for the division manager, and the approval form for the computer manager.

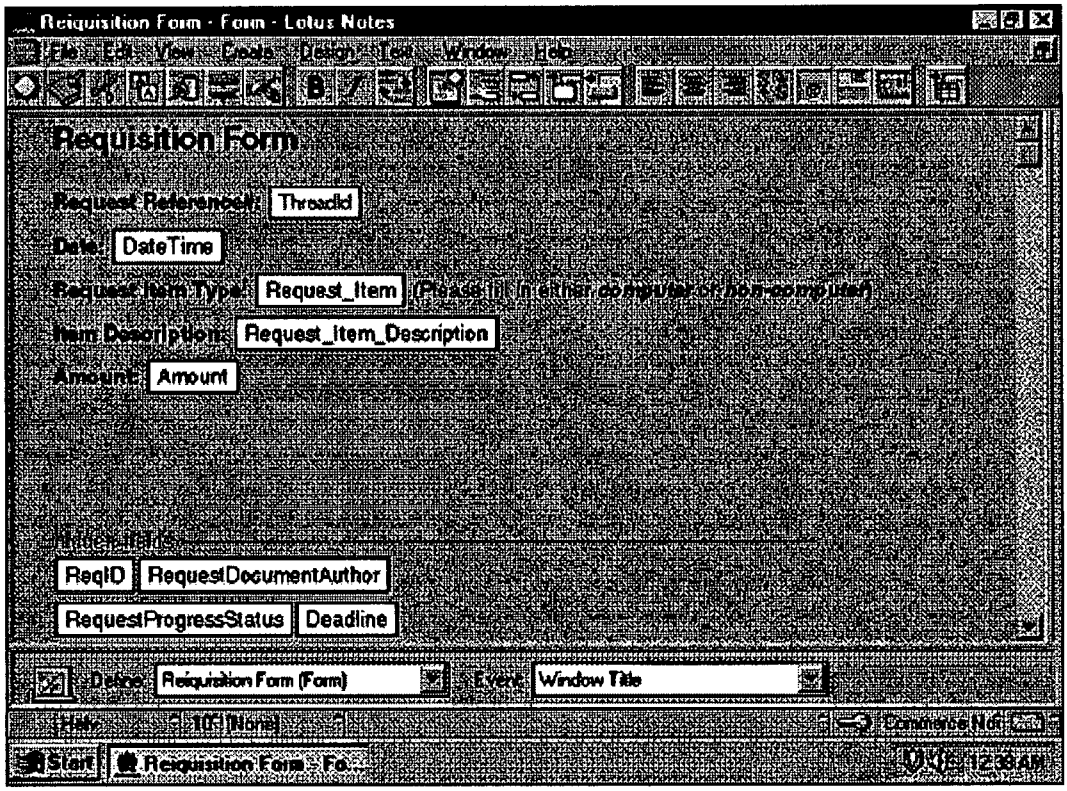


Figure 5-6 A Requisition Form

The requisition form carries information ranging from a requested item, the amounts of the item to the process state information. The approval forms belongs to a response type of the Notes documents. The documents of these forms cannot be created alone; they must be based on parent documents which, in this case, are the documents of the requisition form. The approval form for the computer manager can only be accessed by a user whose role is a computer manager in the database. The role of users can be defined in the Access Control List, as shown in Figure 5-7.

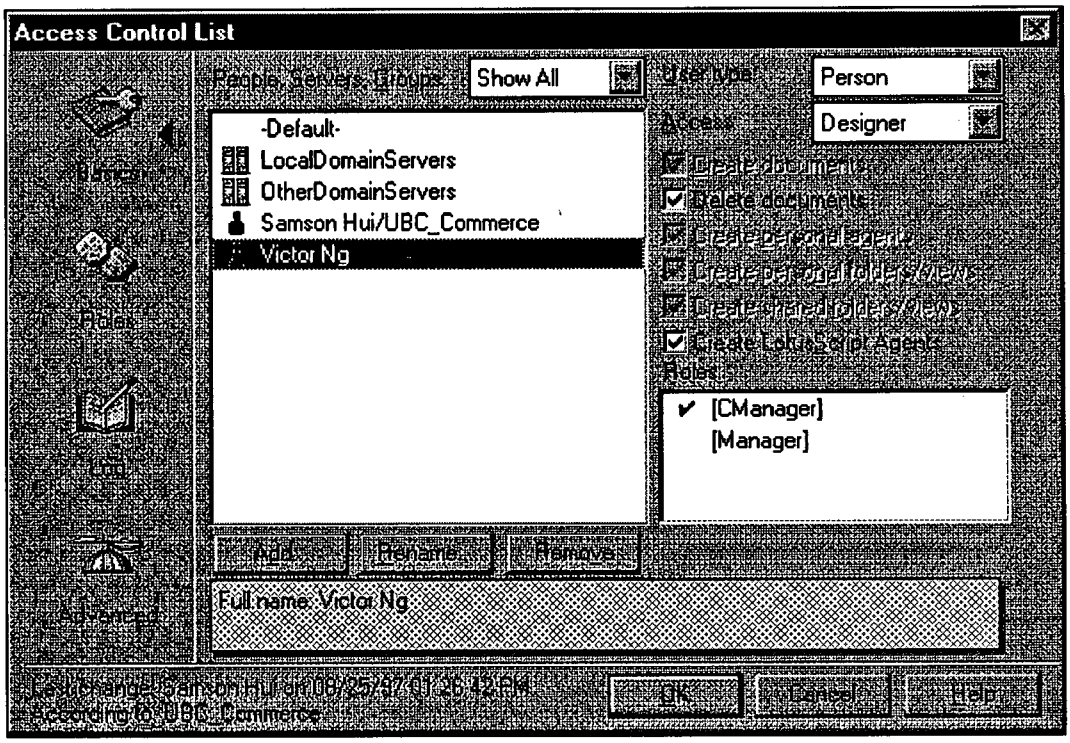


Figure 5-7 The Access Control List Dialogue Box

In the requisition process, when a user submits a PR form, a requisition request should be generated to either the division manager or the computer manager according to a requested item. Figure 5-8 shows that a PR form whose requested item is a computer-related item was created; the deadline for approving it was also calculated by the Business Controller Object Agent.

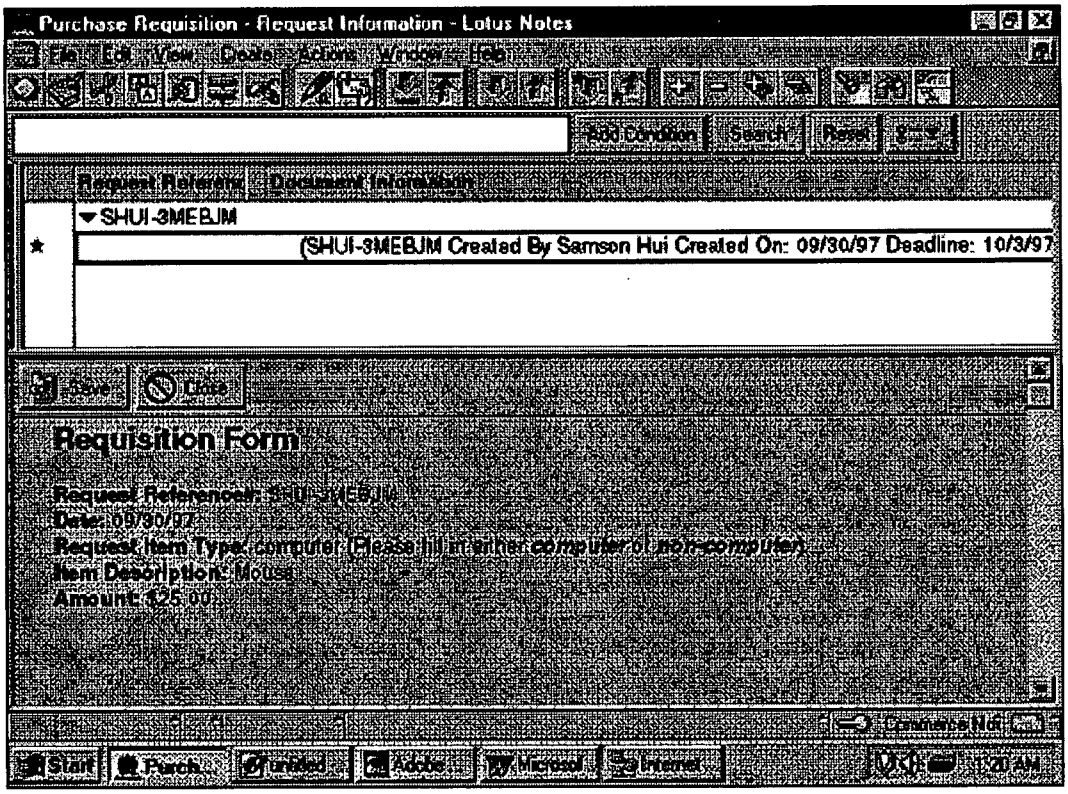


Figure 5-8 A Requisition Form for a Computer-Related Item

The Agent also sent a message to the computer manager according to business rules (See Figure 5-9) and created a Workflow State document (See Figure 5-10). The computer manager can go to the form by clicking on the icon in the message. Once the computer manager reads the form, he/she can create an approval document. If he/she approves the request, he/she can simply fill in "Approved" in the Approved field and submit it to the Controller Agent which sends another message to the division manager and updates the Workflow State information (See Figure 5-11). The manager issues the final approval of the request and submits the approval to the Controller Agent which informs the requester of the decision by electronic mail.

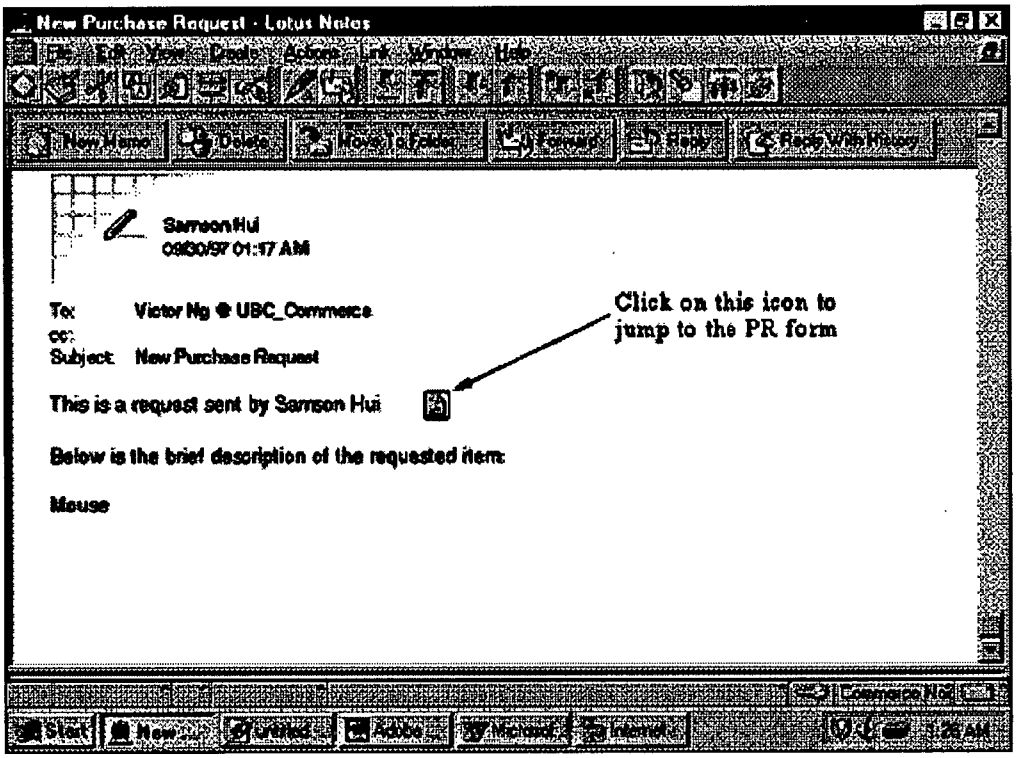


Figure 5-9 A Message Generated by the Business Controller Object Agent

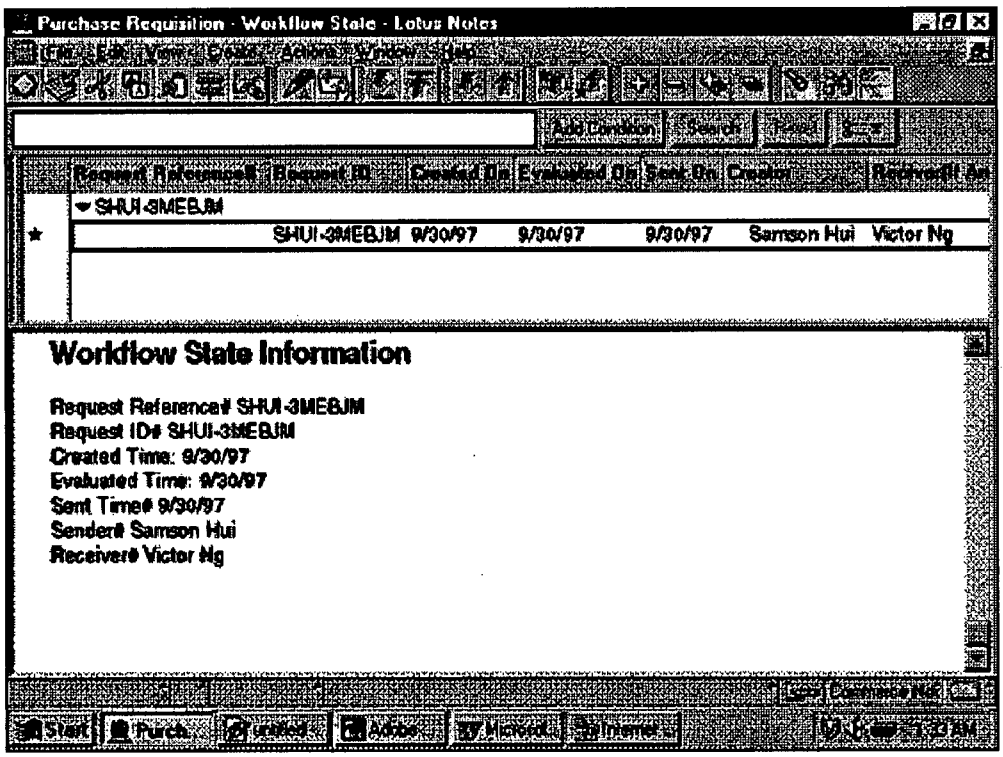


Figure 5-10 A Workflow State Document Created by the Controller Agent

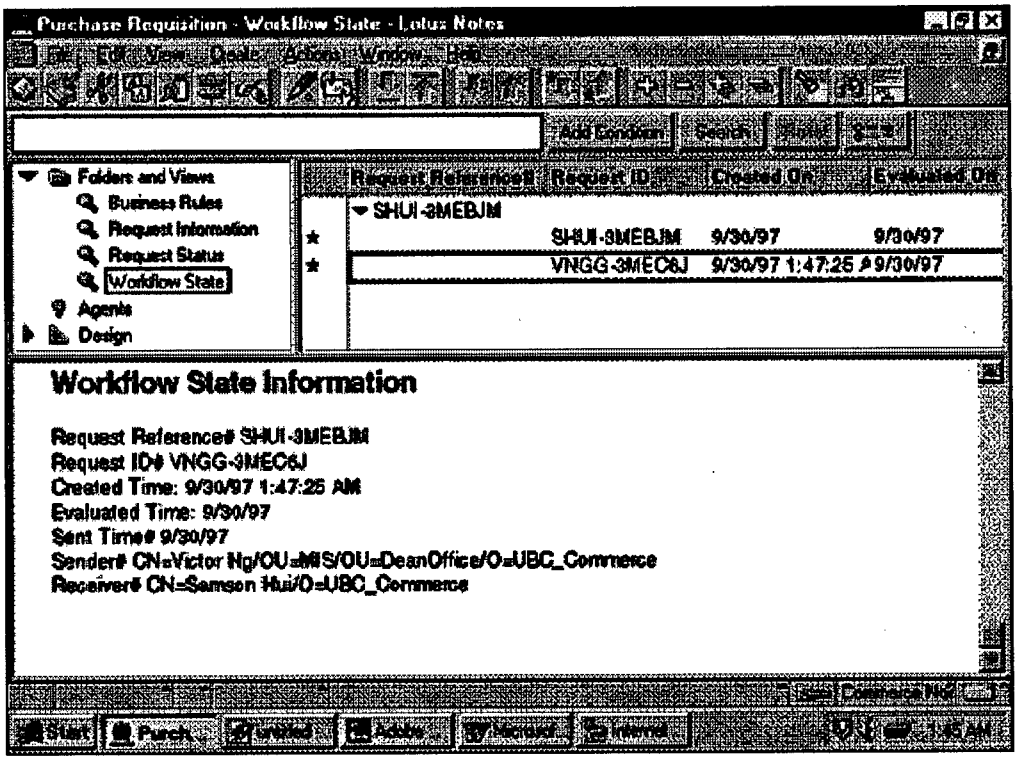


Figure 5-11 A Workflow State Document After a Message Was Sent to the Division Manager

It is important to note that the Controller Agent is triggered by the Clock Object Agent once a day. The Controller Agent compares the deadlines of all PR forms to the current system date to determine if any forms are due. If a form is due, the Controller Agent, according to a business rule, will reject the form, inform the requester, and update the Workflow State Information.

5.5. Limitations of the Implementation

There is still a gap between the system and the architecture. First, our architecture allows communication between the Clock object and the Controller object. The Controller can request the Clock to remind it of specific time occurrences. However, the Clock Object Agent in our system cannot be triggered at a specific time defined by the Business Controller Object Agent. Instead, the Clock Object

Agent triggers the Business Controller Object Agent at certain time intervals specified by application developers but not by the Business Controller Agent. Second, the Business Rule Evaluator in our architecture may refer to the state information about a process from the Workflow State Repository when it evaluates business rules. In our workflow system, the Business Controller Object Agent can simply update the state information in the Repository, but it cannot cross-reference the information to evaluate the business rules. All relevant state information is carried by a request. Finally, the conjuncts and disjuncts in a business rule in our system are limited to the field names in the **same** document type. For instance, a rule may refer to a certain value of a document, but it cannot simultaneously refer to another value of another document of a different document class.

5.6. Summary

We have discussed the pros and cons of Web-based development platforms and Lotus Notes in this chapter. The reason that we chose Notes as our implementation platform is that Notes provides a mature development environment and facilities that mesh well with the architectural components of our Object-Oriented Workflow Management System (OOWMS). To test our system, we developed a workflow application to automate the purchase requisition process. In spite of the fact that there are limitations to our implementation, the demonstration of the application proves that our architecture, derived from a set of well-formulated ontological concepts and principles, suggests a new way of building a workflow system.

6. Conclusion and Future Research

6.1. Thesis Summary

The central theme of this thesis is to suggest an architectural blueprint for a workflow management system. We developed this blueprint by exploring the concepts of workflow management and ontologically developed modelling methodology, the Object-Oriented Modelling (OOEM) method. We discussed what workflow management is and examined some common workflow terminology by following the specifications proposed by Workflow Management Coalition (WfMC). We also reviewed the WfMC's Workflow Reference Model so that we could achieve a better understanding of what workflow products should offer. Different workflow modelling techniques were compared. These techniques can be divided into two types: the traditional approach and the object-oriented approach. While the traditional approach focuses on the informational and functional aspects of a process, the object-oriented approach concentrates on the interactions between objects and captures the organizational aspects of the process.

Since OOEM provides a "natural view" of an organizational process, and since it offers no support for workflow constructs, we extended it by introducing two different workflow constructs: *activity* and *business rule*. The extension of OOEM, namely the Object-Oriented Workflow Model (OOWM), reflects our view of the organizational process in an object-oriented context. We argued that a service consists of an ordered set of activities which are governed by business rules defined by an organization. These rules only control when and by whom a specific activity

should be performed. We also extended the notion that an activity within a service can generate requests or responses to other objects.

The process of creating an OOWM was introduced; this process is referred to as the OOWM method. The method consists of two general steps which include the construction of an OOEM model for a process under study and the representation of the internal characteristics of objects by means of Object Activity Templates (OAT). In the first step, a request propagation algorithm, proposed by Zhao [1995], can be used to identify objects, their services, interface and internal attributes, and request of a process. The second step divides services into activities and identifies information about company policies which govern these activities. All this information can be represented in the OAT. The OAT is derived from the Zhao's Internal Object Template [Zhao, 1995] and introduces three additional columns: pre-condition, activity, and termination-condition columns. With these columns, the OAT is able to show the execution sequence of work within an object. Because of the ability of the OAT to describe the task structure, we also drew a relationship between an OOWM and an activity-based diagram. We found that an OOWM can be used to derive an activity diagram, but the reverse is not true.

We proposed an implementation model for an Object-Oriented Workflow Management System (OOWMS) which enacts a process described in our workflow model. The objective of the implementation model is to identify the general functionality and critical components of the OOWMS. We introduced the Controller object which monitors and controls the interactions among objects based on business

rules. Because objects can independently interact with each other, we presented a control schema which specifies which requests should be controlled by the Controller under organizational policies. Based on the implementation model, we proposed the architecture of the Object-Oriented Workflow Management System (OOWMS). We presented a request processing cycle, an algorithm taken by the Controller object to process an incoming request. In the request processing cycle, the Controller fetches a request, decodes it, retrieves information carried by the request, evaluates business rules based on the request information and state information about a process, and takes actions in accordance with the result of this evaluation. The Controller object consists of two components: the Business Rule Evaluator and the Workflow Executor. The Evaluator is responsible for evaluating the business rules and instructing the Executor to perform work according to the rules. Different kinds of information are also required for the request processing cycle; such information includes the state information about a process, the information carried by requests, the business rules, and request types.

Finally, we used Lotus Notes to build a simple workflow system by following the architecture. The Notes facilities seem to mesh well with the identified architectural components. We showed that the system is functional by applying it to a hypothetical purchase requisition process.

6.2. Contributions

This thesis continues previous research efforts focused on developing an ontologically-based Object-Oriented Enterprise Modelling (OOEM) method. It

addresses the inability of OOEM to capture the task structure at an object level by proposing workflow constructs to OOEM so that a more complete model is formulated. We believe that our OOWM captures the informational, functional, and organizational aspects of a process.

Another major contribution of the thesis is the architecture of the OOWMS which enacts an OOWM. Since the architecture is derived from purely object-oriented thinking, a workflow system following this architecture can be very flexible and adaptable to fit into a constantly changing business environment. The architecture also sheds some light on how a workflow system can be developed in a heterogeneous business environment in which business divisions or departments are autonomous.

Finally, our prototype of a workflow system suggests another approach to building workflow applications in the Notes environment. Traditionally, Notes developers have been hardcoding business logic into their applications. Our prototype shows that it is possible to separate the business logic from programming codes and to allow non-technical users to customize the rules.

6.3. *Limitations and Future Research*

Several research issues need to be addressed. First, the number of cases to which the OOWM method and the architecture has been applied is very limited. Case studies should be conducted to further examine the practicality of the method and the architecture. Second, this thesis does not address the technical aspect of how the objects should communicate with each other. How can the Controller object be introduced into a technologically heterogeneous environment? Even though some

technological initiatives such as CORBA and DCOM are currently being taken by research institutions and computer vendors, the question of how these initiatives may be applied to our architecture should lead to future research. Finally, CASE tools can be developed to support the construction of an OOWM and to generate workflow implementations on the basis of workflow specifications.

Bibliography

1. Allaire Corporation. (1997). Cold Fusion 3.0 Home Page. http://www.allaire.com/products/coldfusion/30/index.cfm.

2. Amberg, M. (1997). "The Benefits of Business Process Modeling for Workflow Systems". *Workflow Handbook 1997.* Edited by Peter Lawrence. England: John Wiley & Sons Ltd. p. 61-68.

3. Angeles, P.A. (1981). *Dictionary of Philosophy.* New York: Harper Perennial.

4. Coad, P., and Yourdon, E. (1991). *Object-Oriented Analysis*, 2nd edition. Englewood Cliffs, New Jersey: Yourdon Press/Prentice Hall.

5. Coad, P., and Yourdon, E. (1991). *Object-Oriented Design*, 2nd edition. Englewood Cliffs, New Jersey: Yourdon Press/Prentice Hall.

6. Curtis, B., Kellner, M., and Over, J. (1992). "Process Modelling". *Communication of the ACM*, Vol. 35, No. 9. p. 75-90

7. Georgakoopoulos, D., Hornick, M., and Sheth, A. (1995) "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure", *Distributed and Parallel Database*, No. 3, p. 119-153.

8. Ginsburg, M. and Duliba, K. (1997) "Enterprise-Level Groupware Choices: Evaluating Lotus Notes and Intranet-Based Solutions". *Computer Supported Cooperative Work: The Journal of Collaborative Computing,* No. 6. p. 91-115.

9. Grasso, A., Meunier, J.L., Pagani, D., and Pareschi, R. (1997). "Distributed Coordination and Workflow on the World Wide Web", *Computer Supported Cooperative Work: The Journal of Collaborative Computing,* No. 6. p. 175-200.

10. Greene, C. (1995). "Workflow Management 101", *Lotus Notes Advisor*, Vol. 1, no. 1, Premiere Issue, p. 34-39.

11. Hales, K., and Lavery, M. (1991). *Workflow Management Software: the Business Opportunity.* London, UK: Ovum Ltd.

12. Integrated Definition for Function Modeling (IDEF0). Federal Information Proceeding Standard Publication 183, December 12, 1993.

13. Jablonski, S., and Bussler, C. (1996) *Workflow Management: Modeling Concepts, Architecture and Implementation.* UK: International Thomson Computer Press.

14. Jacobson, I., Ericsson, M., and Jacobson, A. (1995). *The Object Advantage - Business Process Re-engineering with Object Technology*. ACM Press. Addison-Welsey Publishing Company.

15. Jung, D. (1997*). Object-Oriented Modeling: From Analysis to Logical Design*. M.Sc. Dissertation. The University of British Columbia.

16. Kobielus, J. G. (1997). *Workflow Strategies*, CA: IDG Books Worldwide, Inc.

17. Laamanen, M. T. (1994). "The IDEF standards", in: *Methods and Associated Tools for the Information Systems Life Cycle*, A.A. Verrijn-Stuart and T.W. Olle (Editors), Elsvier Sciences B.V. (North Holland). p. 121-130.

18. Lakin, R., Capon, N., and Botten, N. (1996). "BPR Enabling Software for the Financial Services Industry", *Management Services*, March , p. 18-20.

19. Lotus Corporation. (1995). Lotus Notes "White Paper". http://www.lotus.com/bible/.

20. Lotus Corporation. (1997). Domino Home Page. http://www2.lotus.com/domino.nsf.

21. Mano, M. M. (1993). *Computer System Architecture*, 3rd edition. New Jersey: Prentice Hall.

22. Medina-Mora, R., Winograd, T., Flores, R., and Flores, F. (1992). "The Action Workflow Approach to Workflow Management Technology," *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW)*, New York:ACM. p. 281-288.

23. Netscape Communications Corporation. (1997). Netscape Server Central - SuiteSpot Home Page. http://www.netscape.com/comprod/server_central/product/suite_spot/index.html.

24. Novell Inc. (1997). Groupwise Home Page. http://www.novell.com/groupwise/.

25. Oracle Corporation. (1997). Web Developer Suite Home Page. http://www.oracle.com/products/tools/WDS/.

26. Orfali, R. Harkey, D., and Edwards, J. (1996). *The Essential Distributed Objects Survival Guide*. Canada: John Wiley & Sons, Inc.

27. Ruiz, D. (1997). "Growth and Challenges in Enterprise Workflow". *Workflow Handbook 1997*. Edited by Peter Lawrence. England: John Wiley & Sons Ltd. p. 223-230.

28. Rumbaugh, J., Blaha M., Premerlini W., Eddy F., and Lorensen W. (1991). Object-Oriented Modeling and Design. Englewood Cliffs, New Jersey: Prentice Hall.

29. Sprague, R. H., Jr., and McNurlin, B. C. (1993). *Information systems Management In Practice*, 3rd edition. New Jersey: Prentice Hall.

30. Stark, H. (1997). "Understanding Workflow". *Workflow Handbook 1997.* Edited by Peter Lawrence. England: John Wiley & Sons Ltd. p. 5-25.

31. Tan, W. (1997). *A Semantically-Enhanced Object-Oriented Case Tool For Enterprise Modeling.* M.Sc. Dissertation. The University of British Columbia.

32. Wand, Y. (1989). "A Proposal for an Formal Model of Objects", *Object-Oriented Concepts, Language, Applications, and Database.* Kim, W., and Lochovsky, F.H.. New York: ACM Press/Addison-Welsey Publishing Company. p. 537-599.

33. Wand, Y., and Weber, R. (1990). "An Ontological Model of an Information System", IEE Transactions On Software Engineering. Vol. 16. No. 11. p. 1282 - 1292.

34. Wand, Y., and Woo, C. C. (1993). "Object-Oriented Analysis - Is It Really That Simple?", *Proceedings of the 3rd Workshop on Information Technology and Systems.* December. Orlando, Florida. p. 186-195.

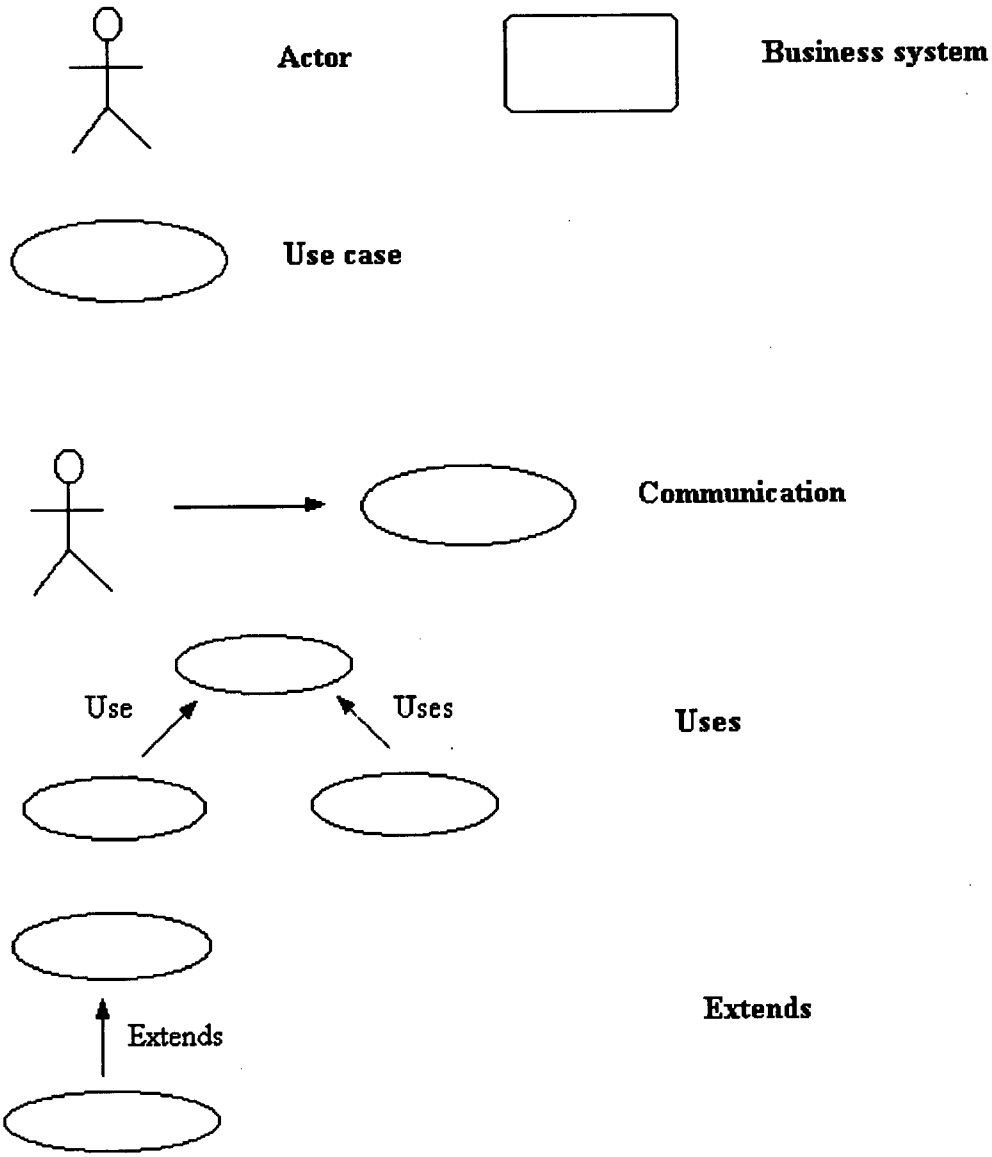
35. Wang, S. (1994). "OO Modeling of Business Processes", *Information Systems Management*, Spring, p. 36-43.

36. Workflow Management Coalition. (1997). *Workflow Handbook 1997.* Edited by Peter Lawrence. England: John Wiley & Sons Ltd.

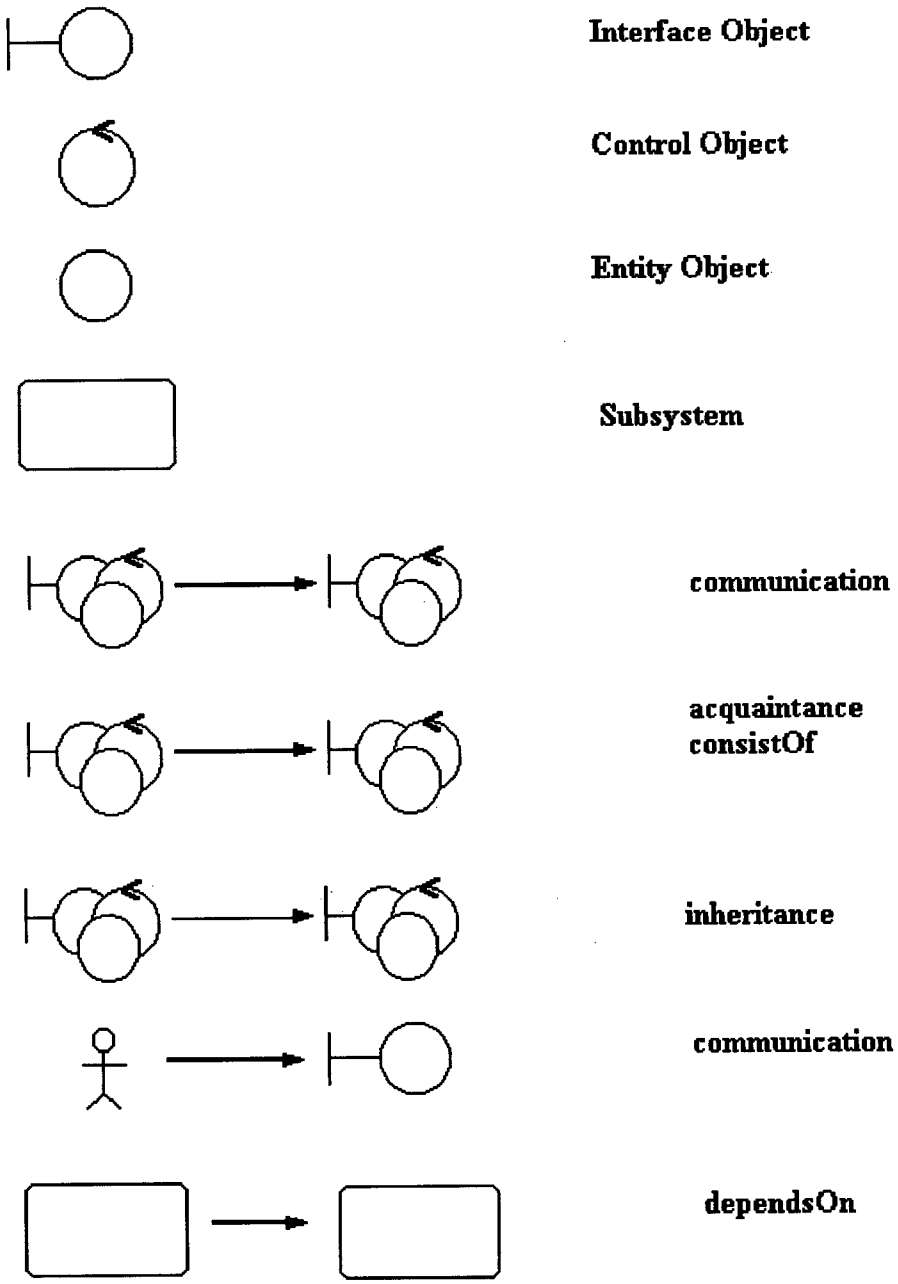
37. Yourdon, E. (1989). *Modern Structured Analysis.* New Jersey: Prentice Hall. p. 259-274.

38. Zhao, H. (1995). *Object-Oriented Enterprise Modeling.* M.Sc. Dissertation. The University of British Columbia.

Appendix A - Graphical Constructs of the Use-Case Model



Appendix B - Graphical Constructs of the Object Model



Appendix C - Guidelines for Constructing the Models of Rumbaugh's OMT*

1. **Determine the problem domain**

2. **Construct an Object Model**

   - Identify object classes
   - Begin a data dictionary containing descriptions of classes, attributes, and associations
   - Add associations between classes
   - Add attributes and links
   - Organise and simplify object classes using inheritance
   - Test access paths using scenarios and iterate the above steps
   - Group classes into modules, based on close coupling and related functions

3. **Develop a Dynamic Model**

   - Prepare scenarios of typical interaction sequences
   - Identify events between objects and prepare an event trace for each scenario
   - Prepare an Event Flow Diagram for the system
   - Develop a state diagram for each class that has important dynamic behavior

4. **Construct a Function Model**

   - Identify input and output values

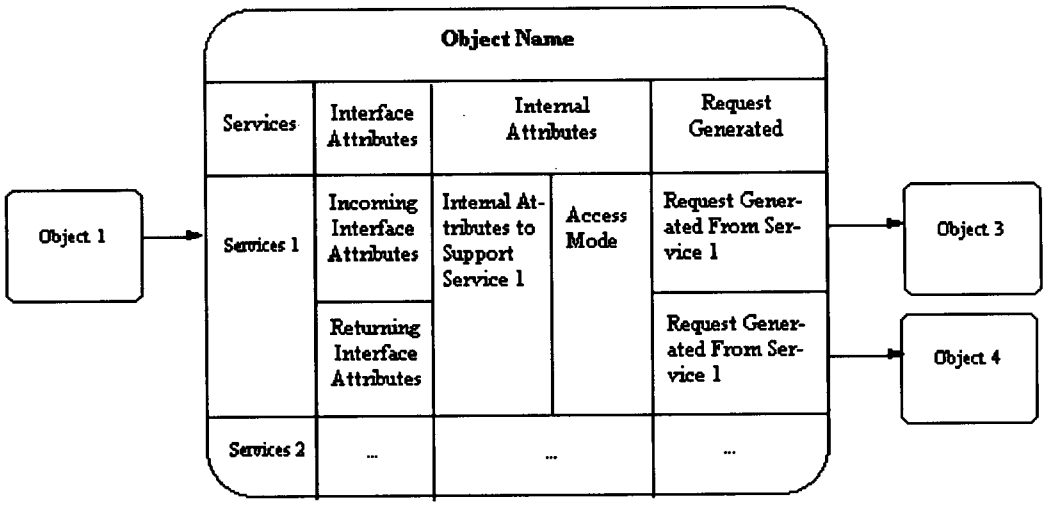
---

* The guidelines are directly taken from Zhao's *Object-Oriented Enterprise Modeling* in which the steps to constructing the models of OMT are summarised.

- Use data flow diagrams as needed to show functional dependencies
- Describe what each function does
- Identify constraints
- Specify optimization criteria

5. **Verify, iterate and refine the three models**

Appendix D - Appendix D - An OOEM Internal Object Template (IOT)



Appendix E - Summary of Wand and Woo's Modeling Rules

The modelling rules proposed by Wand and Woo [1993] are summarized in this section. They provide fundamental guidelines for constructing an Object-Oriented Enterprise Modelling (OOEM) model.

1. **The scope identification rule**

   This rule defines the boundary of an enterprise model. It separates activities within the system from those in the external environment. The environment is represented by the external objects or clients of the system. The interaction between the environment and the system is modelled by an external request. When the system receives the request, its state becomes unstable until such a request is satisfied.

2. **The object identification rule**

   This rule identifies things that should be modelled as objects. An object is included if and only if it provides or request at least one service. The rule reflects the principle that every change is tied to a change of state of things and that everything changes. An internal object is an object that is part of the system and provides at least one service. An external object belongs to the environment that interacts with the system.

3. **The service inclusion rule**

   A service is included in an object if and only if it is invoked by at least one request in the system as defined by Rule #1. Such a request can be generated by either an

external object or an internal object. Services reflect internal transformations (state changes) of things.

4. **The attribute inclusion rule**

   This rule determines which object attributes should be included in a model. An interface attribute must be used or affected by at least one service, and known to at least one other object. An internal attribute must be affected by at least one service and be unknown to other objects.

5. **The attribute ownership rule**

   This rule reflects the ontological principle that properties always belong to things. This rule identifies the owner of the attribute. When an object modifies the attribute via its service, that object is known as the *custodian* of the attribute. Other objects can only obtain or modify the value of the attribute through the actions of the custodian object. This rule ensures that every attribute belongs to a specific object.

6. **The aggregation and decomposition rule**

   This rule determines when objects, as defined in Rule #2, should be combined (aggregated) and decomposed in the model. A composite object refers to the aggregation of the objects. It is included if and only if it provides services that are not provided by any of its components (its aggregate objects). When modelling the properties of the composite object, one must include those properties not modelled in components. The rule echoes the ontological principle that a composite thing must have emergent properties.

7. **The generalization and specialization rule**

This rule states that a general object class can be created if and only if two or more object classes provide one or more common services. This general object class is called a *super-class* of the original object classes. The original object class is referred to as a *sub-class* of the general object class. All services provided by the super-class should be eliminated from the sub-classes which can inherit those services from their super-class and which entail different services and attributes from their super-class.

Appendix F - Bunge's Ontological Constructs

This appendix summarizes Bunge's ontological constructs [Bunge, 1977, 1979]. The summary of the constructs is taken from Zhao [1995], much of whose work is largely based on Bunge's.

Static Model of an Substantial Individual

- **Thing**. A thing is defined as an entity or substantial individual endowed with all its properties. The world is made of things that have properties.

  Bunge distinguishes thing and constructs. Constructs are creations of the human mind. There are four basic kinds of constructs: concepts, propositions, contexts, and theories. Constructs do not have all the properties of things. For example, sets add and intersect but do not move around, have no energy and no causal efficacy, etc. Constructs, even those representing things or substantial properties, have a conceptual structure, not a material one. In particular, predicates and propositions have semantic properties, such as meaning, which is a non-physical property.

- **Properties, Attributes, and Functional Schema**. Properties of substantial individuals are called substantial properties. Properties of things can be intrinsic or mutual to several things, e.g. if a person is employed by a company, employment is a property of both the person and the company. A property is modelled via an attribute function that maps the thing into a set of values. Attributes are characteristics assigned to things by humans; therefore, they reflect the view point of an individual observer. An attribute can be represented as a function from a set

of things and a set of observation points into a set of values. This is the basis for defining a model of a thing as a functional schema: a functional schema is a set of attribute functions defined over a certain domain, usually time. Similar things can be modelled using the same functional schema.

- **Composite things**. Composite things are things composed of other things. More precisely, an individual is composite if and only if it is composed of individuals other than itself and the null individual. A composite thing has *hereditary* properties and *emergent* properties. A property of a composite thing that belongs to a component thing is called a hereditary property; otherwise, it is called an emergent property. A composite thing must have an emergent property. The notion of emergent property is an important assumption in Bunge's ontology. According to him, every concrete system is assembled from, or with the help of, things in the same or lower order genera but possesses properties not available in the components of the system. The hierarchy of system genera can be characterised as: physical, chemical biological, social, and technical.
- **State and Conceivable State Space**. Every thing is - at a given time associated with a given reference frame - in some state or other. The vector of values for all attribute functions of a thing is the state of the thing.

  The set of all states that the thing might ever assume is the conceivable state space of the thing.
- **State Law**. A state law restricts the values of the properties of a thing to a subset that is deemed lawful because of natural laws or human laws. A law is also

considered a property of the thing.

- **Class, Kind, and Natural Kind**. A class is a set of things that possess a common property.

  A kind is a set of things that possess two or more common properties.

  A natural kind is a set of things that share the same laws.

  Things come in natural kinds, i.e. classes of things possessing ("obeying") the same laws. A natural kind constitutes a natural grouping because it rests on a set of laws, but it is not a real thing: it is a construct.

Dynamic Model of a Substantial Individual

- Event. An event is a change in the state of a thing.

  In order to keep track of the changes undergone by things, we need the principle of nominal invariance which states that a thing, if named, shall keep its name throughout its history as long as the latter does not include changes in natural kind - changes which call for changes of name.

- **Event Space**. The event space of a thing is the set of all possible events that can occur in the thing. Let S(x) be a state space for a thing x. Any pair of points in this set will unambiguously represent a conceivable event in x.

- **Transformation and Lawful Transformation**. A transformation is a mapping from a domain comprising states to a co-domain comprising states.

- **History**. The chronologically ordered states that a thing traverses are the history of the thing.

Static Model of System

- **Coupling**. A thing acts on another thing if its existence affects the history of the other things. The two things are said to be coupled or to be interacting.
- **System**. A set of things forms a system if and only if for any bi-partition of the set, coupling exists among things in the two sets.
- **System Composition**. A decomposition of a system is a set of subsystems such that every component in the system is either one of the subsystems in the decomposition or is included in the composition of one of the subsystems.
- **System Environment**. Things that are not in the system but interact with things in the system are called the environment of the system.
- **System Structure**. The set of couplings that exists among things in the system and among things in the system and things in the environment of the system is called the structure of the system.
- **Subsystem**. A subsystem is a system whose components and structure are subsets of the components and structure of another system.
- **Level Structure**. A level structure defines a partial order over the systems in a decomposition to show which subsystems are components of other subsystems or of the system itself.

Dynamic Modelling of System

- Stable State and Unstable State. A stable state is a state in which a thing, subsystem or system will remain unless forced to change by virtue of the action of a thing in the environment (an external event).
- **External Event**. An external event is an event that arises in a thing, subsystem or

system by virtue of the action of some thing in the environment on the thing, subsystem or system. The before-state of an external event is always stable. The after-state may be stable or unstable.

- **Internal Event**. An internal event is an event that arises in a thing, subsystem or system by virtue of lawful transformations in the thing, subsystem or system. The before-state of an internal event is always unstable. The after-state may be stable or unstable.
- **Well-Defined Event**. A well-defined event is an event in which the subsequent state can always be predicted, given that the prior state is known.
- **Poorly Defined Event**. A poorly defined event is an event in which the subsequent state cannot be predicted, given that the prior state is known.

Appendix G - The Complete OOWM for the Purchase Reimbursement Process

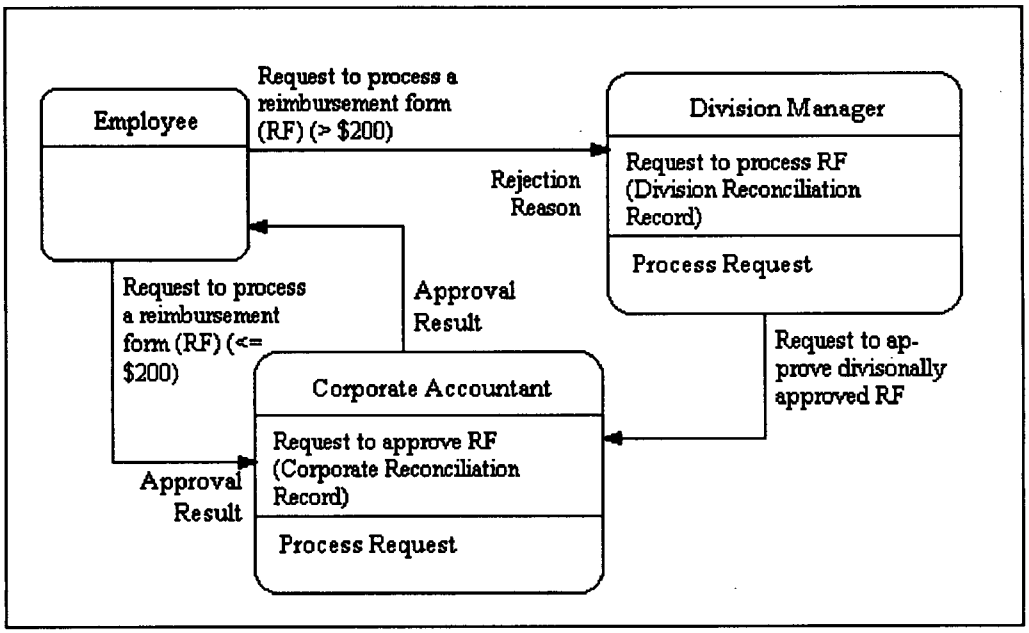
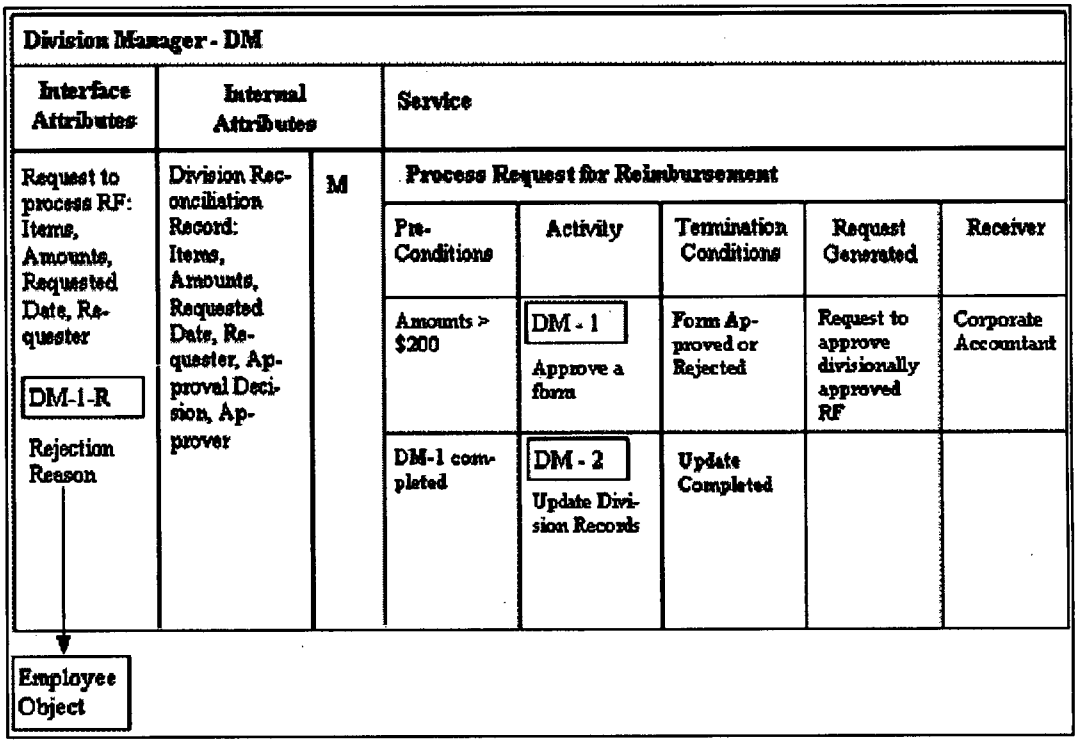


Figure G-1 Complete OOEM for the Purchase Reimbursement Process

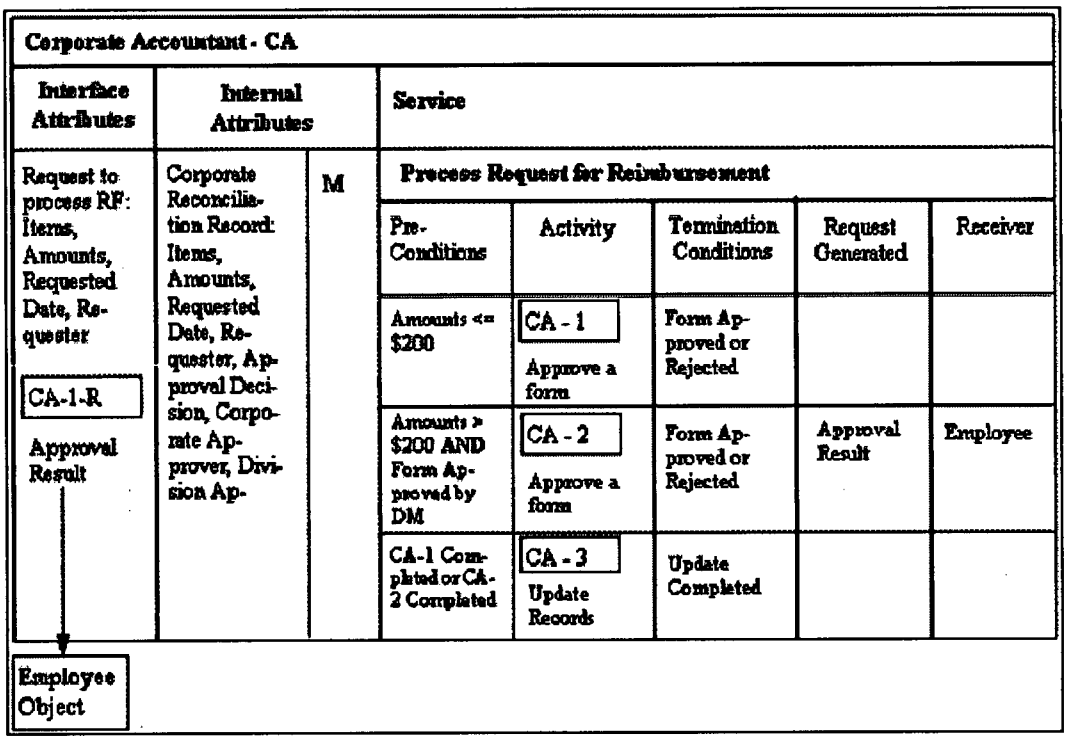


Division Manager - DM

Interface Attributes	Internal Attributes		Service				
Request to process RF: Items, Amounts, Requested Date, Requester	Division Reconciliation Record: Items, Amounts, Requested Date, Requester, Approval Decision, Approver	M	Process Request for Reimbursement				
			Pre-Conditions	Activity	Termination Conditions	Request Generated	Receiver
DM-1-R Rejection Reason			Amounts > $200	DM - 1 Approve a form	Form Approved or Rejected	Request to approve divisionally approved RF	Corporate Accountant
			DM-1 completed	DM - 2 Update Division Records	Update Completed		

Employee Object

Table G-1 An OAT for the Division Manager



Corporate Accountant - CA

Interface Attributes	Internal Attributes		Service				
Request to process RF: Items, Amounts, Requested Date, Requester	Corporate Reconciliation Record: Items, Amounts, Requested Date, Requester, Approval Decision, Corporate Approver, Division Ap-	M	Process Request for Reimbursement				
			Pre-Conditions	Activity	Termination Conditions	Request Generated	Receiver
CA-1-R Approval Result			Amounts <= $200	CA - 1 Approve a form	Form Approved or Rejected		
			Amounts > $200 AND Form Approved by DM	CA - 2 Approve a form	Form Approved or Rejected	Approval Result	Employee
			CA-1 Completed or CA-2 Completed	CA - 3 Update Records	Update Completed		

Employee Object

Table G-2 An OAT for the Corporate Accountant

Appendix H - The OOEM, OATs for the Internal Objects, and the Control Schema for the Purchase Requisition Process

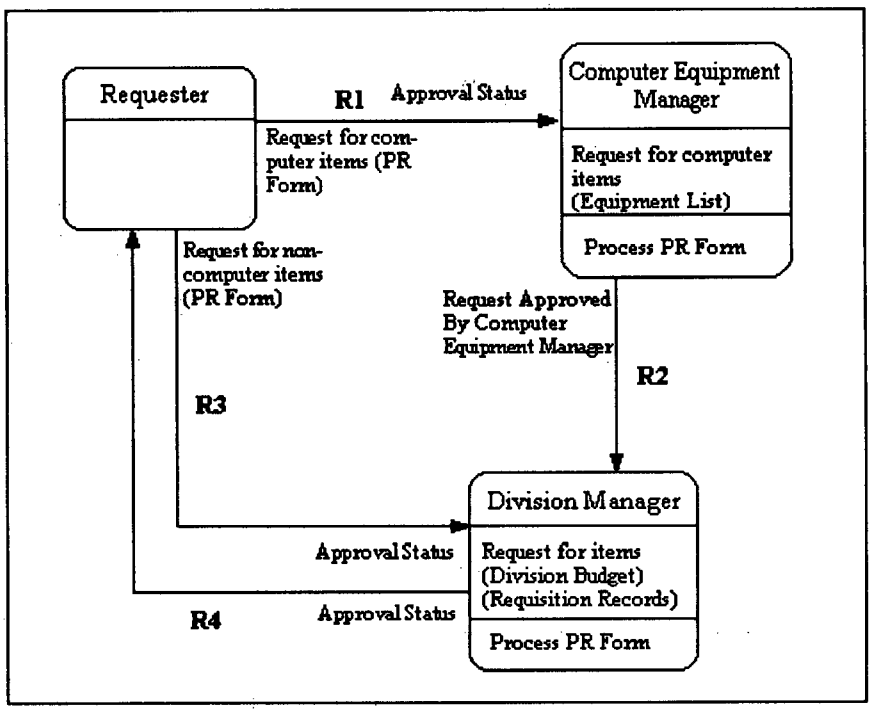


Figure H-1 The OOEM Model for the Purchase Requisition Process

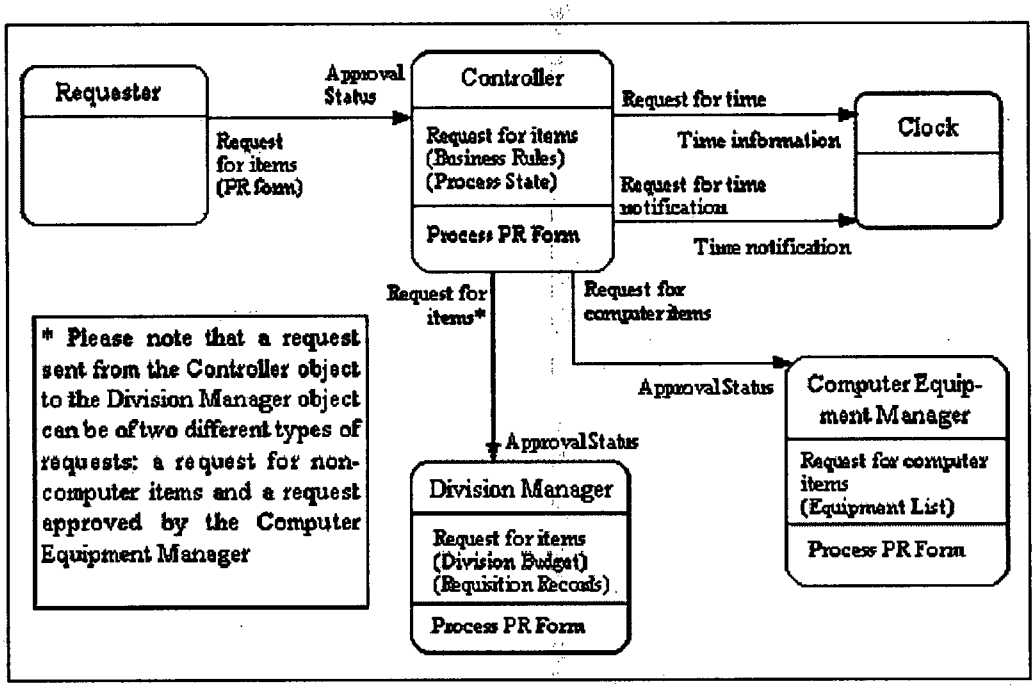
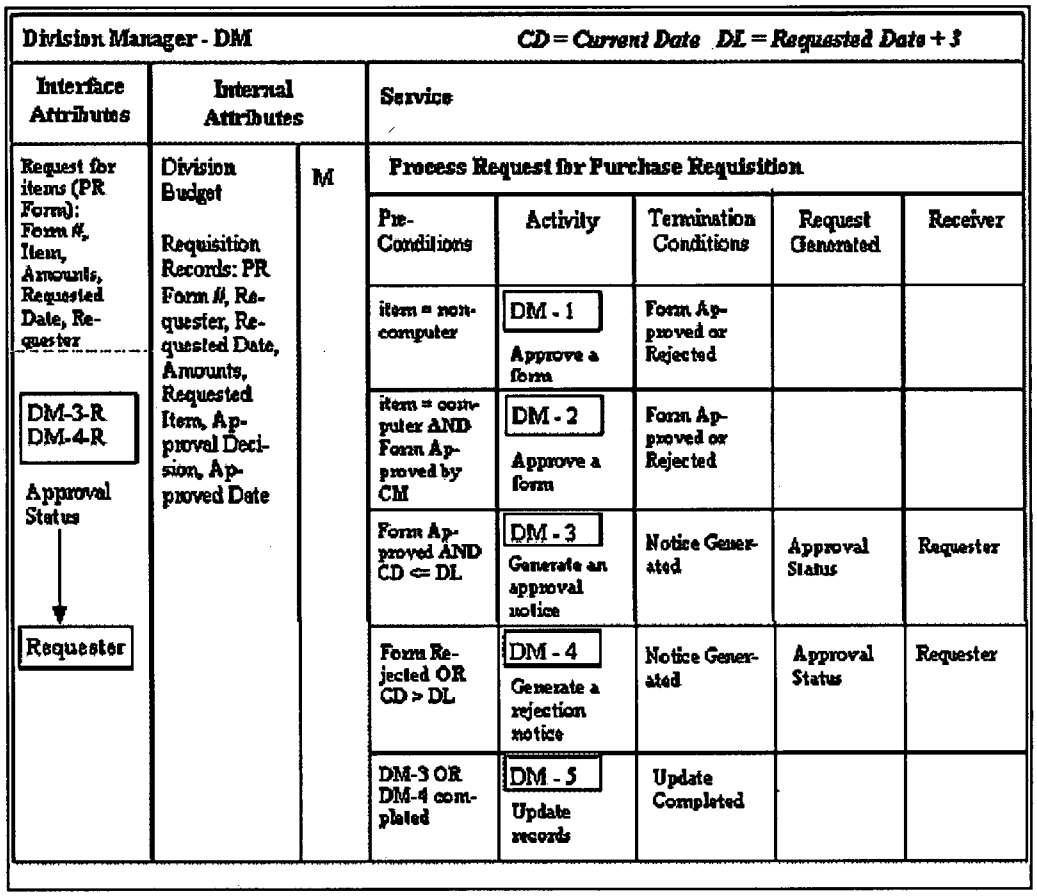


Figure H-2 The OOEM Model with the Controller Object for the Requisition Process

Computer Equipment Manager - CM			CD = Current Date DL = Requested Date + 3				
Interface Attributes	**Internal Attributes**		**Service**				
Request for computer items (PR Form): Form #, Item, Amounts, Requested Date, Requester	Equipment List: Bar Code #, item, location	U	**Process Request for Purchase Requisition**				
			Pre-Conditions	Activity	Termination Conditions	Request Generated	Receiver
			item = computer	CM- 1 Approve a request	Request Approved or Rejected		
CM-2-R CM-3-R Approval Status ↓ Requester			Request Approved AND CD <= DL	CM- 2 Generate a request	Request Generated	Approved Request	Division Manager
			Request Rejected OR CD > DL	CM- 3 Generate a rejection notice	Notice Generated		

Table H-1 The Object Activity Template for the Computer Equipment Manager



Division Manager - DM			CD = Current Date DL = Requested Date + 3				
Interface Attributes	**Internal Attributes**		**Service**				
Request for items (PR Form): Form #, Item, Amounts, Requested Date, Requester DM-3-R DM-4-R Approval Status ↓ Requester	Division Budget Requisition Records: PR Form #, Requester, Requested Date, Amounts, Requested Item, Approval Decision, Approved Date	M	**Process Request for Purchase Requisition**				
			Pre-Conditions	Activity	Termination Conditions	Request Generated	Receiver
			item = non-computer	DM - 1 Approve a form	Form Approved or Rejected		
			item = computer AND Form Approved by CM	DM - 2 Approve a form	Form Approved or Rejected		
			Form Approved AND CD <= DL	DM - 3 Generate an approval notice	Notice Generated	Approval Status	Requester
			Form Rejected OR CD > DL	DM - 4 Generate a rejection notice	Notice Generated	Approval Status	Requester
			DM-3 OR DM-4 completed	DM - 5 Update records	Update Completed		

Table H-2 The Object Activity Template for the Division Manager

Request	Workflow Business Rule(s)	Action(s)
R1	item = computer	Sends a form to the Computer Equipment Manager
R1' (immediate response to R1)	(Request approved or rejected by the Computer Manager) **OR** (the current date > the R1 submitted date + 3 calendar days)	Informs the requester of the Approval Status
R2	The request is approved by The Computer Manager **AND** the current date <= R1 submitted date + 3 calendar days	Sends the approved request to the Division Manager
R3	item = non-computer	Sends the form to the Division Manager
R3' (immediate response to R3)	(Request approved or rejected by the Division Manager) **OR** (the current date > the **R3** submitted date + 3 calendar days)	• Informs the requester of the Approval Status • Updates the Requisition Records
R4	(Request approved or rejected by the Division Manager) **OR** (the current date > the **R2** submitted date + 3 calendar days)	• Informs the requester of the Approval Status • Updates the Requisition Records

Table H-3 The Control Schema for the Purchase Requisition Process