

Implementing the CoSAWoE model in a commercial workflow product

by

Carmen Erwee

Implementing the CoSAWoE model in a commercial workflow product

by

Carmen Erwee

Dissertation

submitted in fulfillment
of the requirements
for the degree

Magister Technologiae

in

Information Technology

in the

Faculty of Engineering

of the

Nelson Mandela Metropolitan University

Promoter: Prof. Reinhardt A. Botha

January 2005

Declaration

I, Carmen Erwee, hereby declare that:

- The work in this dissertation is my own work.
- All sources used or referred to have been documented and recognized.
- This dissertation has not previously been submitted in full or partial fulfillment of the requirements for an equivalent or higher qualification at any other recognized educational institute.



Carmen Erwee

Abstract

Workflow systems have gained popularity not only as a research topic, but also as a key component of Enterprise Resource Planning packages and e-business. Comprehensive workflow products that automate intra- as well as inter-organizational information flow are now available for commercial use. Standardization efforts have centered mostly around the interoperability of these systems, however a standard access control model has yet to be adopted. The research community has developed several models for access control to be included as part of workflow functionality. Commercial systems, however, are still implementing access control functionality in a proprietary manner.

This dissertation investigates whether a comprehensive model for gaining context-sensitive access control, namely CoSAWoE, can be purposefully implemented in a commercial workflow product. Using methods such as an exploratory prototype, various aspects of the model were implemented to gain an understanding of the difficulties developers face when attempting to map the model to existing proprietary software.

Oracle Workflow was chosen as an example of a commercial workflow product. An investigation of the features of this product, together with the prototype, revealed the ability to affect access control in a similar manner to the model: by specifying access control constraints during administration and design, and then enforcing those constraints dynamically during run-time. However, only certain components within these two aspects of the model directly affected the commercial workflow product. It was argued that the first two requirements of context-sensitive access control, *order of events* and *strict least privilege*, addressed by the **object design**, **role engineering** and **session control** components of the model, can be simulated if such capabilities are not pertinently available as part of the product. As such,

guidelines were provided for how this can be achieved in Oracle Workflow. However, most of the implementation effort focussed on the last requirement of context-sensitive access control, namely *separation of duties*.

The CoSAWoE model proposes **SoD administration** steps that includes expressing various business rules through a set of *conflicting entities* which are maintained outside the scope of the workflow system. This component was implemented easily enough through tables which were created with a relational database. Evaluating these conflicts during run-time to control **worklist generation** proved more difficult. First, a thorough understanding of the way in which workflow history is maintained was necessary. A re-usable function was developed to prune user lists according to user involvement in previous tasks in the workflow and the conflicts specified for those users and tasks. However, due to the lack of a central access control service, this re-usable function must be included in the appropriate places in the workflow process model.

Furthermore, the dissertation utilized a practical example to develop a prototype. This prototype served a dual purpose: firstly, to aid the author's understanding of the features and principles involved, and secondly, to illustrate and explore the implementation of the model as described in the previous paragraphs.

In conclusion the dissertation summarized the CoSAWoE model's components which were found to be product agnostic, directly or indirectly implementable, or not implemented in the chosen workflow product. The lessons learnt and issues surrounding the implementation effort were also discussed before further research in terms of XML documents as data containers for the workflow process were suggested.

Acknowledgements

I would like to dedicate this dissertation to my father who was in the process of doing his own master's degree when he died. I know this would have made him immensely proud. He taught me that "when you do something, you do it properly"; an attitude which will continue to inspire me in all my endeavors.

I would also like to thank the following people, without who this research would have been an impossible undertaking:

- My promoter, Professor Reinhardt Botha, for his astonishing wealth of knowledge which he made available to me so enthusiastically, patiently and constructively.
- My family and friends for always being there; supporting and encouraging me when I needed it, and reminding me of my own strength during those long hours in front of the computer screen.
- My colleagues in the faculty of Computer Studies, for their advice and encouragement. Special thanks goes to Breyten, Elderige and Stephen for helping me out when I got stuck in the technical quicksand.
- The Port Elizabeth Technikon and the National Research Foundation for administrative and financial support.

Trademarks

The following product names appear in the dissertation: Oracle 9i, Oracle Workflow and PL/SQL. Oracle is a registered trademark, and any of these product names, as well as any product features, are trademarks or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

Contents

Declaration	i
Abstract	iii
Acknowledgements	v
Trademarks	vii
1 Introduction	1
1.1 Motivation for this study	2
1.2 Problem Definition	4
1.3 Objectives	6
1.4 Methodology	7
1.5 Layout of the Dissertation	7
2 Workflow	9
2.1 Understanding Workflow	10
2.2 Workflow Standards	13
2.3 Functional Aspects of Workflow Systems	15
2.3.1 Process Design and Definition	16
2.3.2 Process Instantiation and Control	17
2.3.3 Interaction with Users and Applications	19
2.4 Trends in Workflow	20
2.5 Conclusion	23
3 Access Control Requirements	25
3.1 Secure Information in a Workflow	26
3.2 Access Control Service	28
3.2.1 Access Control Service: Administration	28

3.2.2	Access Control Service: Run-time Enforcement	32
3.3	Access Control in Workflow Systems	34
3.3.1	Order of Events	35
3.3.2	Strict Least Privilege	35
3.3.3	Separation of Duty	36
3.4	Conclusion	37
4	The CoSAWoE Model	39
4.1	CoSAWoE Overview	40
4.1.1	The Administration/Design Aspects	41
4.1.2	The Run-time Enforcement Aspects	43
4.2	Commercial Perspective	44
4.2.1	Separation of Duty Administration	45
4.2.2	Worklist Generation	49
4.2.3	Integration with Commercial Systems	51
4.3	Conclusion	53
5	Oracle Workflow	55
5.1	Workflow Terms Used in Oracle Workflow	56
5.2	Oracle Workflow Architecture	58
5.3	Process Design and Definition	61
5.4	Process Instantiation and Control	68
5.4.1	The Workflow Engine	68
5.4.2	The Notification System	71
5.4.3	The Business Event System	72
5.5	Interaction with Users and Applications	73
5.5.1	Viewing Notifications and Processing Responses	76
5.6	Conclusion	79
6	CoSAWoE: Admin in Oracle Workflow	81
6.1	Access Control Features	82
6.2	Specifying Access Control	83
6.2.1	Role-based Access Control at Activity Level	84
6.2.2	Strict Least Privilege at Item Attribute Level	85
6.3	Specifying SoD	86
6.4	Lessons Learnt	89

6.5	Conclusion	90
7	CoSAWoE: Run-time in Oracle Workflow	93
7.1	Worklist Generation in Oracle Workflow	93
7.2	Pruning the User List for DSoD in Oracle Workflow	94
7.3	Session Control in Oracle Workflow	95
7.4	Lessons Learnt	96
7.5	Conclusion	97
8	Insurance Claim Example	99
8.1	Requirements of a Suitable Example	99
8.2	The Insurance Claim example	100
8.3	Administrative Access Control in the Example	103
8.4	Demonstration of the Example	108
8.5	Conclusion	117
9	Conclusion	121
9.1	Revisiting the Problem Statement	122
9.2	Implementation Issues and Lessons Learnt	125
9.3	Future Research	126
9.4	Final Word	127
A	The WFSOD PL/SQL Package	129
A.1	FilterUsers Procedure	129
A.2	DeleteAdhocRole Procedure	132
B	Accompanying Material	135
	References	137

List of Tables

4.1	Static SoD interpretations for the business rule “Auditors should act independently”	47
4.2	Dynamic SoD interpretations for the business rule “An order should not be approved by its initiator”	48

List of Figures

1.1	Subject areas under discussion	5
1.2	Specialization areas under discussion	5
1.3	Layout of the dissertation	8
2.1	Processing a purchasing requisition	11
2.2	Typical components of a workflow	12
2.3	Workflow Management System	14
2.4	Partial process definition for a purchasing requisition process.	16
2.5	Process instantiation and control	18
2.6	Process interaction with users and applications	20
3.1	Role-based Access Control	30
4.1	CoSAWoE: Conceptual view	40
4.2	Dynamically pruning the user list during run-time	50
4.3	Scope of implementing the CoSAWoE model in a commercial workflow product	52
5.1	Oracle Workflow Terms	57
5.2	Oracle Workflow Architecture	59
5.3	Oracle Workflow Builder: Object Navigator and Process Dia- gram windows	62
5.4	Drilling-down into a process activity(“Notify Approver”) . . .	63
5.5	Workflow tables that store process definitions in the database	66
5.6	Workflow tables that store history information of process in- stances	69
5.7	How the Workflow Engine interacts with the Business Event System	73

5.8	Workflow users can access workflow features via a Workflow Homepage	75
5.9	The “Notifications Web page” shows open notifications for the current user	78
5.10	The “Notification Detail” page.	79
6.1	Specifying conflicting entities via database tables in Oracle Workflow	88
6.2	Inserting a function activity in the process definition to enforce DSoD	89
8.1	The Insurance Claim process as shown in Oracle Workflow Builder’s process window	102
8.2	Examples of Insurance Claim process instances	103
8.3	Users for the Insurance Claim example	104
8.4	Roles for the Insurance Claim example	105
8.5	User to Role assignment for the example	105
8.6	Assigning a Performer Role to an activity	106
8.7	Attribute Type settings for messages	107
8.8	Conflicting Tasks for the Insurance Claim example	107
8.9	Conflicting Users for the Insurance Claim example	108
8.10	The Insurance Claim process with SoD function activities	109
8.11	Starting a new Insurance Claim process	110
8.12	Monitoring the status of the Claim via the “Activities List” webpage	111
8.13	The “Prepare Claim” notification appears on Pauline’s worklist	111
8.14	Pauline prepares a new claim by supplying values for the item attributes presented by the notification message	112
8.15	The worklists of Pauline, Kenneth, Ben, Alan and Tom after Task 1 has been completed	113
8.16	The process diagram in the web monitor shows the execution of “Filter Users” before the “Complete Customer Profile” task	114
8.17	A graphical illustration of how the “Filter Users” function prunes the user list before Task 7	115
8.18	Tom prepares his assessor’s report in task 6	116

8.19 Kenneth prepares a customer profile before submitting it for
task 7 116

8.20 The worklists of Alan, Ben, Kenneth and Pauline after task 6
and 7 have been completed. 118

8.21 A summary of the activities that were completed in this example. 119

Chapter 1

Introduction

The corporate landscape has undergone significant changes in the last few decades. Increased globalization, supported by the Internet phenomenon, has forever changed business philosophy and culture. The advent of the Internet and the World Wide Web ushered in a new era, appropriately termed the “Information age”. Modern-day consumers are more “informed” than ever about the products and services that they buy.

They have also become accustomed to a high level of convenience and customer service. Companies have expanded to become virtual corporations who can conduct business on every continent, 24 hours a day, 7 days a week. Therefore, the increased need for speed and distribution has become the driving force for companies seeking to gain a competitive advantage in a global arena (Sheth, van der Aalst, & Arpinar, 1999).

Not just big business, but organizations of all sectors are changing to streamline their operations. Even governmental agencies are starting to realize the potential of using the Internet to provide essential information and services to the people they serve (Caldow, 1999). This transformation process includes business process re-engineering (BPR) and the increased use of automated workflows to facilitate business processes:

“Success in this dynamic business environment demands robust end-to-end business process solutions that are flexible, scalable and adaptive. The critical success factor is Workflow.” (IBM, 2000).

Although the use of technology to enable information-flow brings about several advantages, it also introduces new and challenging problems. Automated

business processes often involve complex routing of tasks not only between humans, but also between different information systems. Furthermore, these workflows may be distributed and implemented on a variety of IT platforms and may have life cycles ranging from minutes up to months.

Some of these problems also manifest themselves in the field of information security. Information assets, like any other asset, must be protected from exploitation by competitors and malicious forces. With the proliferation of electronic information and computer networks, the focus shifted from the physical protection of tangible assets to the logical protection of intangible information assets.

This study was primarily motivated by the need for controlling access to the increasingly important information assets, specifically those under the control of workflow systems.

1.1 Motivation for this study

The research undertaken for this study was motivated by the following realizations.

The realization that workflow is a critical success factor for e-business

With global competition becoming a real threat, organizations need to consider how information technology can be used as an enabling business component. Several trends can be observed when studying information technology over the past decade. These include a widespread adoption of Enterprise Resource Planning (ERP) systems and a strong move to e-commerce (or lately e-business) (Ash & Burn, 2003). The importance of workflow technology in all of these activities is widely recognized. ERP vendors are starting to integrate workflow into their systems and touting it as a key technology in their e-business frameworks (Sheth et al., 1999). Advocates of workflow systems are quick to point out that, if critical business processes are facilitated by workflow systems, a return on investment can be realized in a short time. The biggest contributing factor to this is the saving of time that results from the streamlined information flow (Teng, Jeong, & Grover, 1998; Duchessi &

Chengalur-Smith, 1998; Amoroso, 1998).

However, workflow systems are vulnerable as far as information security is concerned, and their advantages can just as easily be turned to vices if they cannot perform “with the accuracy and assurance that the customer expects” (IBM, 2000). This led to the following realization.

The realization that an access control service is necessary to protect the workflow and the information it manages

In any automated process the need for control and security is paramount. The physical checks that were so common in paper-based systems now need to be replicated electronically. Users of the workflow should not be burdened with the responsibility of ensuring that the business rules are enforced correctly or that the data is kept secure. It is exactly this kind of “red tape” that workflow systems try to avoid.

A security service is therefore necessary to protect the data from a *logical* perspective, as opposed to *physically* securing the transmission and storage of that data via methods such as encryption. Logical protection would entail ensuring that information stay available, confidential and maintain its integrity (Botha & Eloff, 2002). Availability is gained by controlling access to information in such a way that a user’s work is not interrupted. Confidentiality and integrity, which relates to allowing access to information according to the context of the user’s current task in the workflow, is also indirectly attained with the assistance of an access control service.

Information security, including access control in workflow systems, have received much attention from the research community, yet companies have been slow to recognize and adopt the necessary precautions (von Solms, 1999). The secure flow of information in a workflow is not gained without effort. Workflows must be carefully planned to anticipate where and when information in the workflow might become vulnerable. Often, the implementation of much of the desired security requirements, is still left to each individual developer.

Therefore, further investigation into the development of workflow systems and the products available commercially, also revealed the next realization.

The realization that commercial workflow products employed by most e-businesses may not provide adequate access control mechanisms

The main player in the standards arena regarding workflow systems is the Workflow Management Coalition (WfMC). The main purpose of the WfMC effort is to facilitate information interchange between heterogeneous workflow systems. This produced a common reference model for workflow management systems (Hollingsworth, 1995). Unfortunately, no specific reference to security is made. A “Security Considerations White Paper” published 3 years later (Workflow Management Coalition, 1998a) also does not identify access control as a priority. Instead, access control is upheld as a facility that can add value to products, thus serving a product differentiation role.

Therefore, most commercial products, such as IBM WebSphere MQ Workflow, Handysoft Bizflow, TIBCO InConcert, Staffware and Oracle Workflow to mention but a few, include only limited, and at times no distinctly recognizable access control features. Most companies seeking to expand their e-business capabilities via workflow, rely on one of these established workflow products to implement their solution. If access control is considered as a requirement when evaluating their options, they will need a standard model on which to base their assessment. Even if such a model is available, a candidate product which exactly matches this model will be hard to find. Most likely, this functionality will need to be “added on”, or the product customized in order to incorporate the customer’s specific access control requirements.

It is this realization that strengthened the author’s motivation to investigate the considerations, and ease with which an access control model can be implemented in a commercial workflow product.

1.2 Problem Definition

The problem addressed in this dissertation stems from two areas of interest, namely workflow and access control (see figure 1.1).

The initial question asked was, “What role does the access control service play to create a secure workflow?”. This is shown as the overlap of the workflow and access control spheres on figure 1.1. This question has already been

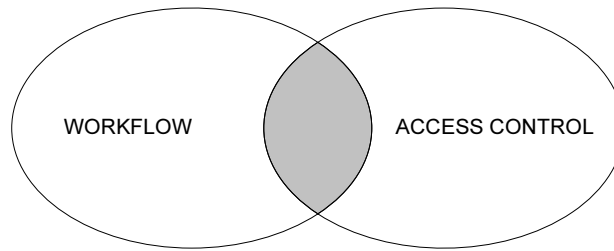


Figure 1.1: Subject areas under discussion

answered by researchers such as Botha (2001), Bertino, Ferrari, and Atluri (1999), and Wei-Kuang Huang and Vijay Atluri (1999), who suggest that the access control service is crucial to protect information from a logical perspective according to the organizational rules of the business. Various models have also been proposed to implement access control requirements. The interest, therefore, is not *what* should be done, but rather *can* it practically be done.

Also, as discussed in the previous section, most companies have realized the need for workflow systems to create effective e-business solutions. However, when choosing a technical solution, they will prefer to use one of the many established products with proven track-records and readily available support, as opposed to in-house development. Therefore the question becomes whether a particular access control model can be implemented, not as part of some specially built prototype as developed with such models, but in a commercially available workflow product. Figure 1.2 indicates that the CoSAWoE model and the Oracle Workflow product, in particular, will be investigated. The integration of CoSAWoE into Oracle Workflow is shown as the shaded area.

This narrows the scope and poses the following question:

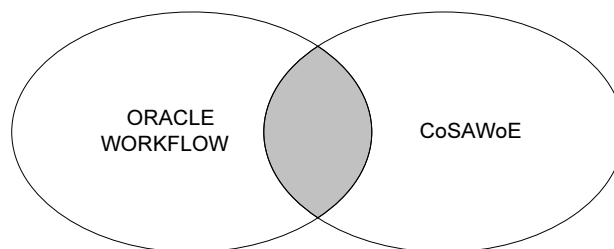


Figure 1.2: Specialization areas under discussion

“Can the CoSAWoE model be purposefully applied to a commercial workflow product such as Oracle Workflow?”

In order to answer the question stated above, and determine whether a “purposeful” implementation is indeed possible, some additional questions will also form part of the discussion:

- What are the functional requirements of an access control service for workflow systems?
- Which aspects of the access control model is product agnostic?
- Which aspects of the model can directly be implemented and which will require customization in order to achieve similar results?
- Which aspects are not implementable for the chosen workflow product?

The next section will discuss what objectives this dissertation will attempt to achieve in order to answer the questions above.

1.3 Objectives

The principal objective of this study is to determine whether a particular access control model, CoSAWoE, can be purposefully implemented in a commercial workflow product.

In order to do this, this study will first aim to integrate existing knowledge in the fields of workflow and access control. In this discussion, the protection requirements of information in a workflow, from a logical perspective, will be established. Subsequently, the suitability of the CoSAWoE model will be investigated, and the components effecting commercial workflow products will be identified. The chosen commercial product must also be dissected, and the features relevant to an access control discussion will be highlighted. A theoretical discussion of these features will ensue to determine whether the objectives of the CoSAWoE model can be met within the chosen workflow product. The suggested implementation will then be demonstrated by building an exploratory prototype. Finally, this demonstration will reveal the difficulties and “lessons learnt” from implementing the CoSAWoE model in a commercial workflow environment.

In order to reach these objectives the following methods will be used.

1.4 Methodology

The primary methodology followed in undertaking this research was a thorough literature study. The literature study examined the problem by investigating the topics of workflow and access control. Attention was especially given to the specialization areas of Oracle Workflow, the CoSAWoE model, and their integration as shown on the overlapping area in figure 1.2.

The aim of this literature study is to reveal the key components of the model that will be applicable to commercial systems. These aspects are also part of a general discussion as to how they can be implemented in one such system, namely Oracle Workflow. In aid of this discussion, a prototype was built to explore how the main services of the model are implemented within a particular workflow scenario. The prototype, built with Oracle Workflow, highlights how the chosen product can be used to enforce certain application-level access control requirements, and what problems were encountered.

The following section will show how the results of the literature study and the exploratory prototype is presented in this dissertation.

1.5 Layout of the Dissertation

The layout of the dissertation is depicted in figure 1.3. Following on this introduction, **Chapter 2** discusses the role of workflow within modern e-businesses and the functionality included with typical workflow management systems. Then **Chapter 3** investigates the access control requirements of information within a workflow environment. The components of the CoSAWoE model are investigated next in **Chapter 4**, with the features of Oracle Workflow discussed thereafter in **Chapter 5**. Both these chapters will concentrate on features which will have a direct bearing on **Chapters 6 and 7** in which the implementation of the model in the chosen workflow product will be discussed. **Chapter 6** will suggest administrative/design steps that must be followed, while **Chapter 7** discusses the run-time execution of the proposed functionality. Thereafter, in **Chapter 8**, a practical example is used to illustrate how a prototype can systematically be built, incorporating the methods discussed in chapter 6, and demonstrating their effects as discussed in chapter 7. Finally, the dissertation is concluded in **Chapter 9** where the lessons

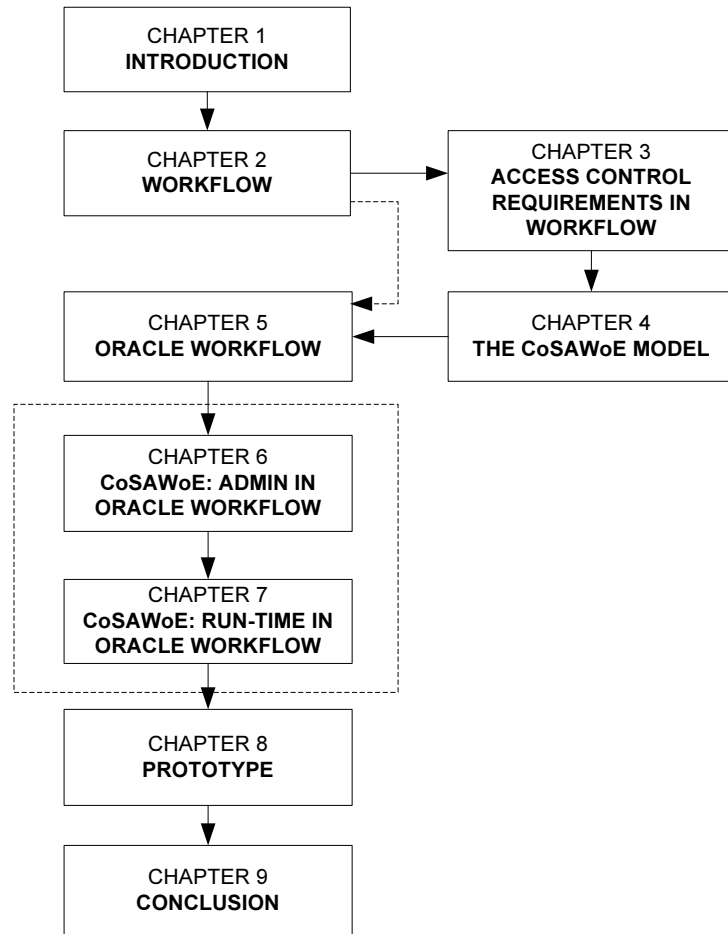


Figure 1.3: Layout of the dissertation

learnt from this implementation effort are presented, and recommendations for further research are proposed.

Chapter 2

Workflow

Workflow systems have proliferated during recent years. Workflow grew out of the need to re-engineer business processes so that the flow of information and tasks between various participants are optimized. The interest in such systems was fuelled by the increasing competitiveness of businesses operating in the global, networked economy facilitated by the Internet. The combination of the information communication capabilities of the World Wide Web, used together with the strategic business process automation capabilities of workflow systems, have spawned many new research avenues as far as e-commerce is concerned.

According to Wei-Kuang Huang and Vijay Atluri (1999), “Web and workflow management systems together serve as an ideal combination to integrate the distributed processes that are across or within enterprise boundaries”. The Internet has indeed brought several advantages to traditional workflow systems, but more importantly, companies who make strategic use of this medium to integrate all areas of their business (and thereby becoming e-businesses), are realizing the importance of workflow as a key technology. Workflow management systems also support the definition, execution, controlling and documentation of the business processes automated by such workflows.

This chapter will conceptualize workflow and workflow management systems. Extensive research have already been done in this field and groups such as the Workflow Management Coalition (WfMC) (Hollingsworth, 1995) have attempted to standardize many aspects of such systems. The following sections will explore some of these definitions and standards, culminating

in a discussion of the functional aspects of workflow systems. Towards the end of the chapter, the relationship between workflow and the Web will be explored. First, however, it is necessary to understand the basics of a typical workflow.

2.1 Understanding Workflow

In order to streamline their operations, many businesses have become more and more process-focussed. BPR has fuelled the need to identify and understand critical business processes. These processes are often routine in nature, such as reordering stock, and will typically follow the same sequence of steps every time the process is repeated. Decisions will be made by various people along the way until the desired result is achieved. In many cases, the whole process will be paper-based, with the completion of a form triggering the next step in the process (Leyman & Roller, 2000). This system of “pushing paper across desks” is still in use in many organizations today. However, information technology offers an alternative, where processes can be controlled by a computer system and information can flow from point to point in an instant.

The concept of workflow¹ is best described by the Workflow Management Coalition (1998b): “The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.” There are various types of workflows, including image processing, groupware applications, transaction-based applications and project development systems (Hollingsworth, 1995). However, most formal business tasks are based on, or are driven by, document flows (Sprague, 1995).

Document-based workflows are concerned with completing electronic documents by routing it to the various people concerned so that they can access and update their specific parts before sending it along. These electronic documents often resemble paper-based

¹The term “workflow” can apply to any business process, whether it is completely computerized or not. However, for the purposes of this dissertation it will be used in the context of an automated process.

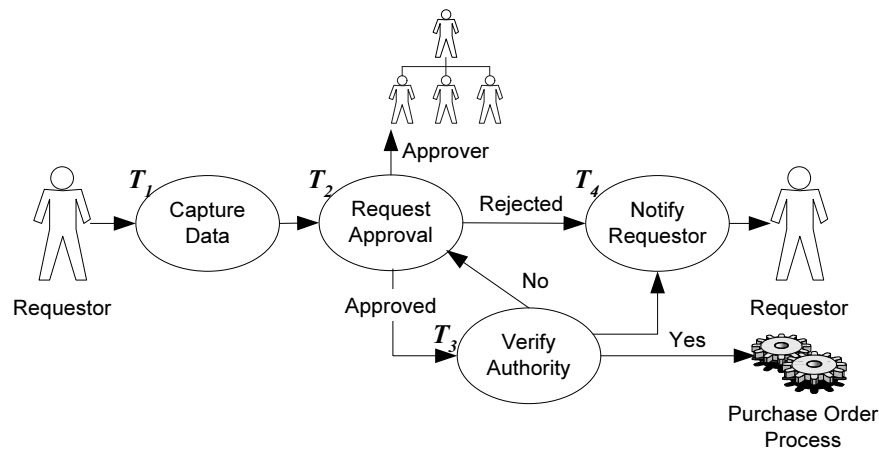


Figure 2.1: Processing a purchasing requisition

forms, with various data fields organized into some structure (Bae & Kim, 2002).

Consider a simplified purchasing request. A requestor will fill in all the relevant details pertaining to the items requested on a paper form. This information will have to be captured using some electronic document. The request will then need to be evaluated and approved by a manager with the right level of authority. If it is approved, the request will likely become a purchase order which will in turn be processed by another workflow process.

Although there are many types of workflows, and just as many ways of modelling them, most representations will include these three basic components:

- *tasks* or activities that have to be done;
- human *users* or *applications* that are needed to complete them;
- and *rules* that will determine which tasks to do next.

These concepts can be illustrated in the context of the purchasing requisition example discussed earlier. Figure 2.1 shows four tasks or activities numbered T_1 to T_4 . T_1 is initiated by the requestor who completes and submits an electronic purchasing requisition form. The form is then forwarded to the approver for approval, as shown by T_2 . The approver is selected according to some management or role hierarchy. Details of this user selection for each task will form part of the access control discussion later in chapter 3.

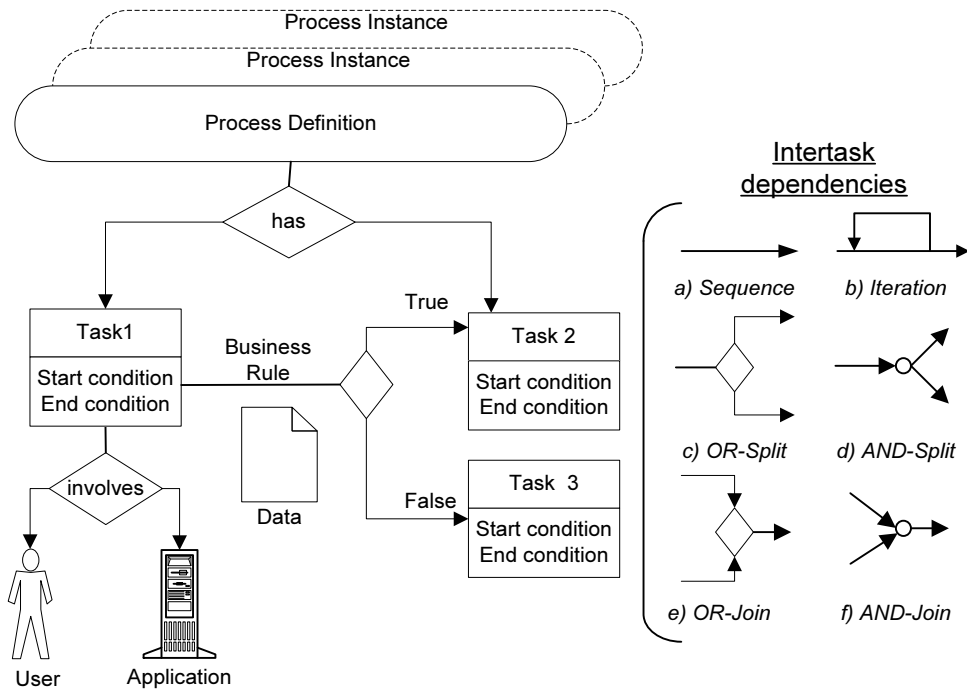


Figure 2.2: Typical components of a workflow

From T_2 the workflow can follow different paths. If the claim is approved, the next task (T_3) is to verify that the approver has the authority to approve the requisition amount. If the approver does not, the form is looped back to another approver further up in the management hierarchy until a manager with the correct level of authority approves or rejects the requisition. Whether the purchase requisition is approved or rejected, a message is sent to the requestor notifying him or her of the decision. This is shown as T_4 on figure 2.1 on the page before. From here the workflow can join up with another process to generate and fulfil the purchase order for the approved requisition. Notice that a requisition can only continue to the purchase order process if it was approved by an authorized approver. This represents a business rule for the processing of a purchasing requisition.

The example just discussed is quite simple, but business processes that are automated often involve complex routing of tasks not only between humans but also between different information systems. Furthermore, these workflows may be distributed and implemented on a variety of IT platforms and may have life cycles ranging from minutes up to months (Hollingsworth, 1995).

Figure 2.2 on the facing page gives an overview of the typical components of a workflow including tasks, business rules, data in the form of documents, users and applications. It also shows the different intertask dependencies that will determine the flow of data from one task to another (Workflow Management Coalition, 1996). These can broadly be categorized into sequence, iteration, split and join dependencies. “OR” splits and joins mean that only one of the defined paths can be followed by a process instance. “AND” splits and joins facilitates what happens when two parallel paths are followed simultaneously by the same process instance. At any one time there can be several instances of the same workflow process running, each with its own data and path through the process (indicated by the dotted lines in figure 2.2 on the preceding page).

Workflow management systems (WfMSs) provide tools to support the definition, administration and monitoring of workflow processes. Using workflow systems to automate and monitor processes effectively in modern organizations increases productivity, improves quality and customer service, and enhances operational control (Workflow Management Coalition, 1998b).

All workflow systems exhibit certain common characteristics, which will be described in more detail in section 2.3. The WfMC has developed a workflow reference model with the aim to ensure integration and inter-operability between workflows implemented on different systems. It also forms a consistent starting point from which to discuss workflow management systems, since most workflow systems will conform to this model.

2.2 Workflow Standards

The Workflow Management Coalition (WfMC) is an international standards organization whose aim it is to establish standards for workflow management systems. By defining the commonalities between these systems, an architecture can be developed that will be independent of the different implementation methods. This will allow greater flexibility and interoperability between different workflow management products.

The efforts of the WfMC has already produced a Workflow Reference Model (Hollingsworth, 1995) which defines the basics of workflow management, a reference architecture and the interfaces between different compo-

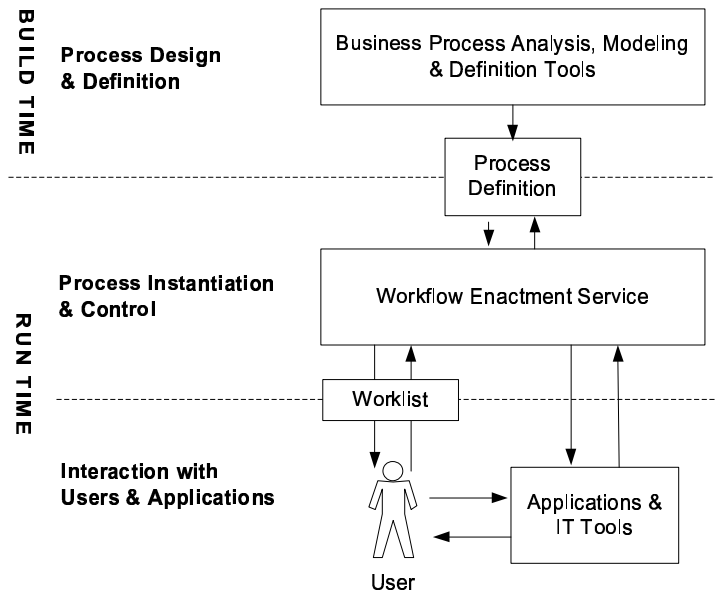


Figure 2.3: Workflow Management System (based on Hollingsworth (1995))

nents of that architecture. It places a significant amount of emphasis on the interfaces, since they will have the biggest impact on interoperability between components. For the purposes of this dissertation, the detailed description of each interface and component is unnecessary. The simplified diagram shown in figure 2.3 will be used as a basis for section 2.3 (functional aspects of workflow systems).

However, no one standard can be sufficient to standardize all aspects of a typical e-business workflow process. Other standards have also been developed, most notably the OMG's Workflow Management Facility standard (or *jointFlow* as it is also known). This standard is based on the WfMC's standards, but presents them as a unified object model (Siemens Nixdorf Informationssysteme, 1998). This facilitates a modular approach when dealing with distributed workflow applications. *JointFlow* allows interoperability not only between workflow applications at a higher process level, but also at component level, including process monitoring, execution and resource handling. It also provides the opportunity to develop more specialized standards which concentrate on certain functionalities. Two such standards are the Simple Workflow Access Protocol (SWAP) and the Wf-XML message set.

The SWAP standard focusses on process execution, from the instantiation

of process instances, to the monitoring and control of each individual instance (Swenson, 1998). Furthermore, it proposes the use of the HTTP protocol as a vehicle for XML messages to travel between workflow components. This is purposely done to ensure integration of distributed workflows on the *Internet*. SWAP was first introduced in December of 1998, and has since been used by in the U.S. Department of Defence and IBM's MQseries workflow engine prototypes.

Crucial to the SWAP standard is the use of XML encoding to send messages between components. This led to the development of the Wf-XML specification, a subset of SWAP (Hayes et al., 2000). It provides a few advantages over its predecessor, like the independence from transport mechanisms. Although the HTTP protocol has been accepted by the WfMC as the core transport mechanism to send and receive Wf-XML messages, solution providers are free to choose their own method for delivering these messages (like e-mail or CORBA for example). The transport mechanism would be specified in an interoperability contract. This contract between systems is key to the interoperability of Wf-XML messages since it defines each system's expectations and requirements. It would typically include information about connectivity, security, the different process definitions and their formats. The Wf-XML specification was released as an official WfMC standard in May 2000 (Workflow Management Coalition, 2000).

All the standards referred to above were developed with one aim: to create interoperability between workflow products from different vendors running on different platforms. As mentioned previously, a detailed explanation of how this interoperability is achieved will have no bearing on this dissertation. More importantly, it is necessary to understand the common functional aspects of typical workflow systems.

2.3 Functional Aspects of Workflow Systems

The architectural model as described by the WfMC (Hollingsworth, 1995) focusses on gaining compatibility between heterogeneous workflow environments via five different interfaces. However the simplified diagram, shown in figure 2.3 on the preceding page, shows more clearly the three distinct functions of a workflow management system. These functions also refer to the

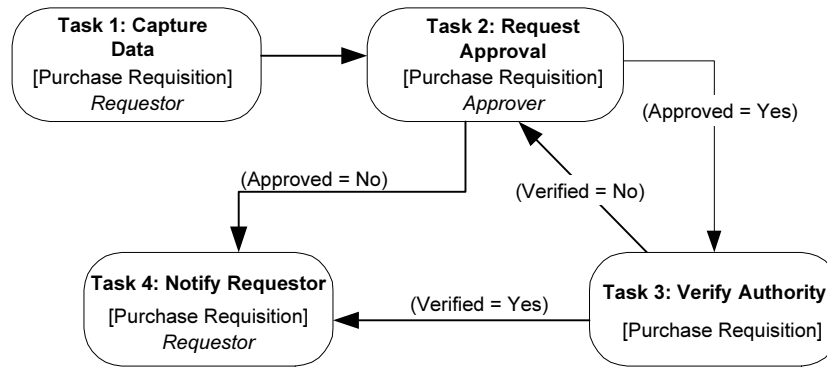


Figure 2.4: Partial process definition for a purchasing requisition process.

stages a typical business process would follow when facilitated by a workflow management system.

Firstly, the process is defined and formatted so that the computer can understand it. The output from this stage is a process definition document which is then interpreted during run-time to instantiate different process instances. During the execution of each instance, the workflow management system will route tasks and data between various participants in the workflow based on conditions encountered along the way. The workflow management system will also maintain workflow related data as well as control data for each instance.

The following discussions will reveal more detail about the various components of a workflow management system that are needed to perform each function.

2.3.1 Process Design and Definition

The WFMC refers to this stage as “build-time” (Hollingsworth, 1995, p. 7). During this phase the workflow, as explained above in conceptual form, must be designed using analysis and modelling tools. These designs are then implemented in a format the computer can understand, called the Process Definition. This process definition will typically include task definitions, references to outside programs and/or participants, navigation rules and business rules. A process definition tool can be used to capture these definitions using a process definition language, object-orientated models or scripts.

Figure 2.4 shows a partial process definition for a purchasing requisition

process based on the notation used in Botha and Eloff (2001a). Each task on the diagram is divided into three sections: 1) the task description in bold, 2) the document being used is placed in square brackets, and 3) the user or application program involved in completing that particular task is shown in italics.

Most tasks follow a straight sequence from one task to the next specified task in the process definition (e.g. from capturing the requisition data in task 1 to performing the approval in task 2). However, as shown in the diagram, there are cases where the flow can be split in two directions depending on whether a condition is satisfied or not. Tasks can also be executed in a loop until an exit condition is satisfied. These business rules/conditions are shown in rounded brackets on the connecting lines between tasks. For example, depending on whether the purchase requisition is approved or not, the requestor will either be notified straight away of the rejection or it will be routed to some internal function checking the approver's authorization to approve the amount. Here it enters a loop, where the purchase order is repeatedly approved by managers in the hierarchy until a manager with the suitable authority is found.

This example can obviously be expanded to include several more tasks, documents and participants, but it is not required to illustrate the concepts discussed later. As can be seen from this diagram, the process definition serves as a template for all possible routes through a certain process. The next section, process instantiation and control will describe how each individual process instance will be interpreted and directed according to the process definition.

2.3.2 Process Instantiation and Control

During "run-time" several background processes must occur to create, control and monitor the workflow. These are executed on the server-side and is not seen by the user. Central to this phase is the workflow enactment service (see figure 2.3 on page 14). This service creates a new process instance each time a user starts a new workflow process. It then interprets the process definition and uses it to execute and control tasks that form part of that process instance. Each task will go through several states and as one finishes, the enactment service will determine which task to trigger next. Figure 2.5

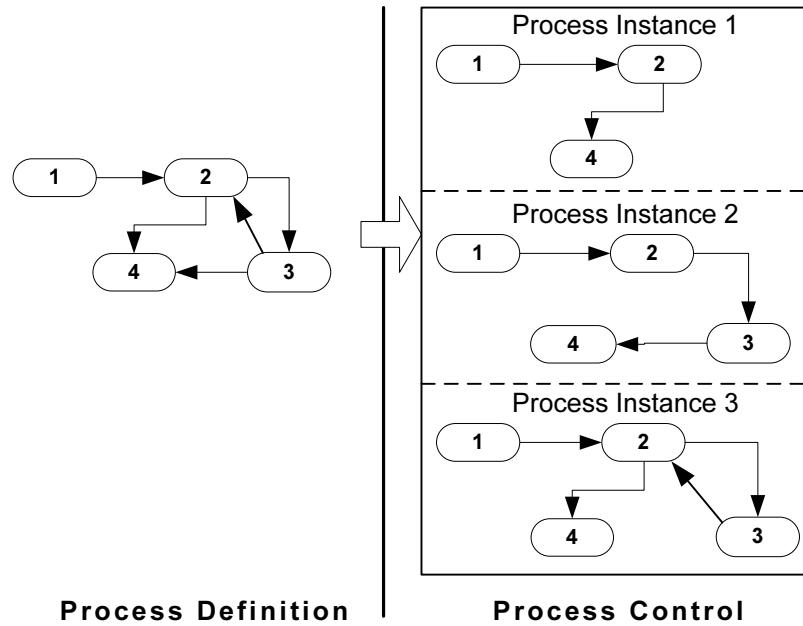


Figure 2.5: Process instantiation and control

shows how the process definition is interpreted differently for each process instance, by creating task instances only as they are needed. Process instance 1, for example, does not complete task three, but rather goes directly to task four. Both process instance 2 and 3 trigger all tasks to be executed, but in a different sequence resulting in different end results. In process instance 2, the requisition is approved the first time around by an approver with the correct authorization level. Process instance 2 enters the loop back to task 2 when the first approver's authority is not sufficient for the successful completion of the process. The process exits the loop when the requisition is subsequently rejected by an approver with a higher level of authority.

Throughout the workflow data will be accessed and updated by users and applications. The workflow enactment service must manage the relevant data related to each process instance. It is also important to note that tasks do not always occur in a neat uncomplicated sequence. They may occur in parallel, be split between participants or share the same resource, as was suggested in figure 2.2. The workflow enactment service must be able to navigate all of these flow structures for each separate process instance. In addition, a process can be distributed between several enactment services and can often take several days, if not months, to run to completion. In view of these complexities, control data, security logs and recovery information gain new

importance. The enactment service will keep its own internal control data associated with each process instance. This enables the enactment service to recover data and continue processing if the process is interrupted for any reason.

2.3.3 Interaction with Users and Applications

As mentioned previously, the workflow will often require some involvement from users or external applications to complete certain tasks. This happens on the client-side, and it is the only time when the users are allowed to interact with the workflow system, with the exception of the administrator who has supervisory privileges to alter tasks and monitor activity. In order to interface with users, the enactment service will place tasks on each user's individual worklist (as shown by the worklist for the approver in figure 2.6 on the following page). A worklist handler will then place the tasks according to priority or sequence onto the user's desktop. Using the push method, the worklist handler will schedule tasks in the order that a user *must* perform them. However, sometimes the pull method is used, which presents the users with all the tasks that they must perform and then lets them *choose* the order in which they want to perform them. As the user completes a task it is removed from the user's worklist.

External applications are often invoked indirectly via the user interface, for example when a user needs to complete a form using a spreadsheet or word processor. However, sometimes it is also necessary for the enactment service to directly access applications, for example when the rejection notification is sent via an e-mailing system or when the approved purchase requisition is uploaded to a persistent document store. The enactment service will also need to integrate the workflow process with external business applications, triggering a task instance based on the occurrence of some business event, or allowing such applications to access process instance data.

Now that the functionality of a typical workflow has been described, the following section will examine the recent move of workflow technologies to the Web.

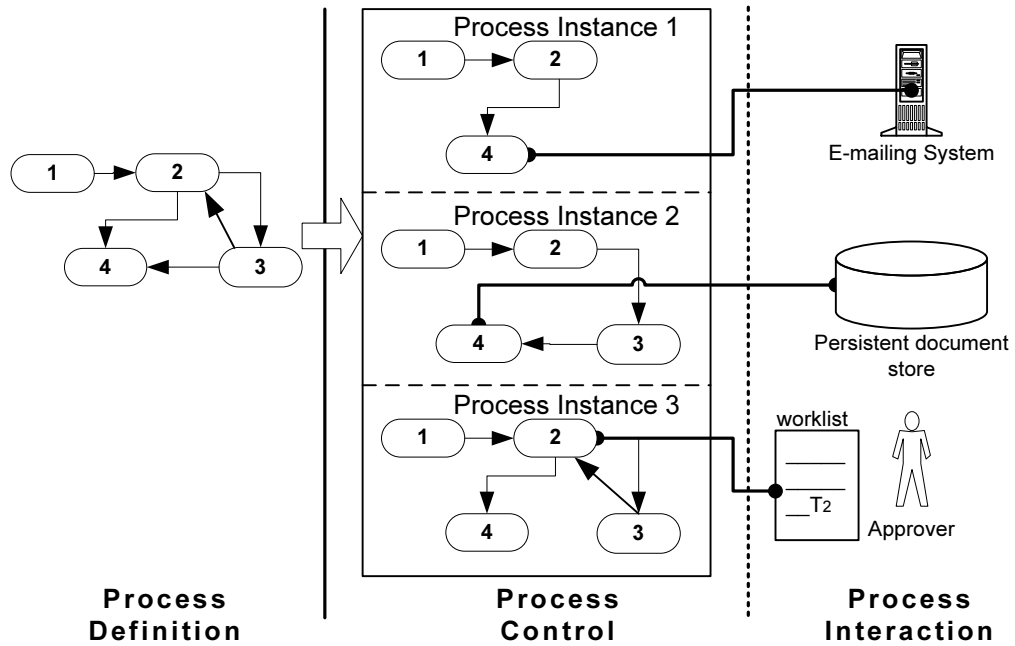


Figure 2.6: Process interaction with users and applications

2.4 Trends in Workflow

The introduction to this chapter identified the need for speed and flexibility as the main reasons for the use of workflow systems to streamline a company's business processes. Many e-commerce applications rely on workflow management systems to effectively and reliably process consumer requests in the background, while providing a virtual storefront to potential customers on the World Wide Web. The business-to-business (B2B) model for e-commerce is also seeing companies forming short-term casual trading relationships to facilitate a particular process. Therefore, workflow systems must be able to cross company boundaries to build virtual enterprises (Workflow Management Coalition, 1998b).

This trend towards inter-organizational, heterogeneous workflow environments is driven by more than just the technical requirements for different systems to communicate workflow data. According to van der Aalst (1999), providing interoperability between the different "ways of doing business" is just as important. The WfMC has been working on standards to provide interoperability between workflow products from different vendors. To this end they have just released a draft version of their updated Wf-XML standard based on ASAP (Asynchronous Services Access Protocol) to facilitate

message sending of workflow control data between systems (Workflow Management Coalition, 2004). In addition, a new standard called XML Process Definition Language (XPDL) (Workflow Management Coalition, 2002) was released to address the problem of transporting the process design (the “way of doing business”) across different workflow products, or from specialist modelling tools.

Kim, Kang, Kim, Bae, and Ju (2000) proposed the “WW-Flow” system, which takes a modular approach to support flexible workflow management in a distributed heterogenous environment. Their system allows different workflow engines to interoperate using nested process models and runtime encapsulation: “Run-time encapsulation facilitates cooperation among different departments and organizations, and it improves WfMS scalability by enabling distributed workflow engines to manage subprocesses independently.” Manolescu (2001) also advocates a modularized architecture for workflow systems, although his solution was developed to aid object-orientated developers in creating customized workflow solutions for integration with their existing OO applications. Nevertheless, adopting an object-orientated or modular architectural style has many other advantages. Separating workflow functionality into a series of components is also in keeping with the recent trend towards service-based development (i.e. providing workflow services as interchangeable software components) (Gottschalk, Graham, Kreger, & J.Snell, 2002). Such web-enabled workflow services utilize Web technologies such as XML (eXtensible Markup Language) and Java to take full advantage of the Internet’s platform independent, distributed architecture for a workflow enactment service (Kim et al., 2000).

However, the Web has probably had the most significant impact on the user interaction side of workflow. Client interfaces are utilizing traditional Web features such as hypertexts and Internet protocols such as HTTP, which means that workflow users have ready access to the workflow system from anywhere in the world. The Web has become the ubiquitous user interface for most computer systems, and workflow systems are quickly following suit (Brambilla, Ceri, Comai, Fraternali, & Manolescu, 2002). Administrators now also have a convenient and familiar interface to track and monitor individual process instances, review resource productivity and analyze work volume and performance issues from any location. It has even made it pos-

sible to provide an on-line, self-service functionality for a company's clients who want to query the status of their requests. The Internet has also provided various platform independent ways of exchanging process data. EDI is a common business-to-business tool for exchanging transaction data. However, XML documents provides a more flexible method to exchange process data which is more secure than using the more traditional HTML pages (Workflow Management Coalition, 1998b).

The shift toward becoming more streamlined e-businesses have put corporate executives under pressure to ensure they coordinate their resources, disparate applications and business activities as well managed processes. As such, another major trend has seen workflow vendors integrating their products into mainstream technology for enterprise application integration (EAI) (Moore, 2000). EAI systems, such as inventory control, human resource management, customer care, manufacturing and accounting systems, are integrated solutions that span multiple departments. Workflow is seen as an integration tool for such systems and is used to drive data across functional boundaries and manage a common information set. Increasingly, workflow systems are being marketed as the core component of EAI suites featuring these packaged solutions, rather than stand-alone products. Workflow systems are also being used to gather statistical workflow data for decision making purposes. Bonifati, Casati, Dayal, and Shan (2001) suggests that information can be extracted from the the process instance history and placed in a data warehouse. This will allow managers to utilize the workflow data through a decision support systems and to identify trends in the business process.

This section discussed several trends relating to how workflow management systems are being used, their architectures and related technologies. The following bulleted list summarizes these trends:

- Workflow systems should be able to support distributed, heterogenous workflows that span departments and allows for inter-organizational processes. As such, a high level of technical and functional interoperability between different workflow products are required. XML is increasingly being used by business partners to exchange information about their business processes in an automated way (van der Aalst & Kumar, 2003).

- Workflow management systems are increasingly being developed according to a modularized approach. Components are developed using OO techniques (Manolescu, 2001) and some are used as interchangeable web-services (Nanda, Chandra, & Sarkar, 2004).
- Web-based user interfaces are becoming the norm. Clients, end-users, administrators and managers can initiate, monitor, control, query and respond to workflow data from any location via a standard web browser (Lu & Chen, 2002).
- Workflow systems are being used less as stand-alone products and more as background products that are seamlessly integrated into packaged EAI systems that service multiple business units (Moore, 2000; Wu, Deng, & Li, 2004).
- The potential for workflow systems to deliver strategic business information has been recognized and workflow data is now also being utilized within data warehouses (Bonifati et al., 2001).

2.5 Conclusion

Today's business enterprises must deal with global competition, reduce the cost of doing business, and rapidly develop new services and products. This chapter introduced workflow as a means to automate and support evolving business processes. Processing information in the form of paper-based documents is often a major part of most business processes. Workflow aims to reduce the time wasted while filling in these forms and waiting for them when they are delayed at some point in the system.

As discussed, a typical workflow process is comprised of *tasks* that must be completed by either human *users* or external *applications* according to a set of business *rules* which determine the flow. Automated workflow systems must be able to handle many of these processes that can be nested within one another, that may involve hundreds of users and other information systems, and that may have a time-span ranging from minutes to months. This complexity necessitates the use of a comprehensive workflow management system and close adherence to several standards recommended by the Workflow Management Coalition.

The functional aspects of workflow management systems formed the crux of this chapter. Firstly, business processes must be captured using process definitions modelled during build-time. These process definitions are implemented by the workflow enactment service during run-time as different process instances. Each of these process instances represent a unique flow of information through the possible routes set out in the process definition. The flow will be influenced by internal workflow control data and variables as well as external users and applications. The interaction with users is an important function of most workflows. It is typically facilitated by a worklist handler, which is responsible for managing how users complete their assigned tasks.

Workflow is no longer the sole domain of traditional ERP systems that make use of EDI to exchange information between static, well-defined business entities. The emergence of electronic commerce has called for workflows that span dynamic, “virtual” enterprizes. Workflow management systems are in existence, and are still emerging, that take advantage of the many possibilities offered by the Internet and the WWW. Web-based workflow inherits all the advantages of the web such as the standard client, the ability to run on heterogenous systems, and the use of all the supporting technologies, such as XML for information exchange. However, web-based workflow also inherits all the security threats that web-based application suffer from. Moreover, workflow systems suffer the typical security threats that come from insiders. The access control service, discussed in the following chapter, will therefore prove crucial to ensuring that users’ access to workflow tasks and information is regulated according to business rules.

Chapter 3

Access Control Requirements in Workflow

The previous chapter discussed the use of workflow systems to facilitate the flow of information throughout and beyond the organization. Information is seen as a valuable asset by most companies operating in the current “Information Age”. Increasingly the Internet is being used to facilitate communications and the flow of information between distributed business functions. Workflow systems have not been unaffected and both internal functions such as asking for vacation leave or external functions such as requesting a loan from a bank are being automated by these systems through Web interfaces. However, exchanging business-critical information over such an insecure medium raises serious information security concerns. Workflow systems often target the core business processes of the organization, and as such the information it manages is sensitive to the organization and its clients.

Information security in workflow systems can be modelled on the five security services suggested by ISO8498-2 (ISO, 1989) in much the same manner as other systems making use of the Internet. The ISO8498-2 standard proposes the following information security services: identification and authentication, authorization (access control), confidentiality, integrity and non-repudiation. For businesses to make effective and safe use of a workflow system, the system must provide mechanisms for each of these services in the appropriate position in the workflow environment.

Three of the services mentioned above, namely identification and authentication, confidentiality and non-repudiation are implemented similarly in

workflow and non-workflow environments. As such these services, and their associated mechanisms such as encryption and digital signatures, will not receive as much attention in this chapter. They provide a workflow environment in which the data can be physically transported with relative confidentiality. More important in a workflow is the need to protect the information from a logical perspective. An access control service provides such logical protection. It can also be argued that the access control service assists with integrity in the sense that it prevents users from changing data for which they do not possess modification privileges. Therefore, the access control service will be the main focus of this chapter.

3.1 Secure Information in a Workflow

In order to secure information in a workflow, it is first necessary to understand exactly what the potential threats are. Workflow systems typically operate in a distributed manner, with participants accessing workflow tasks and information over a company wide network. Many workflow systems are also now making use of the Internet to allow its users access from any location via a web-based interface. Information must therefore be secured from threats outside as well as inside the organization. Some of the threats suggested by Stallings (1995, p. 7) are:

- *Modification*: An unauthorized party gains access to valuable data and modifies that data. An example of this could be the alteration of the contents of messages that have been transmitted across the network.
- *Fabrication*: An unauthorized party adds fake objects to the system. For example, when fabricated messages are sent across the network.
- *Interception*: An unauthorized party gains access to an asset of the system. An example would be the monitoring of messages on the network.

All three of these threats could come from inside as well as outside the organization. Protecting information in the workflow from outside threats are discussed by Valia and Al-Salqan (1997). They propose the use of the Secure Sockets Layer (SSL) and a Public Key Infrastructure (PKI) to create a

secure web-enabled workflow environment. What is more difficult to achieve is the protection of information within this secure workflow environment from a logical perspective. This would entail ensuring that information is accessed according to the organizational requirements of the organization. Here we could introduce the threat of fraud:

- *Fraud*: An authorized party executes actions on the system that violates business rules in order to benefit that party. An example of this would be when person creates a request and by-passes the approval hierarchy to sign his own request.

Although fraud is a major concern, other user actions (some quite accidental in nature) may also compromise the integrity of data within a physically secure workflow environment. Whatever the threat, information in a workflow needs to stay available, confidential and maintain its integrity (Botha & Eloff, 2002). Availability is gained by allowing access to information in such a way that a user's work is not interrupted. However, it should also be kept from unauthorized users, i.e. it should be confidential. Confidentiality thus implies that the information accessed by users are restricted according to its context within the workflow. Lastly, integrity of information should be ensured by controlling what changes are effected, when, and by whom.

Leyman and Roller (2000) name three kinds of integrity. Firstly, *physical integrity* is concerned with ensuring that the information is not altered either at the storage points or during transmission. Secondly, *operational integrity* is concerned with issues such as concurrent updating. Thirdly, *semantic integrity*¹ states that the alteration of data during a specific task should be consistent with business rules. For example, a business rule might state that "a person may not approve his or her own leave application" or that "purchase orders above a certain amount can only be approved by managers with a certain level of authority".

Availability, confidentiality and integrity is often indirectly attained with the assistance of the access control service. Access control is by no means a "cure-all" for the threats posed to information within workflows, but it does form the basis of a secure workflow and it is also a fundamental concept

¹From this point on in the dissertation the term *integrity* will refer to the *semantic integrity* of the information as described.

of this dissertation. It has been investigated in depth through the work of Botha and Eloff (2001b), Wei-Kuang Huang and Vijay Atluri (1999) as well as Bertino et al. (1999). The following section highlights some of the salient points of a general access control service. Thereafter, the use of such a service in workflow management systems will be investigated further.

3.2 Access Control Service

Access control is a complex security mechanism with various permutations. At its core, it controls access to tasks and data based on a user's permissions associated with a particular object. *Users* may be actual people or software agents (sometimes called alter-egos) acting on their behalf (Gudes, van de Riet, Burg, & Olivier, 1997). *Objects* represent anything of value that requires protection and forms part of an information system. Documents, directories and database records are examples of objects. The access *permissions*² may be specified according to the semantics of the objects that it relates to. For example, the access permissions associated with an account object may be debit and credit, whilst a file may be read, write or delete. Different methods can be used to map access permissions between users and objects statically during administration as well as dynamically during execution.

3.2.1 Access Control Service: Administration

von Solms (1999) stated that “technical security controls alone cannot enforce a secure IT-environment, it needs to be supported by proper operational controls”. From an access control perspective this means that restricting access to objects through technical measures at run-time can only be effective if sufficient operational policies are in place. These policies will specify the procedures to be followed and the requirements to be met when implementing and deploying the technical measures of an access control service.

An administrator's first step in implementing an access control service, is to decide what users may and may not do while in the system, and what

²The terms “privileges” and “rights” are synonyms sometimes encountered in other literature for “permissions” as used in this dissertation

strategy he or she is going to follow in order to restrict their actions. Recall that “permissions” govern what may be done with a particular object. Permissions may then be allocated to users to allow them a certain “scope” within which to operate on objects. The allocation of access permissions to users during build-time is governed by three *organizational policies*. Organizational policies describe the “normal operating procedures” of the company. The first policy will determine how permissions are specified and administered. Here a *discretionary*, *mandatory*, or *role-based* access control administration paradigm can be used. The other two policies, *least privilege* and *separation of duty*, allow the administrator to evaluate and refine the combination of permissions users may receive based on their job function and access to other objects.

Administration Policy

Classically, administering access control to objects are specified as being either discretionary or mandatory.

A **discretionary** administration policy is based on the premise that the owner of the object controls the access to the object. Such an owner is typically interpreted as the person who created the object (Lamson, 1971). However, in most organizations the object actually “belongs” to the company and may be the responsibility of several employees. Therefore, even though a specific employee created the object it does not necessarily mean that he or she will have sole responsibility for granting access to that object through-out its use. This results in the administrator granting “owners” the right to pass on their access and granting rights to other users who can then in turn propagate their rights to other users at their discretion. From an administrative point of view, this type of owner-centered administration policy makes it difficult to know and control exactly who has which permissions to which objects and when they will be used.

Mandatory administration policies address the issue of who has access to what by imposing strict information flow. This information flow is based on the premise that all users and objects are classified and assigned an appropriate security label by system administrators. Information flow is then restricted according to a set of strict rules on the resultant lattice of security labels. For example, a user with the security label of “Secret” may access

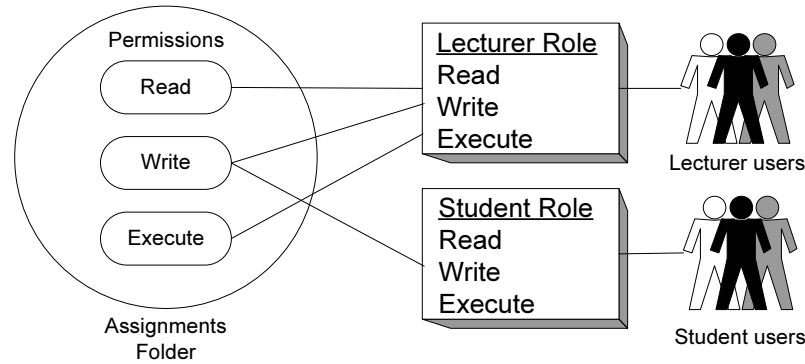


Figure 3.1: Role-based Access Control

all objects with a similar “Secret” label or lower. No restriction is placed on when or where a user may do so as part of his or her job description. Also, the level at which these labels are specified and the terms in which they are expressed can often be very generalized and vague. Therefore additional policies may be needed to put more specific constraints on user access.

From the previous discussions it is clear that both these classical administration policies have functional as well as administrative deficiencies when applied in a commercial environment. Both discretionary and mandatory access control policies are expressed from an object-perspective. An alternative is to administer the environment from a user-perspective.

Role-based Access Control (RBAC) provides the use of a *role* to create a level of abstraction between the specific users of a system and the permissions they have to access objects of that system. A role-based administration policy acknowledges a user’s job function and responsibilities by associating a user with a role, which is then in turn associated with a set of permissions necessary to perform a specific set of tasks. Users can be assigned to one or more roles as the need arises. Permissions can also be attached to more than one role. In addition, permissions may either be specified as positive or negative for a particular role. If a role receives a positive permission for an object, it means that that access will be granted accordingly. If a negative permission is assigned, users belonging to that role will be specifically barred from performing the action. Therefore, various organizational policies can be enforced by constraining the associations between users and roles, and between roles and permissions.

In figure 3.1 the folder “Assignments” is an object in a system. This

folder has permissions to view, write or execute file objects contained in the folder. Separate roles are defined for “Student” and “Lecturer” users. The “Student” Role will receive permissions to write files to the folder thereby enabling students to upload their assignments, but not view each other’s assignments. The “Lecturer” role will receive all three permissions for the folder.

To ease the administrative burden, roles are structured in a role-hierarchy and permissions are inheritable upward in the hierarchy with parent roles having the accumulative permissions of it’s subordinate roles. This role-hierarchy often reflects the management structure of the organization (since individual roles represent job functions). In the previous example, the “Lecturer” role can fall under another role “Head of Department”. This would mean Heads of Department will have the same permissions to the folder object as normal Lecturers in addition to any permissions of their own.

Least Privilege Policy

Least privilege is an access control policy whereby permissions are assigned selectively to users in such a manner such that no user is given more permissions than is necessary to perform his or her job (Ferraiolo, Barkley, & Kuhn, 1999). A manager, for example, may only view the salaries of staff reporting to him or her. The least privilege policy avoids the problem of an individual having the ability to perform unnecessary and potentially harmful actions merely as a side-effect of being granted certain general permissions. Ensuring adherence to the least privilege policy is largely an administrative challenge that requires identification of job functions, specification of the minimum set of permissions required to perform each function, and restricting the user to those permissions and nothing more. With role-based administration this is typically done by associating permissions with a role that describes a specific job function. Users can then be associated with the role that describes their job function.

Separation of duty policy

Separation of Duty (SoD) policies are aimed at the prevention of fraud. In principle SoD policies attempt to reduce the likelihood of collusion by distributing the responsibilities for tasks in a business process between multiple

participants. This distribution of responsibilities could be achieved in a number of ways. Literature makes a distinction between static and dynamic SoD principles (see, for example, Ahn and Sandhu (1999); Nyanchama and Osborn (1999) as well as Simon and Zurko (1997). Static SoD principles govern the administration/design-time associations between users and permissions, while dynamic separation of duty principles govern the way in which permissions are granted at run-time.

An access control service restricts a user's access during run-time dependent on the administration/design-time specification. The next section, therefore, considers the run-time enforcement aspect of access control.

3.2.2 Access Control Service: Run-time Enforcement

At run-time the access control service uses information from the underlying environment and the administration/design-time specification of access control policies to govern a user's access. As a minimum requirement the access control service needs the user identity and the object identity. Depending on the administration policy the access control service would use other information. For example, when a mandatory administration paradigm is ascribed to, the security labels will represent essential information.

Based on the required information, the access control service must be able to answer a variety of access control related questions. For example, when an access request is made, the access control service needs to answer a question of the form "May user u perform permission p on object o ?". For this question to be answered the user must be active in a session and the relevant access control policies must be evaluated.

Determining permissions in sessions

An access control decision must be enforced by the access control service. Before any access control requests can be made to the access control service a user must first initiate a "session" to establish the user's identity for the duration in which request are made. A session is typically started when the user authenticates him or herself through the use of a username and password that are supplied by the user when he/she signs on to a system.

Users typically receive permissions for the entire duration that they are

signed on until such time that they end their session by signing off from the system. Some systems, such as relational databases that support role-based administration, may allow a user to activate/de-activate permissions based on the activation/de-activation of a specific role.

The permissions that are active within a session should be determined by evaluating the access control policies that are in place.

Evaluating Access Control Policies

Certain access control policies are mainly enforced by careful administration. Least privilege, for example, is enforced by ensuring that users are not assigned permissions that are not required by their job function. The separation of duty policy can also be enforced statically by assigning permissions to users in such a fashion that they will not be able to undermine business rules. If RBAC is used then much of the static enforcement of these policies will depend on careful role administration to logically group the permissions according to the job function.

However, constraining a users actions administratively is not always sufficient. The nature of the role hierarchy in RBAC systems, make it possible for users to adopt many possible roles and their associated permissions during a particular session. This could lead to a particular user inadvertently being allowed to activate a combination of roles that would allow him or her to bypass access control rules. For example, a “manager” would be able to approve his own application for leave that he was able to create using permissions inherited from the “clerk” role lower down in the hierarchy.

Therefore, some of these policies, like separation of duty, may need to be evaluated at run-time in order to constrain the permissions accordingly. The separation of duty constraint that was violated in the previous example may, for example, be enforced during run-time by ensuring that a user may not activate the “manager” and “clerk” roles together in the same session. Enforcement during either administration or run-time alone thus proves to be insufficient and therefore a hybrid approach is recommended. First, during administration, permissions must be assigned to appropriate roles and the administrator must specify which roles may not be activated together. Then, at run-time, the access control service must determine which roles are active, or were activated, and restrict which roles may now not be activated as

well. It is important that the above decisions are made and enforced in an unobtrusive manner.

Unobtrusive enforcement

In order for the run-time enforcement to be unobtrusive to the user, the access control service must also be able to answer questions of the form: “Which objects may user u access?” and “Which permissions do user u have to object o ?”. The answers to the questions can be used by the user interface to facilitate the unobtrusive use of an access control model by only making appropriate objects and methods available.

These general access control principles just discussed can now also be applied to workflow systems.

3.3 Access Control in Workflow Systems

The functionality of workflow management systems (WfMS), as discussed in section 2, can be separated in design-time and run-time functionality. *Design-time* functionality includes the conceptual specification of the process definition. The access control service must take cognisance of any static access control requirements as described in section 3.2.1. End-users will have access to the workflow during *run-time*. This phase requires the background execution and monitoring of processes as well as enforcing access control requirements as described in section 3.2.2).

Traditionally it is the responsibility of an administrator to capture these business processes and operational security requirements during design-time. Tasks can be considered to be the principle building-blocks of most workflow systems, and tasks are executed by performing multiple actions which require interdependent permissions. As such it makes sense that tasks are assigned to roles during design-time and not individual users. However these roles are usually designed with particular job functions in mind, and a task may require a more restrictive set of permissions. Therefore, when making such assignments the administrator must still consider the “Least Privilege” and “Separation of Duties” policies and enforce them if necessary by supplying appropriate mechanisms during run-time.

Section 3.2.2 showed that the static interpretation of roles and permissions

alone can lead to an abuse of the system if not monitored and enforced at run-time as well. In a workflow sense, the particular object's status or context within the workflow must, therefore, also become a deciding factor in whether a user is granted certain permissions to that object. This idea of context-sensitive access control stipulates that three requirements must be considered when designing secure workflow systems as apposed to non-workflow systems. They are: "order of events", "strict least privilege" and "separation of duty". Three comprehensive access control models exist that incorporate these requirements to varying degrees. The models of Atluri and Huang (1996) and Bertino et al. (1999) broke new ground with regards to context sensitive requirements and will, therefore, be used as reference points for the discussions below. The following chapter will also evaluate how the CoSAWoE model developed by Botha (2001) supports these requirements.

3.3.1 Order of Events

Order of events must be enforced by only granting permissions if tasks follow a set sequence. For example, an order cannot be approved until filled out completely; similarly, once an order has been approved, it may not be re-edited. Enforcing this requirement was the main focus of the "Workflow Authorization Model" (WAM) developed by Atluri and Huang (1996). It recognizes that permissions change as tasks are finished throughout the workflow. Permissions are, therefore, connected directly to a task through an authorization profile. These profiles specify the minimum permissions necessary to execute that task, and therefore it can be said that such profiles also support the "least privilege" requirement.

3.3.2 Strict Least Privilege

Although the authorization profiles of WAM links tasks with particular permissions, it does not necessarily enforce the *strict* least privilege requirement. This requirement specifies that a user is given the least amount of privileges needed to perform the current task irrespective of the overall permissions associated with his or her role. For example, a manager who initializes a purchase order should not, at initialization stage, receive the permission to approve the order, even though he or she has the permission according to

the role to do so later in the process. It also states that a user should only receive such permissions when they are busy with the task. When work on a task is suspended at arbitrary times those permissions should subsequently be revoked from the user while the task is inactive. In the WAM model permissions are granted and revoked only at the beginning and ending of a task, with no regard to what a user may do with those permissions while they are not working on that particular task. This is however better than the BFA model (named after its authors) that does not even relate permissions directly to tasks. This model's aim was primarily to specify conflicts that exist within roles and to relate those roles to tasks. What this means is that a user will receive permissions purely based on his or her role in the organization with no regard for the current task being executed. Therefore, strict least privilege is not supported by the BFA model.

3.3.3 Separation of Duty

The separation of duty requirement enforces semantic integrity through adherence to business rules designed to prevent participants from inadvertently making mistakes or deliberately committing fraud. Separation of duty ultimately comes down to restricting users from doing certain tasks in the workflow. Role-based access control has sought to alleviate the administrative burden of allocating permissions to users by introducing the role abstraction. It also allows roles to inherit permissions from roles below it in the role-hierarchy. The danger in a workflow is that this can lead to the situation where a user in a manager role can approve the same order he created with permissions inherited from the clerk role. It is therefore necessary to distinguish between SoD specifications enforced during design-time and those enforced during workflow sessions.

Static Separation of Duties (SSoD) is mainly an administrative policy to make sure that roles are allocated to tasks in such a way as not to violate organization rules (who is supposed to do what). Preferably this responsibility should not be left solely to an administrator's ability to discern possible conflicts, and a mechanism must be provided to assist the administrator and prevent wrong assignments from taking place. However this could prove very restrictive to smaller organizations who may not have many users to fill a specific role. **Dynamic Separation of Duties (DSoD)** will allow

certain conflicts between roles and tasks to exist statically. However, during run-time, DSoD will resolve these conflicts by carefully selecting individual users, from the list of potential users assigned to the role, to participate in that instance of the workflow tasks.

Separation of duty is supported in the SecureFlow prototype developed as part of the WAM model. The expression of separation of duty constraints is limited, however, to tasks. No provision is made for static separation of duty constraints or constraints based on the roles of users. All these separation of duty constraints are however fully supported by the BFA model.

From this section it is clear that the access control service plays a crucial role in securing data in a workflow. Still, access control must be tailored to the language used to describe the data and the types of actions that can be executed on such data. In this respect, investigating an access control model and related mechanisms, in terms of the commercial environment in which it is deployed will be crucial to its effectiveness.

3.4 Conclusion

This chapter narrowed the scope of the dissertation to focus on information security, and access control requirements in particular, within a workflow environment. First, the traditional information security requirements of workflow systems were discussed. A distinction was made between protecting the information from a physical as opposed to a logical perspective. A physically secure workflow environment can be created by applying the same techniques (such as encryption and digital signatures) as one would use for other applications where information must travel over a network. This would help to counter the threats of modification, fabrication and interception from unauthorized users to the system. However, fraud may still be committed when authorized users' access to information in the workflow is not strictly controlled. This essentially motivates the need for logical access control.

The access control service plays an important role in securing information. Some of the common administrative as well as run-time enforcement issues relating to an access control service were discussed. In a workflow environment it is also necessary to cater for the dynamic requirements of users based on the context of the tasks they are performing at a particular time.

As such, the access control mechanism must ensure that permissions are not only specified statically, but also constantly checked at run-time to ensure that a user does not contravene “order of events”, “strict least privilege” and “separation of duty” requirements.

The separation of duty function of the access control service ensures that tasks are performed according to the organizational rules of a business. The aim is to distribute the responsibilities to perform tasks between multiple participants so that there is less chance for users to bypass the business rules or commit fraudulent acts.

The CoSAWoE model developed by Botha (2001) can now be investigated to show how these security concepts can be catered for by an access service as part of a workflow environment. Since this dissertation focusses on the implications of using this model in a commercial workflow product (such as Oracle Workflow in chapter 5), only those components that will directly affect such systems will be discussed in more detail.

Chapter 4

The CoSAWoE Model

Chapter 3 discussed the role of an access control service in workflow systems. It was argued that workflow systems require specialized access control functionality. Three policies were identified to ensure context-sensitive access control for workflow systems, namely “order of events”, “strict least privilege”, and “separation of duties”. Discussions of these policies also highlighted how they are supported (or not) in existing access control models.

The aim of this dissertation is to investigate the ease with which a particular access control model might be implemented in an actual commercial workflow product. A model developed by a researcher at this institution was the obvious choice to investigate, simply because of the access it afforded the author of this dissertation to its creator. This chapter will therefore focus on how the afore-mentioned three policies are supported by the “**Context-Sensitive Access control in Workflow Environments**”, henceforth called the “CoSAWoE” model (Botha, 2001).

The model also divides the functional aspects into two: administration or design-time functionality and run-time enforcement functionality. This echoes the manner in which in workflow management systems (chapter 2) and access control requirements (chapter 3) were previously discussed. However, some of the model’s features will not be relevant to commercial workflow systems, and as such will not be discussed beyond the overview of the model that follows next. The main focus will be on the separation of duties and worklist generation aspects. Therefore, the chapter will conclude with a scoping statement to outline the aspects which will be covered in our solution chapters.

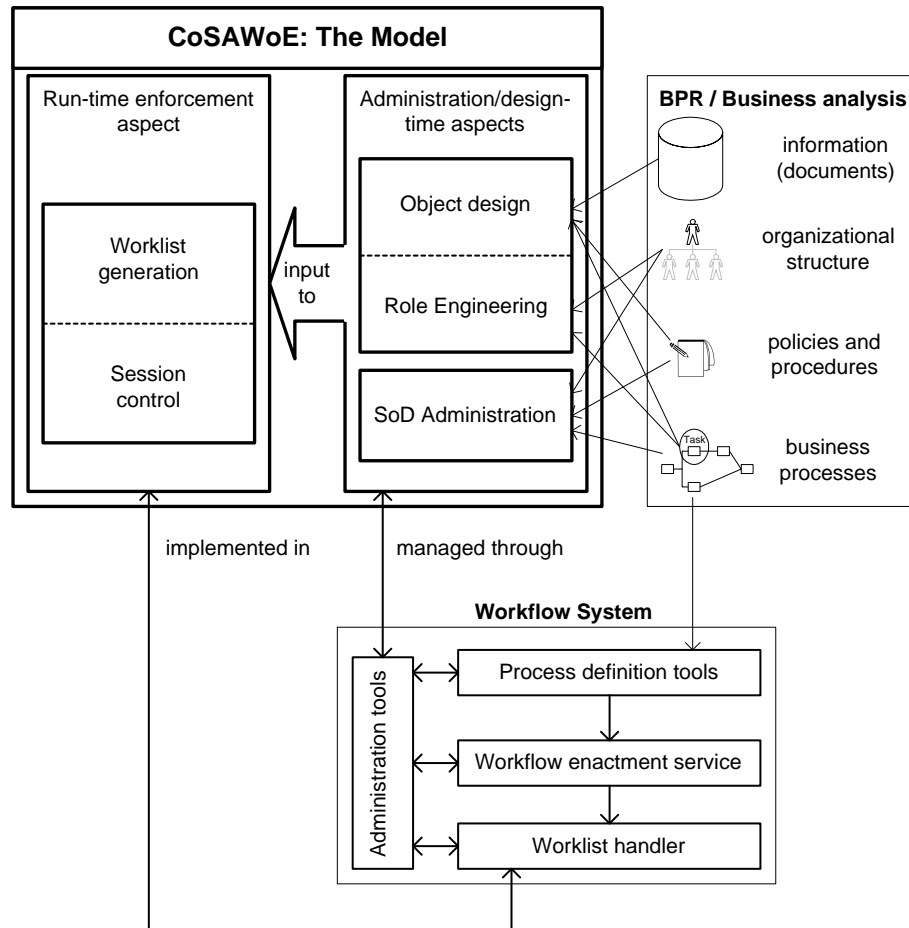


Figure 4.1: CoSAWoE: Conceptual view

4.1 An overview of the CoSAWoE model

An important property of the CoSAWoE model is that it not only addresses the technical enforcement of access control, but that it also provides guidance for the design-time activities that pre-empt the technical enforcement of access control. Figure 4.1 provides a conceptual overview of the administration/design-time and run-time aspects included in the CoSAWoE model. It also indicates that the administration/design-time aspect serves as “input to” the run-time enforcement aspects which will be implemented through the workflow system’s worklist handler. Administration tools, which may or may not be supplied with the workflow system, will draw on information gathered during business analysis that will include relevant documents, organizational structures, policies and procedures, and business processes. This information will guide the administrator in setting up the “secure envi-

ronment” within which users may access workflow processes and data.

Since the workflow system is still responsible for the enactment of the workflow and the access control service suggested by the model is responsible for the access control decisions, it can also be said to be an orthogonal approach. This implies that changes made to the access control service will not require changes to the workflow system and vice versa. This feature of the model will make it particularly suitable for integration with a commercial workflow system especially if such a system does not allow for much customization.

4.1.1 The Administration/Design Aspects

In this dissertation a permission is defined as the capability to execute a method of an object (see section 3.2 on page 28). Section 3.3 on page 34 also stated that tasks are executed by performing multiple actions which require interdependent permissions. Workflow systems allow administrators to assign roles to tasks during design-time. These roles are usually designed with particular job functions in mind, and a task may require a more restrictive set of permissions. Therefore, the desired features of an access control service for workflow systems cannot be supported unless the designers of the system have a clear understanding of what represents “the absolute minimum permissions required to perform the task” (Botha & Eloff, 2001b).

To this end, the components of the CoSAWoE model related to the administration and design-time aspect provides administration guidelines to assist the security administrators in ensuring that permissions of an appropriate granularity are available (**object design**), and that they are associated with roles in an appropriate manner (**role engineering**). They also ensure that the environment is of such a nature that the run-time enforcement mechanism can control access in line with the context-sensitive requirements for workflow identified in chapter 3 (including **separation of duty**).

Object design

Object design relates to how a permission profile, that indicates the exact permissions required for a specific “stage” of the information object’s life, can be constructed. This requires a hierarchical decomposition of the infor-

mation object and an analysis of its lifecycle by constructing a state chart. The lifecycle can only be expressed if the workflow in which it cooperates is properly identified as part of the business analysis.

Designing information objects in the described fashion ensures that the permissions are of a sufficiently fine granularity to meet the requirement evident in the definition of strict least privilege which states that “a user should receive the smallest possible set of permissions for the current task within the business process”. The described object design technique has the further advantage that the number of permissions are also kept under control to ease the administration thereof.

The permissions need to be assigned to roles that can, in turn, be assigned to the workflow tasks. Appropriate roles should thus exist. This represents the main aim of role engineering.

Role Engineering

Role engineering uses information collected as part of the business analysis, particularly information about the organizational structures and the workflow tasks.

This aspect of the model presents a methodology for the systematic construction of a “typed” role hierarchy that uses the information collected during business analysis. The resultant hierarchy has roles, called “task roles” that can be associated with the appropriate permissions.

The way in which the hierarchy is constructed ensures that other control principles, such as delegation of authority and reporting structures, are also supported. The resultant role hierarchy directly supports the principle of strict least privilege in that it ensures that there is a role that will have the absolute minimum permissions required for the task at hand.

Since the “typed” nature of the role hierarchy is only relevant during construction, the resultant role hierarchy supports current ideas on separation of duty.

SoD Specification

The “Conflicting entities” Administration Paradigm (CoAP) is proposed by the CoSAWoE model as a way of specifying separation of duty requirements. When entities in CoAP conflict, it implies that the associations with those

entities must be carefully controlled. Conflicts may be specified between entities such as tasks, users, roles and permissions.

Two strategies for controlling the conflicting entities mentioned in CoAP may be employed, namely static and dynamic. These strategies may be used in isolation or together to complement each other's weaknesses. Static conflicts indicate that certain associations must never be allowed, while dynamic conflicts indicate that the associations may be made but that the use of these entities must be carefully controlled within the context of a process instance. Static conflicts, on the one hand, are evaluated by the administration environment in a bid to ensure that associations that should not be allowed are rejected. Dynamic conflicts, on the other hand, can only be interpreted by the run-time components.

4.1.2 The Run-time Enforcement Aspects

The run-time enforcement of access control must happen according to the specifications of the administration/design aspect of the CoSAWoE model. In a bid to design the access control model in an orthogonal way to the workflow system, as little as possible influence on the workflow systems should be evident. With the CoSAWoE model the influence lies with two components, namely how the worklist is constructed and how the concept of a session is interpreted as a WSession. The text considers the purpose of each component in turn.

Worklist Generation

The tasks that need to be done are communicated to the users through the worklist. Since each task is associated with the minimum role that may perform it, all users that may assume a role equal or superior to that role may perform the task.

However, in order to incorporate separation of duty, the ability to assume a role equal or superior to the role that the task requires becomes a necessary, but not a sufficient condition for receiving the item on the worklist. The user should, furthermore, also not be identified as a user that would cause any separation of duty requirement to be violated. The users that should receive a specific item on their worklist can, therefore, be computed with a series of

set minus operations.

The task would appear on the worklists of those users who may assume the required role and who do not cause a separation of duty requirement to be violated. When a user acts on a task in the worklist, a WSession is created.

WSessions

A WSession is a specialization of the session concept in RBAC models. It is based on the assumption that when a user signs onto the system, that user will be associated with roles based on his administration time associations, but he would not receive any permissions at the time. In other words, the roles won't be activated. Only once a user acts on a task in the worklist, a WSession is created. During the duration of a WSession, a user receives the permissions of the role associated with the task and not of the roles that the user may assume. The WSession is terminated and the permissions revoked as soon as the user stops or suspends work on the task.

It is argued that the CoSAWoE components provide an environment that supports the implementation of context-sensitive access control. However, not all these components will necessarily have a bearing on commercial workflow systems.

4.2 Commercial Perspective on CoSAWoE

As section 2.4 on page 20 pointed out, commercial workflow systems are increasingly being integrated into existing systems and will often be part of packaged suites that deliver specific cross-functional solutions. Therefore, object design and role engineering will be performed as part of the normal security administration of company-wide information, and will not be the particular domain of workflow systems as such. The methodologies suggested by these two administration components does indeed provide access control to objects at a fine level of granularity based on carefully constructed roles that would ensure strict least privilege is maintained. However, where and when this object and role engineering effort occurs is outside the scope of the commercial workflow application and can just as easily be achieved through an administration tool such as SoDA, the CoAP administration prototype

developed for the CoSAWoE model (Perelson, Botha, & Eloff, 2001).

Session control is concerned with creating workflow sessions (WSessions) that would not allow a user to access a workflow object outside the needs of the tasks he/she is currently working on. In many workflow systems, such as the Oracle Workflow discussed in the following chapter, the system does not allow users to directly update the workflow object, but instead the workflow engine act as a proxy. Since users are not given permissions to the actual object, the session control component to manage the granting and revoking of said permissions is of no consequence. If the workflow object is however represented as a hierarchical document to which users would require different privileges according to each task, such WSessions could be implemented through an access control service, as illustrated in the WACC prototype developed by Cholewka, Botha, and Eloff (2000).

The following paragraphs will describe separation of duty administration and run-time worklist generation in more detail. These two model components will have a significant impact on how an access control service will be implemented within a commercial workflow product.

4.2.1 Separation of Duty Administration

In order to determine the separation of duty requirements, a careful analysis of the policies and procedures of an organization will have to be made during the business analysis. The CoSAWoE model attempts to make the administration of what is generally considered to be fairly complex constraints, as simple as possible through its “Conflicting entities” Administration Paradigm (CoAP). Within CoAP, a conflict between entities implies that the risk of fraud increases if the associations with those entities are not carefully controlled and monitored. Note that the term “conflict” does not indicate a state of disharmony between the entities themselves, but that it rather refers to the potential state of disharmony between the actual and desired state of the workflow system that could result from not carefully controlling the associations with those entities.

Fundamental to the interpretation of “risk of fraud” is the concept of permissions. Permissions indicate the ability to execute a certain method on an object. Permissions, therefore, are central to SoD definitions. Permissions are considered as **conflicting permissions** if, together, they provide more

ability than required by a single user.

Conflicting users state that there may be a relationship between users (for example, husband and wife, or brother and sister) that may compromise the integrity of the tasks they perform. However this is an unfortunate naming convention, since this relationship between users is anything but conflicting. In fact, their relationship implies that they can work together, or collude, so that the outcome of their tasks will benefit each other.

Conflicting roles are roles that share conflicting permissions. From a practical perspective, the conflicting permissions might not always be identified, since the identification of conflicting permissions for conflicting roles may negate the administration advantages obtained through the role abstraction in RBAC. However, identifying conflicting roles would be senseless if there are no conflicting permissions involved.

Conflicting tasks, similarly, are tasks that require some conflicting permissions to complete. To ease administration, conflicting tasks may also not always enforce the identification of conflicting permissions.

The various conflicts could apply either at administration time (static separation of duty) or at run-time (dynamic separation of duty). Static conflicts indicate that certain associations must never be allowed, while dynamic conflicts indicate that the associations may be made but that the use of these entities must be carefully controlled within the context of a process instance.

Static Separation of Duties (SSoD)

SSoD policies can be enforced in the administration environment and, as such, has the potential to prevent making assignments that should not be allowed. This will therefore impact on the user and permission assignments made during the role engineering component discussed in previous sections.

The way in which these SSoD requirements are enforced is best explained through an example. Consider the well-accepted business rule that “auditors should act independently”. This implies that auditors should not be able to audit their own doings. There are several interpretations of this business rule. Table 4.1 on the facing page summarizes an interpretation in terms of each of the conflicting entities.

It should be pointed out that SSoD can be very restrictive to business operations, especially in smaller organizations with fewer users or fewer roles.

Table 4.1: Static SoD interpretations for the business rule “Auditors should act independently”

Possible conflict	Interpretation of business rule
Conflicting roles	User assignments to the “Auditor” role and (for example) “Accounts Payable Manager” role must be mutually exclusive.
Conflicting permissions	The same user may under no circumstances receive the “Approve Order” and “Approve Audit” permission. This implies that the two permissions may not be associated with roles that share common members.
Conflicting users	Members of the same family must be considered as the same user and may therefore not be assigned to roles, permissions or tasks to which a single user should not be assigned.
Conflicting tasks	A user who can do the “Approve Order” task may never do the “Approve Audit” task, and vice versa. This implies that the two tasks may not be performed by the same role.

The inflexibility of static separation of duty is evident with the conflicting user interpretation of the business rule “two members of the same family may not belong to the Auditor and Accounts Payable Manager roles respectively”. According to this requirement a person would not be able to ever act in the role of Auditor if a family member of his or hers is in a role of authority such as the Accounts Payable Manager. This is, obviously, very restrictive. Nevertheless, in certain cases it may prove necessary. For example, it might be a realistic requirement that auditors may not do anything but audit and that they should have no family ties that could make them even slightly biased.

The extreme restrictions imposed by static separation of duty can be alleviated, however, by using dynamic separation of duty.

Dynamic Separation of Duty (DSoD)

While static conflicts, on the one hand, are evaluated by the administration environment in a bid to ensure that associations that should not be allowed are rejected. Dynamic conflicts, on the other hand, will allow such conflicts

to exist and will only specify the conflicts so that they can be enforced by the run-time components. How they will be enforced will be explained later during the discussion of the “worklist generation” component.

The dynamic interpretations of conflicting entities, differ somewhat to their static counterparts. Consider the example of the different dynamic interpretations of the business rule “An order may not be approved by its initiator”, summarized in Table 4.2.

Notice that the emphasis is on determining whether conflicts exists based on the activation of permissions in the *same process instance*. Therefore, dynamic separation will allow users with vested interest in one another’s work to adopt roles that would allow them access to two conflicting tasks, provided of course that they do not perform those tasks as part of the same process instance (or session). This also indicates the need to keep some form of access history with regards to who had access to which tasks in a particular process instance. The following section will elaborate on this when run-time components are described. All that is required during administration as far as DSoD is concerned, is the specification of the conflicting entities themselves (e.g. $User_A$ conflicts with $User_B$, or $Task_1$ conflicts with $Task_2$, etc.).

Table 4.2: Dynamic SoD interpretations for the business rule “An order should not be approved by its initiator”

Possible conflict	Interpretation of business rule
Conflicting roles	The “Stock Controller” role and the “Account Payable Manager” role may not both be activated for the same user during one process instance.
Conflicting permissions	The “Create Order” and “Approve Order” permissions must not be exercised by the same user in a single process instance.
Conflicting users	Family members (Conflicting users) must not operate on conflicting tasks or exercise conflicting permissions in a single process instance.
Conflicting tasks	The “Create Requisition Form” task and the “Approve order” task must be done by different people in a single process instance.

4.2.2 Worklist Generation

The worklist forms the primary interface between the user and the workflow system. As such, it is the primary means of controlling access to tasks in the system. In typical RBAC systems each task is associated with the minimum role that may perform it. All users that may assume a role equal or superior to that role may perform the task, and as such it may appear on their worklists as well. Although the roles assigned to tasks are checked during build-time, the individual users who will assume those roles during run-time cannot be determined or verified before execution.

As specified earlier, dynamic separation of duty can only be enforced during run-time based on the conflicts specified during administration. It therefore becomes necessary to identify users, belonging to the assigned role, who may violate any of the separation of duty requirements and then prevent them from receiving the task. This would require information regarding users' past involvement in the workflow process instance. Workflow history is usually maintained by the workflow service in the form of task instances. The model requires that such task instances include information that

- uniquely identifies the process instance,
- identifies the task definition on which the task instance is based,
- identifies the user that acted on the task instance and
- identifies the role that user assumed while he or she acted on the task instance.

This information will be used by the access control service to determine which users may perform a task without violating the separation of duty policy. This is done in a two phased approach. In the first phase, a list is generated of users who may assume the role allocated to the task in question, without taking the effect of separation of duty constraints into consideration. As a second phase, the list is pruned according to the constraints introduced by the separation of duty policies. These constraints determine if the same user (or his or her related users) executed any of the conflicting tasks, specified for the task in question, during the same process instance. Dynamically conflicting users are considered as one user for the purpose of the process

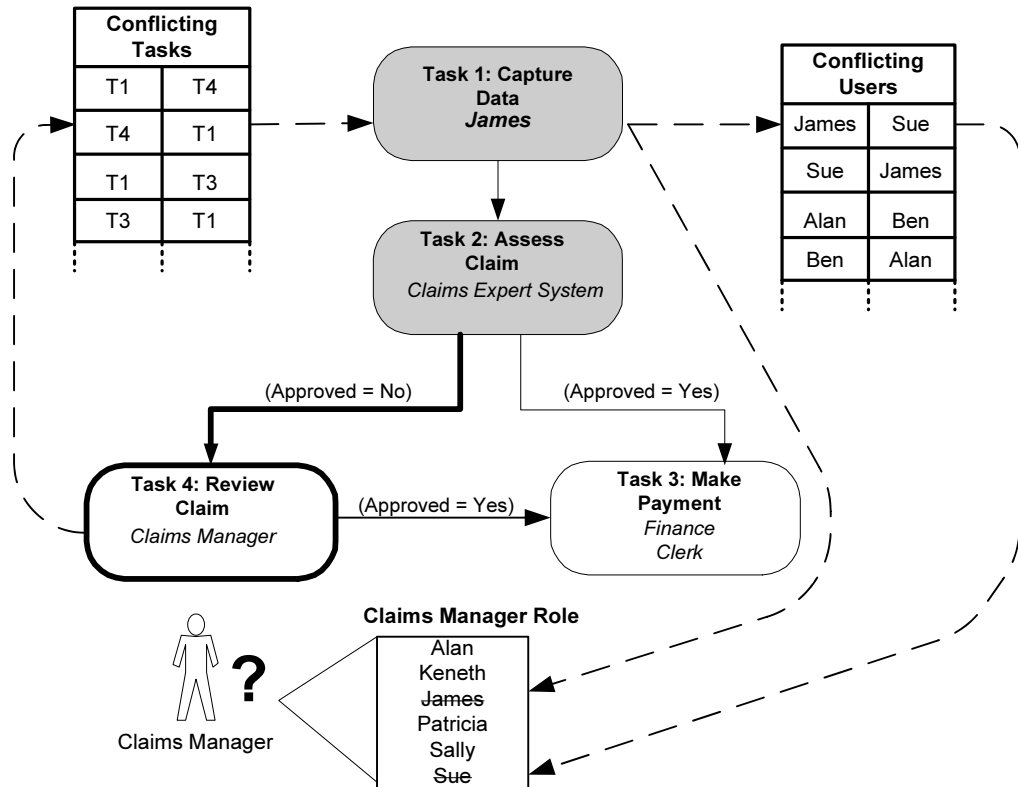


Figure 4.2: Dynamically pruning the user list during run-time

instance and as such it has no influence when looked at on its own. However conflicting users must be specified during administration and interpreted with the other dynamically conflicting entities.

Figure 4.2 shows an example of how separation of duty requirements can be implemented during run-time to prune the potential user list. In the example, task 1 (T_1) has already been completed in this process instance. When task 4 (T_4) needs to be performed, a list of managers will have access to this task. But first, the *conflicting tasks* specified during administration/design are searched and task T_1 is identified as conflicting for T_4 . From the task instance (T_1) it is clear that James has participated in the same workflow session on this task, and therefore he is excluded from the list of possible performers for T_4 . On the other hand Sue, who is also identified as a potential manager, have not taken part at all in this process instance. So at first it would seem as though no conflict exist to exclude her from the list. However, the *conflicting users* are looked-up next to find out if there are any users who could be in a position to collude with James. Sue, his wife, is identified

and therefore she is also excluded from the list of possible performers for T_4 . *Conflicting permissions* and *roles* for the current task can be evaluated in a similar fashion to determine if a particular user had performed other tasks which activated those permissions or roles.

Pruning the user list for each task will prevent users from even being notified of the work item in their worklists if they are not allowed to perform it according to separation of duty requirements. A worklist is therefore also essential to making the access control service as *unobtrusive* as possible to the workflow users. When a valid user acts on a task in the worklist that, task is immediately removed from the other user's worklist denying them concurrent access to the same task.

4.2.3 Notes on integration of CoSAWoE with Commercial Systems

The previous sections described the two model components which will have the most relevance on commercial workflow systems: SoD administration and worklist generation. Should the vision of the Workflow Management Coalition regarding a fully interoperable environment (Hollingsworth, 1995) be realized, the CoSAWoE model will have an influence on two workflow components: the administration tools and the worklist handler. Therefore the scope of the model and workflow components that will be affected is graphically depicted in figure 4.3 on the next page as the shaded areas.

However, currently the distinction between the different components in commercial systems is not so clear. In this respect a commercial workflow management system will have to exhibit the following properties to enable the integration of CoSAWoE:

- The administration side should be exposed to customization. Alternatively the administration side should be separated from the rest of the system with a well-defined interface. This would allow the support of tools based on CoAP in addition to the current administration tools.
- The creation of the worklist should be customizable. Should the worklist be maintained as a physical entity, database-level programming in the form of triggers might enable the appropriate removal of work items

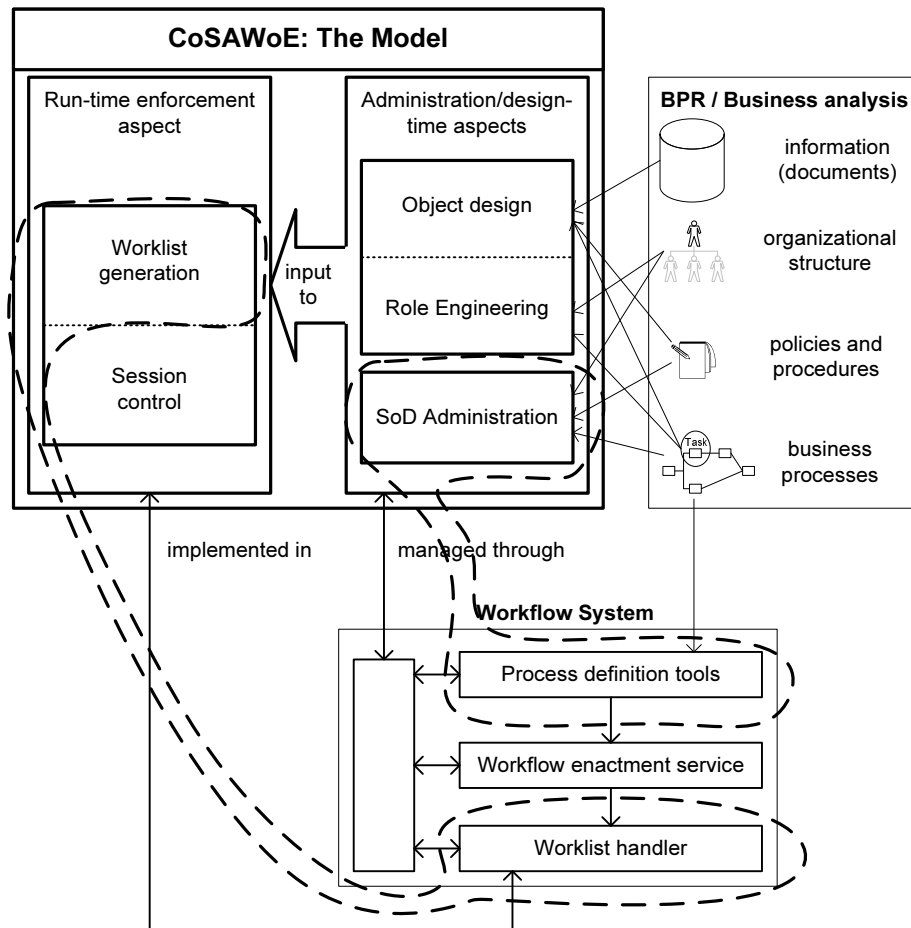


Figure 4.3: Scope of implementing the CoSAWoE model in a commercial workflow product

from the individual worklists. If worklists are generated dynamically, potential users will need to be filtered.

- The timing of the granting and revocation of access rights possibly hold the biggest challenge in current systems. It would require the task abstraction to be clearly separated from the rest of the system. Kang, Park, and Froscher (2001), for example, in their SALSA prototype use a decentralized approach with no central workflow engine. Each task contains a small portion of the workflow specification in that it knows which tasks to interact with. In such a distributed manner the task object itself controls the access to it and the granting and revoking of access largely becomes irrelevant. The task objects need to interact with a monitor service to be able to enforce dynamic separation of duty requirements.
- In systems that don't support role-based access control principles, a role server may also be required to provide that functionality (Ahn, Sandhu, Kang, & Park, 2000). The role server will issue users with role certificates, based on user-role assignments and the role hierarchy, which then have to be evaluated by the workflow engine when the worklists are assembled.

From the brief comments above, it is clear that the integration of CoSAWoE into a commercial workflow system is far from arbitrary. It presents interesting challenges that will have to be addressed through a variety of approaches.

4.3 Conclusion

This chapter showed how the CoSAWoE model, in effect, links an role-based access control service with workflow functionality. The model is based on associating users with roles and roles with tasks, in such a way that the three policies for context-sensitive access control would be enforced. Therefore, each section pertaining to a particular component referred to how it affected the “order of events”, “strict least privilege” or “separation of duty” policies. The components of the model was conceptually divided into two:

those components that provide administrative guidelines to security administrators during design in order to set up access control requirements, and those components which enforced those requirements during run-time.

The **object design** administrative environment required that object permissions are clearly specified according to a strict least privilege policy which states that “a user should receive the smallest possible set of permissions for the current task within the business process”. The **role engineering** component also ensured that there would be a role in the role-hierarchy that will have the absolute minimum permissions required for the task at hand. The run-time enforcement of **session control** through workflow sessions (WSessions) also showed how users are associated with that specific “typed” role once they request access to the task, and how it is subsequently revoked on completion of the task.

Although the model requires the three aforementioned components to ensure context sensitive access control is achieved, their implementation in commercial systems do not represent a significant integration effort. More importantly, **SoD administration** addressed the need to specify the various conflicts of the CoAP paradigm so that they may be applied either at administration time (static separation of duty) or at run-time (dynamic separation of duty). The **worklist generation** component, that fall in the run-time enforcement sphere, was aimed at controlling the allocation of tasks in particular process instances to workflow users based on their previous participation. Pruning the user list for each task prevents users from receiving work items in their worklists if they are not allowed to perform it according to separation of duty requirements.

Prototypes that enforced certain aspects of the model were developed to demonstrate certain concepts, and were briefly mentioned in this chapter. However, such prototypes were specifically developed with the model in mind. Commercial, off-the-shelf workflow systems may not support the model’s proposed functionality as effectively. Some workflow products may include an access control service already, and others may not. Regardless, it is argued that aligning the product’s built-in functionality with the context-sensitive requirements of the CoSAWoE model may indeed prove difficult. Therefore, the remainder of this dissertation will investigate those difficulties with regards to a particular product, namely Oracle Workflow.

Chapter 5

Oracle Workflow

Workflow systems, the information security concerns for such systems, and a context-sensitive access control model that addresses those concerns have now been discussed. The previous chapter also highlighted the issues and requirements surrounding the implementation of the CoSAWoE model in a commercial product. However, there are many commercial systems to choose from on the market with just as many different implementations of the functional requirements presented in chapter 2 for workflow systems. The scope of these systems are also too big to compare the alternative implementations of the model in two or more products simultaneously. Therefore it was decided to base the discussions on a single commercial workflow product that follow the trends in workflow systems as discussed in section 2.4, and which adhere to most of the requirements mentioned in section 4.2.3 for implementing the model. The “Oracle Workflow” product was chosen, as it can be considered a typical mainstream workflow product by a reputable company. The fact that the Oracle environment was already familiar to the author, and the product included a GUI development tool, also meant that development and learning time could be significantly reduced. As an added advantage, PL/SQL could be used to code the necessary additional program logic without involving another development tool.

Unfortunately little documentation about the internal workings of this product exist besides the “Oracle Workflow User Guide” (Chang & Jaeckel, 2002). It was also necessary at times to “reverse engineer” some of the tables structures from the data dictionary since no database model is included to show how the various workflow tables in the database are used to store

workflow definitions and process instances.

The chapter will begin by introducing the reader to the workflow terms used by Oracle Workflow. Thereafter the architecture of its various components, as far as it could be determined from the limited information, will be discussed. The remainder of the chapter will summarize the salient information from the user manual. These discussions will form the backdrop of the administration/design-time and run-time steps of implementing the CoSAWoE model in Oracle Workflow, discussed in chapter 6 and 7 respectively.

5.1 Workflow Terms Used in Oracle Workflow

The workflow terms used by Oracle Workflow differ somewhat from those used by the Workflow Management Coalition's Reference Model (Hollingsworth, 1995) and those defined in chapter 2. The reference model defines a business process as "a procedure where documents, information or tasks are passed between participants according to defined sets of rules to achieve, or contribute to, an overall business goal". As stated in chapter 2, a *workflow* is seen as a representation of the business process in a machine readable format. The Oracle Workflow product can be described as a fully fledged *workflow management system* (WFMS) as it is "a system that completely defines, manages and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic" (Hollingsworth, 1995). The Oracle Workflow terms for workflow components are shown in figure 5.1 on the facing page:

- **Item Types** are used to group all the components needed for a specific *process definition*. This process definition groups together all the activities that occur in the process and the relationship between those activities.
- **Activities**, also referred to as *tasks*, in a process definition can be automated functions, external functions, notifications to users or roles that may optionally request a response, a business event, or process subflows that are made up of a more granular set of activities.

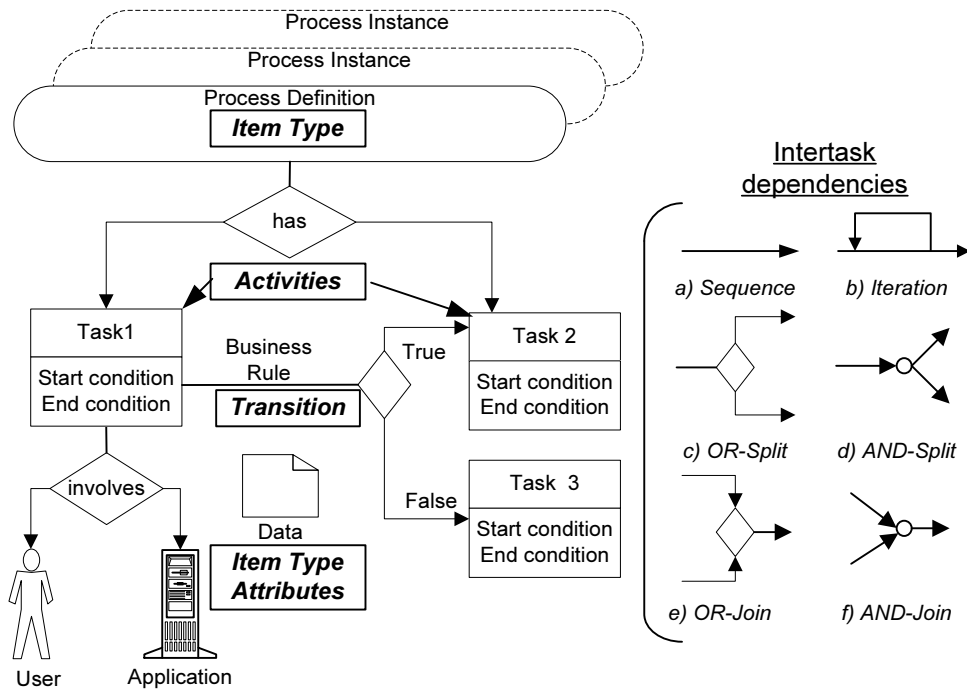


Figure 5.1: Oracle Workflow terms (in terms of figure 2.2)

- **Transitions** are used to connect activities. *Business Rules* or *conditions* may be attached to these transitions in order to decide the route that may be followed when transitioning from one activity to the next. According to these *intertask dependancies* the process may follow a straight sequence, loop back to repeat activities, and split into parallel flows.
- **Item Type Attributes** are data values associated with a given item type. An item type attribute acts as a global variable that can be referenced or updated by any activity within a process. As such they provide the *data container* necessary to propagate and provide information throughout the workflow. These attributes can be of different data types and may even hold an entire document.

Oracle Workflow also recognizes *Users* and *Applications* as components in the workflow and caters for their inclusion via the “Workflow Directory Service” and “Business Event System” features which will be discussed in more detail later. These and other features are part of the architecture for Oracle Workflow, discussed in the following section.

5.2 Oracle Workflow Architecture

According to the reference model of Hollingsworth (1995), a typical workflow management system provides support in three functional areas: *build-time* functions that support *Process Design and Definition*, *run-time* functions that are responsible for *Process Instantiation and Control*, and *run-time Interactions with Users and Applications*. The client/server architecture adopted by Oracle Workflow closely follows these three functional areas as can be seen in figure 5.2 on the next page. Functionality is distributed as follows:

- **Workflow Development Client.** The development client is a PC running MS Windows that will be used as a platform to create and modify process definitions, via *Oracle Workflow Builder*. The *Workflow Definitions Loader* is a utility program that moves workflow definitions between the database and corresponding flat file representations.
- **Oracle Server.** The heart of Oracle Workflow is the rules-based *Workflow Engine* residing in the Oracle database server. The engine uses the process definitions created with Oracle Workflow Builder to coordinate the routing of activities for the process. The *Notification System* is responsible for delivering messages to and from workflow users. The *Business Event System* uses Oracle Advanced Queuing technology to communicate business event data between the engine and external applications or other workflows. The server will also host the business application, integrated with Oracle Workflow.
- **Application Server.** The application server is the environment outside of the database server. This includes ancillary services such as *Oracle Web Application Server*, *WebDB*, and the *Notification Mailer*.
- **End-User Client.** This represents any workstation or PC that the end-user uses to perform their daily tasks. The Notification Mailer will allow normal workflow users to view and respond to their notifications using the *Web Notification Worklist*, as well as any MAPI complaint *Mail Application*. The *Web Process Monitor* graphically depicts the status of a workflow process instance, providing users and administrators access to workflow related information from any computer with

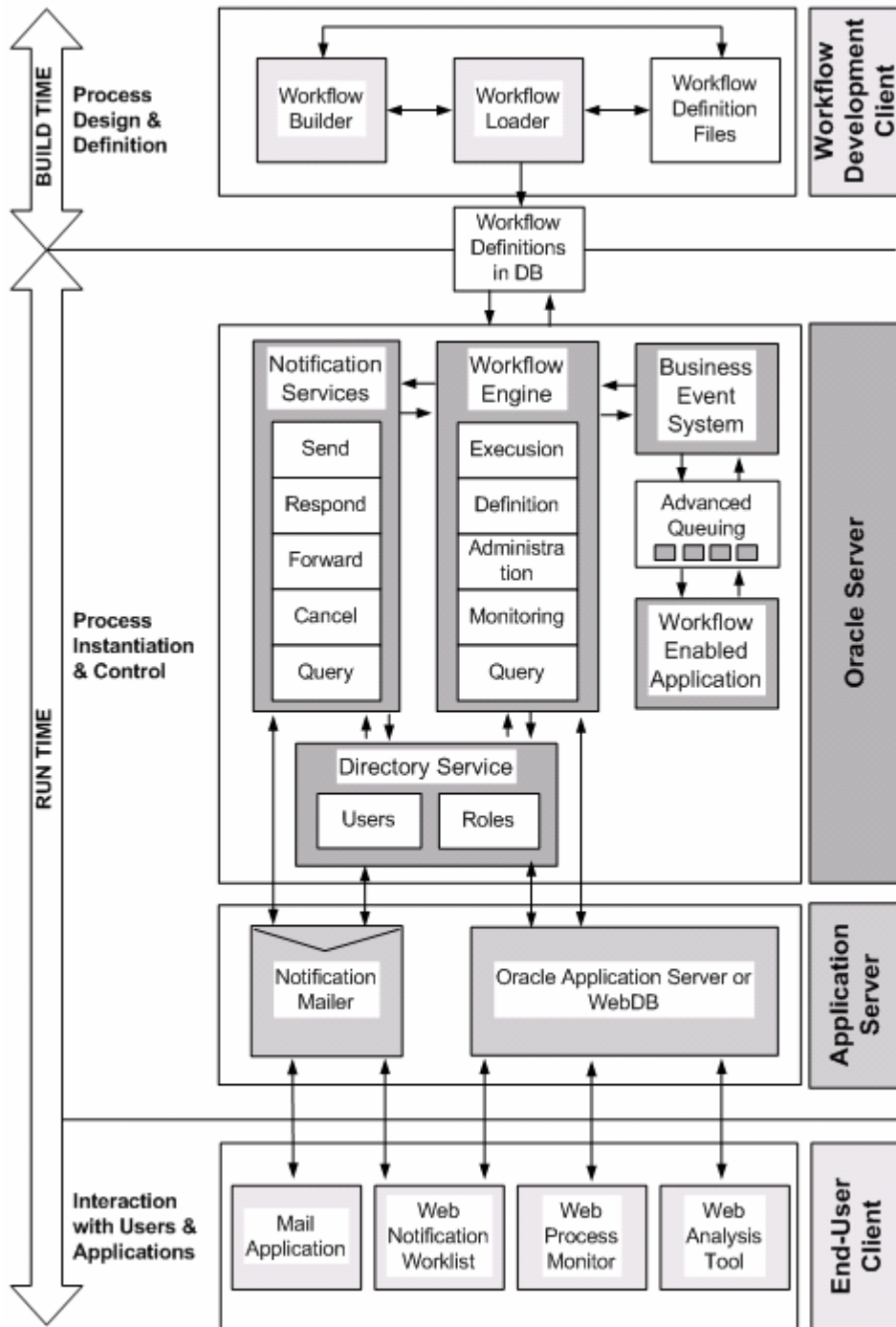


Figure 5.2: Oracle Workflow Architecture

an Internet connection. Administrators may also utilize other *Web Analysis Tools*.

The architecture shown in figure 5.2 was adapted from the architecture provided in Baldock and Seiden (2000) for Oracle Workflow version 2.5. The version that is discussed in this dissertation is the current version shipped with Oracle 9i, Oracle Workflow 2.6.2. This version included the Business Event System as an optional component on the server. The functional discussions in the following section is based on the documentation provided with this version (Chang & Jaeckel, 2002). A newer version, Oracle Workflow for Java (OW4J), leveraging Oracle's J2 Enterprise Edition (J2EE) platform is currently in development. Since only developer previews of this product were available at the time of writing, and as OW4J will only compliment – not replace – the current Workflow Engine residing in the database, it will not be explored further in this dissertation. It is interesting to note, however, that Oracle is following the workflow product trends in section 2.4 by

- modularizing workflow functionality into components and services
- embedding Oracle Workflow into its e-business suite, in addition to providing a stand-alone product, and thereby making it an integral part of core business applications such as Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), and Human Resources Management Systems (HRMS)
- utilizing Internet technologies to interface with workflow users and administrators
- ensuring interoperability with other workflow products through the support of XML event messages

Figure 5.2 on the preceding page showed how Oracle Workflow components can be divided into the three functional areas of workflow described in chapter 2. Some of the architecture components will now be described in more detail according to the particular functional area each one services. The way features are implemented differs depending on whether Oracle Workflow is used as a stand-alone product or embedded in the e-business suite. Therefore, the following section will describe the functionality from a stand-alone perspective.

5.3 Process Design and Definition in Oracle Workflow

Oracle workflow claims to be more than just a tool to simply route documents from one user to another with some approval steps. “Oracle Workflow allows you to model and automate sophisticated business processes complete with processes that loop, branch into parallel flows and rendezvous, decompose into subflows, branch on task results, time out, and more. Expressing business rules in the process model enables model-driven integration” (Oracle Technology Network, 2004). The central application to specify this process model is *Oracle Workflow Builder* which allows a user to create and modify all workflow components, including the item types, activities, and transitions mentioned earlier with simple drag and drop operations. There are two parts to this process definition tool: the *Object Navigator* and the *Process Diagram*. Figure 5.3 depicts a simplified workflow process definition that routes a requisition to a manager or set of managers for approval. The *Object Navigator* window on the left allows a user to define the activities and components of the business process (“Requisition” item type) using a tree structure. These can then be assembled in the *Process Diagram* window (right-hand frame) to create a process model. The central application to specify this process model is *Oracle Workflow Builder* which allows a user to create and modify all workflow components, including the item types, activities, and transitions mentioned earlier with simple drag and drop operations. There are two parts to this process definition tool: the *Object Navigator* and the *Process Diagram*. Figure 5.3 depicts a simplified workflow process definition that routes a requisition to a manager or set of managers for approval. The *Object Navigator* window on the left allows a user to define the activities and components of the business process (“Requisition” item type) using a tree structure. These can then be assembled in the *Process Diagram* window (right-hand frame) to create a process model.

The *Object Navigator* window displays a navigator tree hierarchy for each data store that is opened or loaded into Oracle Workflow Builder. A data store (primary branch) is a database connection or flat file that holds the workflow process definition. The Workflow Definitions Loader is invoked whenever a data store is opened or saved to. It loads process definitions into

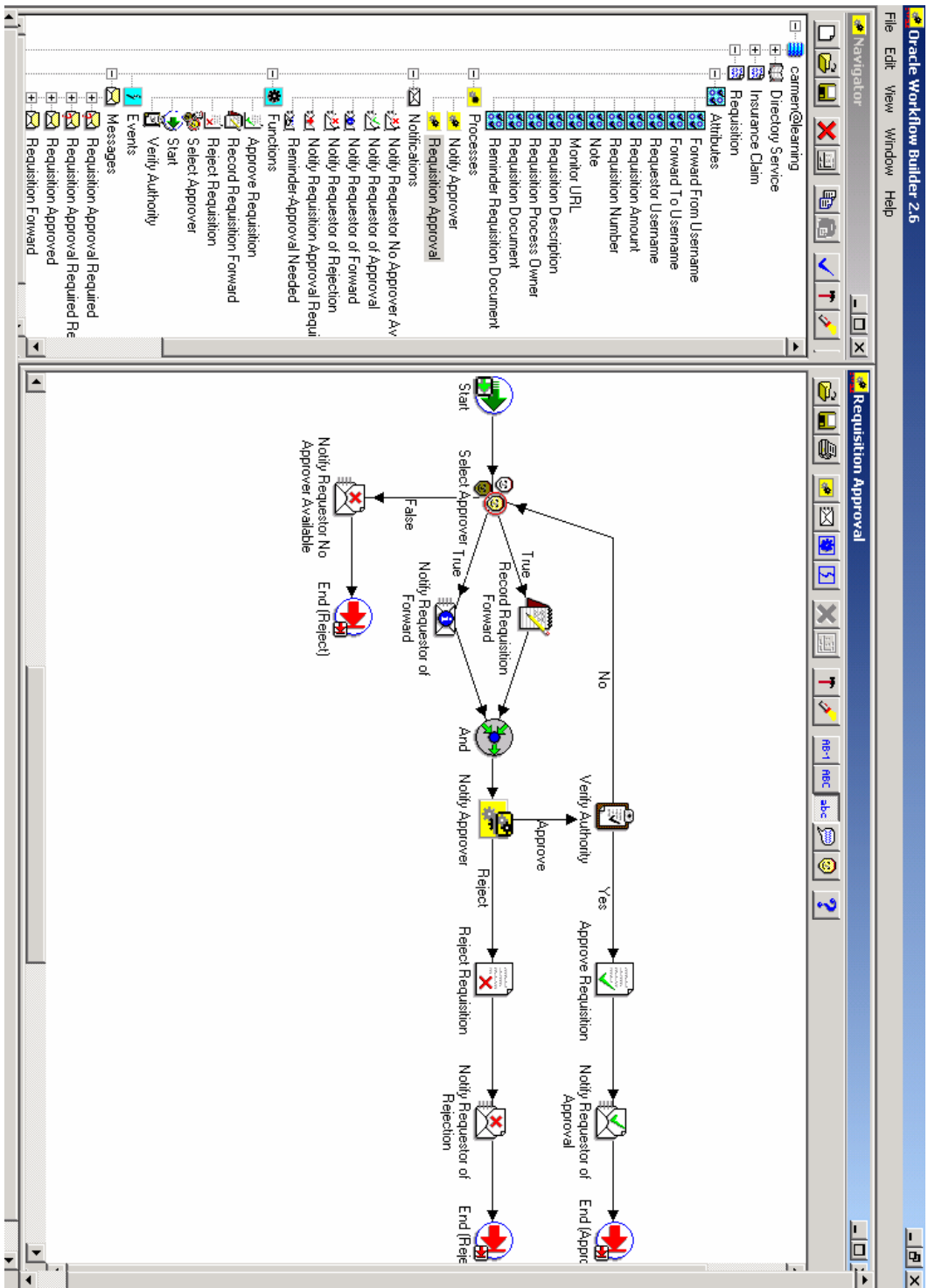


Figure 5.3: Oracle Workflow Builder: Object Navigator and Process Diagram windows

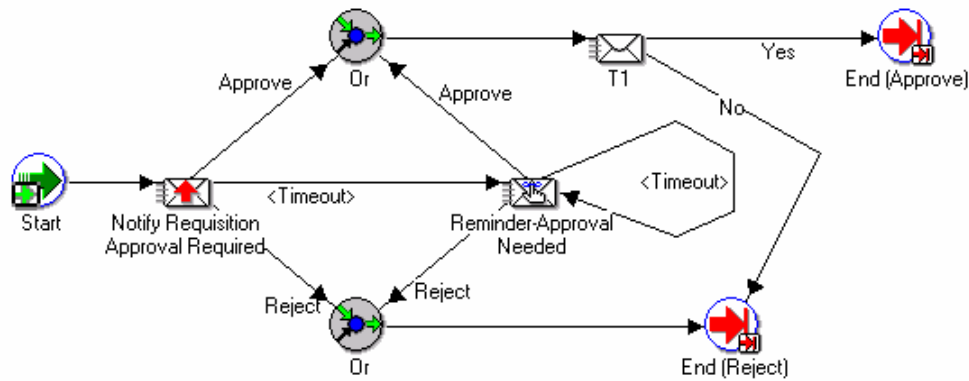


Figure 5.4: Drilling-down into a process activity(“Notify Approver”)

and out of Workflow Builder. Within each data store there is at least one **item type** heading (secondary branch) that represents the grouping of a particular set of processes and its component objects. An item type will contain process, notification, function or event activities. **Notifications**, such as “Notify Requestor of Approval”, use **Messages** that contain data in the form of attributes, and that can be sent to and acted upon by workflow users. The same message template can be re-used by several notification activities. Messages may prompt for a response or may simply provide information. Item Type **Attributes** are visible to all activities in the process and might include dates, numeric values, or roles (the equivalent of human participants or users in workflow terms). **Functions** handle any automated unit of work that do not require user interaction, and they are specified with PL/SQL procedures. The process example shown in figure 5.3 contains several workflow activities implemented as PL/SQL stored procedures, including “Select Approver” and “Verify Authority”. An **Event** activity represents a business event that the process receives, raises, or sends. Such events are particularly useful for integrating the current process with external workflows or applications. **Process** activities are sub-processes that group together other activities at a finer granularity that must be performed as a unit, and will deliver a particular result to the main process. For example, when a user opens the “Notify Approver” process activity, the process diagram shown in figure 5.4 will be displayed.

Notice, however, that the activities do not solely belong to a particular process, but can be part of multiple processes and may be re-used several

times in the same process. This grouping of activities and the different routes between them is only specified when the process diagram is drawn in the process window.

The *Process Diagram* window in Oracle Workflow Builder graphically represents the activity **nodes** (icons) and **transitions** (arrows) for a particular process. Each activity is a node, a logical step that contributes toward the completion of a process. Processes are typically initiated by one or more **start activity** nodes, and concluded with one or more **end activity** nodes. Although any activity can be set to be start or end nodes, standard “Start/End” activities are provided in a “Standard” item type. This standard item type is installed automatically on the workflow server and provides some generic activities that can be used to control processes. It also includes standard activities such as “And/Or”, “Loop Counter”, “Wait”, “Comparison” and “Vote Yes/No”, to name but a few. These activities can be used in any process as many times as necessary.

A **transition** can be specified between any two nodes. The different routes that a process may follow after completing a particular activity node will depend on the **Result Type** specified for the source activity. When a new transition is drawn from that activity node, one of the possible results for that return type is chosen as the **condition** for following that particular transition. Standard result types are specified with the “Standard” item type and does not need to be defined, such as the result type “Approval” (see figure 5.4 on the preceding page) which list two possible results, “Approved” or “Rejected”. The most commonly used return types are specified in the standard definition, however, application specific result types can be added to an item type by defining it under **Lookup Types** through the *Object Navigator*. A result type of <None> may also be specified, indicating that as long as the originating activity completes, the process will transition to the next activity. In addition, a transition may also be self-looping to re-execute the activity node (e.g. the <Timeout> transition on figure 5.4 specifies that “Reminder-Approval Required” messages will keep being sent if no response is received in a particular time frame).

For each activity node specified in the process window, certain node properties need to be specified. A node represents an instance of a specific activity and its property values are unique to that instance. Function and Notifica-

tion activity nodes contain particular properties that will become significant in later discussions.

If an activity is a notification, a **Performer Role** must be specified for that activity node. A performer role is a role that can consist of one or more workflow users. This role can be set to a static (**Constant**) role that is already defined in the database. Alternatively you can specify a dynamic role by selecting an item type attribute that returns a role name during run-time. If this attribute value is not set via some function activity in the workflow (e.g. the “Select Approver” function), the user launching the process is allowed to select any role from all the possible roles in the database for that item attribute. More will be said later about the *Workflow Directory Service* and how it manages and resolves these roles to send notifications to specific users.

A function activity is defined by the PL/SQL stored procedure or external program that it calls. As a PL/SQL stored procedure, a function activity must be defined according to a standard API that accepts standard arguments and can return a completion result. The **Function** property value must be provided as follows: `<package_name>.<procedure_name>`. The coding and testing of this PL/SQL package and its procedures would be outside the *Workflow Builder* environment using an application such as *SQL Worksheet*.

Once the process model has been diagrammed and the appropriate activity and node properties completed, a user has the option of saving the process definition to a flat file (.wft) on a local disk, or to save it directly to the database. In both cases Oracle Workflow automatically validates the process definition for any invalid or missing information and displays what it finds in a *Workflow Error* verification window. Flat files of the process definitions are useful as backups or to integrate definitions from an external source, however the *Workflow Engine* will only be able to interpret definitions stored to the database.

Figure 5.5 on the next page shows an Entity-Relationship Diagram (ERD) of the tables used to store process definitions. There are of course many other tables to specify application specific variables, but, for the purposes here, they are not crucial to the process definition. Dotted lines indicate that formal foreign key constraints were not found, and the relationship is

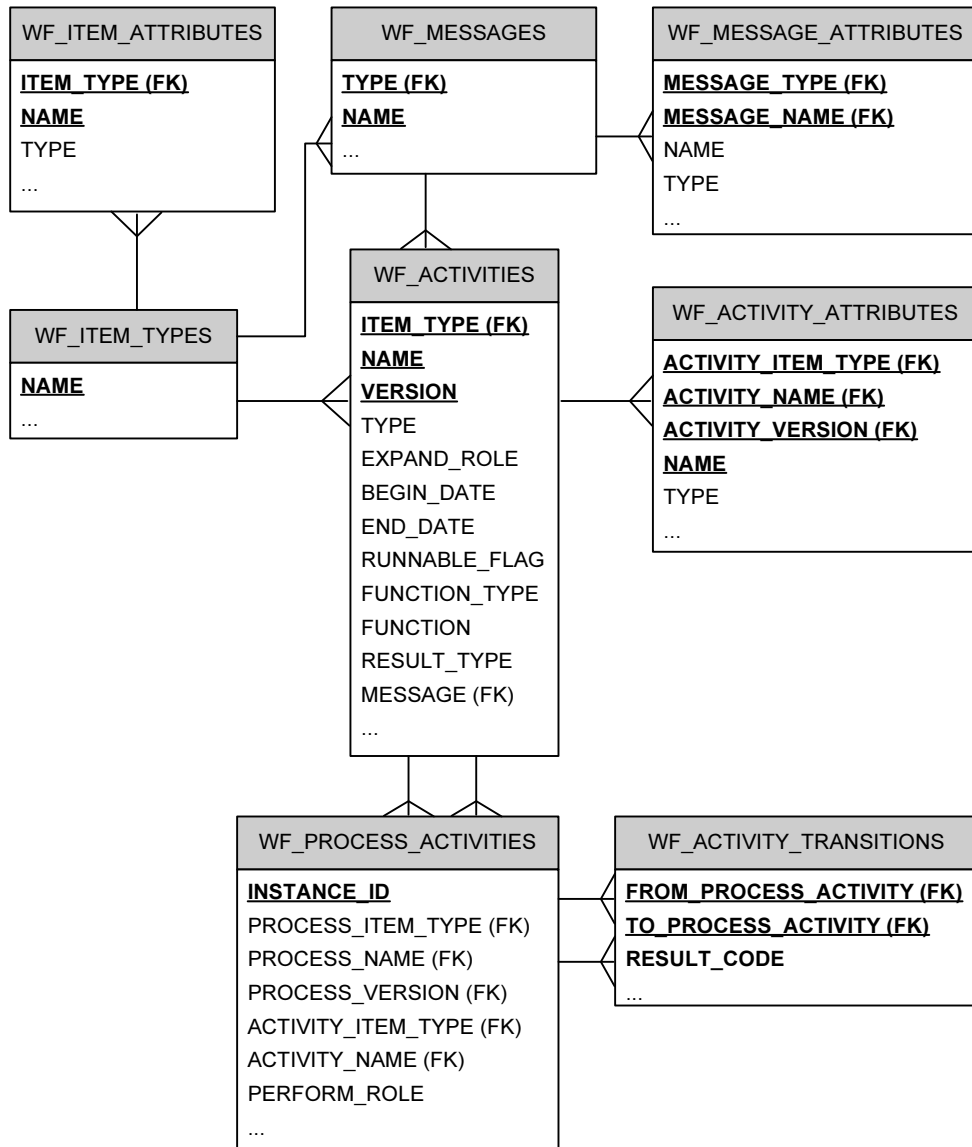


Figure 5.5: Workflow tables that store process definitions in the database

therefore assumed based on the inclusion of reference fields.

Figure 5.5 shows that `WF_ITEM_TYPES` will have multiple `WF_ACTIVITIES` (which includes process activities). Each time an activity is used in a particular process activity, a record is added to the `WF_PROCESS_ACTIVITIES` table to represent node information. Since the same activity can be included many times in the same process, an `INSTANCE_ID` is used to uniquely identify nodes in the process. Note that this is also where the `PERFORMER_ROLE` for notification activities are specified. Each transition between nodes is also captured in the `WF_ACTIVITY_TRANSITIONS` table along with the `RESULT` necessary to to invoke that transition. `WF_ITEM_ATTRIBUTES` represents the **data container** since these attributes will specify the pertinent application data for the item type. `WF_ACTIVITY_ATTRIBUTES` and `WF_MESSAGE_ATTRIBUTES` define application variables of various scope.

Note that the `ACTIVITIES` table also contains a `VERSION` attribute. This allows an administrator to continuously modify and improve the business process, and its definition, without interfering with executing workflows based on older definitions. Oracle Workflow assigns a version number to each new activity that is created. It also updates the version number whenever you make changes to an existing activity. Therefore it saves the new version of the activity to the database without overwriting older versions of the activity. In Oracle Workflow, activities also have dates of effectivity so that at any point in time, only one version of the activity is “in effect”. If a process is running, The workflow engine uses the version of the activity that was in effect when the process was initiated. It does not switch versions of the activity mid-way through the process. The process itself is also defined as an activity with a version number, so the process definition always remains constant until the process instance completes.

Instantiating the correct activity version is only one of the aspects of process instantiation and control for which the workflow engine is responsible. Process definitions provide the blueprints, it is now up to the workflow engine and some ancillary services to successfully drive the process instances and manage the business data associated with each.

5.4 Process Instantiation and Control in Oracle Workflow

Processes are instantiated from the process definitions stored in the workflow tables. Three services, that are part of the Oracle database server, are involved in what usually constitutes the Workflow Enactment Service referred to in the WFMC's Reference model (Hollingsworth, 1995). They are the Workflow Engine, the Notification System, and the Business Event System. These systems manage process instances via PL/SQL APIs and record workflow history data in the shaded tables shown in figure 5.6 on the facing page.

5.4.1 The Workflow Engine

The *Workflow Engine* manages all automated aspects of a workflow process for each item. The engine is implemented in server-side PL/SQL and is activated whenever a call to a workflow procedure or function is made. Since the engine is embedded inside the Oracle database server, if the Workflow server goes down for any reason, the Oracle database server is able to manage the recovery and transactional integrity of any workflow transactions that were running at the time of the failure. Additionally, Workflow Engines can be set up as background tasks to perform activities that are too costly to execute in real time. The Workflow Engine performs the following services for a client application:

- It manages the state of all activities for an item, and in particular, determines which new activity to transition to whenever a prerequisite activity completes.
- It automatically executes function activities (execution is either immediate or deferred to a background engine) and sends notifications (via the Notification System)
- It maintains a history of an activity's status.
- It detects error conditions and executes error processes.

The first API which is invoked to initiate a new workflow process instance is `CreateProcess`. This API creates a new process instance from the process

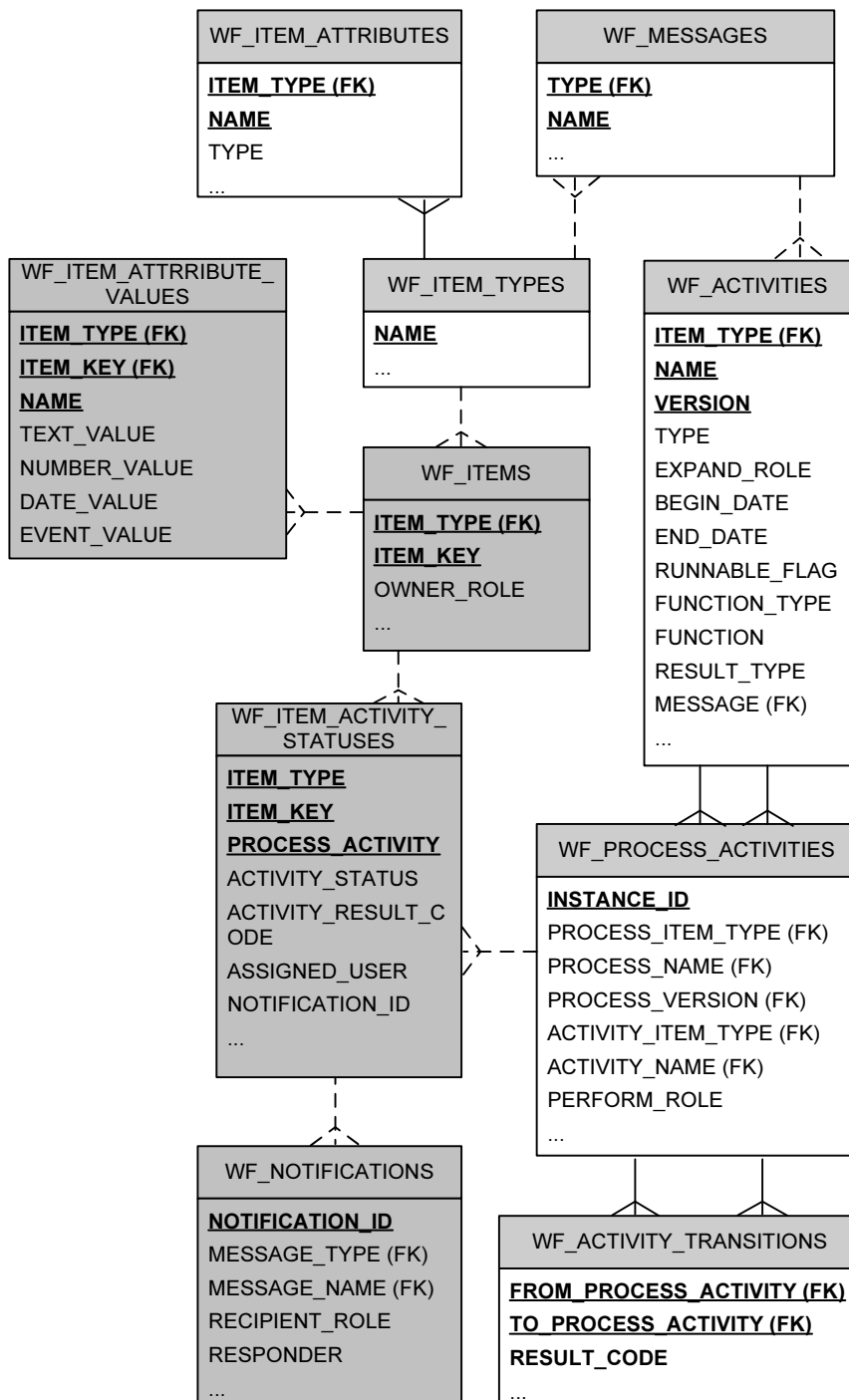


Figure 5.6: Workflow tables that store history information of process instances

definition stored in the workflow tables mentioned in the previous section. At this stage the item attributes must be initialized with start up values (typically supplied by the user) before the next API (`StartProcess`) can be called. Alternatively, the `LaunchProcess` API is a wrapper that combines `CreateProcess` and `StartProcess`. Each new process instance is identified by a unique `Itemkey`, which is either supplied by the user, or is set via a function or procedure before calling the `StartProcess` API.

Upon starting the process, the Workflow Engine identifies and executes the *Start* activity. After each activity completes, the Workflow Engine determines the next activity to transition to and continues to drive through the process, automatically executing all function activities, until it comes to an activity that interrupts the flow such as deferred activities, notifications with responses, blocking activities, and wait activities. When the Workflow Engine encounters one of these activities it sets the audit tables appropriately and returns control to the calling application. In the case of Notifications it also calls on the *Notification System* to notify the recipients. The workflow process is left in an unfinished state until it is started again. The process can be restarted by the *Notification System*, such as when a user responds to a notification; by the background engine, such as when a deferred activity is executed; or by the *Business Event System*, such as when an event message is dequeued from an inbound queue and sent to the workflow process. Once the process is restarted the engine will continue driving through the process in this manner until it encounters an *End* activity. In between the Start and End activities, the state of a workflow item is defined by the various states of all activities that are part of the process for that item. The engine changes activity states in response to an API call to update the activity. The status of the activities are updated by the engine as follows:

- Active - activity is running.
- Complete - activity completed normally.
- Waiting - activity is waiting to run.
- Notified - notification activity is delivered and open.
- Deferred - activity is deferred.

- Error - activity completed with error.
- Suspended - activity is suspended.

In addition the API's already mentioned, there are several other engine API's that manage the process, activity statuses and the item attributes that contain the data passed between activities. The `AssignActivity`, in particular, assigns or reassigns an activity to another performer before the activity is transitioned to. For example, a function activity earlier in the process may determine the performer of a later activity and use this API to set it to its new value (see chapter 7). The `Get/SetItemAttribute` API's are also often used in functions to access process instance data.

5.4.2 The Notification System

When a notification activity is encountered, the engine makes a call to the *Notification System's* `Send` or `SendGroup` API to send the notification. These APIs do the following:

- Confirms that the performer role of the notification activity is valid by querying the *Workflow Directory Service*.
- Identify the notification preference of the performer role for the Notification activity.
- Look up the message attributes for the message. The Subject and Body of the message may include message attributes of source `SEND`, which the API token replaces with each attribute's current value when creating the notification. If a message includes a message attribute of source `RESPOND`, the API checks to see if it has a default value assigned to it. The procedure then uses these `RESPOND` attributes to create the default response section of the notification.
- "Construct" the notification content by inserting relevant information into the Workflow Notification tables.
- Update the notification activity's status to `NOTIFIED` if a response is required or to `COMPLETE` if no response is required.

Once a user responds to a Notification, the `Respond` API is invoked in either the `RESPOND`, `FORWARD` or `TRANSFER` mode depending on the action taken by the user. It executes any post-response processing (associated with a “Post Notification” function), updates the values of any attribute set to `RESPOND` and sets the notification as Complete. If the notification message prompts for a response as specified in the Result tab of the message’s property page, that response value is also set as the result of the notification activity. Finally, `Respond` calls the Workflow Engine’s `CompleteActivity` API to inform the engine that the notification activity is complete so it can transition to the next qualified activity.

5.4.3 The Business Event System

The Business Event System shown in figure 5.7 on the next page is an application service delivered with Oracle Workflow that uses Oracle Advanced Queuing technology to communicate business events between systems. The Business Event System enables point-to-point messaging between systems, messaging hubs, and distributed applications messaging. Workflow administrators can register subscriptions to business events that are significant to their systems to trigger custom code, message propagation, or workflow processes. These events are represented in the process definition by Event activities that can be modelled using Workflow Builder. These activities are often used to start a process, or to continue a process waiting for a specific business event to occur.

When a business event occurs, the Event Manager executes subscriptions on the event. For local events, the subscribing code can be executed synchronously, in the same database transaction as the code that raised the event, or asynchronously, deferring costly subscription processing to a later time in order to return control more quickly to the calling application. Events can also be received asynchronously from external systems. Subscription processing can include executing custom code on the event information, sending event information to a workflow process (using special XML function activities included with Oracle Workflow), and sending event information to other queues or systems. If the event was included as an event activity in the workflow process definition, the subscription code for that event would be executed. Once it has completed, the `CompleteActivity` API must be

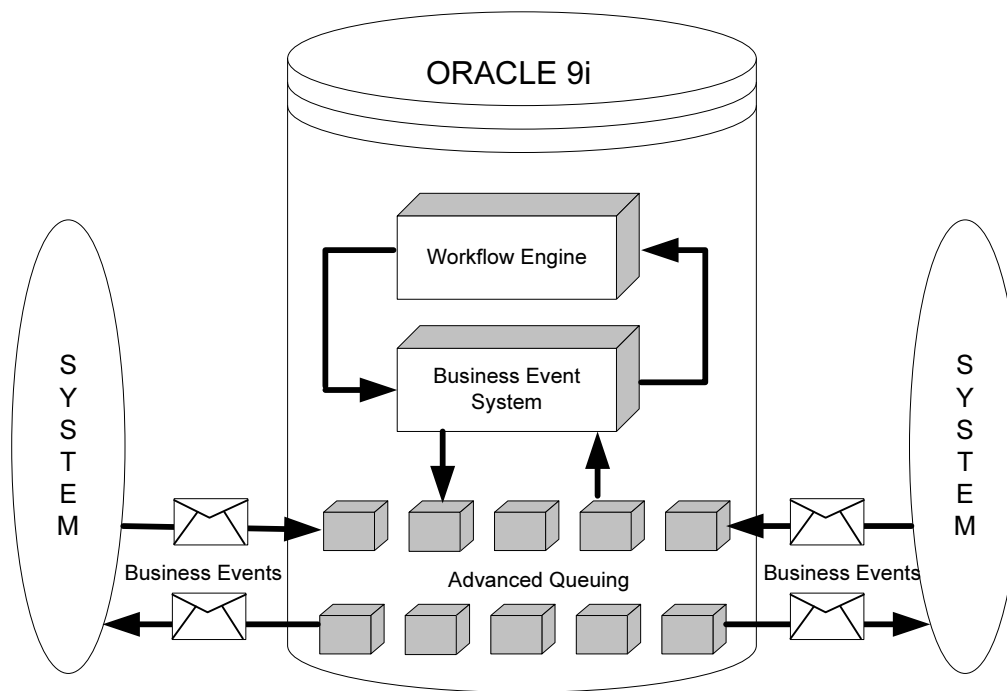


Figure 5.7: How the Workflow Engine interacts with the Business Event System

called to restart the workflow engine.

5.5 Interaction with Users and Applications in Oracle Workflow

There are several ways in which Oracle Workflow may interact with external applications; however, the preferred method is via the Business Event System. As discussed in the previous section, the Business Event System consists of the Event Manager, which lets administrators register subscriptions to significant events, and Event Activities, which model business events within workflow processes. This allows the workflow engine to incorporate information from other systems into the current workflow. However, the workflow may also need to initiate external applications to perform some function outside the scope of the Oracle database server. Once again this can be accomplished by raising an event via the Business Event System and allowing it to interface with the external program via an out-bound queue.

The same functionality could also be achieved using function activities to place items directly into queues.

Oracle Workflow users includes end-users who initiate and execute activities in workflow processes as well as the administrators who define, monitor and control those processes. Both types of users are given access to a variety of information and functionality via Oracle Workflow's web interface which runs on the Oracle HTTP server.

The *Workflow Directory Service* will authenticate users before displaying any secured workflow information such as a user's personal worklist of notifications. It will also only allow administrators access to restricted Event Manager and Workflow Monitor functions. The Workflow Directory Service uses views based on local users already registered on the database. Alternatively the OID (Oracle Internet Directory) service can be used to manage both internal as well as external Internet users, although this service is not available in the stand-alone version of Oracle Workflow.

The *Oracle Workflow Homepage* shown in figure 5.8 on the facing page gives users and administrators central access to all the web-based features of Oracle Workflow. The features can roughly be divided into those pertaining to notifications, managing workflow user preferences, monitoring workflow instances and setting up business events (all those in the right-hand column of the page). Workflow processes are typically not launched from this interface although it does allow administrators to test their definitions via the *Launch Processes* page. Process instances are usually launched within one of the workflow-enabled applications provided with Oracle's E-business suite, such as Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), and Human Resources Management Systems (HRMS). If Oracle workflow is used as a stand-alone product, administrators can set up events using the Business Event Manager to invoke a workflow process instance automatically, or they can design custom web pages¹ that use HTML controls to request process start-up info and call the `CreateProcess` and `StartProcess` API's, from where the Workflow Engine will take over.

Administrators and users can view the progress of a work item in a workflow process by connecting to the Workflow Monitor using a standard Web

¹Oracle Workflow has provided sample workflow processes which are initiated by PL/SQL procedures called from the hyperlinks on the "Demonstration Page"

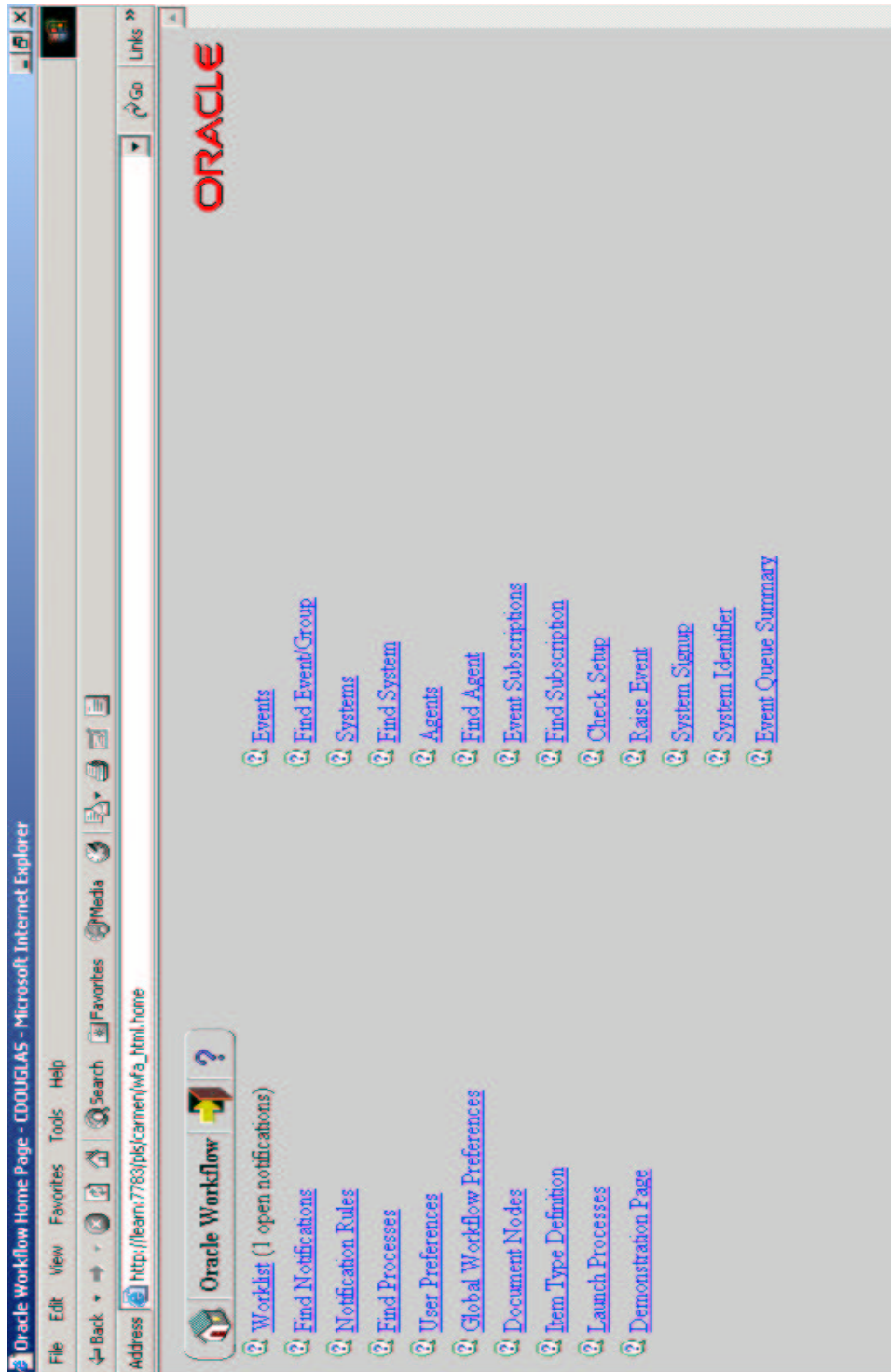


Figure 5.8: Workflow users can access workflow features via a Workflow Homepage

browser that supports Java. The Workflow Monitor displays an annotated view of the process diagram for a particular instance of a workflow process, so that users can get a graphical depiction of their work item status. The Workflow Monitor also displays a separate status summary for the work item, the process, and each activity in the process. Oracle Workflow Manager is an additional tool available with Oracle9i Release 2, that administrators can use to administer workflows.

5.5.1 Viewing Notifications and Processing Responses

Oracle Workflow lets an administrator include users in the workflow to handle activities that cannot be automated, such as approvals for requisitions or sales orders. This is done through electronic notifications which are routed to a role, which can be an individual user or a group of users. Any user associated with that role can act on the notification. Each notification includes a message that contains all the information a user needs to make a decision. The information may be embedded in the message body or attached as a separate document. Oracle Workflow also allows the user to interact with the workflow by responding to a notification. The Notification System will process the response as described in section 5.4.2 and the Workflow Engine then interprets the response to decide how to move on to the next workflow activity. Other users to which the communal task notification was sent, will also no longer have access to that notification.

This section discusses the different ways people involved in a workflow process can view and respond to workflow notifications. A user can view notifications using any one of three interfaces depending on the notification preference setting for the role that user belongs to in the Oracle Workflow Directory Service. Users can receive an e-mail for each individual notification, receive a single e-mail summarizing all their notifications or query the Workflow Notifications Web page for their notifications.

Electronic mail (e-mail) users can receive notifications as e-mail messages if their notification preference is set to “Plain text mail”, “HTML mail”, or “Plain text mail with attachments” in the *User Preferences* web page and the workflow administrator sets up the *Notification Mailer* to run. The preference determines how the message will be displayed and also how response options are presented to the user which will be chosen will depending on the

target mail system's capabilities. All three types of e-mail notifications will allow the user to respond via some template which provides specific sections in the body of the message for the user to enter values and indicate the response action to be taken (e.g. "Approve" or "Reject").

When user responds to a notification by e-mail, the reply message must include the notification ID (NID) and access key from the original notification message. The Notification Mailer can process the response properly only if the correct NID and access key combination is included. The notification access key is a distinct random key generated by the Notification System for each NID. The access key, thus, serves to authenticate users who actually received the notification, thereby allowing them to respond to the notification. If users do not wish to receive and respond to individual notifications via e-mail they can still receive periodic e-mails giving a summary overview of the notifications awaiting their attention in the Notifications Web page.

Web and e-mail users can access the *Notifications* web page (directly by typing in the URL, by clicking the *Worklist* link on the *Oracle Workflow HomePage* or by calling this function from an Oracle Applications form) to see their outstanding work items, then navigate to additional pages to see more details or provide a response. When users view their notifications from the Notifications Web page they are simply querying the workflow history tables from this interface, and the Notification Mailer is not involved. The query will generate a list of open notifications for the current user, as shown in figure 5.9 on the next page.

The user may select any of the notifications to view more detail (figure 5.10 on page 79). In the *Notification Detail* page, the full details of the notification appear in the upper frame, and the response section of the notification appears in the lower frame. The upper frame contains the message specified for the notification activity with the instance variables supplied. The response frame prompts for message attributes that were specified as being of the "Respond" type using fields or poplists. The type of notification and the response type specified for the activity will determine which option buttons are presented to prompt the user for some action. The user can also choose not to respond by simply closing the notification, or re-assign the activity to another user. If the the user should not be allowed to reassign the notification a special attribute, `HIDE_REASSIGN`, is included in the message

Select	Priority	Type	Subject	Sent	Due
<input type="checkbox"/>	1	Requisition	Requisition R52709149, Headset for 121.00 requires your approval	01-FEB-2001	01-FEB-2001
<input type="checkbox"/>	1	Requisition	Requisition R52709146, Airfare for 432.00 requires your approval	01-FEB-2001	01-FEB-2001
<input type="checkbox"/>	1	Requisition	Requisition R52709144, Office Supplies for 245.00 requires your approval	01-FEB-2001	01-FEB-2001
<input type="checkbox"/>	1	Requisition	Requisition R52709139, Reference Books for 189.00 requires your approval	01-FEB-2001	01-FEB-2001
<input type="checkbox"/>	1	Requisition	Requisition R52709132, Office Supplies for 98.00 requires your approval	01-FEB-2001	01-FEB-2001
<input type="checkbox"/>	1	Requisition	Requisition R52709131, Office Desk for 2,599.00 requires your approval	01-FEB-2001	01-FEB-2001

Close Reassign...

Figure 5.9: The “Notifications Web page” shows open notifications for the current user

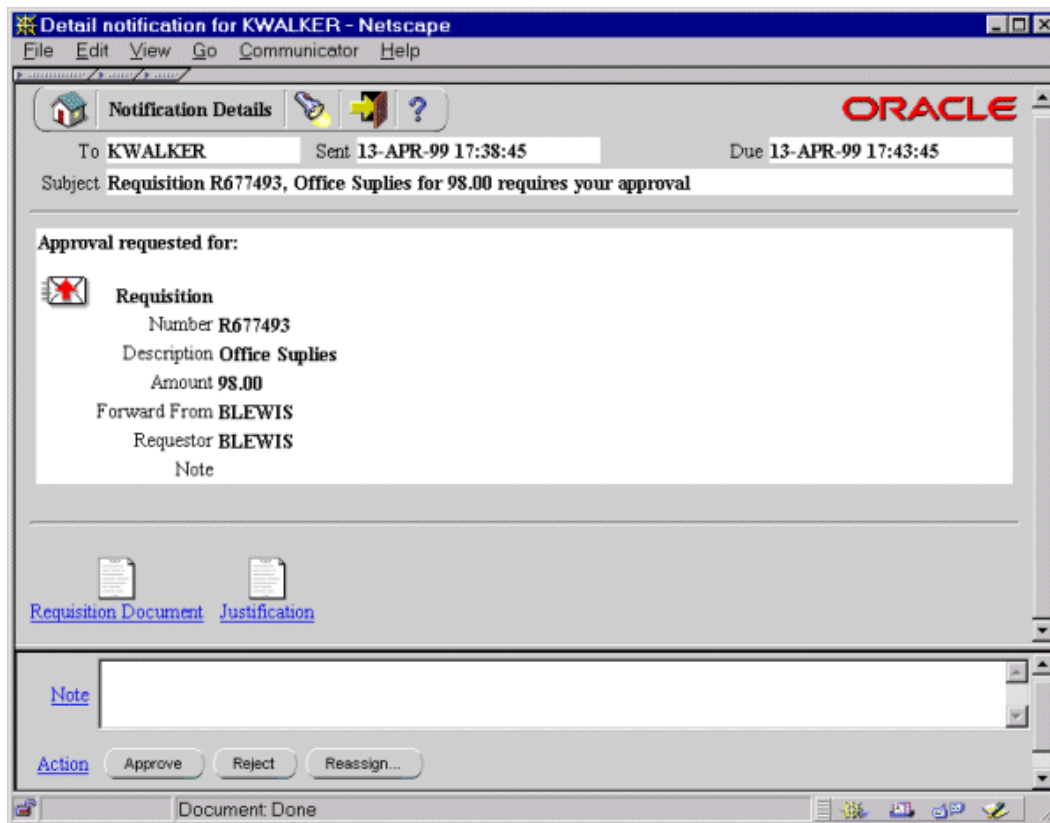


Figure 5.10: The “Notification Detail” page.

definition.

Once a response is submitted, the Notification Detail page returns to the worklist of Notifications, where the notification just responded to now displays a status of Closed and the user is allowed to delete it from his/her worklist. Once closed, users can view the notification details again, but they will not be able to alter their response.

5.6 Conclusion

Although this chapter seem to have presented quite a lengthy discussion on Oracle Workflow’s features, it is in no way representative of all the features supplied with our chosen product. The aim was to provide information on how the product is involved in the lifecycle of a workflow process, from definition to execution.

Oracle Workflow Builder was investigated as the tool with which *process*

design and definition is achieved. This tool provides a GUI process model which is easy to design and alter. Process definitions are subsequently uploaded into database tables from where they are maintained and instantiated as process instances.

The Workflow Engine is primarily responsible for *process instantiation and control* through the use of PL/SQL API calls that are automatically triggered in order to drive the process forward. The Engine also sends and receives API calls to and from the Notification and Business Event systems. These systems halt the workflow process, allowing asymmetric execution of external activities (performed by users and applications) until control is transferred back to the workflow engine that feeds this information into the relevant process instances.

The web interfaces provided by Oracle Workflow allows for the *interaction with users and applications* in a convenient and centralized manner. Users can access their worklists from any location and administrators can register and manage event subscriptions which will actively listen for business events that the workflow engine must respond to. In addition, a monitoring facility is available so that users can track the progress of their requests and administrators can track the performance of workflows and monitor the activities of their users.

As can be seen from the previous paragraph, Oracle Workflow closely follows the three functional areas suggested by the WfMC for workflow management systems. However, the workflow terms used do differ somewhat from those used in chapter 2. Section 5.1 showed the mapping of these terms. Processes become item-types, tasks are activity nodes which could represent a notification, event, or subprocess. These activity nodes are connected to each other with transitions which represent business conditions or rules. Even more significant is the integration of the data container not as a separate electronic document but as attributes of the process itself. Apart from this, Oracle Workflow environment does not have a clear access control service on which to map the CoSAWoE model. These and other implementation issues form the crux of this dissertation and will be discussed in more detail in the following two chapters.

Chapter 6

CoSAWoE: Administration in Oracle Workflow

In the previous chapter, Oracle Workflow was introduced as a commercial workflow product. The features provided with this product was investigated according to the functionality described by the Workflow Management Coalition. This chapter will now show how some of these features will be used according to the administrative guidelines proposed by the CoSAWoE model. As stated in chapter 4, the model components affected will relate mostly to separation of duty administration. The role engineering and object design components of the CoSAWoE model represents methodological issues rather than implementation. Since Oracle Workflow support, albeit not fully, the role-based and object principles on which these methodology aspects are based, the role engineering and object design aspects will not be discussed here.

However, the means whereby Oracle Workflow implements access control policies, such as *order of events* and *strict least privilege*, are less obvious and crucial to the essential functionality prescribed by the CoSAWoE model. Access control can be specified at a task level as well as at a finer level of granularity (data items). *Separation of duties* can be implemented separately in Oracle Workflow, by specifying static conflicts and enforcing them either during design-time or checking them later when the process is executed.

First, however, the access control features of Oracle Workflow that are available apart from any model considerations must be explained.

6.1 Access Control Features provided with Oracle Workflow

Oracle Workflow does provide an “Access Protection” feature, but this is not to be confused with an “Access Control” service as it relates to information security. Oracle Workflow’s access protection is a feature that prevents workflow seed data created by a “seed data provider” from being modified by a “seed data consumer”. For example, a development team at an organization’s headquarters may create a custom workflow process that it wants to deploy to teams at other regional offices. The headquarters team, as seed data providers, may want to protect certain data as “read-only”, while allowing other data to be customized for the regional offices to alter to their offices’ needs. This feature also allows a seed data provider to “upgrade” any existing protected seed data with new versions of that seed data, while preserving any customizations made to customizable seed data.

Oracle Workflow assigns a protection and customization level to every workflow object definition stored in the database and requires every user of Oracle Workflow to operate at a certain access level. The combination of protection, customization, and access levels make up the access protection feature and determines whether a user can modify a given workflow object. The “Access protection” feature, therefore, only protects the workflow definitions and does not control user access to the workflow activity instances or the application data.

No explicit mention is made about “Access Control” in any of the Oracle Workflow documentation (Chang & Jaeckel, 2002). However, this does not imply that there is no control over user interaction with workflow activity instances. Oracle Workflow does provide shades of discretionary and role-based access control (see chapter 3) in the specification and assignment of tasks.

One of the setup steps for Oracle Workflow is to map Oracle Workflow’s directory service to the users and roles currently defined in the organization’s directory repository by constructing views based on those database tables. The views in question are `WF_USERS`, `WF_ROLES` and `WF_USER_ROLES`. The roles can be either individual users or a group of users. Role-hierarchies are also supported. However, since the views are based on users and roles spec-

ified for the database system, those roles may not be sufficient to support access to workflow objects. As such, administrators can specify ad-hoc users, `WF_LOCAL_USERS`, and roles, `WF_LOCAL_ROLES` and `WF_LOCAL_USER_ROLES`, which are more appropriate to the specific workflow environment. The Notification System uses these tables and views to send notifications to the performer role specified for notification activities (see section 5.3).

The level of access control can indirectly be built into the workflow process definition through the way in which the administrator uses some of the Oracle Workflow Builder features (discussed in chapter 5).

6.2 Specifying Access Control with Workflow Builder

Both human participants and other programs can be seen as “users” of the workflow process and any data it propagates. From an access control perspective, administrators would have to ensure that these users can only access the workflow in a predictable and appropriately restrictive way. For other *systems*, the administrator would set up event activities to handle incoming and outgoing data. Event activities will, in turn, be triggered and handled by the Business Event System during run-time. Therefore, the way in which these systems will access workflow data and tasks will be well-defined and automated according to strict rules specified by the administrator. Any deviations will automatically be rejected by the Workflow Engine. The behaviour of *human* users are, however, much less predictable. The inheritance capabilities of the RBAC role hierarchies, in particular, can have unexpected side effects which may compromise the integrity of a workflow process. Therefore, the remainder of this dissertation will view users as *human* users and address problems that might arise from *their* involvement in workflow tasks.

At a task-level, users are given access to a certain notification activity based on the `Performer Role` specified for that notification node. As mentioned in the previous chapter, if a role-type **item attribute** is used, the administrator effectively allows the owner of the process to select the “role-players” who will be involved in that process instance when the process is initiated. For example, a “Requestor” and an “Approver” item type attribute

of the type `Role` can be defined for a “Vacation Request” process which will only be populated during run-time. The process owner then needs to specify the specific roles or individual users who will act as “Requestor” and “Approver” respectively when the process instance is created. This would at least provide some form of *discretionary* access control, since the owner of the process gives permission to certain users to access certain tasks. At worst, an administrator might even design the process in such a way that the “Requestor” can specify the “Approver” during the creation activity and thereby hand-pick the person who will ultimately approve the request.

Some of the obvious access control pitfalls of this approach can be overcome by providing specialized function activities. These functions can programmatically determine the item attribute values based on a pre-defined role/user hierarchy, or at the very least, check the selected role/user against access control policies before executing notification activities for which such a dynamic assignment was made.

Affecting access control in this way, however, places an increased administrative burden on the workflow designer or administrator. Since the item attributes in question is specific to a particular process definition, there would also be issues as far as maintenance, integration and scalability is concerned. Certainly when separation of duties is considered, using a hard-coded manager hierarchy¹ will ensure such separation is achieved, however much of the flexibility will be lost in the process design, and maintenance of the hierarchy itself will be an ongoing task. As such, we would recommend that administrators assign notification activity nodes to constant roles, managed by the Workflow Directory Service, to alleviate the maintenance burden. This would enable *role-based* access control to activities.

6.2.1 Role-based Access Control at Activity Level

The performer role can be set to any **Constant** role that exists in the database. The term “role” is somewhat ambiguously used in Oracle Workflow since individual users must also be specified as roles in order to be assigned to notifications during design or to receive them in their worklists at run-time.

When a constant role is chosen for a notification activity, the role referred

¹Such a hard-coded manager hierarchy is used as part of the “Requisition” demonstration item type included with Oracle Workflow

to here should thus represent a grouping of users according to some job description in order to obtain the maintenance advantage described earlier. The administrator now only needs to ensure that a specific enough role exists and that the users assigned to it is restricted according to who should have access to that notification activity. The CoSAWoE model provides guidelines on how this can be achieved in its role engineering component (see chapter 4).

Note, however, that the role used here only serves to group users, and the permissions to objects usually associated with such roles really holds no value in the Oracle Workflow environment. This is mainly since the “document”, which is the object on which permissions are specified, is non-existent!

In Oracle Workflow the application data propagated by the workflow is resident in a set of item attributes, the instance values of which are stored in `WF_ITEM_ATTRIBUTE_VALUES`. Also, the Directory Service is not involved beyond granting users access to a specific notification. Therefore, once a user responds to a notification, the Workflow Engine and not a specific user is responsible for updating the history tables.

6.2.2 Strict Least Privilege at Item Attribute Level

In order to control access to a finer level of granularity for workflow data, the inclusion and properties of item attributes used in notifications must therefore be controlled. A notification activity presents information to the user through a message template. Specific **Message Attributes** which maps directly onto **Item Attributes**, are contained in messages. These attributes can be specified as **Send** or **Respond**, and the setting will apply to any user who has access to that activity.

The administrator will set message attributes to **Send** if it can only be *viewed* during the notification activity. Whomever responds to the notification first will be allowed to *change* an attribute if it is set to **Respond**. Once someone has responded to the notification, the values of those attributes are saved to the relevant table and the notification is removed from other user’s worklists. The user who responded is also only allowed to view the changes he/she has made. Values of such attributes can only be changed again by another activity in the workflow to which users will once again be granted access based on their role and the performer specified.

It is possible to control individual user’s access to an item attribute via

a “Post Notification” function specified as part of the node attributes. Such a function is only triggered after an individual has responded to a message, but before changes to the database is made. The function can then apply particular access rules to specific item attribute values that were changed and raise an error if the action is not allowed by the responder.

From the discussions above it is clear that it is possible to ensure **strict least privilege** and **order of events**; two of the requirements for access control in workflow specified in chapter 3. The CoSAWoE model caters for these requirements during administration through the use of *object design* and *role engineering*, but since document hierarchies and role-based access control are not fully supported in Oracle Workflow, careful task design and additional programmatical checks are used instead. *Separation of Duties*, the third access control requirement and a critical component of CoSAWoE, is not supported and will also require additional specification by the administrator.

6.3 Specifying SoD in Workflow Builder

Separation of Duties is not enforced in Oracle Workflow other than by the administrator’s selection of abstract roles when assigning the notification activities’ performer role. If sufficiently restrictive roles are available, the administrator can at least make sure that two conflicting notification activities are assigned to mutually exclusive roles. However, there is no guarantee that role engineering was adequately applied, and much of the success of this approach will depend on the administrator’s knowledge of how database roles were created and user assignments were made.

In order to check user assignments to tasks in Workflow Builder, and therefore enforce **Static Separation of Duties (SSoD)**, additional programming will be necessary. First, the static conflicts will need to be specified using database tables. Then PL/SQL triggers will need to be created on these tables, as well as on the process definition tables where task assignments are recorded. For example, a trigger will need to be specified on the `PROCESS_ACTIVITIES` table to check the `PERFORMER_ROLE` against information in the conflict tables each time a new notification activity node is inserted into the process definition. The SoD administration tool developed by Perelson et al. (2001) also uses examples of such triggers to assign users,

roles and tasks without violating business constraints.

In Oracle Workflow, however, these triggers will only be triggered once the definition is saved to the database, and will only raise an error to indicate this to the administrator at that time. Another limitation of this approach is that users may assume multiple roles during run-time based on role-hierarchies. This would place notifications in a user's worklist that are assigned to ANY of the roles he/she belongs to irrespective of his/her involvement earlier in the workflow.

Therefore, **dynamic separation of duties (DSoD)** would be necessary regardless of any static checks that may or may not have been performed. This approach does not prevent user assignment during administration, but rather checks such assignments once the workflow process is running. Still certain steps still need to be followed during process definition to setup the workflow environment for run-time enforcement.

First, the administrator must specify conflicting activities and conflicting users in two database tables as shown in figure 6.1 on the next page. As can be seen from this example, the `WF_CONFLICTING_TASKS` table refers to the activity definitions in the `WF_ACTIVITIES` which are conflicting. Basing the conflicts on this table will imply that once a conflict has been specified between two activities, that conflict will exist regardless of which subprocess they are specified in. If the conflict should be restricted to a particular subprocess, this conflict table should be based on the `WF_PROCESS_ACTIVITY` table instead, and the parent activities included as fields in addition to the activity name fields. The `WF_CONFLICTING_USERS` table relates to the users in the `WF_USERS` or `WF_LOCAL_USERS` tables. The relationship between those users that causes the conflict is also specified for error messaging purposes. The specification of conflicting roles in a separate table will unfortunately have no real value. This is due to the fact that in Oracle Workflow, users do not specify a particular role when logging on to perform workflow tasks. The history of which role a user assumed when performing a particular activity would, therefore, not be available for dynamic checking.

Within Oracle Workflow Builder the administrator must now provide a function activity to eliminate conflicts before notifications can be sent to users. We attempted to create a re-usable procedure that can be used to filter users before *any* notification activity. This procedure does, however, require

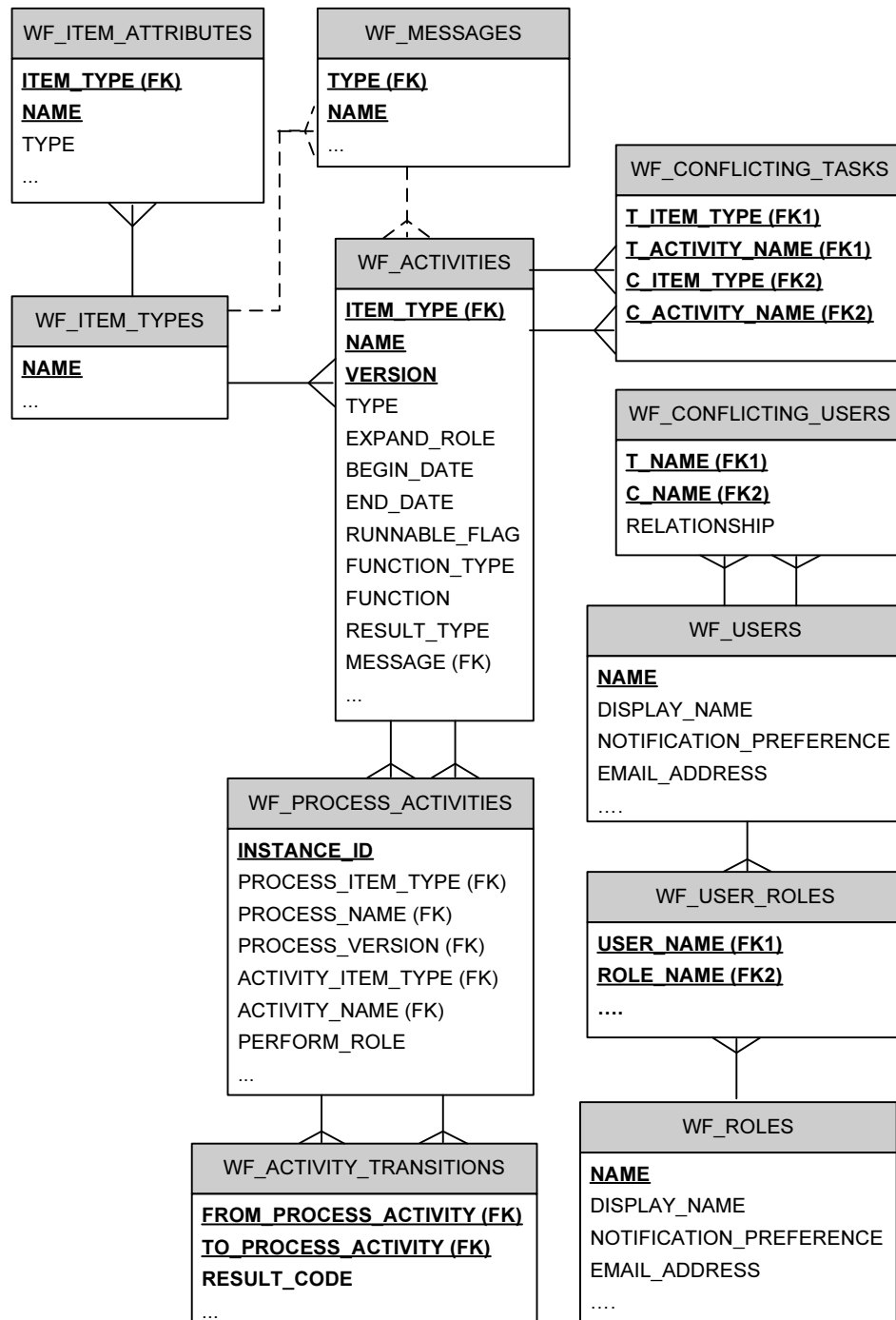


Figure 6.1: Specifying conflicting entities via database tables in Oracle Workflow

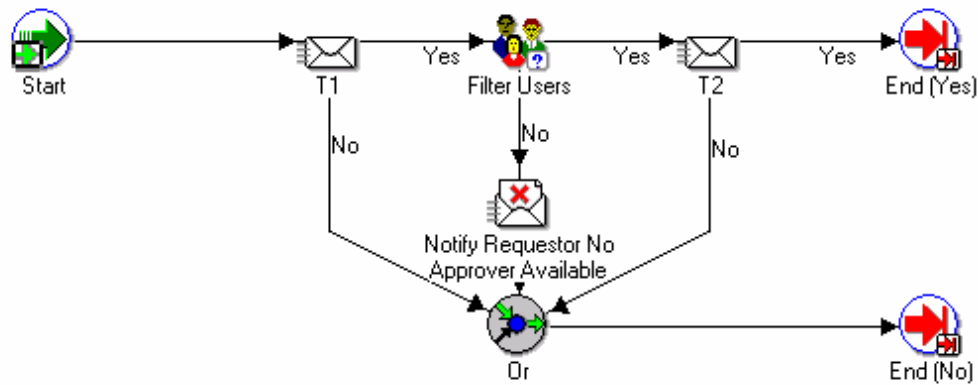


Figure 6.2: Inserting a function activity in the process definition to enforce DSoD

that the administrator use role constants and not dynamic item attributes, when assigning notification activity nodes' performer role.

The actual checking of dynamic conflicts can also be done once a user attempts to respond to a notification. If a conflict does occur an error would be raised and the user will be prevented from completing the activity. However, such an approach would violate good user interface design guidelines by presenting users with notifications to which they in fact will not have access.

Therefore, a more elegant solution can be implemented by inserting a re-usable filter function activity before each notification activity node in the process design, so that user lists can be pruned appropriately before the Notification System sends out such notifications to potential users. Figure 6.2 shows how such a function ("Filter Users") is inserted into a generic process between two notification activities. The following chapter will discuss the internal workings of the PL/SQL procedure attached to this function activity.

6.4 Lessons Learnt

The following lessons could be learnt from the discussions in the previous sections. When using Oracle Workflow Builder to implement access control features:

- Role-based access control can only be implemented at a task level. If a finer level of granularity is needed, to restrict the access to a piece of data so that it may only be viewed or changed during a particular task,

this can only be implemented through item attribute properties which apply to all users. Additional programming would be required if the user's particular involvement in the workflow should also be evaluated for individual pieces of data.

- Users would be able to over-ride such role-based assignments by re-assigning a notification once they have claimed it to another workflow user at their discretion, unless the Re-Assign option is explicitly specified as hidden during message design.
- The explicit specification of Separation of Duties is not supported at all in Oracle Workflow Builder. Role engineering is assumed and the administrator assigns notification activities at his/her discretion.
- Static Separation of Duties can be enforced but additional coding in the form of PL/SQL triggers would be needed.
- Dynamic Separation of Duties can also be achieved, although additional conflict tables must be created and populated outside Workflow Builder and function activities must be inserted into the process definition before each notification activity.

6.5 Conclusion

As a commercial workflow product, Oracle Workflow is quite powerful and well suited to complex workflows. This chapter, however, highlighted some deficiencies as far as access control is concerned. Although, a user directory service is available which allows roles to be assigned to activities, the manner in which these assignments are made greatly depends on the skill of the administrator. Therefore, this chapter's aim was to provide guidelines to administrators and suggest some "best practices" for implementing an access control model based on CoSAWoE.

The CoSAWoE model specifies three functional requirements for context-sensitive access control in workflows, namely order of events, strict least privilege and separation of duties. This chapter showed how the first two policies can be achieved through the design of notification messages and the assignment of such activities to role constants. These role assignments can also be

statically checked against separation of duty conflicts (which are specified in database tables) using database triggers and thereby preventing the assignments from being made. However, it was argued that this approach would be insufficient to enforce dynamic separation of duties. Therefore it also showed how a filtering activity can be inserted before sensitive notification activities to check separation of duty conflicts for the assigned role's users before granting them access to the activity. Only those users who belong to the assigned role, and whose participation would not create a separation of duty conflict, would thus receive a particular activity in their worklist.

The following chapter will discuss the run-time activities and checks that would need to be done in order for this access control method to be successfully implemented in Oracle Workflow.

Chapter 7

CoSAWoE: Run-time Enforcement in Oracle Workflow

The Oracle Workflow Builder administration tool, allows administrators to setup the workflow process in such a way that access control policies can be enforced by the Oracle Workflow Engine during run-time. The previous chapter showed in particular how separation of duties can be catered for. This chapter will focus on how the conflicts specified during administration will now be used to prevent users from receiving such notifications when generating users' worklists. Although chapter 4 excluded the workflow session concept from the CoSAWoE components relevant to this chapter, a short discussion will illustrate the way in which Oracle Workflow achieves similar results using different methods.

The code segments discussed in the following sections is, due to their length, located in Appendix A, and should therefore be consulted while reading this chapter.

7.1 Worklist Generation during Run-time in Oracle Workflow

Chapter 5 discussed how the Notification System is responsible for sending notification messages to individual users in the workflow. When an activity

message is triggered by the Workflow Engine, the Notification System resolves the role specified for the Notification activity node and determines the notification preference for each user attached to the role. E-mail users will be sent notification messages via the Notification Mailer program which will place such messages in the users' e-mail account Inbox. Once a user replies to such an e-mail, the Notification System will verify that only the intended recipient sent the reply (by checking the access key included against the one generated when the notification was sent). Users who access the Workflow Home Page or navigate directly to the Notifications Page can view a list of Notifications. Each user's individual worklist of notifications is generated dynamically by querying the `NOTIFICATIONS` table for any open notifications allocated to one of the user's roles. Once a notification has successfully been responded to, the Notification System closes that notification, which would exclude it from being returned in other user's worklists.

Therefore, worklists do not exist as separate entities within Oracle Workflow's history tables, but are generated dynamically based on the roles assigned to a particular user. Therefore the procedure that prunes the "worklist" is in fact pruning the potential user list for a specified role attached to a notification. This procedure would be attached to the filtering function activity which is inserted before each notification activity that requires separation of duty checking (see chapter 6).

7.2 Pruning the User List for DSoD in Oracle Workflow

A re-usable PL/SQL procedure, attached to the "Filter Users" function activities shown in figure 6.2 on page 89, is responsible for pruning the user list. The `FilterUsers` procedure, included in appendix A, contains the relevant code and will follow the program logic represented by the following steps:

- **Step 1.** Determine the intended users for the activity based on the performer role specified (lines 25 to 27, and lines 75 to 80).
- **Step 2.** Use Set operators to filter out users who responded to conflicting tasks (lines 31, and lines 35 to 43). The `NOTIFICATIONS` table

can be queried or the **Responder** API invoked to determine the individual users who responded to notifications attached to conflicting tasks. If the process owner should also be included, the **Start** activity is specified as a conflicting task to indicate this (lines 46 to 52).

- **Step 3.** Conflicting users can be seen as the same user for conflicting tasks and must therefore also be eliminated according to lines 56 to 66.
- **Step 4.** Once a set of valid users are ready, a new ad-hoc role must be created based on the original role specified for the activity (lines 93 to 95, and line 100). Note that the itemkey, and activity ID must be included in the new role-name since multiple activity instances can exist which utilizes the same source role.
- **Step 5.** The new role must be loaded with the pruned user list, and the notification activity node must be re-assigned to this temporary role (lines 104 to 130).

These generic steps can be followed to prune the user list before assigning them any notification activity.

7.3 Session Control in Oracle Workflow

Users are authenticated whenever they logon to the Oracle Workflow Home page or when they navigate to a secured webpage such as the “Notifications page”. E-mail users will also be authenticated by the e-mail application they are using, and the Notification Mailer will verify responses coming from those systems by using assess keys as described in section 7.1. However the users are not associated with a specific role at that time, and as such a user may assume all the roles assigned to him/her during role administration. These roles will also remain active until the user logs out of the workflow homepage or e-mail application.

WSessions, the specialized form of workflow sessions described in chapter 4, enforces the concept of strict least privilege. This concept defines a workflow session as the time from which a user accepts a task until he or she stops working on that task. This allowed the access control service to assign and revoke user privileges to a particular task for a limited time period only.

Workflow Sessions, are not directly supported in Oracle Workflow, however, the worklists of users are dynamically created according to the roles associated with tasks. What this means is that users will be able to perform tasks only once the Workflow Engine has created the task instance, and the Notification System have created open Notifications for those tasks. A user will, therefore, not be able to access a task outside of the environment provided and controlled through the worklist. As soon as someone responds to a notification, it is closed, removing it from all users' worklists so that it cannot be updated any further. As explained in the previous chapter, users are not allowed to "go back" to previously completed activities unless explicitly modelled in the process design

Assigning ad-hoc roles to notification activities can also be seen as a form of WSessions. This role is specifically created to give only certain users access to a particular task. Then when someone belonging to that temporary role does respond, the task closes and the temporary role is not used any longer. It can also be explicitly removed via a Post-Notification function as soon as someone replies (a procedure called "DeleteAdhocRole" was created for this purpose and is shown in section A.2 of appendix A). Therefore users are allowed access to a task based on a temporary role which is disabled as soon as the task is complete.

These methods would therefore not allow a user to access a task outside their own worklist. More importantly, history tables are also updated by the Workflow Engine acting as a proxy, and therefore users are typically not granted any privileges to these tables. This ensures that users will not be able to affect changes to task data outside of the workflow environment (for example through an SQL editor), since only the Workflow Engine may access those tables.

7.4 Lessons Learnt

The biggest shortcoming of Oracle Workflow is that its architecture does not contain a centralized access control service. Such a service would have aided the workflow engine, and the notification service in the execution of notification tasks. Instead of placing extra function tasks in the workflow process definition (which implies additional error handling and processing overheads),

an access control service would provide a central component which could be used to implement the separation of duty and other access control policies for all notifications.

Since users do not log in with a particular role, it is also not possible to check conflicting roles dynamically. This would allow groups of users to be excluded from having access to a particular task based on the previous roles they activated in the same process instance. This would also require the role information based on the user's log-on information to be stored in the workflow history table `WF_NOTIFICATIONS` along with the user who performed the activity.

7.5 Conclusion

This chapter concluded the proposed solution for implementing the CoSAWoE model in a commercial workflow product such as Oracle Workflow. The key component for run-time enforcement of this model, is worklist generation. In the previous chapter the administration tasks of assigning tasks to roles were discussed. According to these role assignments, the workflow engine, aided by the directory service and the notification system, will place open notification activities in a user's worklist. As discussed, the worklist does not exist as a separate entity, but is generated through a query to the workflow tables when the user logs into the workflow homepage.

At times it may also be necessary to disallow certain users access to a task based on separation of duty constraints. Since a worklist entity for each user does not exist to be pruned, section 7.2 therefore discussed the pruning of the user list for the assigned role. The discussion centered around the logical steps of the code attached to the "FilterUsers" procedure, as shown in appendix A. These steps of the procedure were implemented in such a way, that they could be re-used to affect separation of duties for any notification activity as long as function activities were supplied where appropriate as hooks into the process definition. The next chapter will demonstrate the re-usability of this function by inserting it into a specific example of a workflow process. The example will guide the user through the entire process lifecycle in Oracle Workflow, and will therefore also be used to illustrate the administration guidelines presented in the previous chapter.

Chapter 8

Implementation Example: Insurance Claim

The previous two chapters discussed how functionality from the CoSAWoE model can be implemented using the *Oracle Workflow* commercial workflow product. This chapter will endeavor to show a practical example of the implementation steps proposed. However, a real-world example of a process definition that suited our requirements for illustrating these steps, were not obtainable. As such, the insurance claim process described in the following paragraphs is a simplified version based on the another example presented in the article by Cholewka et al. (2000).

This process centers around the approval of an insurance claim, and is driven by the decision-making efforts of multiple users. As such this example affords us the opportunity to show how access control policies could be built into the process by following the administration steps out-lined in chapter 6. Thereafter, the effects of these steps as observed by normal and administrative workflow users, are explained.

First, the requirements of a suitable example need to be established.

8.1 Requirements of a Suitable Example

In order to show the access control decisions described in chapter 3, an adequate example with the following features was needed:

- The process had to represent an administrative workflow, that included many user-executed decision steps in order to complete. Preferably one

or more of these steps would amount to some approval activity. Parallel execution paths would also be preferable to show *order of events*.

- The process had to have two or more participating roles. Preferably these roles had to be in a role hierarchy.
- In order to show separation of duties, there should be at least two activities that are conflicting, and that should be performed by different users. Also, users had to be assigned to roles in such a way, that conflicting users existed that could possibly perform two conflicting tasks.
- The process should not include external or distributed processes so that all activities can be executed within a single workflow session on the same workflow engine. Such inter-organizational workflows can of course also be catered for, however, this would only add unnecessary complexity.

Of the demonstration process definitions installed as part of Oracle Workflow, the *Requisition* process came closest to meeting these requirements. However, it only involved two roles, a “requestor” and an “approver”. The individual users who fulfill these roles during various activities in the workflow, is also decided by using a hard-coded management hierarchy. As such it does not create much opportunities for violating access control, or for illustrating the mechanisms with which to enforce such access control policies. The example described next, however, does support all of the requirements mentioned above for demonstration purposes.

8.2 The Insurance Claim example

The following sections will show the step-by-step implementation of the CoSAWoE model based on the handling of an insurance claim case study using Oracle Workflow. The process definition is drawn up in *Oracle Workflow Builder*, and is graphically shown in the process window, as can be seen in figure 8.1 on page 102. For easier reference the activities are numbered from 1 to 8, in addition to their activity names. Unfortunately the viewing options

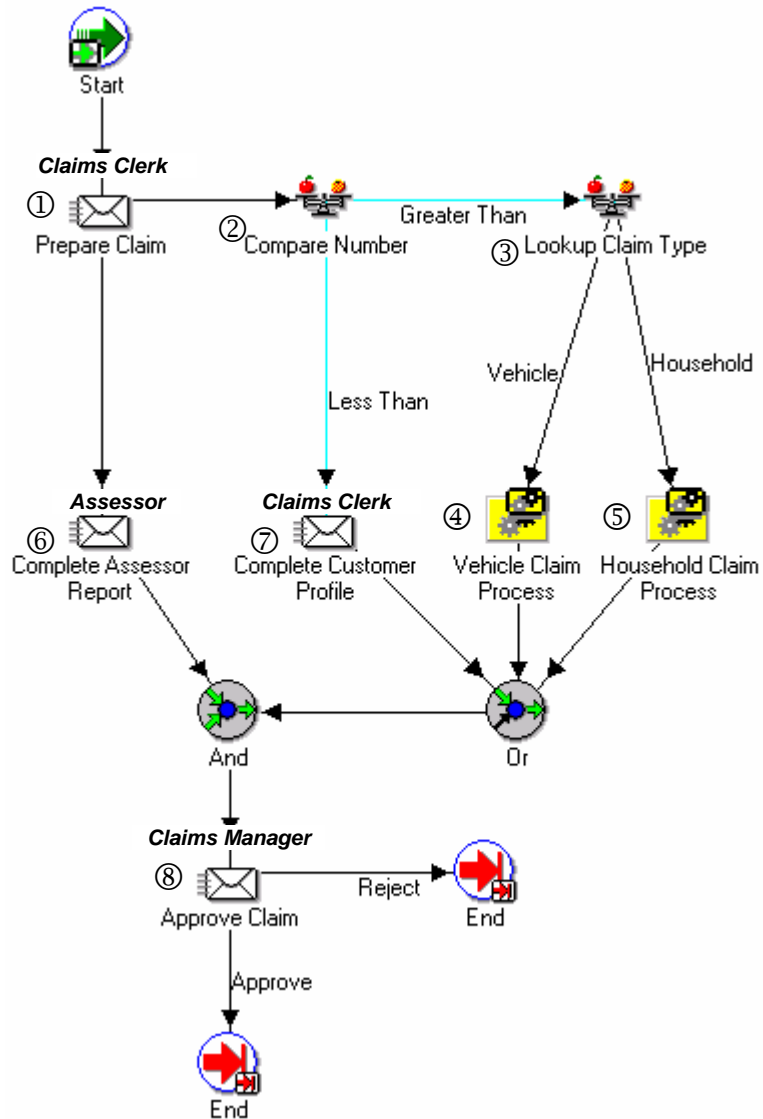


Figure 8.1: The Insurance Claim process as shown in Oracle Workflow Builder's process window

do not show the performers of nodes and the activity names simultaneously. For the sake of clarity the performers were thus also added to the diagram.

The arrows connecting the activities show the order of execution of the process. When more than one arrow originates from the same activity, that means that more than one execution path can be followed. If conditions are attached to these paths, this represents a conditional split where only one path is chosen based on result of the activity. If no conditions exists, all paths are executed in parallel regardless of how the activity completed. Such activities must merge into an “And” join activity to show that both execution paths must be completed before continuing to the next activity. Alternatively, “Or” join activities are used to converge conditional flows. Icons are used to differentiate between the types of activities. Notification nodes are represented as envelope icons. The rest of the activities, except for the two sub-processes, in this example are function activities, with their icons indicative of the work performed by these automated activities.

According to the diagram in figure 8.1 on the next page, once a new claim is started, task 1 is completed by a `Claims_Clerk`. This task could include filling out all the necessary claim related information and assembling the necessary documents¹. Once the claim has been prepared in task 1, a parallel split occurs and task 6 and task 2 are triggered. Task 6 is assigned to any user belonging to the `Assessor` role, who will view the claim information and submit a assessor’s report. Task 2 is an automated decision that checks the value of the claim which was entered in task 1, and based on that decision performs a conditional split. If the claim value is less than 5,000 (the reference constant for the comparison activity), the path to task 7 would be followed. If the claim value is equal to or greater than 5,000 the path to task 3 is followed.

For claims less than 5,000 rands in value, a `Claims_Clerk` only needs to complete a customer profile which involves looking up the client’s previous claim history and some additional information. Claims to the value of 5,000 rand or more requires some additional validation steps. Task 3 determines which type of claim it is (household or vehicle) so that the process can be

¹This information would be included using item attributes, that are typically initialized before the process is started. We will assume that this information is left blank when the process is started, and will only be supplied by the user performing task 1.

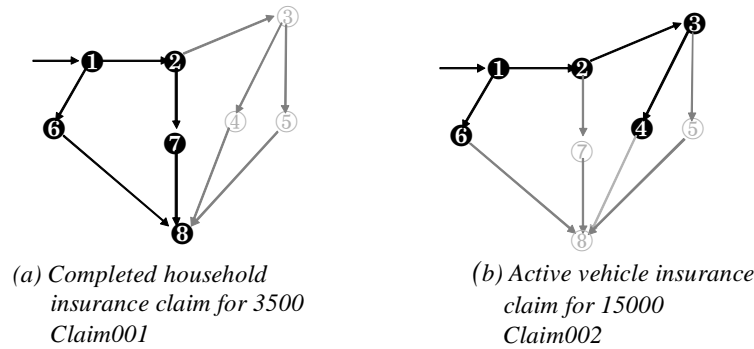


Figure 8.2: Examples of Insurance Claim process instances

continued by a senior clerk in the appropriate department. For brevity's sake, the activities performed by these senior clerks in each sub-process (shown as the process activities 4 and 5) are not discussed here.

No task can be started before all its conditions are met and all prerequisite activities completed according to the order specified. Therefore, task 8 can only be started once the assessor report is completed (task 6), and either the customer profile is completed (task 7) or one of the sub-processes (task 4 or 5) completed successfully. With this information, a `Claims_Manager` can now decide to approve or reject the claim. Figure 8.2 shows two possible process instances, “Claim001” and “Claim002” representing claims for household goods to the value of 3,500 and vehicle accident damage of 15,000 respectively. Claim001 is completed, whilst Claim002 is still in progress. The numbers on the figure correspond with the numbers allocated to the activities in figure 8.1 on the preceding page.

8.3 Building Access Control into the Insurance Claim example

One of the requirements of access control, namely *order of events*, is already indirectly obtained when the process diagram is drawn. Access is controlled from a task perspective and not from a user perspective. What this means is that the right to perform the “Approve Claim” activity is not attached to the Claims Manager role, but instead the Claims Manager role is assigned to the task. Therefore a Claims Manager will only receive task 8 on his worklist

WF_USERS
NAME
Ben
Alan
Pauline
Kenneth
Harry
Sally
Tom

Figure 8.3: Users for the Insurance Claim example

once that activity is triggered by the workflow engine. The permission “Approve Claim” does not exist, and therefore a user will never receive the task based on his role profile alone. Such an implementation could have enabled him to approve a claim before a customer profile has been completed. The way the process was designed with Oracle Workflow Builder shows clear transitions between the activities and the pre-conditions that must exist before the workflow engine can continue to the next activity.

In order to assign a role to an activity, that role must exist within the database and must be loaded into Oracle Workflow Builder. Oracle’s Directory Service described in section 6.1 maps system users onto the **WF_USERS** view to enable those users to access the workflow. Records must also be inserted into the **WF_LOCAL_ROLES** and **WF_LOCAL_USER_ROLES** tables to assign those users to workflow specific roles. For the purposes of the insurance example the users shown in figure 8.3 were created. Figure 8.4 on the next page shows the roles created and graphically depicts the role hierarchy that they are arranged in. The assessor role is not included in the role hierarchy. User assignments were made as shown in figure 8.5 on the facing page.

Performers can now be set for the activities as shown in figure 8.1 on page 102. For each notification activity node, the node properties must be accessed and a Constant role specified as performer (see figure 8.6). This will ensure that only Claims Clerks (Pauline and Kenneth) receive task 1 and task 7, Assessors (Tom) task 6, and Claims Managers (Ben and Alan)

WF_ROLES	
NAME	
Claims Manager	
Snr Clerk (vehicle)	
Snr Clerk (household)	
Clerk	
System	
Assessor	

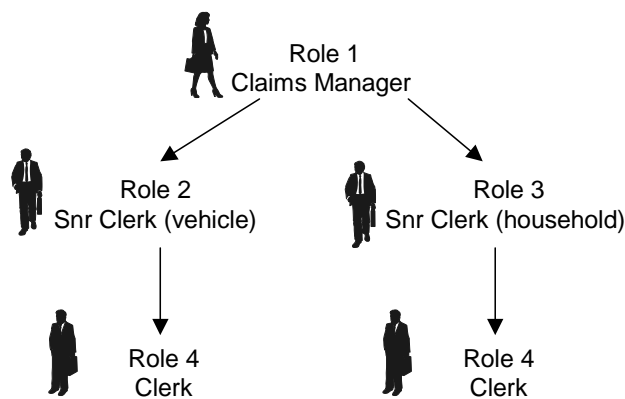


Figure 8.4: Roles for the Insurance Claim example

WF_USER_ROLES	
USER_NAME	ROLE_NAME
Ben	Claims Manager
Alan	Claims Manager
Pauline	Clerk
Kenneth	Clerk
Harry	Snr Clerk (household)
Sally	Snr Clerk (vehicle)
Tom	Assessor

Figure 8.5: User to Role assignment for the example

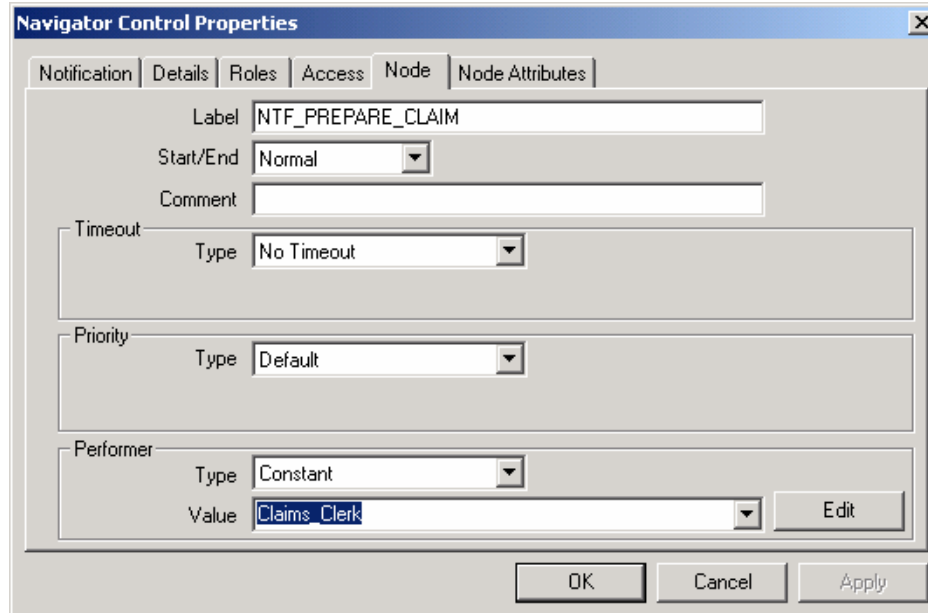


Figure 8.6: Assigning a Performer Role to an activity

task 8 when the workflow process is executed. Because of the role-hierarchy, the Claims Manager role would also be able to perform tasks assigned to the Claims Clerk role. Therefore, Ben and Alan could potentially perform task 1 and 7.

According to chapter 6, *strict least privilege* is again enforced at a task level by including item attributes in the notification activity messages in such a way that it will only be viewable during certain activities, and can be only changed during others. Assuming that the item attributes `Claim_Number`, `Policy_Number` and `Claim_Value` can only be updated during the first activity (“Prepare Claim”), the item attributes will be included in the message templates for each notification activity as shown in figure 8.7 on the facing page. A setting of `Send` means that the attribute can only be viewed in a message attached to that notification activity. A setting of `Respond` will provide users the means to supply a new value for that attribute when they receive such a message. These settings will apply to all users who receive the message irrespective of the role they logged in with. Therefore, a Claim Manager will not be able to update the Claim info during the “Approve Claim” activity, even if he will be able to supply those attributes in the first activity acting as a Claims Clerk.

For this example an administrator may also want to implement some sep-

Figure 8.7: Attribute Type settings for messages

aration of duty constraints. Business rules might state that the same person who prepared the claim may not be allowed to complete the Customer Profile or approve the claim. In order to implement these business rules, the administrator must first specify these constraints in the `WF_CONFLICTING_TASKS` and `WF_CONFLICTING_USERS` tables shown in figure 6.1 on page 88. As no SoD administration tool is supplied by Oracle Workflow, the tables and their records must be created outside the Builder environment. Using SQL statements, conflicts between task 1 and 7 and task 1 and 8 are inserted into the `WF_CONFLICTING_TASKS` table as shown in figure 8.8. In addition, any users who may have a vested interest in one another's work, are specified as conflicting. Effectively conflicting users can be seen as the same user, and as such may not perform conflicting tasks. For the example, we will assume that Ben and Pauline are married. After they are inserted into the `WF_CONFLICTING_USERS` table it will contain the records as shown in fig-

WF_CONFLICTING_TASKS			
T_ITEM_TYPE	T_ACTIVITY_NAME	C_ITEM_TYPE	C_ACTIVITY_NAME
WFINS	Prepare Claim	WFINS	Complete Customer Profile
WFINS	Prepare Claim	WFINS	Approve Claim

Figure 8.8: Conflicting Tasks for the Insurance Claim example

WF_CONFLICTING_USERS		
T_NAME	C_NAME	RELATIONSHIP
Pauline	Ben	Husband
Ben	Pauline	Wife

Figure 8.9: Conflicting Users for the Insurance Claim example

ure 8.9. Therefore Ben will not be able to perform any task that conflicts with a task that Pauline already performed earlier in the process.

Next, the administrator must include a function activity to evaluate the values in these tables, as well as the workflow history, and prune the user list for certain activities. Typically the pruning function will be included immediately before any notification activity that requires a response. However, this example only requires that separation of duties be enforced between task 1 and 7 and task 1 and 8. Therefore the function activity “Filter Users” is included only before task 7 and task 8. Figure 8.10 on the facing page shows the process diagram after these function activity nodes are inserted into the definition. There is no chance that task 1 will be performed before task 7 or 8 (this is only a concern if tasks are performed in parallel), and so the user list can remain unpruned for that task. The re-usable procedure, described in chapter 7, to prune user lists can now be attached to the “Filter Users” function activity.

8.4 Demonstration of the Insurance Claim Example

In this section we step through the insurance example for completing a household claim to the value of R3,500 (the expected execution path of this process instance is shown in figure 8.2 on page 103). We will be logging in with all the users who belong to roles required to complete this claim example. Screenshots of those users’ worklists at different times in the workflow will be accompanied by screenshots of the administrator’s monitor facility to show the execution and progression of the process instance.

The process definition, as shown in figure 8.10 on the preceding page, is

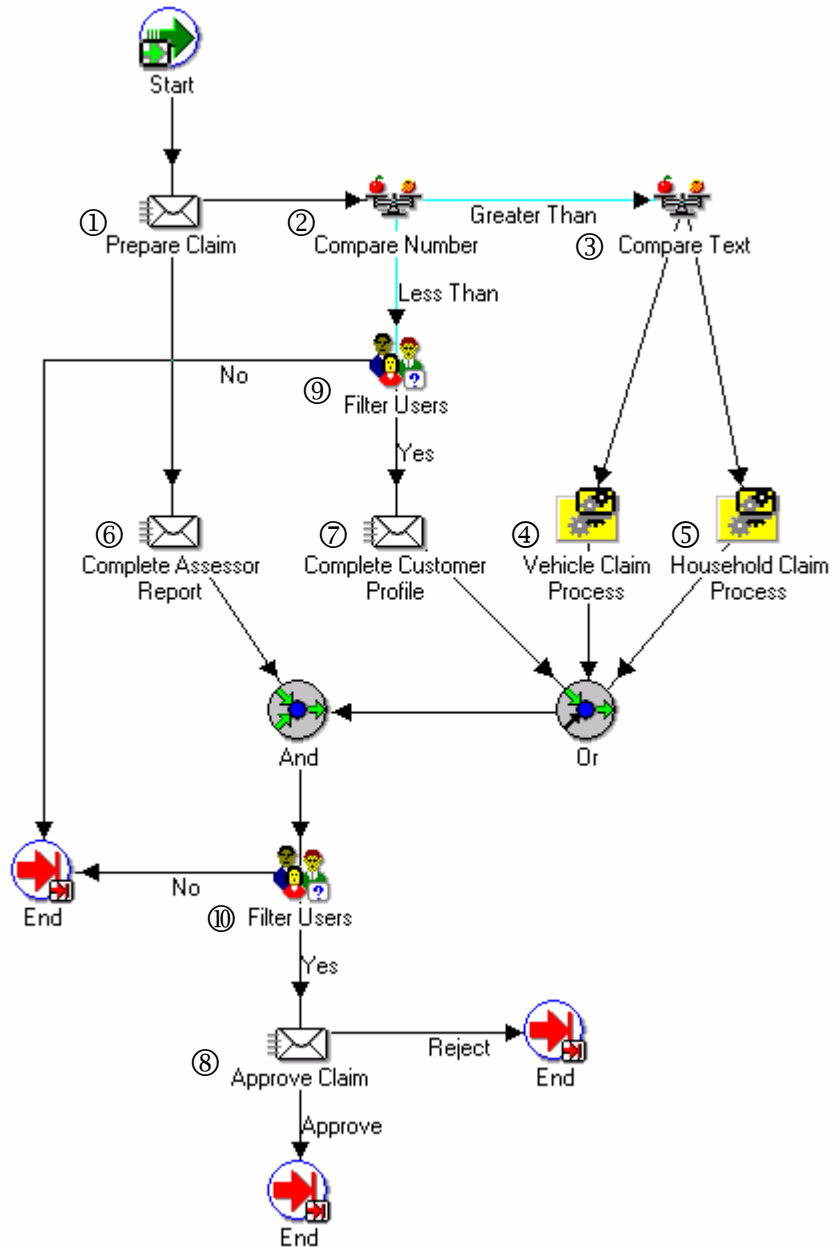


Figure 8.10: The Insurance Claim process with SoD function activities

Figure 8.11: Starting a new Insurance Claim process

initiated when a user with sufficient rights launches that process via the “Launch Processes” webpage (included on the “Workflow Homepage” as shown in figure 5.8 on page 75), or when the `LaunchProcessAPI` is called by a custom webpage or application. The new process instance, shown in figure 8.11, was created via “Launch Processes”, and as such it allows the administrator to specify startup values, although these item attributes are not needed to start the process. Figure 8.12 on the next page also shows the confirmation of the process instantiation, and the creation of the first notification task, “Prepare Claim”, belonging to the `Claims_Clerk` role.

It is important to note that only one notification is created for the “Prepare Claim” activity. Only once a user logs in and is authenticated, this notification is listed in his/her worklist if he/she belongs to the specified role. Therefore, Pauline and Kenneth, will both receive the first task in their worklist since they belong to the `Claims_Clerk` role. Ben, Alan, Harry and Sally will also receive this notification because of their roles which are above the required role in the role hierarchy (see figure 8.4 on page 105). Figure 8.13 shows Pauline’s worklist after task 1 has been created. At this stage in the workflow, everyone’s worklist, except for Tom’s, will look similar to the one shown for Pauline.

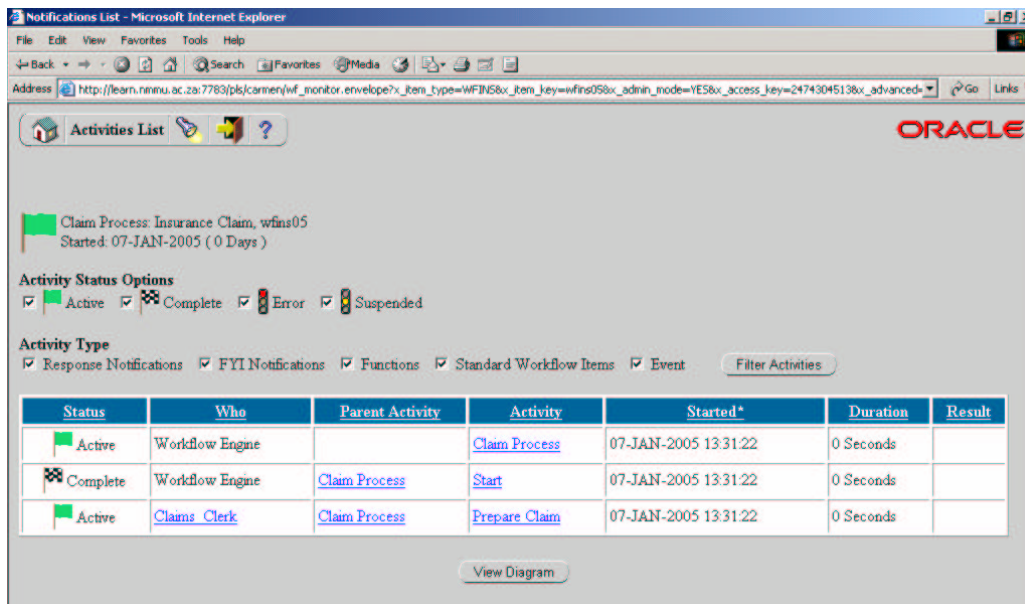


Figure 8.12: Monitoring the status of the Claim via the “Activities List” webpage

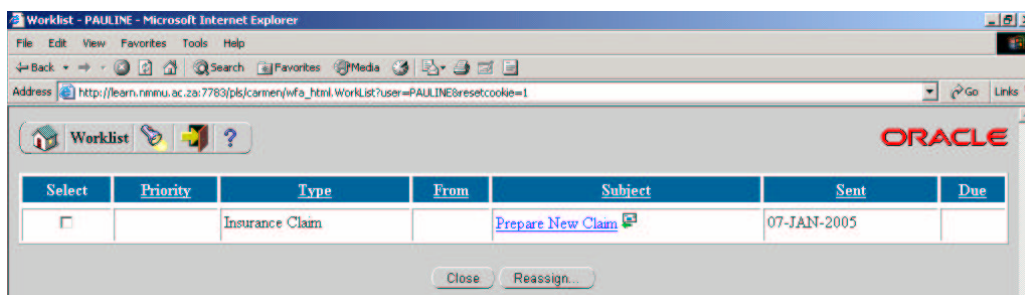


Figure 8.13: The “Prepare Claim” notification appears on Pauline’s worklist

Detail notification for PAULINE - Microsoft Internet Explorer

Address: http://learn.rimnu.ac.za:7783/pls/carmen/wfa_html.detail?id=225

Notification Details

To: Claims_Clerk Sent: 07-JAN-2005 13:31:22

Subject: Prepare New Claim

Please supply the values requested in the section below.

[Return to Worklist](#)

Claim Number: 001

Policy Number: 456789

Claim Type: Household

Police Report:

Witness Report:

Quotation:

Claim Value: 3500

Submit Reassign...

Figure 8.14: Pauline prepares a new claim by supplying values for the item attributes presented by the notification message

We will assume that Pauline, now decides to act on this notification, thereby claiming it and disallowing any of the other users who received this notification from responding to it. Pauline will be able to supply values for all the item attributes which were specified as type **Respond**. These attributes are included in the bottom frame of figure 8.14.

This figure also shows a “Reassign” button. This button would allow Pauline to assign the current task to any user in the database using her own discretion. This option could be removed by simply including the special `Hide_Reassign` message attribute in all message templates.

Once she has pressed the “Submit” button the workflow engine will verify the information she supplied and will update the `WF_ITEM_ACTIVITY_STATUSES` and `WF_NOTIFICATIONS` history tables to indicate that the activity is now closed. This will remove the “Prepare Claim” notification from her worklist as well as the other users who received this task (see the next worklist iteration in figure 8.20 on page 118).

Since the `Claim_Value` of R3,500 that Pauline supplied was less than R5000, the workflow engine will create the “Complete Assessor Report” (task 6) and the “Complete Customer Profile” (task 7) notifications next. The worklists of Pauline (a claims clerk), Kenneth (another claims clerk),

Ben (a claims manager and Pauline's husband), Alan (another claims manager), and Tom (an assessor), shown in figure 8.15 on the facing page, clearly illustrates that the two notifications is not present in all the worklists. We can assume that the senior clerks, Harry and Sally would have received notifications based on the role-hierarchy also. However, since they will not be involved further in this process instance, their worklists are of no interest here.

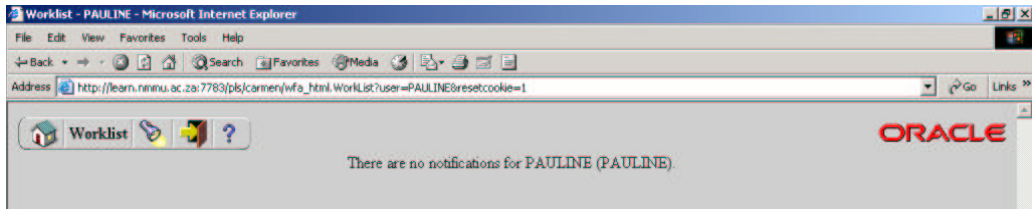
Task 7 is only in Tom's worklist since he is the only user belonging to the **Assessor** role, and this role is also not part of the role hierarchy. Kenneth, a claims clerk, as well as Alan, a manager, receives the "Complete Customer Profile" notification. However, Pauline and Ben, who both belong to roles which should have provided them with task 7, have no such notification in their worklists. The process diagram shown in figure 8.16 reveals the reason for this.

Before task 7 could be created, the "Filter Users" function (task 9) was executed to evaluate the potential users and eliminate those users whose involvement in the following task would have constituted a separation of duty conflict. Figure 8.17 on the facing page graphically illustrates how the function used the **Conflicting Tasks** and **-Users** tables to remove Pauline, based on the fact that she performed conflicting task 1, as well as Ben, since he is her husband, and conflicting users are seen as the same user for the conflicting tasks.

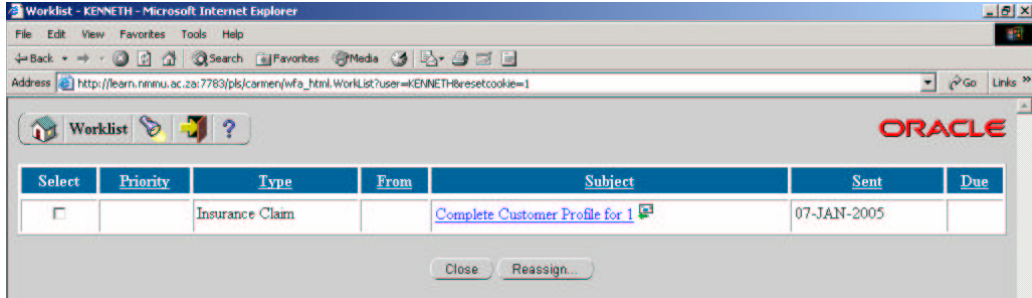
Figure 8.16, shows a new ad-hoc role called `wfins05-1739-CLAIMS_CLERK` which is specifically created to associate valid users with the task. This temporary role only presents the "Complete Customer Profile" task to users who have not participated in conflicting tasks (or are related to someone who has) for the activity identified by "1739" during the process instance "wfins05".

Figure 8.18, and figure 8.19, shows how certain of the item attributes, such as `Claim_Number`, `Policy_Number` and `Claim_Value` can only be viewed in the message body, while allowing the assessor and clerk to submit their reports for task 6 and 7.

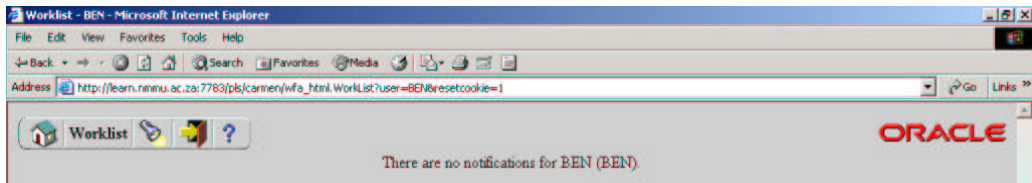
Once both Tom and Kenneth, have completed tasks 6 and 7 respectively, the workflow engine can continue from the "And" join. Figure 8.16 on page 114 shows that another pruning function (task 10) was involved before



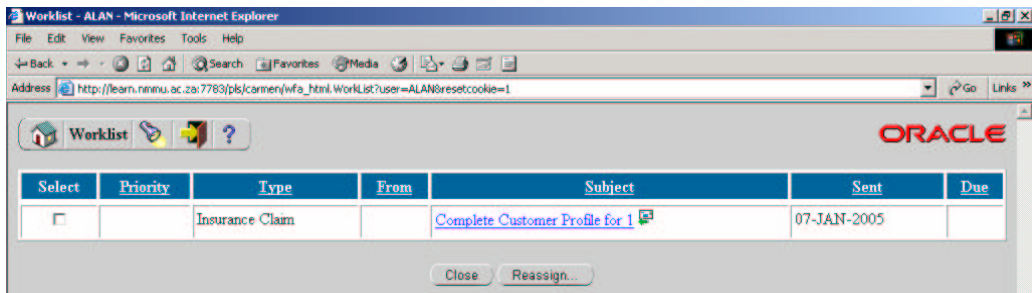
Pauline



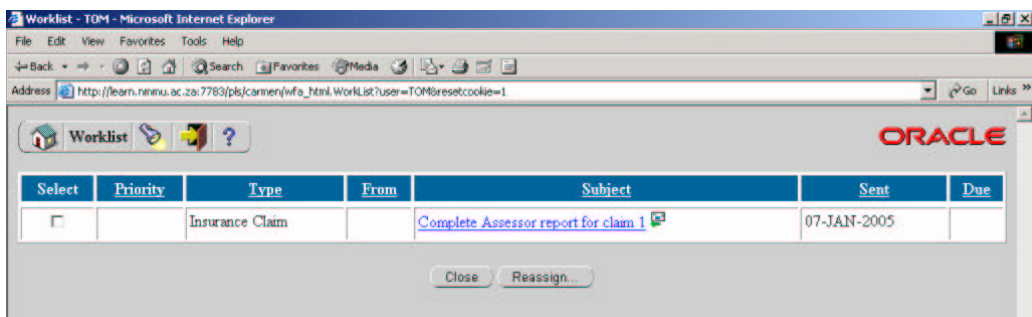
Kenneth



Ben



Alan



Tom

Figure 8.15: The worklists of Pauline, Kenneth, Ben, Alan and Tom after Task 1 has been completed

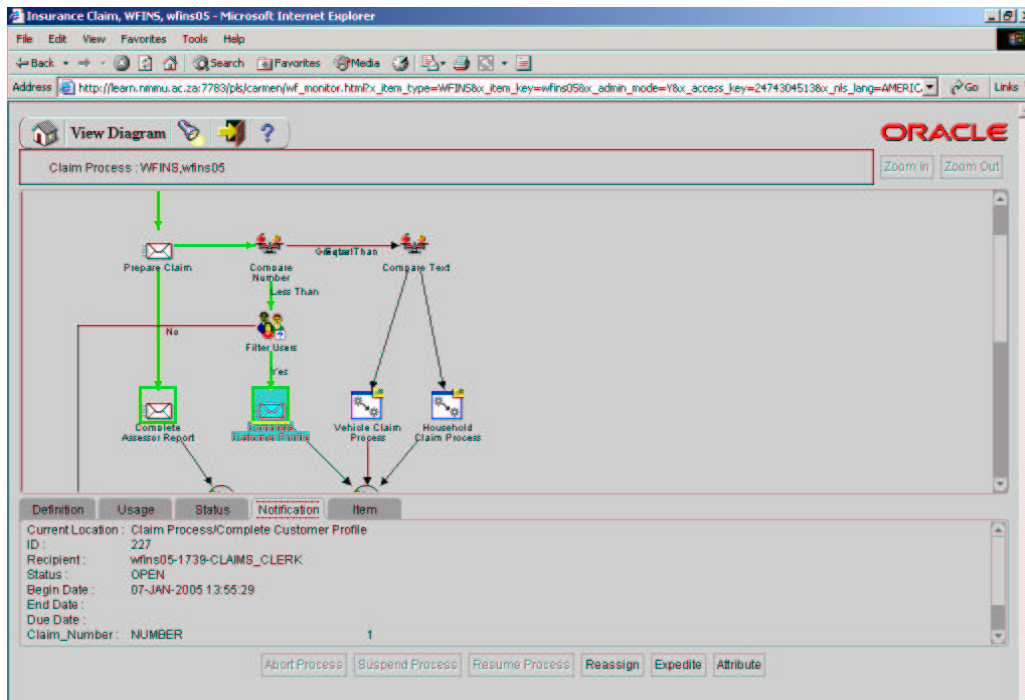


Figure 8.16: The process diagram in the web monitor shows the execution of “Filter Users” before the “Complete Customer Profile” task

the “Approve Claim” (task 8) notification was delivered to `wfins05-1735-CLAIMS_MANAGER`, the ad-hoc role created for that task. Alan is the only user who receives this notification, since only `Claim_Managers` may perform this task, and the other claim manager, Ben, would again be excluded based on the fact that his wife performed a conflicting task (“Prepare Claim”) for task 8. Figure 8.20 on the next page therefore shows the “Approve Claim” in Alan’s worklist but not in Kenneth and Paulines’s (since they do not belong to the Claims Manager role) or Ben’s (who was excluded based on SoD conflicts).

Figure 8.21 on page 119 is an activity list that summarizes the execution steps demonstrated in this example.

8.5 Conclusion

This chapter showed that a practical example could indeed be used to implement the steps and functions proposed in the previous two chapters through an exploratory prototype. First, the requirements of a suitable example for illustration purposes was discussed. This led to the selection of an insurance

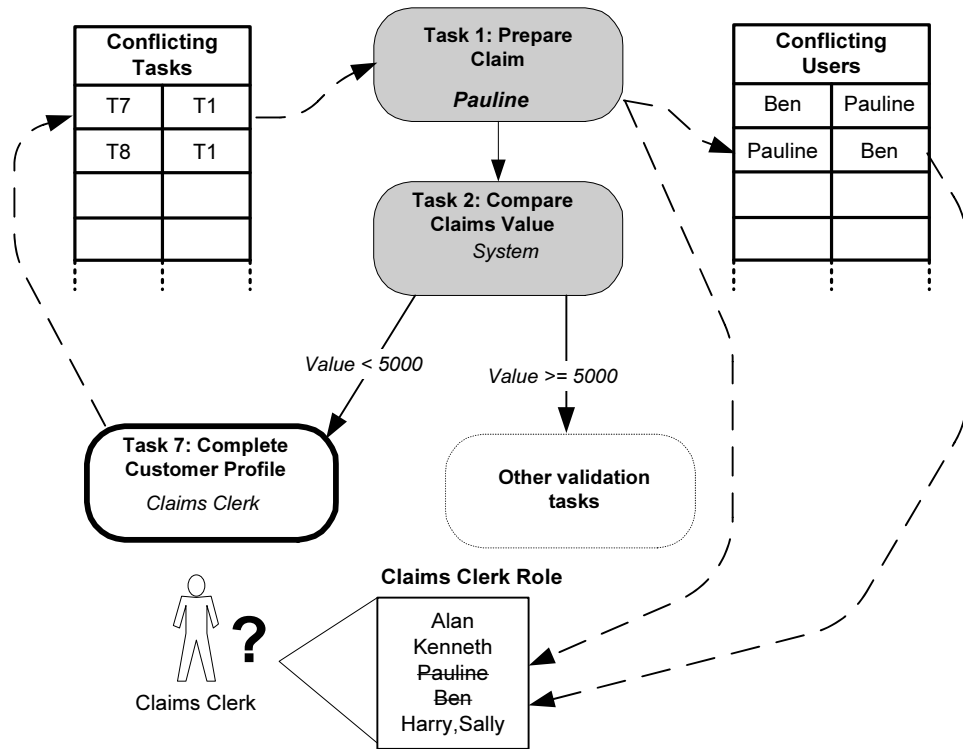


Figure 8.17: A graphical illustration of how the “Filter Users” function prunes the user list before Task 7

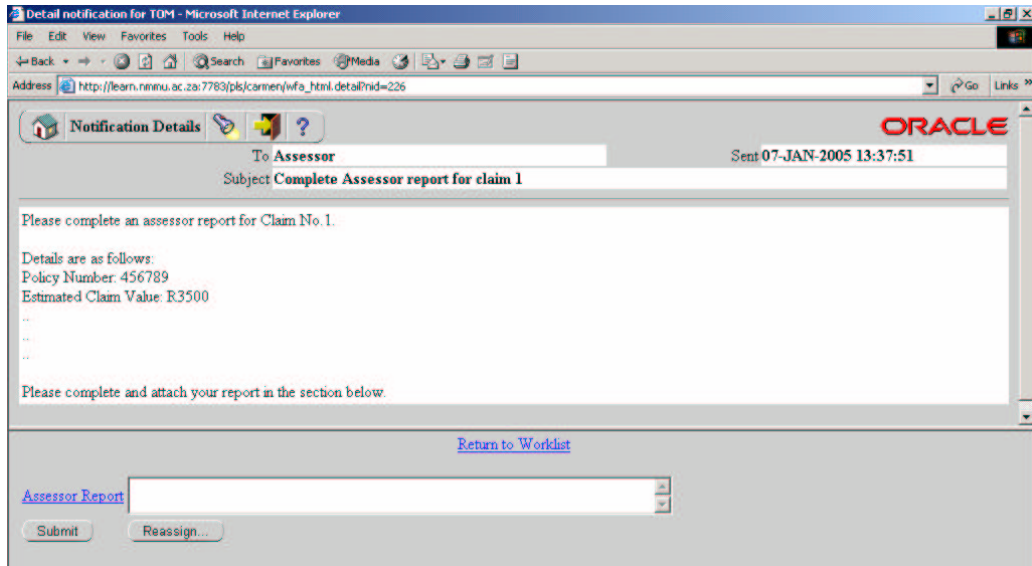


Figure 8.18: Tom prepares his assessor’s report in task 6

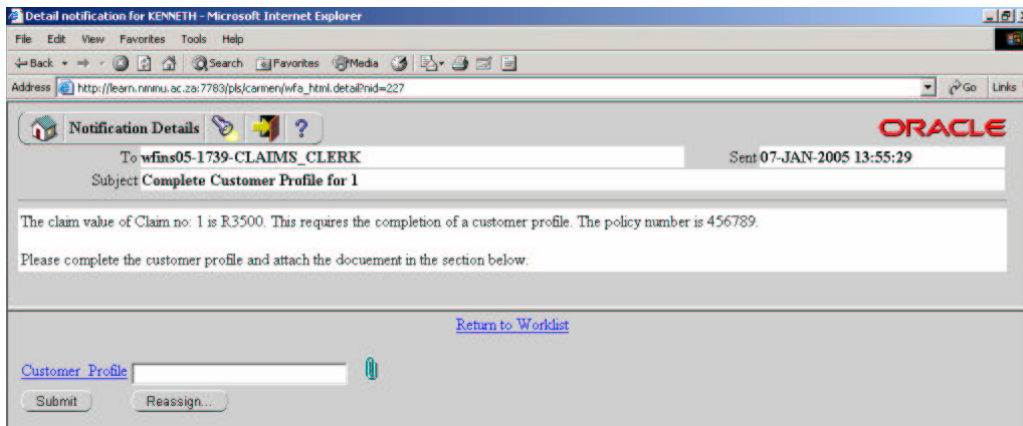


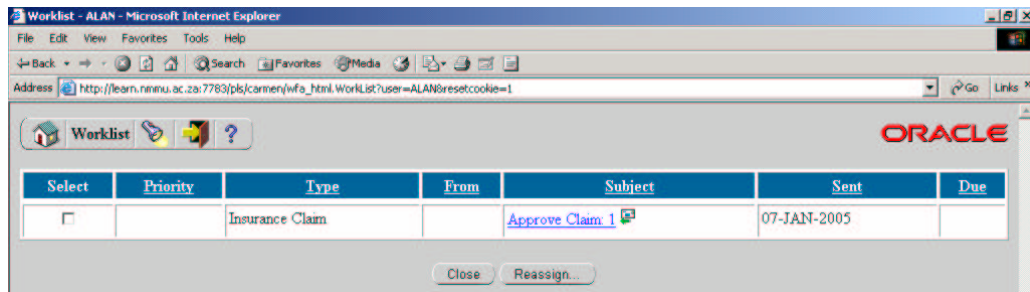
Figure 8.19: Kenneth prepares a customer profile before submitting it for task 7

claim process as our chosen example.

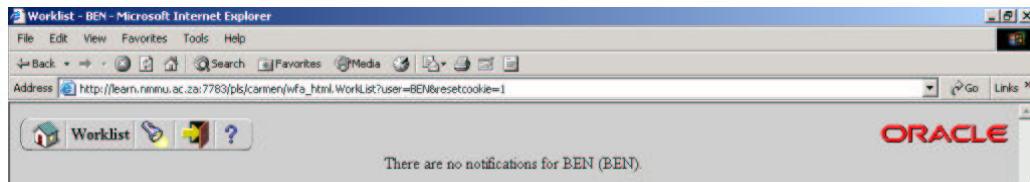
The administration steps were completed mostly in Oracle Workflow Builder. The first step was to create the basic process model for the insurance example (see figure 8.1). This included assigning users to certain notification tasks so that the workflow engine could maintain *order of events*. These users first needed to be created in Oracle's Directory Service and mapped to the relevant workflow views. Secondly, we showed how the inclusion and use of item attributes in message templates could be used to enforce *strict least privilege*. Thirdly, to implement the separation of duty constraints for this example, additional tables needed to be created and populated. The process diagram then needed to be modified to include additional functions which executed the code behind the re-usable "FilterUsers" procedure (shown in Appendix A).

Once a satisfactory process definition existed, the process was instantiated and executed through the web interface supplied with Oracle Workflow. Screenshots of the users' worklists, generated dynamically upon user authentication, as well as administrative views of the process execution at various stages, were used to explain the run-time enforcement aspects described in chapter 7.

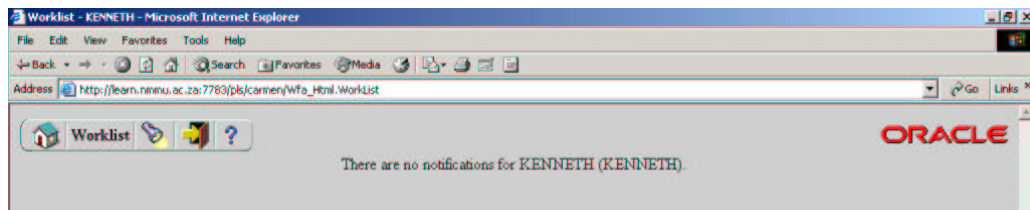
In the following chapter, we will conclude the dissertation by analyzing the problem and the solution that was affected.



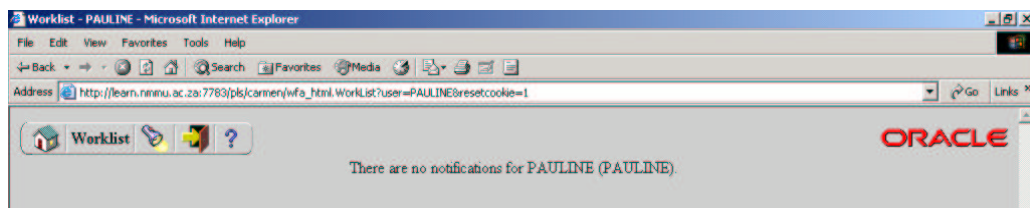
Alan



Ben



Kenneth



Pauline

Figure 8.20: The worklists of Alan, Ben, Kenneth and Pauline after task 6 and 7 have been completed.














<u>Status</u>	<u>Who</u>	<u>Parent Activity</u>	<u>Activity</u>	<u>Started*</u>	<u>Duration</u>	<u>Result</u>
 Complete	Workflow Engine		<u>Claim Process</u>	07-JAN-2005 13:31:22	48 Minutes 13 Seconds	
 Complete	Workflow Engine	<u>Claim Process</u>	<u>Start</u>	07-JAN-2005 13:31:22	0 Seconds	
 Complete	<u>Claims Clerk</u>	<u>Claim Process</u>	<u>Prepare Claim</u>	07-JAN-2005 13:31:22	6 Minutes 29 Seconds	
 Complete	Workflow Engine	<u>Claim Process</u>	<u>Compare Number</u>	07-JAN-2005 13:37:51	0 Seconds	Less Than
 Complete	Workflow Engine	<u>Claim Process</u>	<u>Filter Users</u>	07-JAN-2005 13:37:51	17 Minutes 38 Seconds	Force
 Complete	<u>Assessor</u>	<u>Claim Process</u>	<u>Complete Assessor Report</u>	07-JAN-2005 13:37:51	33 Minutes 7 Seconds	
 Complete	Workflow Engine	<u>Claim Process</u>	<u>Filter Users</u>	07-JAN-2005 13:55:29	0 Seconds	Yes
 Complete	<u>wfins05-1739-CLAIMS_CLERK</u>	<u>Claim Process</u>	<u>Complete Customer Profile</u>	07-JAN-2005 13:55:29	12 Minutes 38 Seconds	
 Complete	Workflow Engine	<u>Claim Process</u>	<u>Or</u>	07-JAN-2005 14:08:07	0 Seconds	
 Complete	Workflow Engine	<u>Claim Process</u>	<u>And</u>	07-JAN-2005 14:10:58	0 Seconds	
 Complete	Workflow Engine	<u>Claim Process</u>	<u>Filter Users</u>	07-JAN-2005 14:10:58	0 Seconds	Yes
 Complete	<u>wfins05-1735-CLAIMS_MANAGER</u>	<u>Claim Process</u>	<u>Approve Claim</u>	07-JAN-2005 14:10:58	8 Minutes 37 Seconds	Approve
 Complete	Workflow Engine	<u>Claim Process</u>	<u>End</u>	07-JAN-2005 14:19:35	0 Seconds	

Figure 8.21: A summary of the activities that were completed in this example.

Chapter 9

Conclusion

This dissertation presented a series of steps to implement the CoSAWoE model in a commercial workflow product. These steps evolved out of the discussion of the model and an extensive investigation into the features provided with our chosen workflow product, namely Oracle Workflow. A prototype further attempted to showcase how these steps can be followed to map the model components to a practical example developed with Oracle Workflow.

This dissertation was based on the premise that workflow has become a key enabling factor for conducting e-business. Information is seen by e-businesses as an asset that requires protection from threats coming from outside as well as inside the organization. This realization prompted investigation into the access control service, and the requirements of such a service when part of a workflow system. The investigation revealed that although several access control models have been developed, implementation of these models in commercial workflow products have been haphazard and unregulated. These realizations motivated the researcher to determine the extent to which a particular model could be implemented successfully within a commercial workflow product.

This chapter will summarize the lessons learnt throughout the dissertation in the effort to integrate aspects of the CoSAWoE model with Oracle Workflow's existing functionality. Thereafter, future research will be proposed.

First, a re-cap of the research questions posed as part of the problem statement is given with a discussion of how each one was addressed.

9.1 Revisiting the Problem Statement

This dissertation was concerned with implementing a access control model within a commercial workflow environment. A decision was made to implement a comprehensive access control model for workflows already in existence, namely CoSAWoE. This model was assumed to be effective in securing the workflow tasks and data from a logical perspective. Therefore the question was not so much “what” needed to be done, as much as “can” it be done. It was also decided that a particular workflow product, Oracle Workflow, would be used to implement the model. This also raised the question of “how efficiently” the chosen product could be used to implement the model.

This dissertation attempted to answer the research questions in the following way.

What are the functional requirements of an access control service for workflow systems?

The first question sought to discover what the functional objectives were that had to be met by the implementation effort. As such the basic functions of a typical workflow system were first discussed in chapter 2. This chapter served as reference for the discussions that followed about the access control requirements in a workflow.

Workflow systems are used firstly to define the process definition of which tasks form the basic building blocks. Task definitions represent, therefore, a unit of work with specific access control requirements as to who can access that task and what information they may and may not view or alter within that task. Workflow systems enact the process definitions by creating and managing task instances for each occurrence of the corresponding business process.

An access control service need to be aware of the context in which a task instance is triggered and presented to users. This idea of the access control service being “context-sensitive” presented three functional requirements that needed to be enforced by the CoSAWoE model: *order of events*, *strict least privilege* and *separation of duties*. This was extensively discussed in section 3.3 on page 34.

Which aspects of the access control model is agnostic to the product?

Chapter 4 discussed the CoSAWoE model components. This chapter identified those components which presented no particular implementation concern for commercial workflow products. Two components, namely object design and role engineering, suggested methodologies for how roles could be constructed that would hold sufficiently few permissions as to be associated with a particular task. As pointed out, the construction of roles are not provided as part of the workflow functionality of commercial systems. Often the roles adopted by the workflow are maintained as part of the ERP system within which the workflow product is embedded. It was assumed that administrators would be able to follow these methodologies outside the scope of the workflow system if needed. As such it did not affect the implementation effort one way or the other, and is considered agnostic to the commercial product.

Which aspects of the model can be implemented directly and which will require customization of the product in order to achieve similar results?

Chapter 4 also discussed two components in detail which were considered fundamental to achieving access control requirements within commercial products, namely SoD administration and worklist generation. Separation of duty administration featured strongly in chapter 6, “CoSAWoE: Administration in Oracle Workflow”. For this component the model suggested the specification of constraints according to the “Conflicting entities” Administration Paradigm. This required that additional conflict tables had be created and populated outside the process definition tool of the product. As pointed out in chapter 6 the static separation of duty policies could be enforced at administration/design-time through the use of database triggers. However, the implementation would also require the dynamic separation of duty approach which only checks the conflicts during run-time when the workflow is actually executed. This approach affected the generation of worklists as described in chapter 7.

The worklist generation functionality described by the model was largely supported with the chosen workflow product, except for the evaluation of separation of duty policies. Considering the lack of a central access control service, a procedure needed to be explicitly invoked with an additional function task before each sensitive task. A re-usable procedure was coded that would generate an approved user list after pruning functions have been applied to enforce separation of duty policies. This procedure's hook into the workflow is provided by the administrator who includes the additional function as part of the process model wherever he/she feels it is necessary.

Session control is indirectly achieved as part of the worklist functionality, since it acts as the primary interface between the user and the workflow. In the case of Oracle Workflow, session control was attained without any additional coding. The workflow engine, acting as proxy, controls access to the task and task data in the workflow tables. Tasks requiring separation of duty is based on temporary roles created by the pruning function described earlier. However, once the sensitive task is reassigned to the temporary role, the user session needed to access that task is controlled as for any other task.

Which aspects are not implementable for the chosen workflow product?

There were no components of the CoSAWoE model which could not be implemented in one form or another. Only one of the conflicting entities suggested by CoAP, namely conflicting roles, could not be *specified* as part of separation of duty administration. The identification and authentication service offered by Oracle Workflow, does not require a user to log in with a particular role. As such this information is also not part of the history tables which provide information about user involvement in a particular process instance. Since the role a user played while performing a particular task instance cannot be identified, specifying conflicts between roles that cannot be evaluated is pointless. Also, dynamically conflicting roles only aims to prevent the same user from performing two conflicting tasks based on his/her role activation. This is already catered for when specifying conflicting tasks (which bars the same user from performing those task regardless of the roles involved). Therefore the conflicts specified in the two conflict tables, for conflicting tasks and

conflicting users, are considered adequate for enforcing separation of duties.

The following section discusses the “how” of the implementation effort of the CoSAWoE model within Oracle Workflow.

9.2 Implementation Issues and Lessons Learnt

As stated in chapter 4, section 4.2.3, the components that has the most impact when implementing the model in a commercial product, are the “SoD Administration” and “worklist generation” components. The components, in turn, affected the process definition and the worklist functionality supplied with the workflow product. The following list summarizes the lessons learnt by exploring these two components in an Oracle Workflow environment:

- Although a dynamic separation of duties policy was adopted, SSoD as well as DSoD required the specification of conflicting entity information as a series of tuples in purposefully created database tables. For the “Conflicting Tasks” table, the `Item_type` was included with the task names to identify the process involved. However, it does not mean conflicts could be evaluated between tasks belonging to different process instances. The “Conflicting Users” table also included a relationship field for error messaging purposes, although no attempt was made to incorporate this information anywhere in the Insurance Claim prototype.
- Also, no attempt was made to enforce Static Separation of duties in the prototype through the suggested method of using triggers. The reasoning behind this was that a similar prototype implementation was already achieved in the SoDA system, and as such no additional value would be added.
- Dynamic separation of duties required the specification of an additional function activity in the process model. This was necessitated by the fact that no central access control model was available to implement the necessary separation of duty functionality. Instead, a re-useable procedure was developed that could be used by all of these function activities, before any notification activity. One limitation, however, was that it could not filter out groups of users based on role activation.

- The development of the re-usable function depended greatly on the understanding of how process instance data is created and where it is stored in the workflow history tables. This information allowed us to retrieve the users who responded to notifications of tasks on which conflicting entities were specified.
- Worklist generation as far as the achievement of session control was also discussed. No additional coding was required although the use of a temporary role was argued to represent the intended workflow session functionality suggested by the model.

9.3 Future Research

During the course of this research XML documents were investigated as the means by which information can be exchanged and propagated by the workflow. The structure and semantic meaning that this markup language lends to the data it describes, gives it the potential to achieve access control at a much finer level of granularity. Message content could also be generated dynamically based on the workflow context, rather than pre-defined and limited message templates, as used by Oracle Workflow. Object design and strict least privilege could therefore also be affected in a more formalized way than can be achieved by the implementation guidelines we suggested. Oracle Workflow does allow for the creation of HTML documents based on the XML data supplied through a PL/SQL document. This requires the use of a stylesheet that is based on the object's (XML document's) permission profile for a specific task in the workflow.

Firstly, therefore, further investigation into the use of XML documents in Oracle Workflow is recommended. Oracle does provide some functionality to evaluate XML tag data received from external systems and business events. The proposed research would, however, aim to provide XML documents as flexible alternatives to the use of item attributes to store and propagate information through the workflow.

Secondly, the dynamic creation of stylesheets based on stored permission profiles for the tasks is suggested as a research topic. These stylesheets could be used to prune the information available XML documents based on the

context of the current task and the user's involvement in previous task in the workflow.

9.4 Final Word

In conclusion, this dissertation developed a method for implementing the CoSAWoE model in a commercial product, such as Oracle Workflow. These steps are aimed at administrators who would like to evaluate access control policies in a consistent manner. To this effect, a re-usable procedure was also developed to prune user lists according to SoD policies for sensitive tasks. The lack of a central access control model necessitated, however, that extra function activities be included in the process model so that this functionality could be executed before users were allowed access to those sensitive tasks. Nevertheless, the author believes that the implementation effort was a successful one in that it achieved the required functionality of context-sensitive access control. However, there is room still for the improvement of this particular product as far as the efficiency with which the said functionality could be achieved. Therefore, the author hopes that this study will contribute to the eventual integration of a standard access control service as a fundamental component in most workflow products.

For those readers interested in pursuing this area of discourse, the author offers the following warning by Jonathan Swift:

“Blot out, correct, insert, refine ...
Be mindful, when invention fails,
To scratch your head, and bite your nails.”

Appendix A

The WFSOD PL/SQL Package

A.1 FilterUsers Procedure

```
1  PROCEDURE FilterUsers (itemtype in varchar2,
2                          itemkey in varchar2,
3                          actid in number,
4                          funcmode in varchar2,
5                          resultout out varchar2)
6  is
7
8  --local variables
9
10 l_next_actid number;
11 l_next_actname varchar2(30);
12 l_next_performer varchar(30);
13 v_user_name wf_user_roles.user_name%TYPE;
14 v_temp_role wf_local_roles%ROWTYPE;
15
16 --l_valid_users is a long string delimited by commas of valid usernames
17 --used as a parameter for the CreateAdHocRole API
18
19 l_valid_users varchar2(100);
20
21 Cursor c_valid_users IS
22
23 --Find all users who are supposed to perform the NEXT task
24
25   Select user_name
26   From wf_user_roles
27   Where role_name = l_next_performer
28
29 --and subtract
30
31   Minus
32
33 --all the users who responded to conflicting tasks specified for the NEXT task
34
35   Select distinct responder
36   From wf_conflicting_tasks ct, wf_notifications n,
37        WF_ITEM_ACTIVITY_STATUSES st, wf_process_activities pa
```

```

38     Where n.notification_id = st.notification_id
39     and item_key = itemkey
40     And st.process_activity = pa.instance_id
41     and pa.activity_name = ct.c_activity_name
42     and t_activity_name = l_next_actname
43     and t_item_type = itemtype
44
45 --as well as all process_owners if "START" was specified as a conflicting task
46     Minus
47     Select distinct owner_role
48     From wf_items i, wf_conflicting_tasks ct
49     where item_key = itemkey
50     and ct.c_activity_name = 'START'
51     and t_activity_name = l_next_actname
52     and t_item_type = itemtype
53
54 --as well as all users who are related in some way to one of the responders
55 --to conflicting tasks
56     Minus
57     Select distinct cu.c_name
58     From wf_conflicting_tasks ct, wf_notifications n,
59         WF_ITEM_ACTIVITY_STATUSES st, wf_process_activities pa, wf_conflicting_users cu
60     Where n.responder = cu.t_name
61     and n.notification_id = st.notification_id
62     and item_key = itemkey
63     and st.process_activity = pa.instance_id
64     and pa.activity_name = ct.c_activity_name
65     and t_activity_name = l_next_actname
66     and t_item_type = itemtype;
67
68 begin
69
70 if ( funcmode = 'RUN' ) then
71
72 --retrieve the actid, name and performer of the next activity
73 --using the WF_ACTIVITY_TRANSITIONS table
74
75     Select TO_PROCESS_ACTIVITY, ACTIVITY_NAME, PERFORM_ROLE
76     into l_next_actid, l_next_actname, l_next_performer
77     from WF_ACTIVITY_TRANSITIONS wft, wf_process_activities pa
78     where FROM_PROCESS_ACTIVITY = actid
79     and result_code = 'Y'
80     and pa.instance_id = to_process_activity;
81
82 --test if cursor does contain at least one valid user
83
84     open c_valid_users;
85     Fetch c_valid_users into v_user_name;
86
87 if c_valid_users%NOTFOUND then
88     resultout:='COMPLETE:N';
89 Else
90
91     --copy values from role on which temp role will be based
92
93     Select * into v_temp_role
94     from wf_roles
95     where name = l_next_performer;

```



```
96
97 --change the name and make it unique by including the itemkey
98 --next activity name and next performer role
99
100 v_temp_role.name:=itemkey||'-'||l_next_actid||'-'||l_next_performer;
101
102 --iterate through the cursor and build string with usernames to assign to the temp role
103
104 loop
105     l_valid_users:=l_valid_users||v_user_name||',';
106     Fetch c_valid_users into v_user_name;
107     Exit when c_valid_users%NOTFOUND;
108 end loop;
109 close c_valid_users;
110 l_valid_users:=substr(l_valid_users,1,length(l_valid_users)-1);
111
112 -- execute API to create the temporary role in a local table
113 -- with users attached as per the cursor;
114
115 WF_DIRECTORY.CreateAdHocRole(v_temp_role.name,v_temp_role.name,
116 v_temp_role.language,
117 v_temp_role.territory,
118 v_temp_role.description,
119 'QUERY',
120 l_valid_users,
121 v_temp_role.email_address,
122 v_temp_role.fax,
123 v_temp_role.status,SYSDATE);
124
125 --reassign the activity to the temporary role
126
127 wf_engine.AssignActivity(itemtype => itemtype,
128                         itemkey => itemkey,
129                         activity => l_next_actname,
130                         performer => v_temp_role.name);
131 resultout := 'COMPLETE:Y';
132 end if;
133 return;
134 end if;
135 if ( funcmode = 'CANCEL' ) then
136
137 --<your CANCEL executable statements>
138 resultout := 'COMPLETE';
139 return;
140 end if;
141 if ( funcmode = 'RESPOND' ) then
142 --<your RESPOND executable statements>
143 resultout := 'COMPLETE';
144 return;
145 end if;
146 if ( funcmode = 'FORWARD' ) then
147 --<your FORWARD executable statements>
148 resultout := 'COMPLETE';
149 return;
150 end if;
151 if ( funcmode = 'TRANSFER' ) then
152 --<your TRANSFER executable statements>
153 resultout := 'COMPLETE';
```

```

154     return;
155 end if;
156 if ( funcmode = 'TIMEOUT' ) then
157 --<your TIMEOUT executable statements>
158     --if (<condition_ok_to_proceed>) then
159     -- resultout := ?COMPLETE?;
160     --else
161     resultout := wf_engine.eng_timedout;
162     --end if;
163     return;
164 end if;
165 exception
166     when others then
167         WF_CORE.CONTEXT ('WFSOD', 'FilterUsers', itemtype,
168             itemkey, to_char(actid), funcmode);
169         raise;
170 end FilterUsers;
171

```

A.2 DeleteAdhocRole Procedure

```

1 PROCEDURE DeleteAdhocRole (itemtype in varchar2,
2                             itemkey in varchar2,
3                             actid in number,
4                             funcmode in varchar2,
5                             resultout out varchar2)
6 is
7     l_adhoc_role varchar2(30);
8     l_itemkeyname varchar2(30);
9     Begin
10
11     if ( funcmode = 'RUN' ) then
12         --<your RUN executable statements>
13         resultout := resultout;
14         return;
15     end if;
16     if ( funcmode = 'CANCEL' ) then
17         --<your CANCEL executable statements>
18         resultout := 'COMPLETE';
19         return;
20     end if; if ( funcmode = 'RESPOND' ) then
21
22         --find the adhoc role name which was crated by FilterUsers procedure
23         SELECT ASSIGNED_USER
24         INTO l_adhoc_role
25         FROM WF_ITEM_ACTIVITY_STATUSES
26         WHERE item_key = itemkey
27         AND process_activity = actid;
28
29         l_itemkeyname := itemkey||'%' ;
30
31         --delete the user-role associations
32         Delete from wf_local_user_roles
33         where role_name = l_adhoc_role
34         and role_name like l_itemkeyname;
35

```

```
36  --delete the adhoc role
37  Delete from wf_local_roles
38  where name = l_adhoc_role
39  and name like l_itemkeyname;
40
41  resultout := resultout;
42  return;
43  end if;
44  if ( funcmode = 'FORWARD' ) then
45  --<your FORWARD executable statements>
46  resultout := 'COMPLETE';
47  return;
48  end if;
49  if ( funcmode = 'TRANSFER' ) then
50  --<your TRANSFER executable statements>
51  resultout := 'COMPLETE';
52  return;
53  end if;
54  if ( funcmode = 'TIMEOUT' ) then
55  --<your TIMEOUT executable statements>
56  --if (<condition_ok_to_proceed>) then
57  -- resultout := ?COMPLETE?;
58  --else
59  resultout := wf_engine.eng_timedout;
60  return;
61  end if; exception
62  when others then
63  WF_CORE.CONTEXT ('WFSOD', 'DeleteAdhocRole', itemtype,
64  itemkey, to_char(actid), funcmode);
65  raise;
66  end DeleteAdhocRole;
```


Appendix B

Accompanying Material

The following additional material is supplied on a CD attached to the back cover of this dissertation.

First, the author has produced an academic paper based on the research undertaken by this dissertation. This paper is available in PDF format on the accompanying CD and is entitled: “Patterns for Dynamic Separation of Duties in Oracle Workflow”

Second, the electronic source to the Insurance Claim workflow example is included on the CD. It is accompanied by a text file with full instructions for setting up the Oracle Workflow client. Certain scripts will also need to be executed to create the necessary users and roles for this example. These scripts are included on the CD under the “Prototype/Scripts” folder, and instructions for running them are included in the text file.

References

- Ahn, G.-J., & Sandhu, R. (1999, 28–29 Oct). The RSL99 language for Role-based Separation of Duty Constraints. In *Proceedings of the 4th ACM Workshop on Role-based Access Control* (pp. 43–54).
- Ahn, G.-J., Sandhu, R., Kang, M., & Park, J. (2000, 26–28 Jul). Injecting RBAC to Secure a Web-based Workflow System. In *Proceedings of the 5th ACM workshop on Role-based Access Control*. New York, NY: ACM Press.
- Amoroso, D. (1998, Jan). Developing a model to understand reengineering project success. In *Proceedings of the thirty-first Hawaii international conference on system sciences: Volume 6* (pp. 500–509).
- Ash, C. G., & Burn, J. M. (2003). Assessing the benefits from e-business transformation through effective enterprise management. *Eur. J. Inf. Syst.*, 12(4), 297–308.
- Atluri, V., & Huang, W.-K. (1996, Sep). An Authorization Model for Workflows. In *Proceedings of the 5th European Symposium on Research in Computer Security* (pp. 44–64). Springer-Verlag.
- Bae, H., & Kim, Y. (2002). A document-process association model for workflow management. *Computers in Industry*, 47, 139–154.
- Baldock, P., & Seiden, R. (2000). *R11i/2.5 Oracle Workflow – Student Guide*. Redwood Shores, CA, USA: Oracle Corporation.
- Bertino, E., Ferrari, E., & Atluri, V. (1999, Feb). Specification and Enforcement of Authorization Constraints in Workflow Management Systems. *ACM Transactions on Information and System Security*, 2(1), 65–104.
- Bonifati, A., Casati, F., Dayal, U., & Shan, M.-C. (2001, 11–14th September). Warehousing Workflow Data: Challenges and Opportunities. In P. M. G. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, & R. T. Snodgrass (Eds.), *Proceedings of the Twenty-seventh Interna-*

- tional Conference on Very Large Databases* (pp. 649–652). Los Altos, CA 94022, USA: Morgan Kaufmann Publishers.
- Botha, R. A. (2001). *CoSAWoE – A Model for Context-sensitive Access Control in Workflow Environments*. Unpublished doctoral dissertation, Rand Afrikaans University.
- Botha, R. A., & Eloff, J. H. P. (2001a). Separation of Duties for Access Control Enforcement in Workflow Environments. *IBM Systems Journal*, *40*(3), 666–682.
- Botha, R. A., & Eloff, J. H. P. (2001b). A Framework for Access Control in Workflow Systems. *Information Management and Computer Security*, *9*(3), 126–133.
- Botha, R. A., & Eloff, J. H. P. (2002, June). An Access Control Architecture for XML documents in Workflow Environments. *South African Computer Journal*, *28*, 3–10.
- Brambilla, M., Ceri, S., Comai, S., Fraternali, P., & Manolescu, I. (2002). Specification and Design of Workflow-driven hypertexts. *Journal of Web Engineering*, *1*(1), 1–21.
- Caldow, J. (1999). *The Quest for Electronic Government: A Defining Vision*. <http://www.ieg.ibm.com/egovvison.pdf>.
- Chang, S., & Jaeckel, C. (2002). *Oracle Workflow Guide* (Vol. 1; Tech. Rep. No. Release 2.6.2). Redwood Shores, CA, USA: Oracle Corporation.
- Cholewka, D. G., Botha, R. A., & Eloff, J. H. P. (2000, 22–24 Aug). A Context-sensitive Access Control Model and Prototype Implementation. In S. Qing & J. H. P. Eloff (Eds.), *Information Security for Global Information Infrastructures: IFIP TC 11 Sixteenth Annual Working Conference on Information Security* (pp. 341–350). Beijing, China: Kluwer Academic Publishers.
- Duchessi, P., & Chengalur-Smith, I. (1998, May). Client/Server Benefits, Problems, Best Practices. *Communication of the ACM*, *41*(5), 87–94.
- Ferraiolo, D. F., Barkley, J. F., & Kuhn, D. R. (1999, Feb.). A Role-based Access Control Model and Reference Implementation within a Corporate Intranet. *ACM Transaction on Information and System Security*, *2*(1), 34–64.
- Gottschalk, K., Graham, S., Kreger, H., & J.Snell. (2002). Introduction to Web services architecture. *IBM Systems Journal*, *41*(2), 170–177.

- Gudes, E., van de Riet, R., Burg, J., & Olivier, M. S. (1997, October). Alter-egos and roles supporting workflow security in cyberspace. In *Proceedings of the IFIP WG 11.3 workshop on Database Security*. Lake Tahoe, USA.
- Hayes, J. G., Peyrovian, E., Sarin, S., Schmidt, M.-T., Swenson, K. D., & Weber, R. (2000, May). Workflow Interoperability Standards for the Internet. *Internet Computing*, 37–45.
- Hollingsworth, D. (1995, Jan). *The Workflow Reference Model* (Tech. Rep. No. TC-00-1003). www.wfmc.org: Workflow Management Coalition.
- IBM. (2000). *e-Business Process Automation*. United Kingdom: IBM.
- ISO 7498-2: *Information Processing Systems — Open System Interconnection — Basic Reference Model – Part 2: Security Architecture*. (1989).
- Kang, M., Park, J., & Froscher, J. (2001, May, 3 - 4). Access Control Mechanisms for Inter-organizational Workflow. In *Proceedings of sixth ACM symposium on access control models and technologies (SACMAT 2001)* (pp. 66–74). Chantilly, VA USA.
- Kim, Y., Kang, S., Kim, D., Bae, J., & Ju, K. (2000, May). WW-Flow: Web-based workflow management with run-time encapsulation. *IEEE Internet Computing*, 4(3), 55–64.
- Lamson, B. (1971). Protection. In *Proceedings of the 5th princeton symposium on information science and systems* (pp. 437–443). (Reprinted in *ACM Operating System Review* 8(1):18–24, 1974)
- Leyman, F., & Roller, D. (2000). *Production workflow: Concepts and techniques*. Upper Saddle River, New Jersey, USA: Prentice-Hall.
- Lu, J., & Chen, L. (2002). An architecture for building user-driven web tasks via web services. In *Ec-web '02: Proceedings of the third international conference on e-commerce and web technologies* (pp. 77–86). Springer-Verlag.
- Manolescu, D. A. (2001, June 15). *An extensible workflow architecture with objects and patterns*.
- Moore, C. (2000, April). *Workflow Goes Mainstream*. www.gigaweb.com: Giga Information Group.
- Nanda, M. G., Chandra, S., & Sarkar, V. (2004). Decentralizing Execution of Composite Web services. In *Proceedings of OOPSLA '04, Oct. 24-28, 2004, Vancouver, British Columbia, Canada* (pp. 170–187). Vancouver,

- British Columbia, Canada: ACM.
- Nyanchama, M., & Osborn, S. (1999, Feb.). The role-graph model and conflict of interest. *ACM Transactions on Information and System Security*, 2(1), 3–33.
- Oracle Technology Network. (2004, November). *Oracle workflow: Feature overview*. www.oracle.com/technology/products/ias/workflow/release262/workflow_fov.html.
- Perelson, S., Botha, R. A., & Eloff, J. H. P. (2001). Separation of duty administration. *South African Computer Journal*, 2001(28), 66–69.
- Sheth, A. P., van der Aalst, W., & Arpinar, I. B. (1999, Jul). Processes driving the networked economy. *IEEE Concurrency*, 7(3), 18–31.
- Siemens Nixdorf Informationssysteme. (1998, July). *Workflow Management Facility* (Tech. Rep.). <http://citeseer.ist.psu.edu/356772.html>: Siemens Nixdorf Informationssysteme.
- Simon, R., & Zurko, M. (1997, June). Separation of duty in role-based environments. In *10th IEEE computer security foundations workshop (CSFW '97)* (pp. 183–194). Washington - Brussels - Tokyo: IEEE.
- Sprague, R. H. (1995). Electronic document management: Challenges and opportunities for information systems managers. *MIS Quarterly*, 19(1), 29–49.
- Stallings, W. (1995). *Network and Internetwork Security Principles and Practice*. Prentice Hall.
- Swenson, K. (1998, Aug). *Simple Workflow Access Protocol SWAP* (Tech. Rep.). <http://www.ics.uci.edu/ietfswap>: Workflow Management Coalition.
- Teng, J. T., Jeong, S. R., & Grover, V. (1998, Jun). Profiling successful reengineering projects. *Communications of the ACM*, 41(6), 96–102.
- Valia, R., & Al-Salqan, Y. Y. (1997). Secure workflow environment. In *Wetice* (pp. 269–276).
- van der Aalst, W. M. P. (1999, December). Process-Oriented Architectures for Electronic Commerce and Interorganizational Workflow. *Information Systems*, 24(8), 639–671.
- van der Aalst, W. M. P., & Kumar, A. (2003). XML-based Schema Definition for Support of Interorganizational Workflow. *Info. Sys. Research*, 14(1), 23–46.

- von Solms, R. (1999). Information security management: why standards are important. *Information Management & Computer Security*, 07, 50-58.
- Wei-Kuang Huang and Vijay Atluri. (1999, October). Secureflow: A secure web-enabled workflow management system. In V. Atluri (Ed.), *Proceedings of the 4th ACM workshop on role-based access control* (pp. 83-94). Fairfax, VA, USA.
- Workflow Management Coalition. (1996, Jun). *Terminology and glossary* (Tech. Rep. No. WFMC-TC1011 Issue 2.0). www.wfmc.org: WorkFlow Management Coalition.
- Workflow Management Coalition. (1998a). *Workflow security considerations - white paper* (Tech. Rep. No. WFMC-TC-1019 Issue 1.1). www.wfmc.org: Workflow Management Coalition.
- Workflow Management Coalition. (1998b, Jun). *Workflow and Internet: Catalyst for Radical Change - a WfMC White Paper* (Tech. Rep.). www.wfmc.org: Workflow Management Coalition.
- Workflow Management Coalition. (2000, May). *Workflow standard- interoperability Wf-XML binding* (Tech. Rep. No. WfMC-TC-1023 Version 1.0). <http://www.aiim.org/wfmc/standards/docs/tc1023v10.pdf>: Workflow Management Coalition.
- Workflow Management Coalition. (2002, October). *Workflow Process Definition Interface - XML Process Definition Language* (Tech. Rep. No. WFMC-TC-1025 Version 1). http://www.wfmc.org/standards/docs/TC-1025_10_xpdl_102502.pdf: Workflow Management Coalition.
- Workflow Management Coalition. (2004, October). *Wf-XML 2.0 - XML Based Protocol for Run-Time Integration of Process Engines* (Tech. Rep. No. Draft document). <http://www.wfmc.org/standards/docs/WfXML20-200410c.pdf>: Workflow Management Coalition.
- Wu, Z., Deng, S., & Li, Y. (2004, September 15 - 18). Introducing EAI and Service Components into Process Management. In *Proceedings of the 2004 IEEE International Conference on Services Computing (SCC'04)* (pp. 271 - 276). IEEE.