# An Extended Role-based Access Control Model for Enterprise Systems and Web Services

A thesis submitted for the degree of

Master of Applied Science by Research (Computer Science)

Wei Shi

School of Computer Science and Information Technology,

Science, Engineering, and Technology Portfolio,

RMIT University,

Melbourne, Victoria, Australia

10th February, 2006

## Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; and any editorial work, paid or unpaid, carried out by a third party is acknowledged.

Wei Shi

School of Computer Science and Information Technology

RMIT University

10th February, 2006

## Acknowledgements

I would like to thank my supervisor Dr. Peter Bertok for all the work he has done during the past year. He has provided enormous support at the most difficult time of my research. His expertise in the area has guided me through many obstacles in the research. I am very grateful to have him as my principle supervisor as I have learnt many essential research skills, problem solving skills and thesis writing skills from him. I also need to thank my second supervisor Associate Professor Jim McGovern for many valuable comments and suggestions, and Associate Professor George Fernandez in the early part of this research.

# Contents

# List of Figures

# List of Tables

# Abstract

This thesis intends to develop application-level access control models to address several major security issues in enterprise environments. The first goal is to provide simple and efficient authorization specifications to reduce the complexity of security management. The second goal is to provide dynamic access control for Web service applications. The third goal is to provide an access control framework for Semantic Web services.

In this thesis, an Authorization-Function-Based Role-based Access Control (FB-RBAC) model is proposed for controlling enterprise systems at the application level. The unique features of the proposed model are authorization-function-based access control and constraint-based fine-grained access control. This model significantly simplifies the management of an access control system by adopting roles and authorization-functions in authorization specifications. An extension of FB-RBAC, Extended FB-RBAC (ERBAC), is applied to Web service applications. New features such as credential-based access control and dynamic role assignment are added to FB-RBAC in order to address user heterogeneity and dynamicity in the Web environment. The proposed ERBAC model is then extended to support Semantic Web services. Each component of the ERBAC model is described by security ontologies. These correlated security ontologies are

integrated with Semantic Web services to form a complete ontology network. Ontology-based role assignment is facilitated so that security information can be queries and discovered through a network of ontologies.

# Chapter 1

# Introduction

## 1.1 General Background

Large commercial organizations rely on enterprise information systems to manage their daily operations. A contemporary enterprise system is often distributed in nature. It brings together heterogeneous and autonomous applications from different locations to deliver specific business functions to end users and to other business applications [COET03, FRAN03]. These individual systems are written in different languages, run on various platforms, and follow a diversity of security models. A typical enterprise system is usually implemented in a Client/Server fashion and consists of client workstations and mainframe server [PERK03]. The application's presentation and business logic resides in the client tier, and its database management logic resides in the server tier [MCGO03]. A middle tier (enterprise tier) may be added in between the client tier and the server tier to encapsulate business logic; this is called the three-tier architecture [EDEL94]. In the three-tier architecture, the client tier manages the user interface and data

presentation only, and the mainframe tier becomes the database tier. Sometimes, the client tier is called the front end, and the database tier the back end [FRAN03].

To provide more interoperability among different sectors of a company, enterprise systems are now supported by Web technologies [COET03]. Web service is a new service paradigm established on top of existing Web technologies such as XML and HTTP, and is designed to support business integration in commercial organizations [WSAC]. It may be implemented in the middle tier or in a separate tier: a Web tier that resides between the client tier and the enterprise tier. Web services have many advantages over traditional communication approaches [BARR03, ERL04]. They can help companies connect standalone computer systems and get isolated applications to share information. Through Web services heterogeneous systems can interact with each other easily and effectively. Web services are easy to deploy and they do not require changes to the existing code and software. Web services are also platform-independent and programming-language-independent, so that they can be implemented in a variety of operating systems and programming environments.

However, Web services also face some major problems. One of them is semantic discrepancy. Data information in the Web environment is described in different ways due to the diversity of hardware platforms, programming languages and network protocols [AGAR04a, AGAR04b]. This so called semantic discrepancy has prevented Web-based information and services become accessible and understandable to a variety of agents and applications [CHEN03, GRIM04]. Recent development in the Semantic Web has resolved this problem by providing a common semantic framework to describe the content of Web pages as well as the properties and the capabilities of Web services [DENK03, SEMA]. The Semantic Web is the idea of the inventor of

World Wide Web, Tim Berners-Lee, and has been developed by the World Wide Web Consortium (W3C) as part of the effort to standardize Web-related technologies and systems [SEMA]. As a part of the semantic Web development, Semantic Web services are supported by ontologies and ontology-based languages like OWL [CHEN03, OWLa]. This allows Web resources and services to be queried and discovered automatically [DENK03]. In the meantime, research on Semantic Web service is still at its early stage, and its security aspects have yet to be extensively investigated.

The above-mentioned systems and services are all designed to serve particular groups of users. To protect them from unauthorized access, access control is required. Access control can be applied to different parts of an enterprise system, such as databases, operating systems, network communication and services, as well as business functions provided by enterprise system [BISH05]. Access control mechanisms for protecting entire enterprise systems are best implemented in the middle-tier of a three-tier client/server system. This is so because that preserves the autonomy of participating server tier applications whose existing access control systems are then not affected by enterprise-level access control mechanisms [BIDA98, VIME96]. In addition, access control mechanisms implemented at the enterprise-level will be independent of front-end user interface applications and back-end systems. Access control involves several important tasks [AMOR94, CAST94]. The first one is to identify what needs to be protected and in what way it should be protected. For example, it needs to know whether it is a database, file, business function or communication channel that the system wants to protect. The second task is to identify who is allowed to access the system. Traditionally, a user of a system is a human. In a broader sense, however, a user can also be a process, an application or a Web service. This

makes user administration more complicated. Thirdly, access control rules need to be established. This includes rules that determine which user can access particular resources or data, and what users can or cannot do with the resources or data, for example a user may be given the privilege to view a file but not to modify it. In addition, some environmental factors may also be included in the access control rules [ALKA02, BERT01c, BERT05, BHAT04b]. For example, the system may only be accessible during normal working hours (9am-5pm), or the system is available only for a certain number of users at a time. The study of access control will be centred on these three aspects.

## 1.2 Problem Statement

An enterprise system consists of multiple applications, resources from heterogeneous and autonomous sources [COET03, FRAN03]. Access control of the enterprise system involves many different aspects. The existing access control models are mainly designed for database or operating systems and are not suitable for application-level access control. These models are required to specify the protection status of individual data items with different access types. Whereas, at the application level, we only need to consider which functions/services of an application are requested/accessed, regardless of what data items or resources are involved. Therefore, it is necessary to develop application-level access control models to address security issues in enterprise environments.

One of the concerning issues is the large number of users and resources present in enterprise systems [ALKA02]. These users and resources are highly diversified. Traditional access control

systems usually rely on user identities to carry out authentication and authorization. As the number of users increases, it becomes difficult to specify and administer them against a large number of protected resources. A major challenge to the security management of an access control system is to specify authorizations that involve heterogeneous users and resources in a simple and efficient manner.

Secondly, providing access control to Web services is a challenging issue [DENK03]. The traditional identity-based access control paradigm does not suit a highly distributed and diversified Web environment where the attributes and properties of individual users and the environmental conditions (e.g. system load) change continuously [ALKA02]. It is necessary to capture and consider these dynamic factors when making access control decisions [BHAT03a, BHAT03b].

The Semantic Web has been developed to facilitate knowledge sharing and automatic service discovery. Semantic Web services are described by ontologies like DAML-S or OWL-S [OWL-S]. They provide common machine-interpretable semantics that allow services specified in such semantics to be queried and discovered automatically [SEMA]. However, research on providing security for the Semantic Web is still at its early stage. A Semantic Web compatible security framework is then required to facilitate security integration and to provide automated access control ability for Semantic Web services.

## 1.3 Aims and Approaches

Access control covers a broad spectrum of computer technologies. In this thesis, our aim is to provide access control at the application level that protects unauthorized access to business functions and services of general enterprise systems. An application-level access control model, Authorization-function-based access control (FB-RBAC) model, is proposed for this purpose. This model is implemented in the middle-tier of an enterprise system to complement the security infrastructure located at the back-end of the system. An important objective of this research is to simplify security administration and specification. An expressive security specification, Authorization-function, are proposed, that are adaptable to access control at the application level. At the same time, the model needs to support a wide range of security policies, which is an essential requirement for a successful access control model. Access control policies may be specified in varying granularity to support different access control scenarios. Some of the existing access control techniques, such as content-based access control and constraint-based access control will be applied and extended to support different types of security policies. Furthermore, since many of the enterprise systems are Web-based, the proposed model also needs to consider access control requirements in a Web environment. The proposed model needs to provide dynamic access control capturing a variety of security aspects, such as user capabilities (specifies what users can do), external factors (e.g. time, location, system status) and predefined access control policies (who can access what resources) in Web environment. Besides, our research also considers the development of the Semantic Web. As the Semantic Web is described by ontologies, this thesis intends to establish a formal access control framework to represent security related concepts using ontologies. The ontology-based access control

framework will allow security services to be in line with the development of Semantic Web. Integration of security ontologies with the Semantic Web can provide sophisticated reasoning ability through a network of ontologies.

## 1.4 Summary of Contributions

The proposed model provides overall access control for business functions and services of an integrated enterprise system. It is especially designed for application-level access control and for avoiding the complexity at the level of individual data items as practiced in database security. It significantly simplifies the security management of an enterprise system by providing a new form of permission: authorization-function. An authorization-function is more efficient and expressive than existing techniques for application-level authorization specifications. In addition, the proposed model is capable of supporting a variety of security policies for different access control scenarios. An extension of this model is applied to Web service applications. This extension provides dynamic access control to address the problem of heterogeneity and dynamicity of users, system resources and environmental conditions in the Web environment. By allowing access control to be dynamic, access control decisions can be synchronized with continuously changing security conditions. As a result, this model will be adaptable to the surrounding environment. Moreover, our research provides access control for the newly developed Semantic Web service. The proposed ontology-based access control framework provides formal semantics for security concepts and security policies. Such a framework can be easily integrated with Semantic Web

services to provide automatic reasoning ability for querying and discovering security-related information.

## 1.5 Structure of the Thesis

Chapter 2 introduces terms and concepts that are used in later chapters. Two well-known security models, Access Control Matrix (ACM) and Role-based Access Control (RBAC) will be discussed in detail.

In chapter 3, an Authorization-function-based access control (FB-RBAC) model is proposed for enterprise systems. We will discuss how this model can support efficient security management and a variety of access control policies at the application level.

In chapter 4, an access control model is proposed for Web services. This model extends FB-RBAC from chapter 3 with a range of access control techniques like credential-based access control and constraint-based access control. This extended model (ERBAC) facilitates dynamic access control decision making according to a range of internal and external factors.

In chapter 5, an ontology-based access control model (OB-ERBAC) is proposed to support Semantic Web services. Ontology, a semantic description technique used in the Semantic Web, is applied to ERBAC described in chapter 4. A bookstore example is used to demonstrate the effectiveness of the OB-ERBAC model.

Chapter 6 summarizes the major results of the thesis, and outlines the research contribution. Some future research topics are also mentioned.

# Chapter 2

# Background

This chapter will introduce the main security concepts and security terminologies used in later chapters. Since this thesis is aimed at providing access control for enterprise systems, access control will be the main focus of the review. Some popular access control policies, access control mechanisms and access control policy management models will be analysed. A summary of the access control requirements for enterprise systems is provided at the end of the chapter.

## 2.1 Security Concepts and Terminologies

### 2.1.1 Security in General

The basic requirements of computer security are to satisfy the following six goals: confidentiality, integrity, availability, privacy, authenticity and non-repudiation. Confidentiality prevents data from being seen by unauthorized parties. Integrity prevents unauthorized changes to data or resources. Availability refers to the ability to use the information or resource desired.

Privacy means information given by a user should only be used for the purpose it was given. Authentication is to prove whether the information given by a user is genuine. Non-repudiation means the sender of a message cannot deny the transmission. These goals are realized through security policies and security mechanisms. Security policies describe what is allowed and what is not allowed by using mathematical expressions. They provide high-level guidelines for design and management of an authorization system [FERN89]. A particular policy or set of policies may be represented by a security model [GOLL1999]. A security model provides high-level guidance towards the implementation of a security system [CAST94]. A security mechanism is a method, tool, or procedure for enforcing a security policy [GOLL1999]. Specific mechanisms include encryption, digital signature, access control, message authentication codes and authentication exchange. There are three different types of security mechanisms [AMOR94, CAST94]. Prevention mechanisms perform access control, detection mechanisms carry out auditing and intrusion detection and tolerance mechanisms ensure practicality. Prevention or access control is fundamental to the proper operation of a security system.

### 2.1.2 Access Control

The main objective of the thesis is to provide access control for enterprise-wide systems. The goal of access control is to protect enterprise systems from unauthorized access. Access control provides a set of policies and mechanisms that permit authorized subjects to access protected resources [BISH05]. Our review will focus on access control mechanisms and access control policies. An access control mechanism is based on three types of information: subject, object and access mode. Subjects are active entities of the system who request to access protected resources.

Objects are passive entities of the system that represent resources to be protected. Access modes (or access types, operations) represent the types of access that subjects can exercise on objects. These three types of information are used to describe the permission for a subject to access a particular object in a given access mode at a particular time in the access control process. This permission is called authorization. Figure 2.1 represents the relationship between the basic elements of a security model.

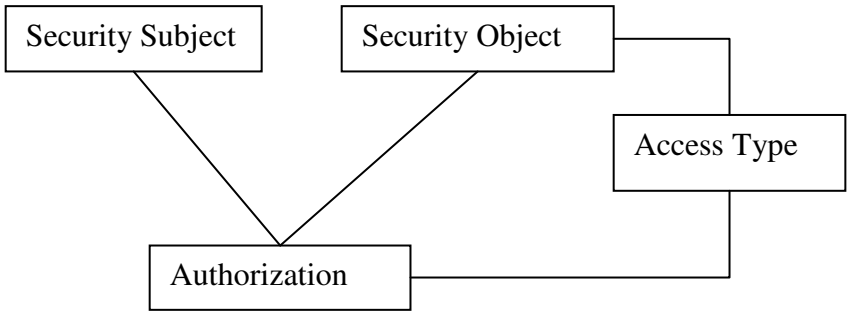



*Figure 2.1 Basic Elements of a Security Model*

Access control is usually performed through a device called a *reference monitor* which grants or denies access. A reference monitor evaluates an access request based on access control policies and ultimately makes a decision whether access is granted, denied or undecidable. A typical access control system is illustrated in Figure 2.2:

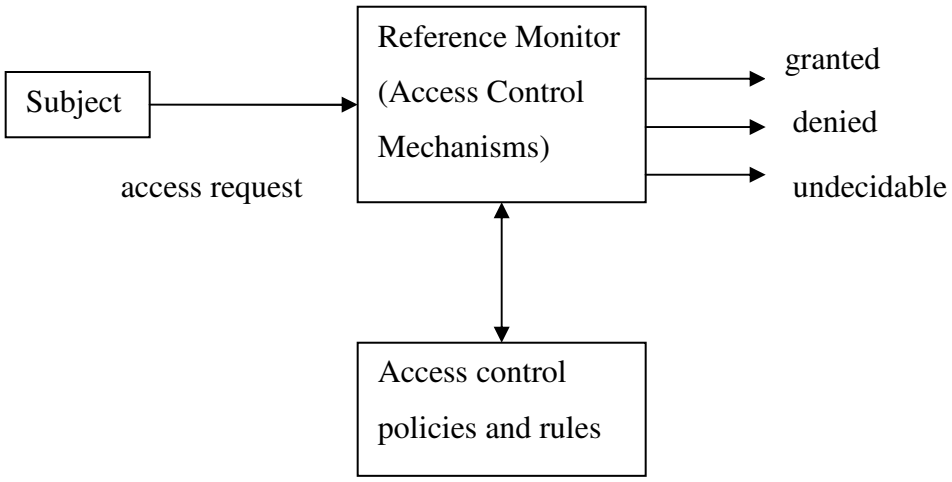



*Figure 2.2 A Typical Access Control System*

## 2.2 Access Control Policy Management Models

### 2.2.1 Access Control Matrix

One of the classic security models used in both operating systems and database systems is the Access Control Matrix (ACM) [HARR76]. An ACM is shown in Table 2.1. In ACM, the authorization state can be described as a triple (S, O, A) where S represents the subject, O represents the object and A is the access mode. The matrix rows represent the subjects, the columns represent the objects, and the entry of A[s,o] stores the access modes. Traditionally, the main task of access control is to protect files and databases, and typical access modes are Read (R), Write (W), Delete (D) and Create (C).

| Subject\Object | O1 | O2 | …… | Om |
|---|---|---|---|---|
| S1 | A[S1, O1] | A[S1, O2] | | A[S1, Om] |
| S2 | A[S2, O1] | A[S2, O2] | | A[S2, Om] |
| …… | | | | |
| Sn | A[Sn, O1] | A[Sn, O2] | | A[Sn, Om] |

*Table 2.1 Access Control Matrix*

Every request from a subject S to access an object O in a given mode M is determined by the corresponding entry A[S, O] in the ACM. The request is granted if M can be found in A[S,O]. However, storing the full matrix as a two-dimensional array is inefficient since many of the entries may be empty and some subjects/objects can be inactive [CAST94]. An alternative solution is to store the ACM by rows or columns. An Access Control List (ACL) stores authorizations by columns. It links each object with its associated subjects and their rights on the object. A Capability List (CL) stores authorizations by rows. It specifies the capability of each subject by identifying the associated objects and the types of access exercised on the object. We provide an example below (Table 2.2 and Figure 2.3) to show how access control information can be stored in ACM, ACL and CL.

| Subject\Object | O1 | O2 |
|---|---|---|
| S1 | R, W | R |
| S2 | R | |
| S3 | | R, W, M |

*Table 2.2 An Example of an Access Control Matrix*

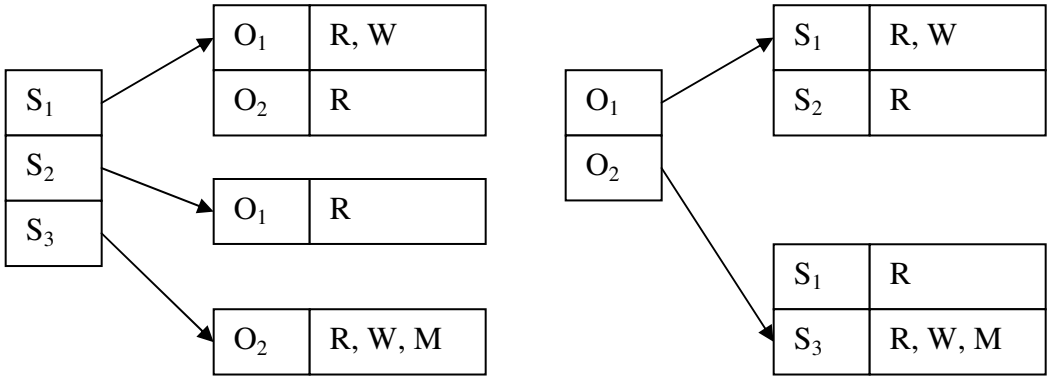



*Figure 2.3 Access Control Matrix (ACM) Implementations: Access Control List (ACL) and Capability List (CL)*

In the CL and ACL representations, the empty entries of the matrix are not stored (refer to Figure 2.3). Hence, the number of entries in the access control matrix can be reduced significantly when there are large numbers of subjects and objects.

An ACM can be extended by adding conditions to the access modes [CAST94]. Possible conditions are:

- Data-dependent conditions, specifying constraints on the value of the accessed data.
- Time-dependent conditions, specifying constraints on the time an access can take place.
- Context-dependent conditions, specifying constraints on combinations of data which can be accessed. E.g. a subject may be able to read “employee name” and “employee salary” separately, but not when they are associated pairs “name-salary”.

- History-dependent conditions, specifying constraints dependent on previously performed accesses.

The access control matrix model is the most precise model for describing authorization states [BISH05]. Many models have been developed on top of the access control matrix model. However, since the ACM is mainly used to control access to data or resources, it may not be the best solution to describe application-level access control. We will discuss this issue in chapter 3.

### 2.2.2 Discretionary Access Control (DAC) and Mandatory Access Control (MAC)

Access control policies can be represented by a variety of models according to their characteristics [AMOR94]. Discretionary Access Control (DAC) and Mandatory Access Control (MAC) are two basic types of access control models [JOSH01]. DAC allows owners of objects to control the granting of access rights. It bases access rights on the identity of the subjects and objects involved. MAC uses centralized control to grant or deny access rights, and individual users cannot alter those rights, access rights are based on the classification of subjects and objects in the system. Compared to DAC, MAC has high security assertions, which makes it less flexible than DAC. These two access control mechanisms are used in different situations: DAC is adopted by many commercial applications; whereas MAC is mostly used by government and military applications [CLAR87, FERR92].

### 2.2.3 Role-based Access Control

Access control of shared information has traditionally been performed by access control lists or capability lists [CAST94]. However, they are inadequate for addressing large number of

subjects and objects presented in heterogeneous and distributed enterprise environments. In the early 1990s, role-based access control (RBAC) [FERR92, SAND94] rapidly emerged and became a proven technology for managing and enforcing security in large-scale enterprise-wide systems. The RBAC model was formally introduced by Sandhu et al. [SAND96] in 1996. A major purpose of RBAC is to provide more efficient and effective security administration [NYAN95].

Role-based Access Control (RBAC) uses three basic entities: users (U), roles (R), and permissions (P) (as shown in Figure 2.4).

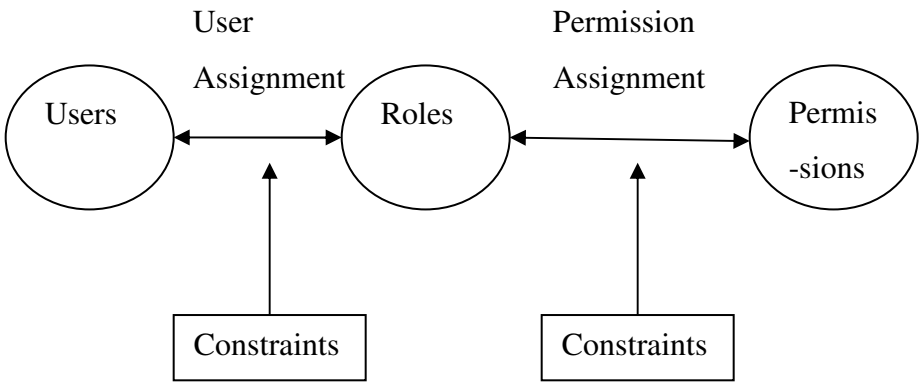



*Figure 2.4 Role-based Access Control Model*

A user can be a human or a computer process. A role is a job function or job title within the organization, and is used for describing a collection of users with the same privileges. A permission is an approval of a particular mode of access to one or more objects in the system, it is equivalent to the term: authorization, access right or privilege. A role is used to associate users with permissions. The fundamental relations in RBAC are user-role assignment and permission-

role assignment, that is, assigning users and permissions to appropriate roles. The idea of role is closely related to the concept of user group in traditional AC models. However, user group are usually defined as sets of users only; without associating them with permissions. The NIST study [FERR93] also indicates that roles in an organization are more stable than user groups, the latter changing more frequently.

In addition to the basic components of the model, RBAC also provides role hierarchies and constraints [SAND96]. Role hierarchies are established when roles can inherit permissions from other roles [BARK97]. Multiple inheritance of permissions is also possible, i.e. one role can inherit permissions from multiple roles. Constraints are used to impose restrictions on acceptable configurations of the different components in RBAC [AHN00, JAEG99]. A common example is mutually disjoint roles, such as the purchasing manager role and the accounts payable manager role. In RBAC, permissions are always positive, and denial of access is modelled as a constraint rather than a negative permission [COVI00]. Constraints can be applied to user-role assignment and permission-role assignment, to users and to roles for various sessions [BARK03]. In addition, constraints can also be applied to the role hierarchy itself. Additional constraints can limit the number of senior (or junior) roles that a given role may have. However, as a result of bringing together constraints and hierarchies, conflicting authorizations may appear. Session is another important concept in RBAC. Each session maps one user to possibly many roles and is only associated with a single user. A user may have multiple sessions open at the same time, each in a different window on the workstation screen for instance.

The RBAC model has many advantages over the traditional AC model [SAND96, COVI00, JOSH01]. It is policy neutral, thus is able to support both DAC and MAC. The use of role

concept in security administration clearly defines the duties and rights for each position in distributed systems. Furthermore, having roles in between users and permissions enables users to exercise much greater control over access configuration and review than by directly relating users to permissions [SAND96]. The establishment of role hierarchies naturally reflect the structure of an organization and the responsibility of individuals in the organization, such as a senior role inherits all permissions held by a junior role [ANDE04, BARK97]. All these advantages make the RBAC model suitable for enterprise security systems [BHAT04b, JOSH01, LIU04]. However, in order to provide access control for distributed enterprise systems at the application level, RBAC needs to be extended and modified to be more flexible and to address specific requirements [ALKA02, BERT05, BHAT03, COVI00, YANG03]. For example, in the RBAC model, permission is an abstract concept specified by objects and access modes (operation-object), and specific meaning has to be attributed to it in different situations. A simple and efficient expression of permission is required to suit application-level access control. The meaning of user in the model should not be restricted to humans that are represented by their identities. A user may be a process, an application or a service, and user credentials may be used instead of traditional user identities. Also, user-role assignment may not be predefined by administrators, but rather it can be dynamic according to access control policies and environmental factors.

## 2.3 Access Control Requirements for Enterprise Systems

Modern enterprise systems are often distributed and take advantage of Internet technology, which allows information stored at different physical or logical locations to be shared across the

entire enterprise system [LOSA02, MCGO03, RICH04]. However, the heterogeneous and distributed nature of enterprise systems, especially Web-based enterprise systems, pose many security challenges. To address these challenges, Coetzee and Eloff [COET03] provided a comprehensive summary of Access Control (AC) requirements for enterprise systems.

1. Identity management -- integrate the existing multiple authentication mechanism to form a unified identity management mechanism.
2. Policy management – access control policy needs to be properly managed by policy specification language and other mechanisms [BIDA98].
3. Trust influences access control enforcement – inter-domain/inter-application communication relies on the trust of the service provider towards the service requester [BLAZ00].
4. Flexible access control decisions are required to support policy changes – access control decision making mechanisms should not be bound to the application [BEZN98].
5. Standards-based implementation – security standards aiming at broad frameworks ought to be independent of technologies and configuration models. The adoption of standards such as Security Assertions Markup Language (SAML), XML Access Control Language (XACML) [ANDE04] and WS-security [WSPO] allow interoperation between different domains in enterprise systems.
6. Independent security domains should be maintained – each domain or application in an enterprise system has its own access control mechanism that is governed by a variety of access control policies. The ability of a domain or application to make authorization decisions is called authorization autonomy [VIME96].
7. Separation of policy from mechanisms – access control policies need to be decoupled from the decision-making mechanisms [BEZN98].

8. Integration of existing access control policies – independent access control policies from component applications need to be integrated into the access control policies of the enterprise system [HALE99, IDRS94].
9. The order of events must be controlled – enterprise users may invoke multiple Web services to communicate with different parties. The result produced by one Web service can influence other Web services [BPEL03].
10. Access control policy composition – The global policy of an enterprise system is composed of independent local access control policies [VIME96]. This process should be performed dynamically according to the changing environments.

The above requirements outlined some of the essential issues in enterprise access control. In this thesis, we will only address the first five topics in the list. The rest of the topics are beyond the scope of this thesis.

## 2.4 Summary

In this chapter, we introduced security concepts and terminologies that would be frequently used in the later chapters. A variety of security policies and security mechanisms were analysed. A classical access control mechanism/model, Access Control Matrix (ACM) and its extensions were presented. ACM is the most precise model for describing authorization states. In addition, many later models were developed on top of ACM. As a popular access control model, the Role-based Access Control (RBAC) model was the main focus of this review. It was governed by two basic relations: user-role assignment and permission-role assignment. The use of role has significantly reduced security administration and made access control more efficient and effective.

RBAC also includes two other features: role hierarchy and constraints. Role hierarchy can reduce the number of authorizations stored in memory, whereas constraints can be used to support complex access control policies. At the end of the chapter, we provided a summary of access control requirements for enterprise systems, and identified issues that would be addressed in this thesis.

# Chapter 3

# Authorization-function-based RBAC

## 3.1 Introduction

Supported by improved communication technologies, an enterprise system can now integrate many heterogeneous and autonomous components, such as numerous database systems, file/document systems or resource management systems, in order to provide unified functionality [PAYT00, PERK03]. However, it also brings together a large numbers of users and a variety of resources. This has caused serious problems with regard to access control and security management [ALKA02, BHAT04b]. In the past, a number of important security models have been proposed for a variety of systems. Earlier models, such as the access matrix model (ACM), [HARR76] are mostly designed for operating systems and database systems. Several influential models, such as Bertino's model [BERT99] for relational databases and Rabitti's model [RABI91] for Object-Oriented databases, have addressed database access control extensively. The development of Web technology has led to recent models that address XML document

security [BERT01a] [BERT01b] [BHAT04a]. These models were all developed in the framework of subject-operation-object, a framework initially introduced by the ACM. There are a number of other access control approaches, including Task-based Access Control (TBAC) [THOM93], Partition-rule based Access Control (PRAC) [ACEV97], and per-method access control [EVER00, EVER02]. These approaches, however, were designed for particular types of systems or to solve particular types of problems, and are not generally applicable. The above-mentioned models are inadequate to address security issues in the new enterprise environment. A generic application-level model is required for enterprise systems.

### 3.1.1 Problem Statement

In enterprise systems, a pressing issue is the lack of an efficient, generic application-level model that supports unified access control frameworks for user-to-system interactions. Traditional security models are not capable of addressing some of the new challenges posed by modern enterprise systems [ACEV97]. One of the problems is that most of the existing security models are influenced by the subject-operation-object paradigm. A typical feature of the paradigm is that privileges or permissions are represented as approvals of access to an object in specified access modes (operation-object). This type of permission representation has complicated the security management of large enterprise systems due to the heterogeneity of resources involved in authorizations. Hence, they are not suitable for supporting a unified access control framework that involves different types of resources from multiple domains.

An essential goal of this chapter is to provide an efficient and generic description of authorization. In a distributed application, user heterogeneity has been resolved by assigning

users to roles, however, the problem of resource heterogeneity in security management has not been properly addressed as mentioned above. This is especially so when the execution of a user request involves a series of actions that access different types of resources in multiple domains, and security management becomes extremely difficult. For example, a user request to "View Customer Record" may ultimately access customer personal details from a company's database system, a customer portrait from an image base, customer contract agreements reached on the phone from an audio repository, and XML-based Customer Orders from the sales department. At the application level, the only concerning issue is whether or not "View Customer Record" can be accessed, not how it is conducted at a lower level. It would be unnecessary to specify and evaluate accesses to individual data items when performing application-level access control

An authorization representation also needs to be expressive enough to describe access control policies for a variety of resources. This requirement is essential to a unified access control framework. Due to the heterogeneity of distributed systems, a unique set of access modes is established for every type of resource. For example, in a database system, access modes may be defined as "read", "write", "update", whereas in a file management system, access modes can be "view", "modify", "create", "destroy". Authorizations specified by heterogeneous resources and a diversity of access modes can be complex. The situation becomes worse if object hierarchy and access modes hierarchy are introduced. Although the hierarchical structures may simplify the authorization specification to some extent, it can introduce many conflicts to the authorization. Consequently, a set of conflict resolutions must be developed. This will not only introduce more complexity to a security system, but also cause possible service repudiation due to authorization

inconsistency. Furthermore, these access types are bound to specific resources or systems, hence they cannot provide support for a unified framework.

Another important aspect of an access control model is the ability to support a wide range of security policies. In a distributed system, a variety of security policies are specified to ensure data confidentiality and integrity, and to convey business rules, some of them must be specified in a fine-grained manner [ACEV97]. For example, if a user requests access to some service, but only part of the service is available to this user, simply permitting or forbidding the user request would be inappropriate. In a similar case, a service can only be accessed within a certain time frame. A security system is required to provide more precise authorization services to satisfy both business requirements and security requirements of the enterprise. However, current business applications/systems do not have a systematic way of defining access control to such fine granularity. The existing RBAC models and their extensions are unable to provide fine-grained access control for enterprise systems, such as controlling the content of user input and system output.

### 3.1.2 Outline of the Proposed Solution

Having identified the problems in the area, this chapter presents an enhanced version of RBAC – Authorization-function-based RBAC (FB-RBAC) model. The proposed model aims to provide an efficient and unified access control framework for enterprise-wide systems. It is designed for application-level access control that resides on top of all the existing lower level (back-end) security systems, such as database security systems and secure file systems. In this model, *authorization-function* (hereafter, *A-function*) is used as the basic unit of access control.

A-functions represent business functionalities exposed by the enterprise application interface and are proxies to access various back-end resources.

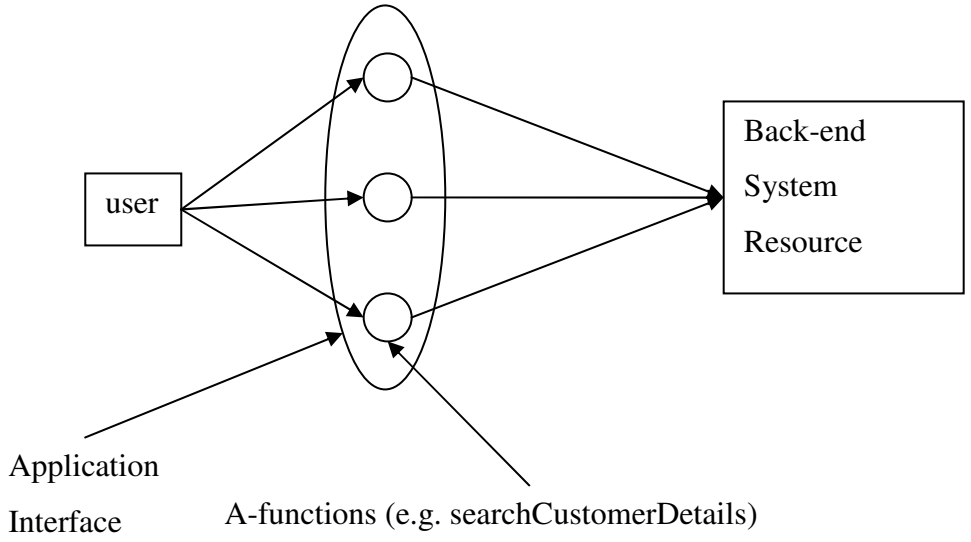



*Figure 3.1 The Application of Authorization-functions in Enterprise Systems*

Accordingly, an authorization is specified as an approval of access to an A-function. This has significantly simplified security specification and administration. A-function is a system and platform independent notion, hence it can support unified access control for an integrated system. However, A-function alone cannot describe a variety of security policies. In this thesis, the concept of constraint defined in RBAC is used to specify complex security policies. When constraints are imposed on the input and output parameters of an A-function, the system will be able to control not only what A-function is accessed, but also the content of the input to the

system and the result of execution. Furthermore, a simple and self-explanatory policy algebra is provided to express authorizations and various constraint-based security policies.

### 3.1.3 Summary of Contributions

The main contribution of this chapter is that it provides an enhanced RBAC – authorization-function-based RBAC (FB-RBAC) for unified application-level access control in enterprise systems. A new concept of permission – Authorization-function (A-function) is introduced for authorization specification. A-function is a simpler expression compared to the traditional operation-object paradigm. Hence, the introduction of A-function in authorization specification can significantly simplify security management in enterprise-wide security systems. The FB-RBAC model is not restricted to enterprise applications and platforms, since the A-function can be used to describe any type of resources and access types. The efficiency and generality of the model also provide support for a unified access control framework, and bring together existing lower level security components within a multi-application/multi-domain enterprise system. Additionally, this model also supports a wide range of possibly fine-grained access control policies using the concept of constraints. It is envisaged that the proposed model can be applied to a variety of enterprise applications such as the emerging Web service applications perhaps with some modification.

### 3.1.4 Structure of the Chapter

The rest of this chapter is organized as follows. Section 3.2 sets the scene by reviewing relevant literature. It discusses and compares the security management in various security models,

especially the RBAC model and its extensions, which are analyzed in detail. It argues that the existing security models are unable to address some of the challenges posed by modern enterprise systems. Section 3.3 will address the problem mentioned in section 3.1 and section 3.2. It presents the A-function-based RBAC model extending the existing RBAC model. A detailed discussion will be provided on how the A-function can be used to support efficient security management. In addition, constraints are used extensively to describe various security policies, and a simple policy algebra is used to describe authorizations and security policies. The contributions of the proposed solution and its advantages over the existing solutions are discussed in section 3.4. Finally, the major arguments and findings of this chapter are summarized in section 3.5.

## 3.2 Background

An enterprise system needs to provide various users with controlled access to heterogenous resources. Traditional access control models designed for operating systems [HARR76] or database systems [BERT99, RABI91] are constrained by the framework of subject, object and access modes (or operation, access rights, access type). This subject-operation-object paradigm has several drawbacks (as mentioned in section 3.1.1). One of them is that this paradigm cannot efficiently describe authorizations and security policies; moreover, this paradigm is tightly coupled with specific applications/systems, hence it cannot support a unified access control framework. This section provides a review of the existing security models with regard to enterprise access control. In particular, RBAC and its various extensions will be discussed in

detail, since they have been the main focus of the recent research in security models, and are widely adopted in many enterprise systems.

### 3.2.1 Access Control Requirements for Enterprise-wide Systems

A fundamental difference between enterprise systems and traditional centralised systems is that the former is extremely diversified and distributed in nature. This feature poses great challenges (as mentioned in section 3.1.1) to access control and security management of enterprise-wide systems. According to relevant studies [ACEV97, CHAN03], an effective enterprise access control framework needs to:

- support efficient access control and security management for enterprise systems,
- provide a generic access control mechanism that is capable of describing heterogeneous resources hosted by various applications,
- specify a variety of security policies and model different access control scenarios in enterprise systems, and
- include a wide range of authorization conditions/constraints in access control decision making.

In this section, models and the aspects of the models in relation to the above-mentioned access control requirements for enterprise security systems are discussed. Their relative merits and drawbacks will be analysed in the following section.

### 3.2.2 Access Control Models for Enterprise Systems

One of the alternative approaches to traditional subject-operation-object paradigm is task-based access control (TBAC) proposed by Thomas and Sandhu [THOM93]. TBAC views access control and authorization not in terms of individual subject and objects, but in terms of long-lived tasks that need to be authorized and managed in information systems. This approach has some advantages over the subject-operation-object approach, because the term "task" is more generic than the combination of operation and object. The TBAC model can very well model task-oriented security policies like Separation of Duty (SoD) and the Chinese Wall, as well as carry out work flow control effectively. However, this model is specifically designed to protect business activities and transactions in a distributed environment. It does not provide generic access control mechanisms for handling a variety of situations in enterprise-wide systems.

With the popularity of the Object-Oriented (OO) paradigm, an Object-Oriented access control model was proposed for general OO-based systems [EVER00, EVER02]. This model uses per-method access control based on the interface methods of an object. At the same time, constraints can be imposed on user-to-system interactions, based on the parameters of the methods. This allows the per-method access control to provide fine-grained semantic protection that the traditional per-attribute (data item) access control models are unable to achieve. In the security specification, the concept of method can describe user-to-system interaction more efficiently than the traditional per-attribute access control. However, per-method access control is only applicable to Object-Oriented database systems. Yet an enterprise access control system should be generic regardless of the underlying system architecture.

Although both models have improved object heterogeneity by using “task” or “method” as the basic access control unit, they are still unable to manage large numbers of users in enterprise systems. Both models fall short of generality as each of them targets at a specific problem or a specific system in the area. Hence, they cannot be used to support a unified access control framework that involves heterogeneous users and resources.

### 3.2.3 Role-based Access Control

The background chapter introduced the main components and contributions of Role-based Access Control (RBAC) [SAND96]. Among all existing models, RBAC is considered to be the most suitable for enterprise systems, due to its generality and expressiveness [AHN00, CHAN03]. This section will focus on how authorization is represented in RBAC and the concept of constraints in RBAC. In RBAC96, a permission is described as an approval of a particular mode of access to one or more objects in the system [SAND96]. Permissions are assigned to roles rather than individual users, since the concept of role provides long-term identity for the system. For instance, in a company, if a user is a system administrator and is assigned to the role “Sys Admin”, then this user is more likely to hold this role as long as they work as a system administrator in the company. Hence, the notion of role has modelled the position and privileges of a user in an organisation concisely. Therefore, permissions assigned to each role are changed less frequently than permissions assigned to individual users. Compared with the huge number of users in a distributed system, the number of roles is much smaller and thus becomes more manageable. This has significantly reduced the size of the authorization base and the level of administration complexity. The RBAC model also allows inheritance so that role hierarchies may

be established. In a role hierarchy, senior roles inherit all privileges from their junior roles. By using role inheritance, only permissions that are specific to the role are defined; permissions that are common to both senior roles and junior roles are specified for the junior roles only. This structure avoids repetition of permission specifications and can further reduce the number of authorizations stored in the authorization base.

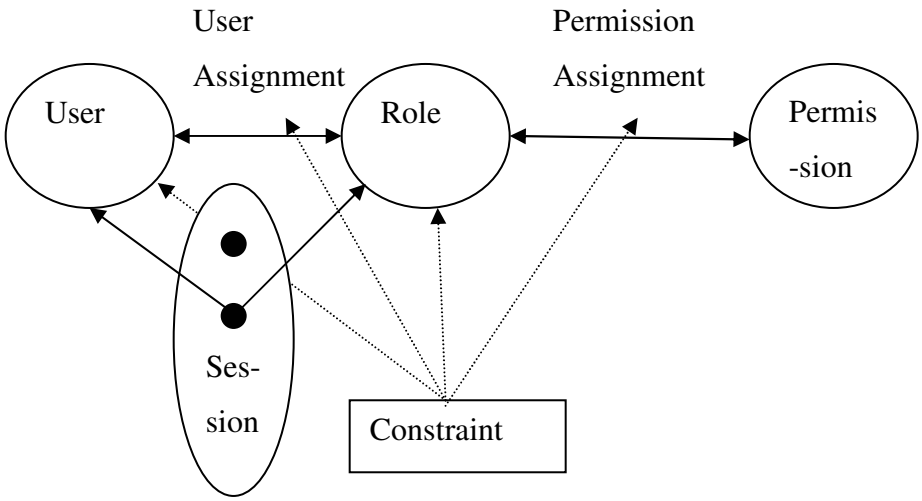



*Figure 3.2 RBAC model (modified from Figure 1 in [SAND96])*

However, in RBAC96 [SAND96], a permission is still represented in the form of operation-object which describes the traditional database access paradigm. In enterprise systems, there are a large number and types of resources, such as database items, files, and heterogeneous hardware. When authorizations are specified by operation-object paradigm, an enterprise system needs to define a set of access modes for each type of resource, such as read, write, update for databases,

view element, view schema, destroy and create for documents – which is inefficient. With the use of access rights hierarchy and object hierarchy, this paradigm becomes even more complicated, hence is no longer suitable for describing heterogenous resources in distributed systems.

RBAC96 uses the concept of constraint to specify conditions that must be satisfied in authorizations. Constraints (as indicated by the Figure 3.2) can be applied virtually to any of the concepts specified in the RBAC model. Yet when the concept of constraint is used in a user/role assignment in an active session directly, it will affect authorization decisions. By using constraints, the authors demonstrated that RBAC is capable of supporting some well-known security policies such as Separation of Duty (SoD) and prerequisite role. However, constraints in the context of RBAC did not receive enough attention in RBAC96, although they carry great importance as they can be used in various situations to specify a wide range of security policies.

### 3.2.4 Extensions of the RBAC model

Since it was first proposed, various extensions have been made to RBAC, aiming to improve one or more aspects of the model. For instance, Covington and Moyer proposed an extension to RBAC -- generalized RBAC [COVI00] that includes not only subject role, but also object role and environmental role. Here an object is classified into a relevant object role according to the properties of this object. Therefore, policy specifications are not based on individual subjects and objects, but on subject roles and object roles. The GRBAC model also allows policy designers to specify system state through environment roles. An environment role can be based on any system state that the system can accurately collect. This role allows various constraints to be specified as

authorizations. Hence the model is able to describe a variety of security policies based on environmental roles. The use of object role has improved object heterogeneity, hence further simplifying the authorization specification. However, the side effect is that it also increased the complexity of policy administration and evaluation, since all system objects need to be classified into object roles for the model to work. Such a classification is not an easy task, because some objects in the system may not have any classifiable property, while others have multiple properties that can be assigned to a number of object roles. In fact, GRBAC's generality makes it even more susceptible to various types of policy conflicts and ambiguities.

The ORBAC model [YANG03] proposed a new role concept -- *task role* in addition to the traditional role concept (which is named *position role* instead). Each position role contains several task roles, and task roles are given privileges to perform specific tasks. It is suggested that the specification of security policies like SoD can be better described using task roles instead of the traditional position role.

RBAC has also been used for Web Services in Business Process (WS-RBAC4BP) [LIU04]. WS-RBAC4BP differs from traditional RBAC in that it takes company as the subject and Web services as the protected object. The authors argue that the identities of individual users are not important in business processes, therefore, in WS-RBAC4BP, security constraints are defined in terms of company. However, both frameworks are designed to resolve specific problems in the area, and do not possess a generic access control framework.

Wang and Li [WANG04] proposed an extensive role-based access control (ERBAC) model. This model substitutes the abstract notion of permission with a set of "tasks" based on organizational task decomposition. This approach has in fact combined the advantages of RBAC

and TBAC [THOM93]. Hence, it can significantly simplify the specification and administration of authorizations and security policies. However, this paper has only discussed the use of business tasks in implementing business oriented separation of duty (SoD) policy. It did not provide a comprehensive analysis on how business tasks can be used to enforce access control in different situations, and to support a variety of security policies besides SoD.

### 3.2.5 Summary of Existing Work

Among the existing security models, task-based access control and per-method access control are two unique approaches that have addressed the problems of object heterogeneity in enterprise systems. However, both approaches focus on specific types of problems without providing a generic framework for various enterprise systems. RBAC has been widely recognized as a promising technology for enterprise-wide systems. Compared with traditional security models, RBAC has effectively reduced the number of subjects as well as the complexity of permission management due to the flexibility with which roles may be configured and reconfigured. Since RBAC is a policy neutral model, it is also able to support a wide range of security policies.

A great deal of effort has been devoted to extending RBAC and improving RBAC in various aspects. Some extended the role concept to include subject role, object role and environmental role [COVI00], others defined task role and position role to facilitate more specific and fine-grained permission assignment [YANG03]. In all the above-mentioned models, permissions are represented in a variety of forms ranging from the traditional operation-object form [SAND96] to various new forms like task [WANG04], method [EVER00], company [LIU04] and object role [COVI00]. However, these representations are inadequate for describing a large number and

types of resources. Some of them target at specific problems or systems, but fall short of supporting a wider range of security policies and/or resources. So far there are no application-level models that can provide efficient security management, and support a unified access control framework and a wide range of security policies in enterprise systems.

## 3.3 The Authorization-function-based RBAC Model

As discussed in section 3.1.1 and 3.2, the existing access control models are limited in their generality as they are purpose specific. In addition, they also fall short in several aspects such as authorization representations and the ability to support a variety of security policies. In this chapter, an authorization-function-based RBAC (FB-RBAC) model that addresses the problems specified in section 3.1.1 is proposed.

### 3.3.1 Motivation

One of the important problems in the existing models is that their permission representations are not suitable for describing application-level authorizations. In this chapter, a new form of permission – Authorization-function (A-function) to express and manage authorizations for enterprise functionality is proposed. The motivation for using the A-function and developing an A-function-based access control model arises from the fact that in an enterprise system, users interact with enterprise applications by executing a prescribed set of functions that control access to the back-end resources. These business functions may contain complex business logic, and are implemented by several simpler functions to carry out specific activities. At the end of function-call chains, resource-accessing functions are called to invoke various operations on resources.

For example, if an enterprise system needs to provide controlled access to a back-end database, then although ultimately users access the data repository at the back-end, they never do so directly, as this would imperil the consistency and integrity of the data. Thus, even though much of the research in database access control has focused on individual database objects in enterprise systems (e.g., attributes, tuples, and tables), users are never allowed to directly access a back-end data repository because of the risks to data integrity. In all cases, a set of A-functions is provided by the application to manage data access. This is also true for all other resource accesses that are conducted in the same manner, that is, through the use of functions. Hence, the problem of enabling different users to access and manipulate a variety of resources can be best described by whether users are allowed to execute functions of enterprise applications in a safe and controlled manner.

### 3.3.2 Authorization-function-based Access Control

#### *3.3.2.1 Formal Definition of Authorization-function*

In order to clearly describe authorization-function-based access control, it is necessary to provide a formal definition for the concept of Authorization-function. An Authorization-function (shown in Figure 3.3) is a function exposed by an application interface to the outside world for the purpose of executing a certain user request. It is the only channel through which an enterprise application/system interacts with users or other applications/systems.

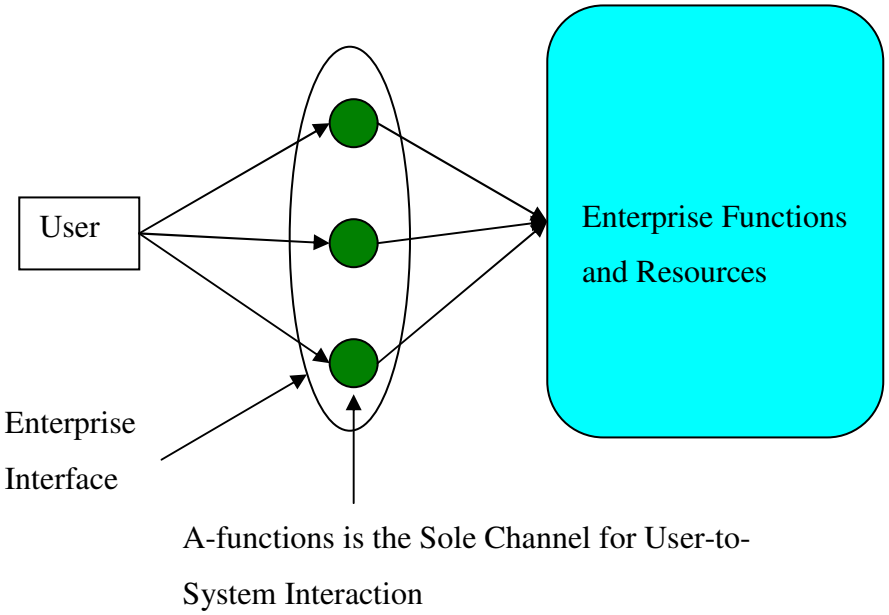



*Figure 3.3 A-function as the Sole Channel for User-to-System Interaction*

In our model, A-function is used as the basic access control unit and a permission representation. An A-function consists of two parts: an interface and an implementation. Its interface exposes functions to the outside world and the potential users, suggesting its functionality, availability and possibly access requirements. The implementation of A-function contains executions of user requests based on pre-defined business logic. The A-function can also be seen as the proxy to access all enterprise resources.

The authorization-function-based access control model maintains the role-permission paradigm from RBAC, but provides more expressive notations for permission instead of the operation-object pair. In the A-function-based approach, the abstract concept of permission is expressed as an approval of access to a particular A-function as shown in Figure 3.4.

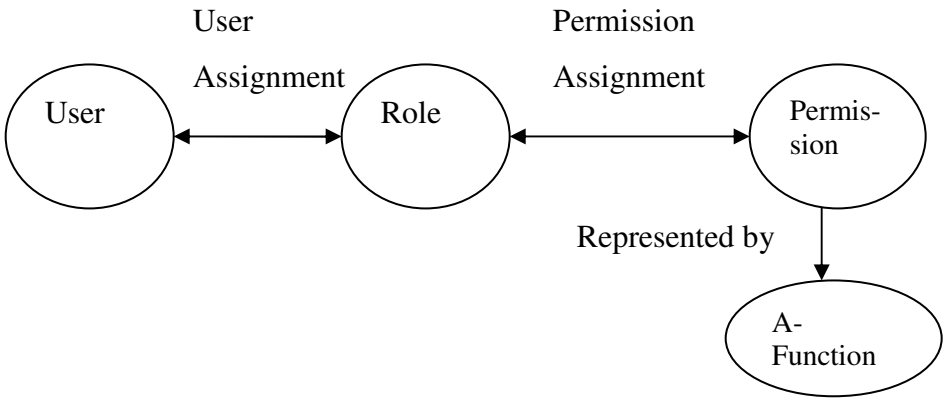



*Figure 3.4 Authorization-Function-Based Role-Based Access Control (FB-RBAC)*

This change of permission representation from operation-object to A-function has made RBAC more suitable for application-level access control, since authorization specifications can be simplified from subject-operation-object to role-A-function.

### *3.3.2.2 An Example of Authorization-function-based Application*

To better illustrate the concept of an A-function, an example is provided in Figure 3.5. In a typical enterprise application, a form delivered by a graphic user interface (GUI) provides users with a list of functions such as searching for customer details and placing an order. They are described by a set of interface objects like text field, buttons or menu items. Each of these functions can be represented by an A-function. By clicking a button or selecting a menu item, a corresponding A-function will be activated.

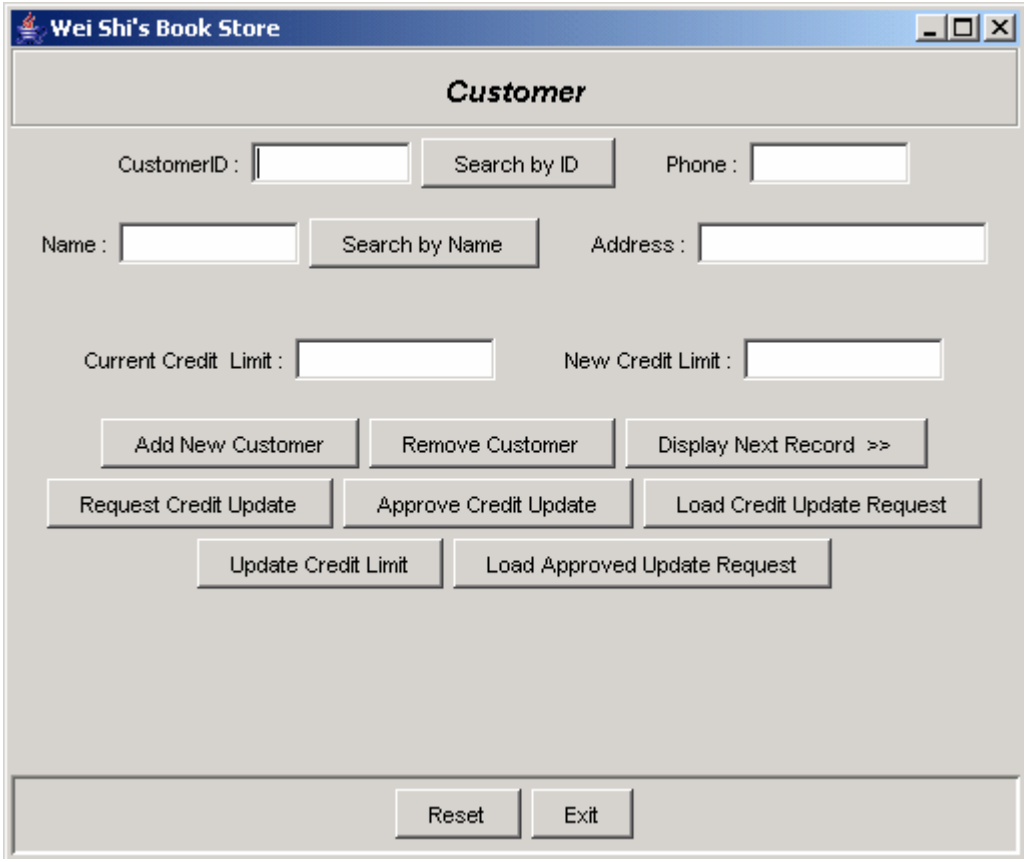



*Figure 3.5 An Example of an Application Interface*

An extract from the list of A-functions corresponding to Figure 3.5 is given below.

- Customer searchCustomerByID(int cust_id) {…..}
- Customer[] searchCustomerByName(String cust_name) {…..}
- int insertCustomer(Customer newCust) {….}
- int updateCreditLimit(Customer cust, int newCreditLimit) {…..}
- void updateCreditLimit(int customerID, int newCreditLimit) {…..}

For example, clicking the "Search by ID" button will activate the searchCustomerByID() function. There is a one-to-one relationship between A-function interface and A-function implementation (Figure 3.6).

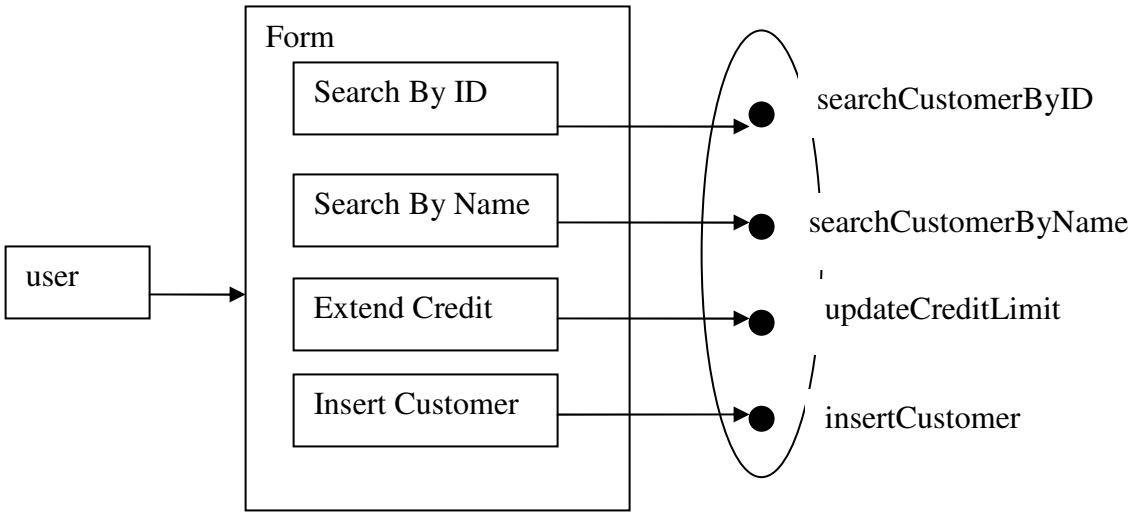



*Figure 3.6 Relationship between Interface and Implementation of an A-function.*

An application interface does not need to be graphical. In some Business-to-Business (B2B) interactions no human user is involved in the transaction as all processes are automated.

### *3.3.2.3 Application of the Authorization-function-based Access Control Model*

The proposed model aims to provide a unified access control framework that can be applied on top of low level access control systems. Figure 3.7 illustrates low-level (conventional) access control and application-level FB-RBAC in an enterprise system.

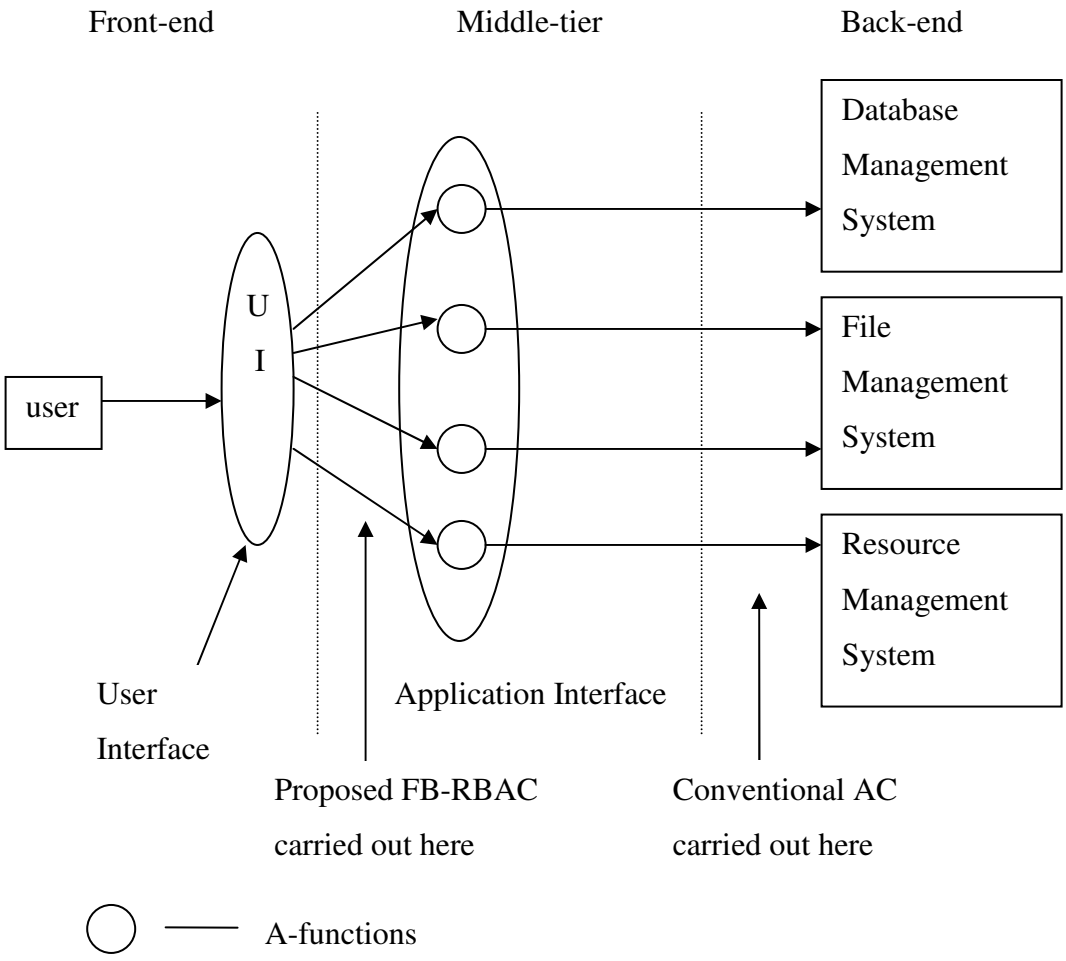



*Figure 3.7 Access Control at Different Levels of a Distributed Enterprise System.*

The proposed model is designed to provide top-level access control for enterprise systems. Underneath, individual security systems are used to protect one or more types of resources. These security systems make authorization decisions based on whether or not a subject can access a particular file or resource in a particular access mode. Hence, authorization decisions are made when systems are accessing specific objects like database items. Traditional access control

models designed for database and operating systems are suitable for these types of security systems, but they cannot provide support for application-level access control due to the reasons mentioned in section 3.3.1.

The real problem, however, for application-level access control is to control the execution of the functions. For example, an enterprise application provides users a function called "View Customer Record", then access control should be conducted to determine whether or not "View Customer Record" can be accessed. It does not need to verify each access to different back-end resources initiated by the "View Customer Record" function. This will be controlled by lower level access control systems such as the database security system or the file management system. Here, an access control decision is made when the user is accessing the interface A-function regardless of what resources are accessed by the A-function.

### 3.3.3 Constraint-based Access Control

One of the important requirements for enterprise access control is to support a wide range of security policies. In this section, we will demonstrate how various security policies can be described by constraint as introduced in RBAC96 [SAND96].

#### *3.3.3.1 Using Constraints to Specify Security Policies*

In a dynamic and distributed environment, access control decisions may be affected by a variety of conditions, such as the input and output of an A-function, time and location as well as history access record. Security policies may be described as constraints imposed on the above-mentioned conditions. Constraints can be used for laying out higher-level organizational policy

[SAND96]. However, it has not been fully utilized in the RBAC96 model (Figure 3.2). In reality, constraints can be imposed virtually on any concepts of RBAC, such as user/role assignment, permission/role assignment, user, role and session. In this section, we will investigate the applications of constraints in a variety of access control scenarios.

As discussed in section 2.2.1, access control conditions (are defined as constraints imposed on various objects) in database systems can be classified into four different types: data-dependent conditions, time-dependent conditions, context-dependent conditions, and History-dependent conditions. However, this classification is specifically designed for database security. Yet it is insufficient to describe the circumstances of contemporary enterprise systems. In order to provide finer-grained access control for user-to-system interactions, three important factors need to be considered: whether the input to the system is a valid input, whether the output returned by the system is verified, and whether the user has the permission to use the functionality provided by the system. In FB-RBAC, access control conditions are classified into three categories.

- A-function attribute conditions impose constraints on attributes of the A-function. For example, the input and output of the A-function can play important roles in user-to-system interaction. The input condition imposes constraints on the data fed into the A-function. The output condition imposes constraints on the information returned by the A-function. The input conditions and output conditions impose restrictions on the content of the input parameters and output parameters. This type of access control is also called content-based access control.

- Environmental conditions specify a variety of constraints related to the operating environment, such as time, location, and system usage. This category of constraints subsumes the third condition specified by [CAST94] (refer to section 2.2.1).
- History-dependent Conditions specify constraints dependent on previously performed access. These constraints remain unchanged from the fourth condition specified by Castano [CAST94] (refer to section 2.2.1).

A complete FB-RBAC is depicted in Figure 3.8.

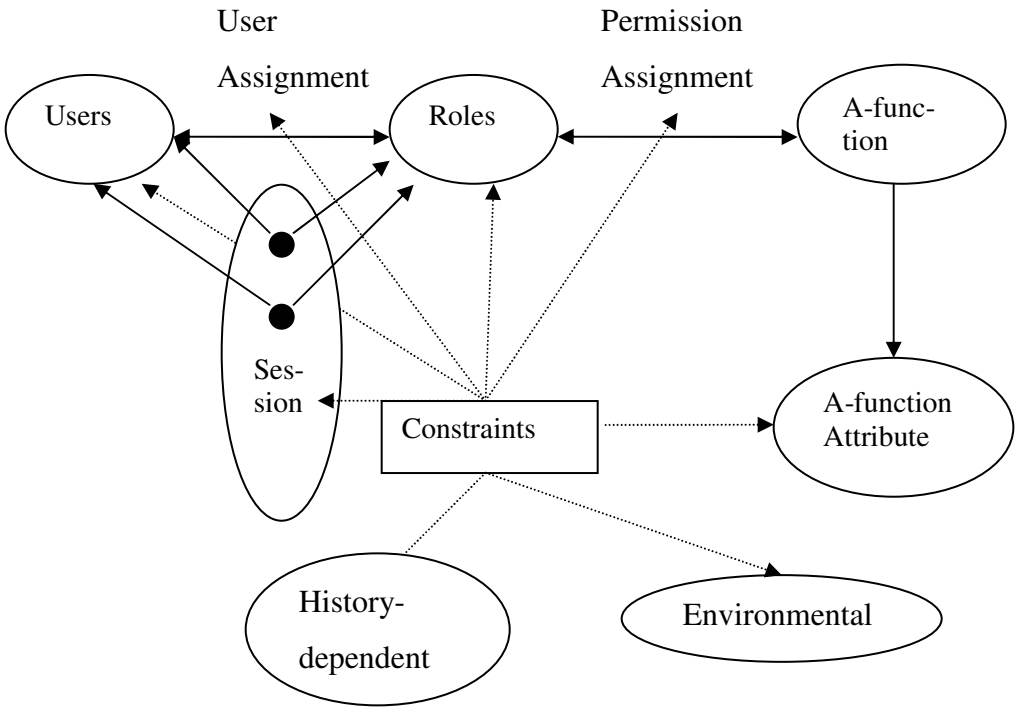



*Figure 3.8 A Complete FB-RBAC Model*

In authorization-based access control, input parameters and output parameters will be evaluated against predefined security policies. By imposing constraints on the input and output

parameters of an A-function, user-to-system interaction in enterprise systems can be controlled in a fine-grained manner. Together with environmental conditions and history-dependent conditions, the proposed model allows a variety of security policies to be specified for various access control scenarios.

#### *3.3.3.2 Policy Algebra for Constraint Specification in FB-RBAC*

This section proposes a context-free policy algebra to express constraints and security policies. This policy algebra is largely self-explanatory.

- Operator set: {AND, OR}, operators are used to join multiple conditions.
- AND: both conditions need to be satisfied
- OR: one of the conditions need to be satisfied
- Predicates Set: {>, <, ≤, ≥, ≠, ⊄, ⊂, ∈, ∉ }, predicates are used to evaluate a particular condition
- =: equal to
- ≠: not equal to
- > greater than
- < less than
- ≤ less than or equal to
- ≥ greater that or equal to
- ∈ : an element of
- ∉ : not an element of.
- ⊂ : a subset of
- ⊄: not a subset of

This policy algebra can be easily extended by adding new operators and predicates. In the next section, several examples are used to illustrate different types of constraints and how they can be expressed using the policy algebra.

Example 1: Input Conditions

Sometimes business rules will specify conditions in which input data can be accepted by an A-function. The evaluation of these rules depends on the value of the input data. The policy algebra can be used to specify and control the value of the data being passed to the A-function. For example, an A-function InsertCustomer() takes a list of user attributes as input and writes them to the database (Table 1). A customer role is allowed to access this function with some limitations. The inputs to the A-function can be limited to a certain value, such as user name, or a list of values, such as a group of user names. Input can be constrained, such as requiring Age < 25.

| Input attributes | Customer-ID | Customer-Name | Age | Address | Telephone |
|---|---|---|---|---|---|
| constraints | | = “Wei Shi” | >25 | Victoria | |

*Table 3.1 Input Constraints*

Constraint on authorization (Customer, InsertCustomer) is expressed as:

```
(CustomerName = “Wei Shi”
AND Age > 25
AND Address belongs to Victoria)
```

The above table indicates the constraint specified on input parameters. If any input value does not satisfy the given rule, then the A-function will not be executed.

Example 2: Output conditions

Constraints can also be imposed on output parameters of the A-function. This may affect the outcome of the A-function execution. The system can choose to block some or all the results produced by the A-function. The following table (Table 2) specifies the conditions that must be satisfied by the output parameter of the A-function searchOrderByID().

| Output Parameters | Customer-ID | Customer-Name | Date | Credit_limit | Purchase Amount |
|---|---|---|---|---|---|
| constraints | | $\notin$ VIP Customers | 1st Jan, 2003 -30th June, 2003 | < 10000 | < 10000 |

*Table 3.2 Output Constraints*

Constraint on authorization (Employee, searchOrderByID) is expressed as:

```
(CustomerName ∉ VIP Customers
AND 1st Jan, 2003 < Date < 30th June, 2003
AND CreditLimit < 10000
AND PurchaseAmount < 10000)
```

If any of the output values does not satisfy the given rule, an administrator will have two options: strict compliance or selective compliance. Strict compliance means no result will be

returned to the user, whereas selective compliance only hides the constrained parameters and users are still able to see the rest of the results.

Example 3: Environmental conditions

Environmental conditions are based on the surrounding environment, such as time, location or system load. For example, the access time has to be within a certain time frame, eg. 9am-5pm. The terminal is located in RMIT, or the number of the current users is less than 1000. These conditions can be expressed as following:

```
Time ⊂{>9am AND < 5pm}
Terminal Location = "RMIT"
Login users < 1000
```

Example 4: History-dependent Conditions

In order to realize history-dependent access control, it is necessary to maintain access records to keep track of activities carried out by users and of the time of invocation. Every time user invokes an A-function, an audit trail creates a tuple that records user name, role assigned, A-function performed, time of access, and possibly a Session ID as well. History-dependent access control can be used to implement well-known security policies like "separation of duty" and "Chinese Wall" [BREW89]. To implement a Chinese Wall policy, it is necessary to keep a record of user history actions. For example, if a group of companies are defined as having a Conflict Of Interest (COI), then a consultant is not allowed to read information from more than one company in any given COI group. The "Chinese Wall" policy in this case relies on the

history of the consultant to decide whether this consultant has already accessed information from companies in the same COI group.

### 3.3.4 Comparison with Existing Access Control models

Among existing models, a variety of expressions are used to represent permissions, such as operation-object [HARR76, BERT99, RABI91], task [THOM93, WANG04], action [WANG04], method [EVER00, EVER02] and service [BHAT04c]. But authorization-function is more suitable for application-level access control than any of these for the following reasons. Authorization-function is a simpler notion than operation-object used in many security models; easier to administer than object role; more meaningful than terms like task and action when describing access control and security policies in enterprise systems; and it also has the generality that method and service do not have.

Below a simple example is used to illustrate why the A-function-based approach is better than the traditional operation-object approach. For example, an A-function called searchCustomerByID() takes Customer ID as the input, and returns a list of customer information. Suppose customer information is stored in several data repositories, for example the customer residential address is stored in the Address database, Customer Credit information is stored in the Credit Detail database, and the Update Credit Policy in the Policy.txt file. Table 3 compares the authorization specifications of three different models for the scenario. The following assumptions are made for user/role assignment in RBAC and FB-RBAC. There are multiple users in the system, and some of them (Mary, John, … ) can be assigned to the role ‘employee’. This role has

permission to access the A-function searchCustomerByID(). A closed world policy is assumed in all three models, which forbids everything that is not explicitly allowed [ESSM99].

| Access Control Matrix (ACM) (S, A, O) | Role-based Access Control (RBAC) (R, A, O) | Authorization-function-based RBAC (FB-RBAC) (R, F) |
|---|---|---|
| (Mary, R, CustomerAddress) | (Employee, R, CustomerAddress) | (employee, searchCustomerByID) |
| (Mary, R, CreditDetail) | (Employee, R, CreditDetail) | |
| (Mary, R, UpdateCreditPolicy) | (Employee, R, UpdateCreditPolicy) | |
| (John, R, CustomerAddress) | | |
| (John, R, CreditDetail) | | |
| (John, R, UpdateCreditPolicy) | | |
| Authorization for more users | | |

*Table 3.3 Authorization Specification in Access Control Matrix in Role-Based Access Control and in Authorization-Function-Based RBAC Model*

As shown in Table 3.3, the traditional ACM specifies authorization as a tuple of three (S, A, O). It is required to define authorizations for each data manipulation. Therefore, authorization evaluation is also carried out three times during invocation of the A-function searchCustomerByID() for each user request. The authorization specification for RBAC is much simpler, it assigns users to roles, hence reducing the number of authorizations by N number of times where N is the number of users assigned to the role employee. With the new A-function-based RBAC model, all three data accesses used for the same task are grouped in the A-function

searchCustomerByID() at the time when user/role assignment is in place. Correspondingly, authorization specification in the new model is reduced to a single tuple (Employee, searchCustomerByID). The comparison clearly indicates that the FB-RBAC approach can greatly reduce the complexity of authorization specification and administration. Authorization evaluation at the application-level will be performed only once, when users are making requests.

Moreover, the FB-RBAC model considers a wide range of access control conditions that existing security models are unable to do. These constraints include those imposed on the result of executions and user requests. An A-function-based access policy suggests that the granularity of data access is at the level of the A-function, i.e. the user is either granted or denied authority to execute an A-function. It is also capable of providing fine granularity access control by controlling the parameter list and the values returned by the A-functions that are executed. The model controls the parameter list and values returned by the A-functions to comply with roles' access rights and the enterprise's business rules.

## 3.4 Discussion and Future Work

In section 3.1.1, 3.2.3 and 3.2.4, we showed that the existing models are not suitable for application-level access control due to the way permissions are represented in authorization specifications. These representations may be inefficient, or are designed for specific problems or systems. The proposed FB-RBAC provides application-level access control that complements the existing lower level access control mechanisms such as a database security system, secure file system and resource management systems. A new form of permission, authorization-function (A-

function), was used in conjunction with the notion of role for authorization specifications. The combination of role and A-function has effectively addressed subject heterogeneity and object heterogeneity in enterprise systems and provided an efficient way of specifying and administering authorizations. The concept of constraints in RBAC96 is used to support a variety of security policies. When constraints are imposed on the input and output parameters of an A-function, it will be able to carry out fine-grained access control based on the contents of input and output. The advantages of FB-RBAC can be summarized as below:

- The proposed model is independent of the types of systems and resources involved, hence it can be used to support a unified access control framework in distributed enterprise systems.
- The A-function is also the sole channel for accessing back-end resources, thus making it a suitable expression for describing permissions.
- The combination of role and A-function-based approaches can efficiently describe enterprise-level authorization and avoid the complications raised by heterogeneous access modes associated with different resources. Hence, the proposed application-level model is able to simplify the security specification, administration and evaluation.
- The basic elements of FB-RBAC model – the Authorization-functions are extracted directly from enterprise applications. They conform to various enterprise applications, so the model can be easily adapted to distributed enterprise applications/systems.
- FB-RBAC is a policy neutral model, so it is not restrained by a particular type of policy, such as both MAC and DAC.
- The model supports a wide range of security policies including fine-grained access control policies

- The implementation of an authorization-function may be modified from time to time as required by database updates, changes in enterprise business policy or the sources of the data, but authorization policy specified in the form of role-A-function will remain unchanged.

When the proposed model is implemented in an enterprise system, there may be some inconsistency between the authorization of the application-level system and the authorization of the low-level system of individual components. To resolve such conflicts, a top-down approach may be used if it is a centralised system so that lower level authorizations can be derived from or made conform to high-level authorization. Bottom-up approaches can be used where low-level security systems are autonomous, so that any high-level authorizations involving resources protected by low-level security system are derived from low-level authorizations. It is also possible to have two separately specified authorization sets where one could override the decision of another. Furthermore, the proposed model is also facing a problem common to all security models. That is to express authorization and security policies in an unambiguous form and for them to be able to be understood by various systems. The policy algebra proposed (section 3.3.3.2) provides limited syntax and semantic and is unable to provide uniform and unambiguous specifications for security policies. Moreover, the proposed model has not discussed how role assignment is conducted in a dynamic and distributed environment. Role assignment will be covered in chapter 4 in the context of Web service applications. In chapter 5, an ontology-based approach for specifying security policies and role assignment will be presented.

## 3.5 Summary and Conclusion

The large number and types of users and resources in an enterprise system pose several challenges to access control and security management. This chapter has reviewed some of the well-known security models with regard to authorization representations. It has found that the traditional access control models are not suitable for enterprise access control either because of administration complexity or lack of generality. There is no high-level access control model that addresses the needs of open distributed enterprise systems.

In this chapter, an application-level access control model -- FB-RBAC, was proposed for distributed enterprise systems. FB-RBAC was an extension of RBAC96 and it inherits all its advantages such as policy neutrality, and platform and programming language independence, so it can be implemented in a variety of distributed enterprise systems. Authorization-function is used as the basic unit of access control and to represent permission in authorization specifications. Authorizations are specified using roles and A-functions. The combination of role and A-function significantly reduces the complexity of security management, hence will increase access control efficiency. This chapter also discussed the application of A-function using a real life example. This example shows that A-functions are the only channels for users to access back-end resources, so they are most suitable for controlling user-to-system interactions. The proposed model may be implemented in the middle-tier, hence complements the lower level protection mechanisms to provide complete security protections for enterprise systems. Moreover, the FB-RBAC also extends the concept of constraint to describe various access control conditions and support a wide range of security policies. Constraints imposed on attributes of an A-function such as input and output provide finer-grained access control for enterprise systems. The

proposed model also provides a simple policy algebra in which different types of security policies can be formally expressed. At the end, a comparison between FB-RBAC and two well-known models (ACM and RBAC96) was provided. This comparison shows that FB-RBAC is more efficient than some of the most influential models. At the end the discussion section demonstrated that the generality and efficiency of the FB-RBAC makes it a promising access control model for enterprise-wide systems.

# Chapter 4

# An Extended RBAC Model for Web Service Applications

## 4.1 Introduction

Internet and web related technologies have advanced at tremendous speed in recent years, providing powerful functions for business processes. One group of the new services provided by the Internet and Web is Web services which are the programmatic interfaces used in application-to-application communications [WSAC]. Web services are developed for business integration and process automation, and have become the most popular applications in Business-to-Business (B2B) transactions due to their world wide applicability and remote accessibility. They use World Wide Web Consortium (W3C) standards and are supported by core XML technologies. The development of Web services has added new features and functionality to the traditional client/server architecture.

Web services operate in an open distributed, decentralized, dynamic and interoperable environment, and are designed to be highly autonomous and distributed in nature, therefore access control of Web service applications is a challenging issue. A variety of solutions have been developed to address specific security problems, such as Security Assertions Markup Language (SAML), X.509 public key certificate and the security part of Web Service Specification (WS* -Specification). SAML provides an XML framework for exchanging authentication and authorization information for securing Web services, and relies on third-party authorities for provision of "assertions". SAML is mainly used for authentication. X.509 public key certificates are currently used to protect SOAP-based messaging. For Web service applications, the security related part of WS*- specification, including WS-Security, WS-Trust, WS-SecureConversation, WS-SecurityPolicy, and WS-Federation, provides a range of security services such as message integrity, authentication and confidentiality, security token exchange, message session security, security policy expression, and security for a federation of services within a system. However, the above-mentioned security services are mainly used to protect Web services at the communication level. There are few application-level access control models for Web services.

### 4.1.1 Problem Statement

Providing efficient and reliable access control for dynamic and anonymous users is an essential requirement in the Web service environment. Users in the Web environment are highly dynamic as they enter and leave Web service applications continuously and their profiles change frequently over time. Users can be anonymous due to privacy concerns as they do not wish to

disclose their identity to strangers. As a result, the identities of many Web service users are unknown to the system at the time of the request. However, traditional models are mainly authentication based and require proof of identity or registration for effective centralized control. Such access control paradigm has many limitations due to spontaneity and privacy concerns, hence are not suitable for Web service applications [AGAR04a, AGAR04b]. Therefore, it is necessary to develop none-identity-based access control models in order to overcome such problems.

The second challenge of Web service access control comes from the heterogeneity of subjects and objects involved in security specification, administration and evaluation of a Web service application, as Web services interconnect numerous systems, applications, user groups and resources. However, administration of authorizations and security related information is still performed manually in many traditional security systems. Manual security management is not a feasible solution in Web service environments because of the large number of subjects and objects involved. Access control should be performed according to a set of predefined rules, and carried out automatically for Web service applications.

Providing support to a wide range of security policies is always important to a successful access control model. Security policies can be dynamic in a Web service environment. For example, security policies may refer to dynamic conditions such as user profiles, user attributes, environmental factors and various other conditions, and these security policies may be temporal and fine-grained in nature. On the other hand, traditional security policies like separation of duty (SoD), Chinese Wall and least privilege principle can still play important roles in Web service

access control. Hence, access control models designed for Web services must be able to describe a variety of security policies.

### 4.1.2 Outline of the Proposed Solution

In this chapter, we propose an Extended Role-Based Access Control (ERBAC) model to address the challenges presented by the Web environment (refer to 4.1.1). ERBAC extends the FB-RBAC proposed in chapter 3 to provide application-level access control for Web services applications. In the ERBAC model, access control is no longer dependent on user identities, but on provided user credentials and a variety of access control rules. It is assumed that each user credential contains authorizations required for user requested access, and user credentials are verified by a trusted third party. This credential-based paradigm provides a flexible and secure way of handling dynamic and anonymous users in the Web environment.

ERBAC improves FB-RBAC by introducing dynamic role assignment for access control in a Web environment. User credentials are assigned to roles automatically according to the user capability encapsulated in user credentials and a set of security policies. FB-RBAC is capable of simplifying security management of an enterprise system. As dynamic role assignment is integrated with FB-RBAC model, we can further reduce the complexity of heterogeneous subjects and objects involved in manual role assignment of the traditional RBAC models. The new role assignment approach also captures the continuously changing user profiles and environmental conditions. Hence, the extended model is more adaptable to the dynamic Web environment.

The ERBAC model provides support for a variety of security policies using constraint-based access control and content-based access control. These aspects are inherited from the FB-RBAC

model. Constraints can very well describe classic security policies like SoD and least privileged principle as well as access control requirements in a Web environment.

Furthermore, this chapter also provides formal specifications for each concept involved in the proposed model and relationships between concepts using a set of mathematical symbols and predicates. This allows security policies to be expressed in formalized notations.

### 4.1.3 Summary of Contributions

The main contribution is that the ERBAC provides an application-level access control for Web services on top of the security infrastructure that was designed to solve specific security problems. ERBAC complements the security services that are used for communication level access control, such as ensuring message confidentiality and authentication. ERBAC provides credential-based access control that resolves the problem of user dynamism and anonymity that cannot be resolved in identity-based access control models. The proposed model inherits all advantages of the FB-RBAC, thus it is capable of addressing intensified subject heterogeneity and object heterogeneity in Web environments. Moreover, dynamic role assignment is enabled in the model, so it is more adaptable to the changing environment. At the same time, a set of formal semantics provided in ERBAC can specify authorizations and security policies in a clear and unambiguous form. This formalism can be mapped to an XML-based specification language to provide interoperable security specification.

### 4.1.4 Structure of the Chapter

In section 4.2, we will first identify the access control requirements of Web services as presented by other researchers. Then, a variety of none-identity-based role assignments and role assignment rules are introduced and analysed. Their advantages and disadvantages in Web service applications are presented. Section 4.3 proposes an enhanced version of FB-RBAC -- ERBAC model. This model provides dynamic role assignment for Web service applications, and it supports a variety of security policies. In addition, formal specification is provided to represent concepts and relations used in the model. Section 4.4 discusses the contributions of ERBAC model and possible drawbacks. Section 4.5 provides a summary of the chapter and draws concluding remarks.

## 4.2 Background

Nowadays, an increasing number of service-providing enterprises make their services available via the Internet. User profiles and environmental conditions change frequently over time in a Web environment. This has raised new challenges to access control as well as user and resource administration [ALKA02]. Web services are protected by a variety of security services such as firewalls, encryption, access control mechanisms and network protocols. They can be classified into two main categories: access control services that protect Internet resources from unauthorized use, and communication security services that ensure confidentiality and integrity of data transmitted over the network [JOSH01]. This section will introduce the key issues and challenges of access control in open and distributed environments. We will also cover different

aspects of Web service access control including authentication, trust management, credential-based access control, access control automation and policy specification.

### 4.2.1 Access Control Requirements for Web Services

Researchers [AGAR04a, AGAR04b, BHAT03a, BHAT04b, INDR04, JOSH01, YAO01] have proposed a set of requirements for Web service access control models. We highlight some of the important aspects as follows.

- A generic and adaptive access control framework is required for Web service applications

A Web service access control framework must be generic so that it can be applied to various Web service applications. Implementation of the model should be easy and have minimal effect on the existing security mechanisms designed for specific problems.

- Provide support for heterogeneous subjects and objects in Web service environment.

Object heterogeneity refers to the diversity of system resources. Subject heterogeneity is due to the dynamic nature of web-service users. An Access control model must be able to support a large number and types of subjects and objects.

- Access control should not rely on user identities.

In Web environments, there are many new users and anonymous users. Hence, user identities are often unknown to the service provider, thus traditional authentication-based access control that requires the knowledge of user identity is not suitable for protecting Web services.

- Enable automatic access control decision making based on dynamic conditions.

Since Web-based enterprise resources are being exposed to a dynamic and distributed environment via Web services, it is necessary to have a dynamic and flexible access control model that captures changing security relevant information, such as user profiles and various environmental constraints at the time the access requests are made, and incorporate these into access control decisions.

- Access control policies should be autonomously specified by service providers.

Due to the distributed nature of the Web environment, security policies for Web services should be specified and administered autonomously by providers. In other words, service providers should be able to determine the access control policies of the provided Web services independently, without any centralized control.

- The framework should allow end users to check and prove their eligibility to use a Web service.

This will be useful if a user has not yet provided enough information for the request to be granted, then he/she should be given an opportunity to provide the missing information.

- Access control framework should support a range of access control policies.

A wide range of security related conditions may affect authorization decisions. In order to support a variety of access control requirements, the ability of specifying more specific and complex security policies is becoming an indispensable part of successful access control.

### 4.2.2 Access Control in Web Environments

Most of the existing access control models are designed for operating systems and databases. Their adaptability in Web environments has been under investigation. A good summary of existing access control frameworks was provided by Joshi et al. [JOSH01]. It compared a range of existing access control frameworks including, MAC, DAC, RBAC, WFMS (work-flow management system), TBAC (Task-based Access Control), agent-based approach and certificate-based approaches, in order to find ideal access control models for Web-based applications. RBAC was found to be an attractive candidate for Web-based applications because of several well-known advantages [JOSH01]. The RBAC model is known to be policy neutral, therefore it is capable of supporting a wide range of security policies including both MAC and DAC. Most importantly, the RBAC model can be easily managed and adapted by enterprise applications, since the role concept can reduce the complexity of subject heterogeneity and security management works [SAND96, JOSH01]. However, Web-based applications have two salient features: they have a large number of users, and many users are previously unknown to the system. Researchers have discovered that in such environments the traditional manually conducted identity-based user-to-role assignment is infeasible [ALKA02, AGAR04a]. The issues that face RBAC are assigning roles to previously unknown users, and automating role assignment based on a wide range of conditions apart from user identity.

### 4.2.3 Credential-based Access Control and Dynamic Role Assignment

Some alternative role assignment approaches have been proposed recently to address the challenges in multi-domain environments. An appealing solution is to assign users to roles based

on credentials provided by users. A credential provides security critical information in order to establish connections between different security entities. Traditionally, a credential consists of user identity for authentication, and user privileges for authorization. To ensure the authenticity of a credential, it may be digitally signed by a trusted third party. By using a trust management approach [BLAZ00], the reliance on user identity can be resolved. A trust management framework can bind digitally signed certificates with user privileges directly instead of user identities [BLAZ00]. Such a framework is supported by Public-Key Infrastructure (PKI) to ensure the authenticity of a credential. User identities are then replaced by public keys. This allows role assignment to be conducted according to the content of user credentials. The trust management framework by itself can only provide authentication, it cannot perform authorization since it is not designed to control user-to-system interactions. Nevertheless, it can provide security relevant information for the subsequent authorization process.

Herzberg et al. utilize trust management approach to support access control in a proposed Trust Establishment (TE) framework [HERZ00]. This framework automatically assigns business roles to strangers according to certificates provided by a trusted third party. Each certificate is expressed as a predicate of the form *cert (Issuer, Subject, Type, Field)* indicating issuer of the certificate, the name of the subject, the type of the certificate and the attributes field of the certificate. The TE system is built on top of PKI. Later on, an extension of TE framework [ZHON01] was developed to include trustworthiness of the user and certificate issuer in role assignment. The trustworthiness of the user is accumulated gradually over times and drops if the system believes that the user has done harmful actions or attempted harmful operations. The trustworthiness of the certificate issuer is classified by the security administrator according to a

set of evaluation rules. The authors argue that these trust-based factors should be used to determine which role is assigned to users. Although this approach can simulate the real world assessment of trustworthiness, it also can lead to major security frauds in the system, since a malicious user may commit malicious acts after his/her trustworthiness has reached a certain level.

Both the Trust Establishment (TE) framework [HERZ00] and its extension [ZHON01] rely on PKI infrastructure. The problem is that the implementation of PKI infrastructure may affect the performance of the system, as the public-key operations involved are computationally expensive. Furthermore, a PKI-based system can become the target of Denial of Service (DOS) attacks and be forced to repeatedly perform computationally expensive public-key operations needed for certain authentication or key exchange protocols. Additionally, both the TE framework and its extension focus on role assignment based on the trust level of the user or certificate issuer omitting other types of role assignment conditions.

A wider range of role activation[1] rules is identified in OASIS (Open Architecture for Secure Inter-working Services) role-based access control framework [YAO01]. In this framework, role activation rules are classified into three categories: prerequisite roles, appointments, and constraints.

- A prerequisite role must be activated before a target role can be activated.
- Appointment occurs when a member of some roles grants a credential that will allow some users to activate another role.
- Constraints can be used to define security policies such as SoD.

[1]Role activation only refers to role assignment performed by the system in a session; whereas role assignment refers to both role assignment performed by a human administrator and role assignment automatically performed by the system in a session. When no administrative role assignment is performed, role assignment and role activation mean the same thing.

In addition, roles can be activated and deactivated dynamically according to the conditions associated with role membership. A first-order predicate calculus is developed for the specification of the role activation rules. However, among all role activation rules, the paper focuses on the discussion of appointment, and no detailed analysis on other types of activation rules is given.

Rule-based RBAC (RB-RBAC) [ALKA02] also provides dynamic role assignment and revocation according to a set of rules. Unlike OASIS RBAC [YAO01], it focuses on the importance of user attributes in the role assignment process. The authors argue that user attributes can provide classifiable information for role assignment, thus they should be used as determining factors in role assignment. A context-free grammar is used to express assignment rules along with the model. Although both approaches suggest that a wide range of role assignment rules can be employed in role assignment, neither has provided a comprehensive analysis on the full ranges of role assignment rules.

In the above extensions of RBAC, role assignments are no longer dependent on user identities but on a range of security relevant information like user attributes, enterprise policy and various constraints. Compared with traditional role assignment approach, they are more flexible and adaptable in distributed environments. They have provided non-identity-based access control for enterprise systems. User credential and user attributes are used in role assignment to simplify security administration and to automate the authorization process in enterprise environments. However, since OASIS RBAC [YAO01] and RB-RBAC [ALKA02] are designed for general enterprise systems, they have not considered the unique requirements of Web service access control.

A generic XML-based RBAC framework, X-GTRBAC (XML-based Generalized Temporal Role-based Access Control) [BHAT05], has recently been developed for multi-domain environments. An XML-based specification language is used to represent security policies in order to achieve interoperability among different domains. X-GTRBAC follows a series of earlier results [BHAT03a, BHAT03b, BHAT04a, BHAT04b, BERT01c, BERT05] in the area of RBAC (particularly role assignment) and XML-based specification language.

The first work in the series is X-RBAC -- an XML based RBAC policy specification framework [BHAT03a]. This framework aims to address two key challenges in Web service security: (i) to develop access control models that provide content-based and context-aware access; and (ii) to handle the heterogeneity of subjects and objects involved. In order to achieve that, an XML-based specification language is used to define role, user, permission, user-to-role assignment and permission-to-role assignment as well as user credentials. Roles are dynamically activated for users based on their credentials supplied at the time of login, and content and context of security relevant information such as time, location and environmental state. This allows role assignment for unknown users as long as they can supply the required security information and comply with a set of role activation rules.

Based on a previous work [BHAT03a], a trust enhanced version of XML-based RBAC (X-RBAC) framework was proposed [BHAT04b]. This extended X-RBAC incorporates context-based access control with PKI infrastructure. To ensure authenticity of the context information, this framework uses trusted third parties (such as any PKI Certificate Authority) to provide trust management (TM). The trust enhanced X-RBAC provides access control for the services requested by users. “A service is an abstraction of the operations provided by system on its

resources" [BHAT03a]. A service request is defined as a triple (role, service, context); authorization is defined as a (role, service) pair. Each context parameter is given as a structure (format) definition. Algorithms are provided for access control decision making, expression evaluation and condition checking.

The trust enhanced X-RBAC pays particular attention to the context of the security information. The context parameters are formalized and are used in role assignment. The XML-based authorization framework allows easy application of the existing XML-based architecture to Web service security. Hence, the X-RBAC model provides a flexible and adaptive access control mechanism for Web services. However, neither of the papers [BHAT03a, BHAT04b] discussed the use of environmental conditions such as time, location and environmental state in role assignment.

Temporal constraints can play an important role in dynamic role assignment. It was introduced in the Temporal Role-based Access Control (TRBAC) model [BERT01c] that provided dynamic role assignment based on temporal constraints like time periods, and periodic role enabling and disabling through role triggers. This ensured that permissions assigned to the roles were only exercised within an appropriate time frame. TRBAC addressed temporal constraints for role activation only, and a more general model Generalized Temporal RBAC (GTRBAC) [BERT05] has been proposed for a wide range of temporal constraints. These temporal constraints can be applied to roles, user-role-assignment, role-permission assignment as well as role activation. In general, both models allow more flexible security policies to be specified, thus providing fine-grained dynamic access control for enterprise systems. However, these two models have not addressed other types of authorization constraints besides duration

constraints. Moreover, both TRBAC and GTRBAC rely on user identities for authentication, they do not support authentication and role assignment through user credentials provided by trusted third parties.

X-RBAC has been further extended to include temporal constraints to form the initial version of X-GTRBAC [BHAT03b]. It also provides an interoperable, context-free XML-based grammar – X-Grammar to specify all concepts in GTRBAC. A comprehensive version of X-GTRBAC [BHAT05] includes credential-based access control, context aware access control, temporal constraints and XML-based policy specification. The series of research conducted by Bhatti et al. [BHAT03a, BHAT03b, BHAT04b, BHAT05] covered many aspects of role activation conditions in a Web environment, and provided an XML-based specification language. The XML-based web-service policy specification has enabled information sharing between different domains that were previously unachievable due to incompatible information formats.

### 4.2.4 Summary of Existing Work

Applying security models to Web services is a great challenge (refer to section 4.1). In order to resolve the problem of subject and object heterogeneity in Web environments, and provide an expressive and generic access control model, RBAC has been applied to Web service applications. RBAC model has several well-known advantages: it is policy neutral, and it can significantly simplify the administration and specification of security policies (refers to section 3.2, section 4.2). However, many researchers have suggested that the traditional manually conducted identity-based role assignment is difficult in an open distributed environment.

Hence, in some extensions of RBAC such as OASIS RBAC [YAO01], RB-RBAC [ALKA02], X-RBAC [BHAT03a] and X-GTRBAC [BHAT05], role assignment is automated according to user credentials and a wider range of role activation rules. In these models, user credentials do not contain user identity any more, instead, security critical information such as user attributes, user capability and environmental conditions at the time of request are included. To ensure authenticity, user credentials may be supported by a trusted third party using a PKI-based framework for trust management [BLAZ00, HERZ00, ZHON01]. This approach enabled the use of identity in Web environments. Role activation rules are diversified: they may specify constraints on user attributes [ALKA02], trustworthiness of users or credential issuers [HERZ00, ZHON01], and environmental conditions (e.g. time, duration, location) [BHAT04b], or it can impose security policies (e.g. SoD, prerequisite role, cardinality) on roles and resources [YANG03, LIU04]. Since security information, like user attributes and environmental conditions, change frequently over time in Web environments, role assignment also becomes highly dynamic and adaptable to the changing environments.

## 4.3 Authorization-function-based RBAC for Web Service Applications

As we have discussed in the background section, the existing access control models are inadequate for addressing the unique security challenges in Web-services applications. In this section, we propose an Extended Role-based Access Control (ERBAC) framework for Web Services. ERBAC is an extension of the authorization-function-based role-based access control

model proposed in chapter 3, but includes new features like dynamic role assignment. In addition, the proposed model will be presented in formal notations.

### 4.3.1 Motivation

In Web service environments, the problem of subject heterogeneity and object heterogeneity is more severe than in conventional enterprise environments. The notion of user is generalized. It can be anything from a human user to a process, an application or a Web service. In fact, computer-to-computer interaction has become an increasingly popular approach as well as traditional human-to-computer interaction. User identity cannot provide an accurate measure of the privileges of the users. In addition, many of these users are new to service providers. This has posed a great challenge to security management in Web service applications. Traditional authentication-based access control systems that require manual security administration would be inadequate in such situations due to the large number of subjects and objects involved.

The job of security administration must be automated in order to improve operation efficiency. The FB-RBAC model proposed in chapter 3 is an ideal model for Web service applications. It not only inherited all the advantages of RBAC96 but also simplified authorization specification using roles and authorization-functions. This has relieved the complexity raised by subject heterogeneity and object heterogeneity. In addition, FB-RBAC utilized the concept of constraint to support fine-grained content-based access control. However, both RBAC96 and FB-RBAC are authentication-based access control models. In authentication-based access control, role assignment is manually performed by administrators according to user identities. As we discussed in section 4.1.1 and 4.2.3, this type of role assignment is not suitable in the Web

environment, a credential-based approach was proposed in some of the recent models [HERZ00, YAO01].

Credential-based role assignment has several advantages over the traditional identity-based approach. First of all, users do not need to register with the provider in order to use the service it provides. Secondly, user-to-role assignment is performed according to the security information provided in the credential, and no manual role assignment is required. Therefore, if credential-based access control is integrated with the FB-RBAC model, we can further reduce the administration work, and most importantly we will no longer rely on user identities for access control. Moreover, credential-based approaches also play an important role in access control automation for a Web service application.

### 4.3.2 A Credential-based Approach towards Web Service Access Control

#### *4.3.2.1 Access Control for Web Service Applications*

This section illustrates the structure of a Web service application, and indicates where application-level access control is applicable (Figure 4.1). A Web service application can be divided into front-end, middle-tier or back-end according to the classic three-tier architecture. The front-end includes user and user interface, the middle-tier contains a list of Web services that are connected to back-end system resources through a set of functions, and the back-end usually consists of different types of data repositories. Therefore, in order to provide controlled access to back-end system resources, it is necessary to provide access control mechanisms at the middle-tier. A credential-based FB-RBAC (is thereafter referred as extended RBAC) will be used to support application-level access control mechanisms in a Web service application. Hence, the

problem of enabling different user communities to access and manipulate back-end system resources is now reduced to a simple problem, i.e. whether user credentials have enough capability to execute the A-functions of Web service applications in a safe and controlled manner.

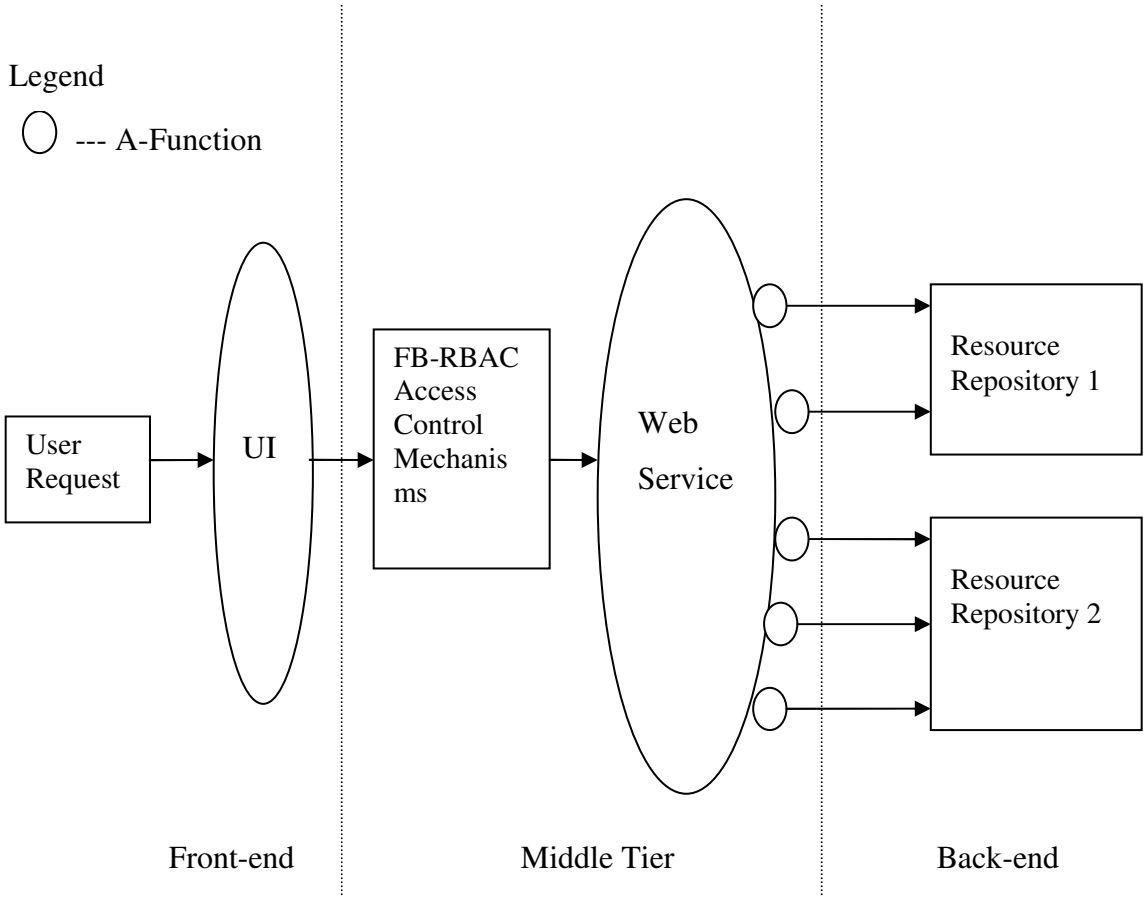



*Figure 4.1 Access Control Mechanisms in a Web Service Application*

#### *4.3.2.2 Important Features of the Extended RBAC Model*

In this section, we will outline the important features of the ERBAC model, some of which are inherited from the FB-RBAC model. These inherited features are closely related to the new features presented in the ERBAC model.

- Authorization-function-based (A-function) access control – a Web service can be described by a group of A-functions. Therefore, A-function is still the basic unit of access control in ERBAC, and it is also used to represent permissions in authorization specifications.
- Constraint-based access control – constraints form the basic components of access control policies. Security policies such as Separation of Duty (SoD) can also be described by constraints. They are heterogeneous: time, location, number and temporal conditions all can be used as constraints. On the other hand, constraints can be applied to a variety of concepts such as role, A-function, attributes of the A-function and user capability.
- Content-based access control – content-based access control can be specified in many contexts. We focus on the content of the input and output parameters of an authorization-function. The contents of input parameters is provided upon user requests, and it determines what resources are accessed at the back-end. The content of the output parameters may reveal confidential information to the user. Therefore, in order to provide controlled access to back-end resources, the contents of the input and output parameters should be included in access control decision making. In addition, by controlling the contents of input and output parameters of an A-function, fine-grained access control can be realized for user-to-system interactions.
- Fine-grained access control – a Web service can be described by a number of A-functions, and an A-function has a number of parameters. However, a user may not have access to all A-

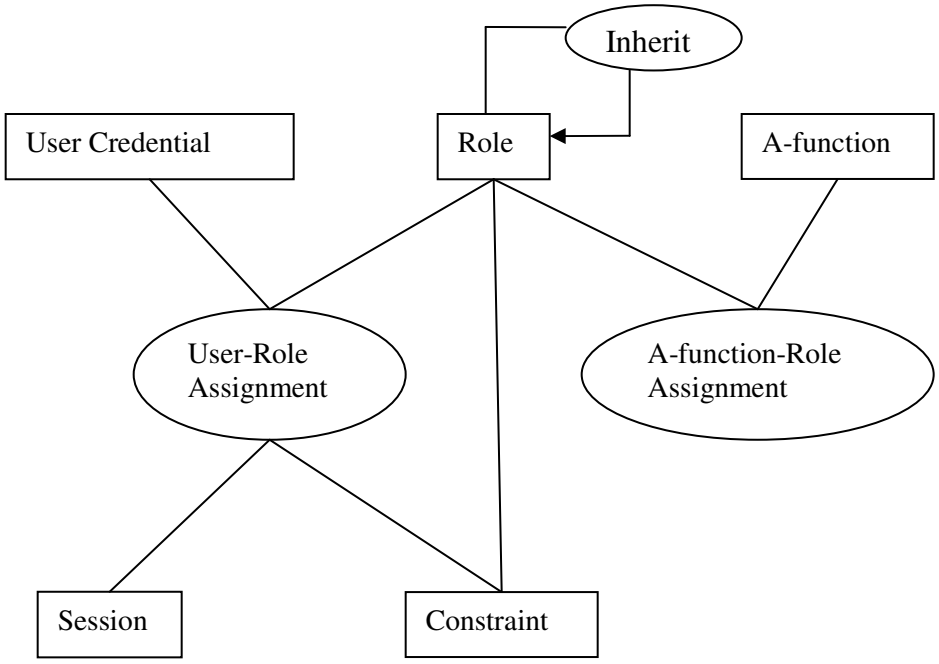
functions of a Web service or all parameters of an A-function. Partial access can be provided by introducing Web service instances and A-function instances. The granularity of access control will not be limited to complete Web services or A-functions, but instances of Web services and instances of A-functions. Together with content-based access control, Web service access control can be performed in a fine-grained manner.

- Credential based access control – a possible alternative to authentication based access control is to use credentials provided by a trusted third party. It can enforce access control without knowing the user's identity. In Web service environments, the user is represented by user credentials. In our proposed model, a user credential must contain user capabilities for the target Web service. In order to access system resources, user capability will be used to activate different instances of the Web service. As a result, different users will observe different instances of the same Web service in different sessions.
- Least Privilege Principle – many access control models do not enforce the least privileged principle. The principle of least privilege states that users should only be granted privileges when they have justifiable needs. The proposed ERBAC model will strictly enforce the least privilege principle in role assignment, so that only the least privileged role is assigned to the user request.
- Dynamic role assignment – dynamic role assignment is dependent on all the above-mentioned features. In Web environments, access control decision making ought to be automated according to a variety of dynamic conditions. These conditions include user credentials, the contents of the input and output parameters of the A-function and a variety of constraints (least privilege principle can be seen as a type of constraint). Dynamic role assignment will save manual administrative work of specifying authorizations for each security subject against each security object, hence making security management simpler and more efficient.

Out of all these features, the A-function-based access control, content-based access control and constraint-based access control are inherited from the FB-RBAC model, and fine-grained access control is the aggregating result of the three. These features will also play important roles in access control of Web services. Credential-based access control, enforcement of least privilege principle and dynamic role assignment are the new features added to the proposed ERBAC model. They will be discussed in detail in this chapter.

#### *4.3.2.3 Credential-based Access Control for Web Service Applications*

In RBAC models, role assignment is an essential part of the access control process, since its outcome directly affects whether or not a request is granted. Security policies and enterprise business rules are also enforced through role assignment. When no user identities are available for authentication and authorization, we will assume that a digitally signed credential [BLAZ00, HERZ00] is provided, which includes user capability towards the target Web service. In such circumstances, a user request will be granted if the user capabilities provided can be mapped into a privileged role -- this role has permission to access the requested Web services/A-functions. On top of that, all other role assignment rules should be satisfied. In the FB-RBAC model, role assignment is based on a wide range of rules, such as the input and output of user requests, constraints imposed on role and A-functions, environmental conditions and well-known security policies. It is essential that the dynamic environmental conditions are taken into consideration during access control decision making in Web service applications. This allows access control to be more flexible and adaptable to the environment. When the FB-RBAC model is applied to the Web environments, the notion of user is replaced by user credentials (see Figure 4.2).




[1] User credential has replaced the notion of user in traditional RBAC

[2] The constraints imposed on role assignment will be used to specify role assignment rules (section 4.3.3.1).

*Figure 4.2 Credential-Based FB-RBAC Model for Web Service Applications*

## 4.3.3 Dynamic Role Assignment in Web Environments

### *4.3.3.1 Terms and Concepts Involved in Role Assignment*

This section introduces the concepts and terms that are used to define role assignment.

- Web Service

A *Web service* application consists of a set of Web services. Each Web service is designed to perform a group of related tasks. For instance, a typical Web service would be "Order Book

Online". Each Web service consists of a set of A-functions, such as searchBook(), searchCustomer() and submitOrder() to carry out designated tasks.

- Authorization-function

Each Authorization-function (A-function) performs one or more operations on back-end resources. It contains a set of input parameters and a set of output parameters. For example, A-function searchBook() takes Book Name as input and returns Book ID, Book Price and Stock as outputs. In our extended RBAC model, the basic unit of access control is set to A-function instead of individual resources at the back-end.

- Web Service Instance

A Web service instance contains a subset of the A-functions that are assigned to the Web service. It is possible that different instances of the Web service differ not only in A-functions provided but also in the parameters of the A-functions. We therefore use the term A-function instance to represent A-functions that differ in their parameter sets.

- Authorization-function Instance

The instances of an A-function may only have a subset of the parameters of the A-function. For example, in the case of Web service "Order Book Online", A-function searchCustomer() takes Customer Name as the input, and returns Customer Address, Telephone number and Credit Limit. However, a fine-grained access control policy may specify that the user cannot view the customer's Credit Limit. This can be achieved by controlling the output parameters of the A-

function. Therefore, a particular instance of the Web service "Order Book on the Internet" can be described with a set of attributes including the A-function it provides and the parameters of the A-function. These descriptions will allow the system to carry out fine-grained access control based on the attributes of the A-function.

- User Credential (or User Capability)

User credential can be used to represent the service requesting agent. It contains user capability and some other security related information. In this thesis, we will only consider user capability since it is the determining factor in role assignment. It provides a set of permissions to access Web services and the A-functions in each of the Web services.

- Role

A set of Web service instances specified by A-functions and A-function parameters is assigned to each role. Inheritance relationships may exist between different roles, such as a senior role inherits all privileges of a junior role.

- Constraint

Constraints are conditions that must be satisfied in authorizations. They can be imposed on various concepts of RBAC model, and are used either as preconditions or obligations. Preconditions are conditions that must be satisfied before any action is taken, such as granting access to a user. Obligations are rules that must be followed by an action, such as constraining the way a request is executed.

#### *4.3.3.2 Role Assignment in Web Service Applications*

In the ERBAC model, role assignment will be carried out twice to ensure the least privileged principle is strictly followed (refer to Figure 4.3).

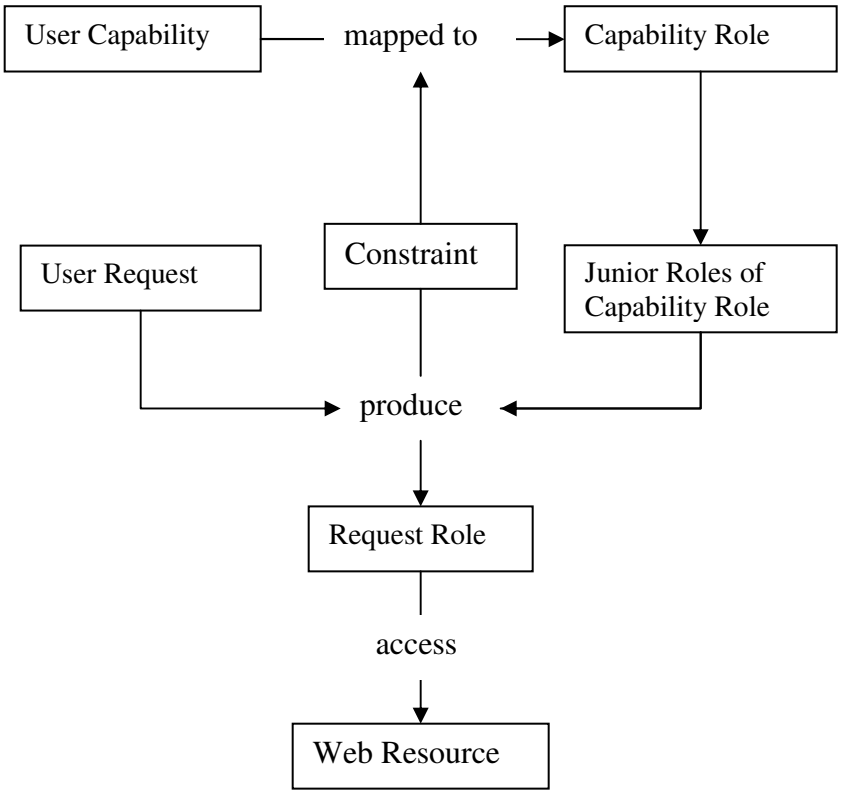



*Figure 4.3 Role Assignments in Web Service Applications*

The first role assignment provides a role according to user capability before any requests are made. It is designed to identify the capability of the user. The role assigned to the user capability will then indicate the maximum privilege that the user will have in this session. This role is called "capability role". However, not all privileges provided in the user capability need to be exercised for each user request. The second role assignment is conducted at the time of the

request. It is mainly based on the content of the request hence is called "request role". The role assignment engine will assign a less privileged role (compared with the role assigned to the user capability) to each user request. This ensures that privileges are not misused. Both role assignments are constrained by a set of role assignment rules. In the following section, we take an algorithmic approach to describe the role assignment process in detail.

#### *4.3.3.3 Assigning Roles to User Capabilities*

In the initial role assignment (shown in Figure 4.4), matchmaking is performed between the user capabilities and the privileges assigned to the role.

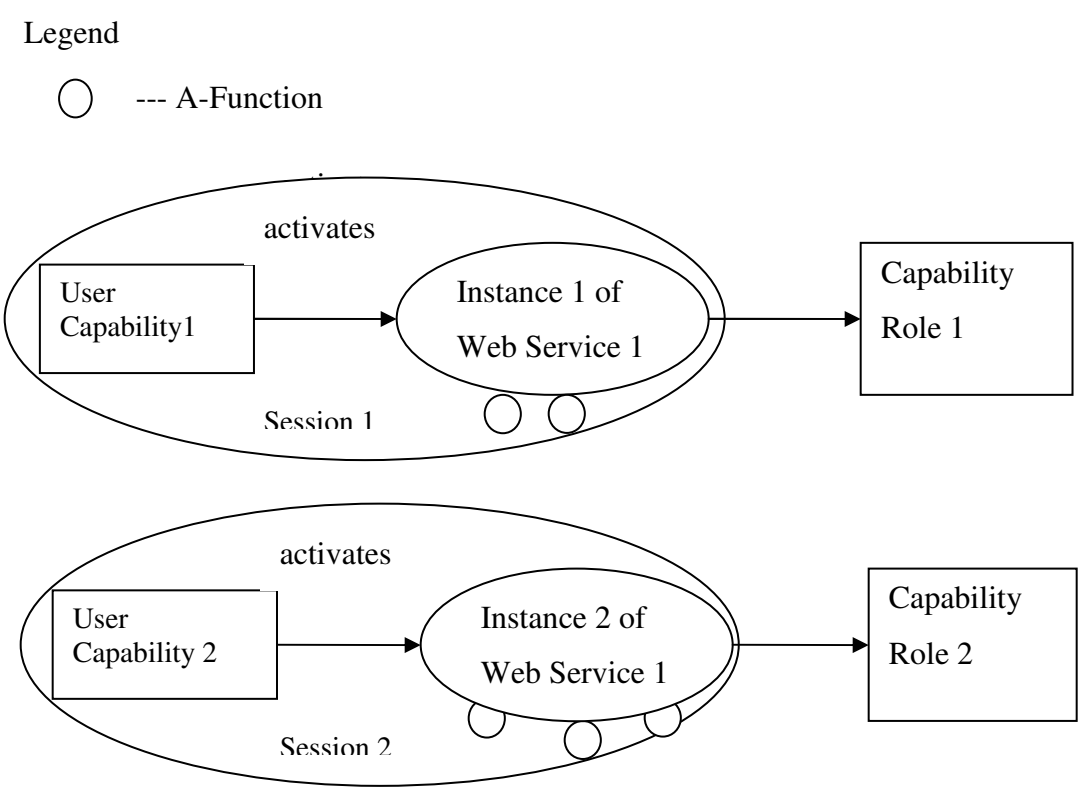



*Figure 4.4 Role Assignment for User Capability*

Both user capability and role privileges are specified as permissions to access Web service instances and/or A-function instances. The rule of this initial role assignment can be specified as follows. If the user capability subsumes the privileges of a role, then this role may be assigned to a particular user capability, and it becomes a candidate of the capability role. While there may be more than one candidate role, the process attempts to find the most privileged role among all candidate roles. The most privileged role is calculated by comparing the weight of the role. The total weight of the role is the sum of the weight of A-functions assigned to that role. Each A-function is assigned a weight according to its importance in the applications. Usually A-functions that write to the back-end database are assigned more weight than A-functions that simply read data from the database. The resultant role will be the role that has the most weight assigned to it. It is assumed that a role hierarchy is established among different roles. The full permission of a role includes all permissions inherited from its junior roles. The following algorithm generates a capability role for a particular user capability.

Algorithm 1 – Generating a Capability Role

```
Input: user capability, a set of predefined roles (role set_p);
Output: the capability role;
{
      add all permissions specified by the user capability be permission
      set_uc;
      for each role (role_k) in role set_p
      {
            add all permissions of role_k to permission set_r;
            if permission set_uc subsumes permission set_r and all
            constraint checks are satisfied,
            then add role_k to the candidate role set (role set_c);
```

```
        }
        let the most privileged role (role_p) to be role_1 in role set_C;
        for each role (role_i) in role set_C
        {
              if the weight of role_i is greater than the weight of role_p,
              then let role_p to be role_i;
        }
        return role_p (the capability role);
}
```

We call the resultant role generated from the initial role assignment "capability role", since it is mapped to user capability. Nevertheless, the capability role is also constrained by a set of rules (discussed in section 4.3.3) when exercising its privilege such as SoD and cardinality. These constraints determine whether a role will be assigned to the user and which one of these roles will be assigned to the user.

#### *4.3.3.4 Assigning Roles to User Requests*

The subsequent role assignment (shown in Figure 4.5) is conducted at the time of request. It is mainly based on the user request and the capability role. In order to strictly enforce the least privilege principle, a least privileged role will be assigned to each user request. We call this role a "request role". This role is selected from the junior roles of the capability role. It grants users with the permissions to access the requested A-function. Since the capability role has identified the maximum privilege of the user, all request roles possess only a subset of the privileges owned by the capability role. This second role assignment is also constrained by a variety of security rules (more details will be presented in section 4.3.4). The generation of the request role strictly

follows the least privilege principle. This ensures no excess privileges are granted to each user request. Figure 4.5 shows how a request role is generated.

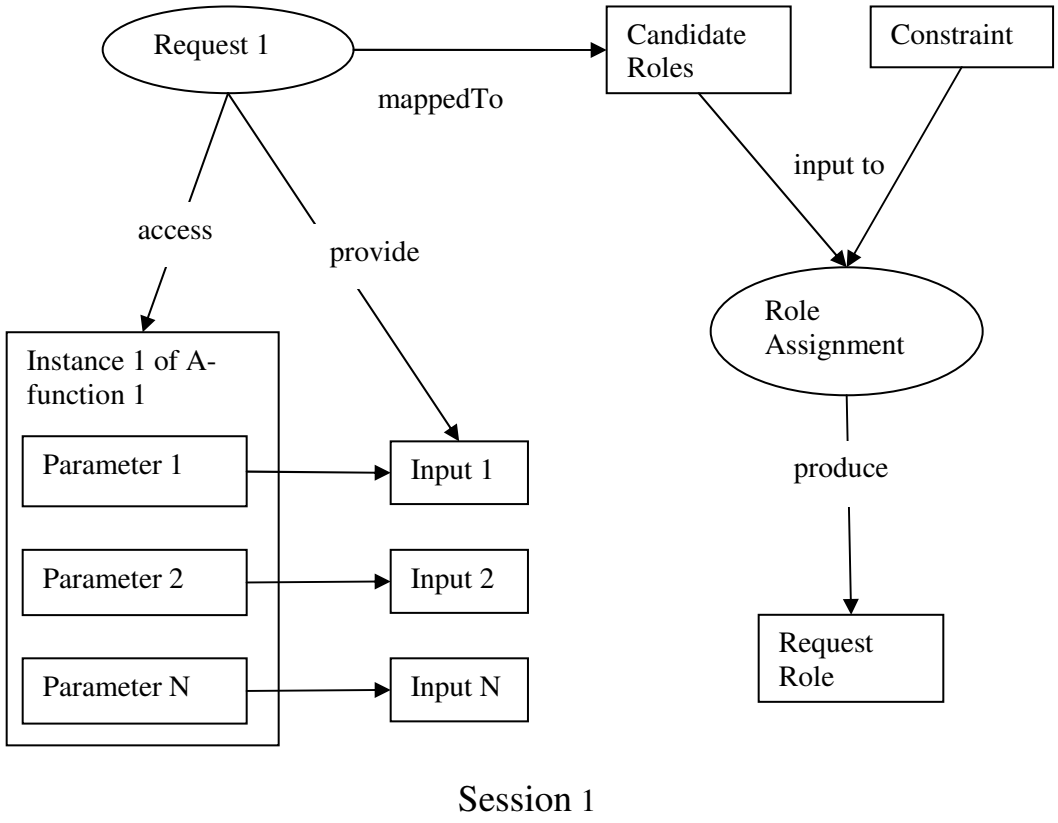



*Figure 4.5 Role Assignment for User Request*

The generation of the request role can be described in two basic steps. The first one is to find a set of candidate roles that satisfy the request. The second one is to apply constraints such as the least privileged principle to candidate roles to find the most appropriate roles. The whole process can be described in Algorithm 2 below.

Algorithm 2 – Generating of a Request Role

```
Input:  User request (to a Web service/A-function), capability role
(role_p);
Output: the request role;
{
      add all junior roles of the capability role (role_p) to role set_j;
      for each role (role_i) in role set_j
      {
            add all privileges of role_i to privilege set_i;
            for each privilege (privilege_k) in privilege set_i
            {
                  if privilege_k is equivalent to the user request;
                  then add role_i to role set_r;
            }
      }
      let the least privileged role (role_s) be role_1 in role set_r;
      for each role (role_m) in role set_r
      {
            if the weight of role_m is less than the weight of role_p;
            then let role_s to be role_m;
      }
      return role_s (the request role);
}
```

### 4.3.4 Constraint-based Access Control

#### *4.3.4.1 Using Constraints to Support Access Control of Web Service Applications*

In the previous chapter (section 3.3.2), we discussed how various constraints were used to provide access control for generic enterprise applications. Constraint-based access control describes a variety of security policies by imposing constraints on different components of the

RBAC model and on a range of environmental factors. For example, SoD and Chinese wall can be implemented by imposing constraints on user-to-role assignment, or on permission-to-role assignment. In this section, constraints are used to provide fine-grained access control for Web service applications. The existing RBAC models [ALKA02, BHAT03a, BHAT05, YAO01] have identified a variety of role assignment rules ranging from the properties of a user to environmental conditions at the time of request. In the ERBAC model, apart from user capability and user request, access control in a Web environment is also constrained by a variety of conditions. All the aforementioned access control requirements and conditions can be described as RBAC constraints. Constraints are classified into different categories according to their definitions, such as content-based, environmental-based, history-based and temporal constraints as shown in section 3.3.3. They can be applied to different parts of the extended FB-RBAC model, such as user-capability, role, A-function and role assignment. In this thesis, constraints are generally used for three purposes.

- They can decide which role should be activated. For example, a particular set of user capability may be assigned to a day shift role between 9am – 5pm, and a night shift role for the rest of the day.
- They can determine whether or not the role should be activated. For example, a constraint may specify that a particular role cannot be activated for more than ten users at the same time. This is an example of cardinality constraint.
- They can constrain what roles can or cannot do. For example, a constraint may specify that a particular role can only access its own information record.

In the first two cases, constraints are used in role assignment. They may be called pre-conditions. In the third case, the constraints are applied after a role has been assigned. They may be called post-conditions. For example, in a Web-based bookstore service, a particular role can only access its own information record. Constraints are imposed on the content of the input parameters of the A-function in the same way that constraints are applied in the FB-RBAC model. We argue that the contents of user input and output are essential to realize fine-grained access control. For example, we may specify that an ordinary user can only read his/her own account information. The input parameter – customer name, of the A-function searchCustomer() is then restricted to a certain value, for example to "John". Whereas if Mary, an employee of the online book store, will be given full privileges for the Web service, including access to all customer account information.

#### *4.3.4.2 Using Constraints and Authorization-functions to Support Separation of Duty*

Constraints can also be used to define well-known security policies such as Separation of Duty (SoD). SoD is designed to prevent users from performing tasks that have conflict of interest. It can be classified into static SoD and dynamic SoD. In static cases, mutually exclusive roles cannot co-exist in a set of roles assigned. With dynamic SoD, a user may be assigned mutually exclusive roles, but they cannot be activated at the same time in a session.

Static SoD is used when administrators manually assign users to roles, which is not suitable for dynamic role assignment in Web environments. In this section, we focus on dynamic SoD as it can be applied to the role assignment process described in the ERBAC model. The traditional SoD policy is represented by defining mutually exclusive roles. However, not all privileges

assigned to mutually exclusive roles are in conflict. Such implementation may end up preventing users from exercising some of their legitimate privileges. For example, a clerk role and an accountant role cannot be activated at the same time, because these two roles perform the conflicting tasks of "request for credit update" and "update credit limit". But mutual exclusivity between the conflicting roles of clerk and accountant also means that none of the other permissions granted to these two roles can be exercised at the same time. Hence, we argue that mutually exclusive roles are not suitable for defining SoD.

ORBAC [YANG03] uses a less privileged role -- task role to specify SoD (refer to section 4.2). Task role is more precise than conventional role, but it is still not the best option for describing SoD for a particular task. Compared with roles, the notion of A-function is a more straightforward term for describing SoD, since it represents a single permission rather than a set of permissions owned roles. By adopting an A-function-based approach, the principle of SoD can be realized by defining mutually exclusive A-functions. This new way of describing SoD not only prevents conflicting tasks from being performed at the same time, but also avoids possible service repudiation during the access control process.

### 4.3.5 Formal Definition of the Concepts

The existing models have provided a range of formal specifications for security policies, such as first-order predicate logic [YAO01], context-free grammar based RB-RBAC language [ALKA02], policy algebra [WANG04] and constraint logic programming [BARK03]. These policy specification formalisms are specified in high-level logic expressions using predicates and mathematical symbols. In this section, we will also provide formal definitions for each of the

concepts and relation using mathematical symbols and predicates. A set of security objects involved in access control is defined below. In this specification, an upper case letter denotes a set of elements, a lower case letter denotes single element.

- Web Service

  Web Service Set: WS = $\{ws_1, ...., ws_k\}$, where $ws_i$, $1 \leq i \leq k$ is a Web service.

- Web Service Instance

  Web Service Instance Set: WSI = $\{wsi_1, ...., wsi_k\}$ where $WSI_i$, $1 \leq i \leq k$ is a Web service instance.

- A-function

  A-function Set: F = $\{f_1, ...., f_k\}$ where $f_i$, $1 \leq i \leq k$ is an A-function, each A-function $f_i$ has weight $w_{fi}$ according to its importance.

- A-function Instance

  A-function Instance Set: FI = $\{fi_1, ...., fi_k\}$ where $FI_i$, $1 \leq i \leq k$ is a A-function instance.

  The relationships between the above defined concepts are presented below:

- Web service $ws_i$ consists of a set of A-functions: $F_i = \{f_1, ...., f_m\}$, where $f_i$, $1 \leq i \leq m$ is an A-function.
- Web service $ws_i$ has a set of instances $WSI_i = \{wsi_1, ...., wsi_k\}$, where $WSI_i$, $1 \leq i \leq k$ is a Web service instance.

- Web service instance $wsi_i$ is an instance of Web service $ws_i$, it consists of a set of A-functions instances: FI = $\{fi_1, ...., fi_n\}$ where $fi_i$, $1 \leq i \leq n$ is a A-function, n is the number of A-functions in Web service instance $wsi_i$, $n \leq m$, $FI \subseteq F$.
- Each A-function $f_k$ has a set of input parameters IP = $\{ip_1, ...., ip_m\}$ where $ip_i$, $0 \leq i \leq m$, is the name of the input parameter, and a set of output parameters OP = $\{op_1, ...., op_n\}$ where $op_i$, $0 \leq i \leq n$, is the name of the output parameter.
- Each instance of A-function $f_k$, namely $fi_k$ has as a set of input parameters SIP, where $SIP \subseteq IP$, and a set of output parameters SOP, where $SOP \subseteq OP$.

The definition of the basic components of role-based access control such as user (now represented by user capability and user request), role and permission will be based on the concepts and relations defined above.

- User Capability

User capability contains a set of (permissions to) Web service instances $WSI_{uc} = \{wsi_1, ...., wsi_k\}$, where $wsi_i$, $1 \leq i \leq k$ is a Web service instance. Each Web service instance $wsi_i$ corresponds to a unique Web service $ws_i$, and so no two Web service instances in $WSI_{uc}$ are the instances of the same Web service.

- User request

A user request can be defined as a pair (rf, IPV), where rf is the A-function that the user requests to access, and IPV is a set of input = $\{ipv_1, ...., ipv_m\}$, where $ipv_i$, $0 \leq i \leq m$, is the value of the input parameter.

- Role

A set of roles can be defined as R = {$r_1$, …., $r_k$) where $r_i$, $0 \leq i \leq k$, is a role, each role $r_i$ is assigned a weight $w_{ri}$, where $w_{ri} = w_{f1} + w_{f2} + \ldots + w_{fk}$, which is the total weight of the A-functions assigned to the role. Each role $r_i$ is assigned a set of permissions, which is also defined as a set of Web service instances, $WSI_{rp}$, and a set of constraints applied to the role $C_{rp}$, thus $r_i$ can be described as a triple ($WSI_{rp}$, $C_{rp}$, $w_{ri}$).

- Seniority of the role

A role $r_s$ is a senior role of role $r_j$, if $WSI_j \subseteq WSI_s$, where $WSI_s$ is the permission assigned to $r_s$ and $WSI_j$ the permission assigned to $r_j$, they are denoted as $r_s$ SeniorTo $r_j$. At the same time, role $r_j$ is a junior role of role $r_s$, which is represented by $r_j$ JuniorTo $r_s$ Seniority of the role is used to represent the inheritance relationship among predefined roles.

- Service request

A service request is acting on behalf of a particular user towards an A-function or a Web service. It is represented as a triple (R, F, IPV), where R is the role assigned to the user, and F is the A-function to be accessed, IPV is the set of input parameters initiated by the user which is defined as {$ipv_1$, …., $ipv_m$}, where $ipv_i$, $0 \leq i \leq m$, is the value of the input parameter,

- Constraint

Constraints are heterogeneous, we will define constraints using the following parameters: (1) constraint ID (CID); (2) the security concept it has imposed on (CC), such as constraints on role,

constraints on A-function and constraints on permission; (3) what purpose it is used for (CP) (refer to section 4.3.4) such as to decide which role is activated (category 1), to determine whether a role should be activated (category 2), constrain what roles can or cannot do (category 3); and (4) the content of the constraints (CO). A set of constraints can be represented as C = $\{C_1, ...., C_k\}$ where $C_i$, $1 \leq i \leq k$ is a constraint, it can be defined by a tuple (CID, CC, CP, CO).

Having formally defined all the concepts and relations involved in the proposed model, we will elaborate the access control of Web service using the set of notations provided.

In a generic case, user A requests to access A-function $f_k$ in Web service $ws_i$, the capability of the user is defined to be $WSI_{uc}$, and user request is defined as (rf, IPV). The first step is to identify Web service instance $wsi_i$ that is an instance of the requested Web service $ws_i$ in set $WSI_{uc}$.

Once Web service instance $wsi_i$ is found, it will be mapped to the capability role which indicates the capability of user A in a particular session. Given a set of predefined roles R, each assigned with a triple ($WSI_{rp}$, $C_{rp}$, $w_{ri}$). The role assignment engine generates a set of roles denoted by $R_c = \{r_1, ...., r_k\}$, each $r_i$, $0 \leq i \leq k$, is assigned with a triple ($WSI_{rpi}$, $C_{rpi}$ , $w_{rii}$), where $wsi_i \in WSI_{rpi}$, and $C_{rpi}$ is satisfied. Within this set of Roles $R_c$, the most privileged capability role $r_c$ can be found by identifying the role that has the largest weight among all in $R_c$.

For each user request defined by (rf, IPV), a request role $r_r$ will be assigned to it. Role $r_r$ has to be a junior role of the capability role $r_c$. Role $r_c$ has a set of junior roles $R_s$, in which a subset of them, $R_{sj}$ can access the requested A-function rf, and also satisfy the constraints defined. The least privileged role to the user request will be derived by calculating the weight of each role $r_i$ in

set $R_{sj}$. The resultant role $r_r$ are assigned to the user request. Thus, the service access request will be formed by a triple ($r_r$, rf, IPV).

The service access request triple ($r_r$, rf, IPV) will be evaluated against a set of constraints (post-conditions), before the access to the A-function rf is allowed.

## 4.4 Discussion and Future Work

The proposed ERBAC model provides application-level access control for Web service applications. Since traditional security infrastructure requires proof of identity and centralized control, they are not suitable in Web environment. The proposed role assignment in ERBAC is based on user credentials provided at the time of request instead of user identities registered with the system. By taking a credential-based approach, ERBAC is capable of handling dynamic and anonymous users typical in Web service applications. Role assignment is automated in ERBAC and uses a set of predefined rules. These role assignment rules include dynamic security information such as user credentials, temporal constraints and environmental conditions. The automation of role assignment has not only relieved security administration work such as manual assignment of users to roles, but also allowed access control to be consistent with frequently changing user profiles and environment profiles. The problem of heterogeneous users and resources involved in the identity-based manual role assignment has now been looked after by an automated role assignment mechanism. Moreover, ERBAC has inherited all the advantages of FB-RBAC proposed in chapter 3. One of these advantages is that roles and authorization-functions used in FB-RBAC and ERBAC are more expressive and meaningful representations of

security subjects and security objects involved in authorizations and security policies than in other methods. This allows security administration and specification to be greatly simplified. Therefore, when credential-based dynamic role assignment is integrated with authorization-function-based access control, the problem of subject and object heterogeneity involved in both role assignment and security management can be resolved. The credential-based role assignment has avoided the dependency on user identities in the access control process, and at the same time it has automated access control decision making.

ERBAC is more comprehensive than the existing credential-based access control models [HERZ00, ZHON01, YAO01, ALKA02, BHAT05] since it has used constraints to support dynamic role assignment. The notion of constraint can be used to describe any types of security policies, but its full usefulness is not exploited in the existing RBAC models. In our model, a constraint is applied to many areas of access control for various reasons. For example, a constraint can be imposed on the input and output parameters of the authorization-function to facilitate fine-grained access control for Web service applications. In addition, the model also strictly enforces the least privilege principle on role assignment to ensure only necessary privileges are granted to users.

ERBAC provides fine-grained access control for Web service applications by presenting different users with different interfaces according to the credential provided at the time of request and various environmental factors. These user interfaces not only differ in the A-functions they provide but also in the parameters of the A-functions. By imposing constraints on the content of the input and output parameters, we can enforce finer-grained control based on the information being provided. Moreover, security policies like SoD are also implemented in a fine-grained

manner. Mutually exclusive authorization-A-functions are defined instead of defining mutually exclusive roles in SoD. Authorization-function is more specific and meaningful than the notion of role for describing conflict of interest.

ERBAC provides a set of formal notations to define concepts and relations involved, so that authorizations and security policies can be specified formally. This formal notation set can be easily extended and updated. It is capable of describing a variety of access control scenarios. However, this formal notation uses high level logic expressions; hence it is not machine-understandable. In order to achieve interoperability, it must be represented by policy specification languages like XML-based languages or ontology-based languages.

Certain issues are beyond the scope of this thesis. For example, we have not addressed security information apart from user capability in the discussion of user credentials. A user credential can carry more security information than just user capabilities such as features of the user, credential issuer. In a dynamic Web environment, the content of user credentials also should vary according to the access control requirements of a particular Web service application. Furthermore, there is no comprehensive analysis on different types of constraints. In this chapter, we have only considered the simplest case of a Web service. In reality, a Web service may be composite in which one Web service interrelates with a number of other Web services. In such situations, access control needs to consider the interrelationship between the Web services. Moreover, this model has not taken into the consideration the latest developments of the Semantic Web. The Semantic Web is a new form of Web content that can be interpreted by software. It provides a common framework for data to be shared and reused across application, enterprise, and community boundaries. Thus it may be widely adopted by future Web service applications.

In the future, we may consider different types of security information that can be stored in user credentials in addition to user capability. Moreover, the dynamic composition of user credentials based on access control requirements of target Web services may be investigated. Further investigation is also required for different types of constraints and their applications. Due to the increasing popularity of the Semantic Web, there is a need to develop access control models that are in line with the development of the Semantic Web. We will address the access control problem for Semantic Web services in chapter 5.

## 4.5 Summary and Conclusion

In this chapter, we presented an application-level access control model, ERBAC, for Web service applications. We described the application of the proposed model to Web service applications. This model has inherited the features of the FB-RBAC model proposed in chapter 3. Web services are described by A-functions and A-functions are used as the basic access control unit and to represent permissions. The new features of ERBAC include credential-based access control and dynamic role assignment. The notion of user was represented by user credentials instead of user identities. The user credentials provided user capabilities for subsequent authentication and authorization. Role assignment was made dynamic according to a variety of conditions including user credentials provided by a trusted third party and a set of constraint-based security policies. Security policies are represented by a variety of constraints and involve dynamic environmental factors. This model also provides fine-grained access control by using Web service instances and A-function instances to realize partial access. The least privilege

principle is strictly enforced throughout the role assignment process to eliminate security. We also showed that constraints like separation of duty policy can be more precisely implemented under ERBAC. Finally, formal specifications were used to define concepts and relations involved in the proposed model.

# Chapter 5

# An Ontology-based Access Control Model for Semantic Web Services

## 5.1 Introduction

This chapter is an extension of the research conducted in chapter 4 and focuses on providing access control for Semantic Web services. The Semantic Web has attracted a great deal of attention, since it describes Web resources in machine-processable metadata and these metadata will allow software agents or search engines to find and interpret Web content more quickly and precisely. The core of the Semantic Web is ontology, a proven description technique that is used to represent concepts and their relations. "Under the Semantic Web, Web pages are annotated by concepts and relations that are formally defined by ontologies" [QIN03]. The Semantic Web has many advantages over the traditional Web. For example, it supports automatic reasoning ability for Web resources and Web services. Moreover, by using Semantic Web pages, it will be

possible to automatically query and extract meaningful content from these Web pages. Due to the extreme variation in the information presented by different systems/services, the Semantic Web can play an important role by providing common frameworks for data to be shared and reused across applications, enterprises, and community boundaries [SEMA01]. Semantic Web service is an essential part of the Semantic Web development. It is built on top of the OWL-based Web Service Ontology, ***OWL-S*** (formerly DAML-S), and is used to support tools and agent technology to enable automation of services on the Semantic Web. It is envisaged that semantic Web services can be used in the following areas [SWSA]:

- **B2B** – Facilitate business to business (B2B) communication in Enterprise Integration Systems.
- **Web** – Provide ability to query, discover and combine Web services dynamically based on the requirements of software agents.
- **Agent** – Semantic Agent-oriented organization can provide planning and control or coordination for large heterogeneous organizations, such as logistics.
- **Grid** – Semantic Web services can be useful in Scientific Grid Computing, an emerging technology for widely distributed large-scale heterogeneous computing.
- **Ubiquitous** – Ubiquitous Computing efforts focus on the emerging area of wireless computing, and the use of portable devices to tap into a variety of spatially proximate computing services in a very context-specific fashion. Semantic service descriptions can provide common understandings between spatially proximate computing services.

However, security, trust and privacy issues of Semantic Web services are mainly unsolved [SEMA]. In the next section, we will identify some of the main security issues related to semantic Web services.

### 5.1.1 Problem Statement

Access control of Semantic Web services is a challenging issue, as it not only faces the problem of conventional Web services but also the problems arise from the Semantic Web. We mentioned in chapter 4 that access control of Web services should be performed automatically according to a set of conditions such as user credentials, user requests and some environmental conditions in the Web environment. The development of the Semantic Web has made information sharing and automatic service discovery possible through the use of ontology. The problem with Semantic Web services is that they are specified by ontologies like DAML-S or OWL-S. Access control automation cannot be achieved without a Semantic Web compatible security framework. However, current research on Semantic Web service security has focused on matchmaking between the capabilities of Web service requesters and the requirements of Web service providers. Research in the area of semantic Web security is still at its primary stage.

### 5.1.2 Outline of the Proposed Solution

To provide automated access control for Semantic Web services, we need to establish a security framework that is in line with the Semantic Web. In this chapter, an ontology-based access control model is presented for Semantic Web services. The proposed model applies ontology to the Extended Role-based Access Control (ERBAC) model (presented in chapter 4)

with the objective of bridging the gap between access control services and the concepts represented in ontologies such as the Semantic Web, and to provide automated reasoning ability in the access control decisions-making process. ERBAC model has two important aspects: credential-based access control and dynamic role assignment. These two aspects enabled dynamic access control according to user capabilities and a set of access control policies. Since ontology is core to the Semantic Web, providing an ontology-based description for ERBAC model will allow security services to be integrated with Semantic Web services. Such integration will facilitate automatic reasoning for access control of Semantic Web services.

This ontology-based framework represents concepts and relations involved in ERBAC in several different types of ontologies. Each component of ERBAC will be defined in a separate ontology, for example, role ontology, policy ontology or process ontology. These ontologies describe the attributes of the component and its relationship with other concepts in the model. The role assignment process will be represented by a process ontology that implements the two role assignment algorithms presented in sections 4.3.2.3 and 4.3.2.4. The role assignment ontology will take input described by the user ontology, the role ontology, and the permission ontology, and return roles as an output. The process is also restricted by the constraints specified by the constraint ontology. This process ontology provides a semantic description for the dynamic role assignment process that allows role assignment to be performed by semantic reasoning.

In addition, a formal semantic policy representation framework is developed as an integral part of the ontology-based RBAC framework. This policy representation framework is derived

from ontologies developed for the ERBAC model. It uses the syntax and vocabulary of a Web ontology language (OWL) to specify access control policies represented by ontologies.

### 5.1.3 Summary of Contributions

The proposed Ontology-Based ERBAC (OB-ERBAC) model employs ontology and ontology language to support semantic Web services and resources represented in ontologies. It is the first step towards a complete ontology-based access control model. By applying ontology to the ERBAC model, in particular to the role assignment process and to the role assignment rules, we provide automatic reasoning ability for access control decision making. Moreover, an ontology-based semantic policy framework has been developed that allows security policies to be understandable across multi-domain environment.

### 5.1.4 Structure of the Chapter

In section 5.2, we analyse related works in the area of semantic Web security. Ontology and ontology-based language are introduced to the readers as background information with the focus on an access control model for Semantic Web and semantic policy representation framework. Section 5.3 presents the Ontology-Based Extended Role-based Access Control (OB-ERBAC) model. We will analyse the Extended Role-based Access Control (ERBAC) model and apply ontology techniques to describe it. A formal policy representation framework is developed along with the model. Section 5.4 discusses the advantages and limitations of the OB-ERBAC model, and possible improvements in the future. Section 5.5 provides a summary of the chapter.

## 5.2 Background

There are several challenges in developing an access control model for the semantic Web, as was described in section 5.1.2. The first one is to specify and enforce security policies over concepts defined in ontologies. The second one is to represent security policies in a formal policy representation framework, in order to resolve syntactic and semantic discrepancies in a multi-domain environment. This section will first provide some background knowledge on the Semantic Web, ontology and ontology languages. It will then review some important work in the area of access control for the semantic Web and formal policy representation framework in relation to the challenges mentioned above.

### 5.2.1 The Semantic Web

The Semantic Web has been introduced to define a machine-interpretable web, and it is targeted for automation, integration and reuse of data across different applications [QIN03]. The Semantic Web is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URI (Universal Resource Indicators) for naming. Ontology is the basic component of the semantic web, since it can define and connect the concepts used to describe security relevant information on the Web. By using Semantic Web annotations, Web-based information and services become accessible and understandable to a variety of agents and applications.

### 5.2.2 Ontology

Ontology is an ancient Greek philosophical term meaning "a specification of a conceptualization" [GRUB]. In the Artificial Intelligence (AI) world, ontology is defined as "the term used to describe and represent an area of knowledge" [OWL-a]. Due to the semantic discrepancies between services within scalable, open distributed environments, ontology has been used to provide formal semantic descriptions for Web content and services. Ontology vocabulary sits on top of an XML schema and RDF schema. Ontologies like DAML-S, OWL can provide a rich description of the terminology, concepts, and the relationships among those concepts relevant to a particular domain or area of interest [SAND]. Different types of ontologies are used for different purposes.

- Role-based ontologies define terminology and concepts relevant to a particular user, person or application.
- Process ontologies define input, output, constraints, relationships, terms, and sequencing information relevant to a particular business process or a set of processes.
- Domain ontologies define the terminology and concepts relevant to a particular topic.
- Interface ontologies define the structure and content restrictions relevant to a particular interface such as an API, database, scripting language, or content type.

Ontology differs from an XML schema in that it is a knowledge representation, not a message format [OWLb]. Moreover, it also provides richer semantics than XML schema for formal representation of knowledge. Since ontology-based specification allows data to be shared, reused and queried among different domains, they can be used to achieve a common understanding

between different formalisms and enable interoperability, and consequently automatic reasoning and querying for Web services can be realized through ontologies [OWLb].

### 5.2.3 The Application of Ontology in Semantic Web Services

As a part of semantic Web development, semantic Web services are described by ontologies like DAML-S and OWL-S. OWL-S (formerly DAML-S) is an ontology of services that describes Web services at the application-level, and complements the various industry efforts at the lower level. By using OWL-S, Web services providers will have a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-intepretable form, and users and software agents will acquire the ability to discover, invoke, compose and monitor Web resources with high degree of automation [DAML-S, OWL-S]. OWL-S consists of three main parts: service profile, process model and service grounding (Figure 5.1).

The service profile is used for advertising and discovering services. It consists of three types of information: a description of service and service provider, the functional behaviour of the service and several functional attributes used for automated service selection [AGAR04a]. The process model provides a detailed description of a service's operation. The two main components of the process model are process ontology and process control ontology. Process ontology describes a service in terms of its inputs, output, preconditions, effects and possible sub-processes; process control ontology describes each process in terms of its state, including initial activation, execution and completion [AGAR04a]. The service grounding specifies how to interoperate with

a service via messages [OWL-S]. OWL-S is capable of providing comprehensive descriptions for features, workflows, operations and interactions of various Web services.

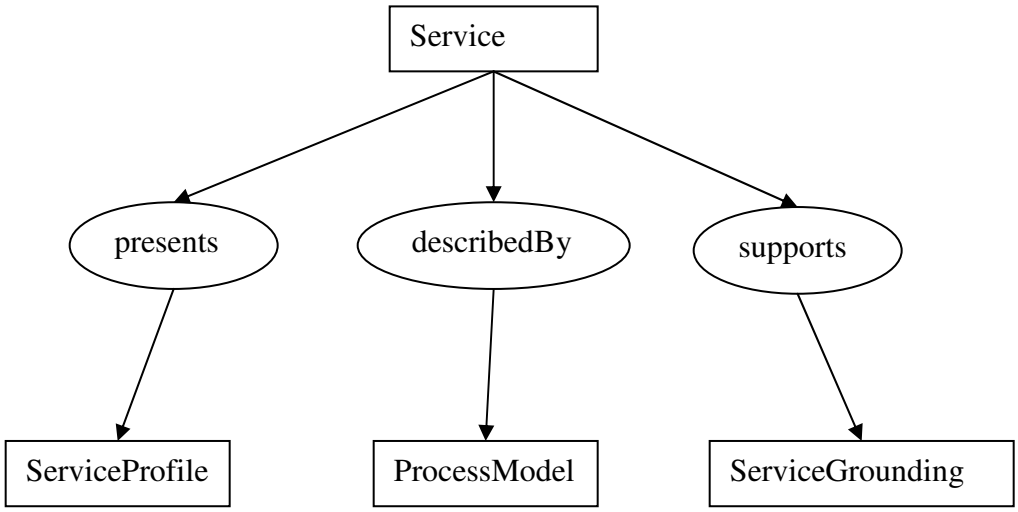



*Figure 5.1 OWL-S Ontology (based on Figure 1 in [OWL-S])*

### 5.2.4 Ontology Languages

Ontologies offer expressive, declarative, system independent languages for specifying richer semantic description of Web documents. The World Wide Web Consortium (W3C) developed a description logic-based Web Ontology Language (OWL) for defining and instantiating Web ontologies based on the Resource Decision Framework (RDF). It is a revised version of the DAML+OIL Web ontology language. OWL is also an important component of Semantic Web

activities. It has several advantages over conventional Web service description languages such as the Web Service Description Language (WSDL) [OWL-S].

- OWL, like all other Web ontology languages, is semantically richer.
- OWL has more facilities for expressing meanings and semantics.
- OWL can support advanced Web search, software agents and knowledge management.
- OWL provides tools to reason and discover Web services.

These advantages make OWL a suitable specification language for representing heterogeneous policies in Web environments.

### 5.2.5 Ontologies for Semantic-Web Security

#### *5.2.5.1 Access Control Model for the Semantic Web*

Research on access control of Semantic Web services is still at its primary stage. Qin and Atluri [QIN03] proposed an access control model for the Semantic Web that was capable of specifying and evaluating authorizations over concepts defined in ontologies. The most important contribution of the paper was that it identified and categorized the semantic relationships among concepts, and provided propagation policies for authorizations. The conceptual level security policies were defined by subject, action, object and sign, which were similar to the subject-operation-object paradigm used by traditional database security models (refer to section 3.1). The subjects are roles and privilege groups; the objects are the concepts annotated by ontologies; the actions can be “read”, “write” and “update”; and the signs indicate either positive or negative authorizations. The security policies are represented in a Semantic Access Control language

(SACL) that is based on the Web Ontology Language (OWL), but also includes some new vocabulary for security concepts. However, as we discussed in chapter 3 and chapter 4, the traditional subject-operation-object approach is unable to provide efficient access control for enterprise systems and Web service applications. Moreover, this model cannot support complex access control policies involving constraints and sessions.

#### *5.2.5.2 Using Ontology to Describe Security Concepts*

Several papers [DENK03, AGAR04a] provided ontology-based semantic descriptions for security concepts like credential and Access Control List (ACL). Credentials are digitally signed documents that specify granted capability for a service or characteristics of the user. Ontology-based credentials can be used to match with ontology-based Web service descriptions, and to query and discover Web services.

A DAML-S based security framework [DENK03] for semantic Web services proposed security-related ontologies to represent well-known security concepts such as credentials. These ontologies were used to describe security requirements and capabilities of web services, so that they could be exploited by an ontology reasoner and matched against agent requests. Security ontologies were merged with the DAML-S ontology for Web services by defining two new properties for Web services, namely "securityRequirement" and "securityCapability" of type SecurityMechanism. The integration of ontology-based security mechanisms with ontology-based Web services was designed to provide security brokerage between agents and services. An important contribution of this paper is that it proposes an ontology-based security framework that integrates various existing security related standards, and addresses general security mechanisms

without redefining all the details for specific implementation choices. This approach enables interoperability among heterogeneous platforms, and it provides a formalism for agents to reason about appropriate Web services. Moreover, DAML-S is extensible so that new components and concepts can be easily integrated into the existing framework. However, the security mechanisms discussed in this paper were mainly used to match user requests with the appropriate Web services and to provide authentication using a credential ontology. The security ontologies are designed to describe the communication level security framework such as security protocols like XKMS and security standards like WS-Security. They have not provided specifications for high-level security policies like SoD and policy models like RBAC.

Agarwal et al. [AGAR04a] provided a credential-based access control for Semantic Web services using DAML-S and Simple Public Key Infrastructure (SPKI)/Single Distributed Security Infrastructure (SDSI). As mentioned in section 5.2.3, DAML-S is an ontology for describing Web services with the objective of making Web services computer interpretable. SPKI/SDSI is a public-key infrastructure suitable for providing credentials for Web service access control. In this framework, the specification of access control policies together with user credentials can be expressed in a unified manner by DAML-S. A Web service is described as a process ontology in which access control policy is defined as the preconditions of the process and credentials are defined as the input to the process. The access control policies are based on Access Control Lists (ACL). By combining DAML-S and SPKI/SDSI, this framework is able to automate access control decision making based on credentials provided by an independent trust structure and described by an ontology. However, this paper only provided a semantic description of

credentials without providing a comprehensive ontology-based description for all security concepts involved in Web service access control such as sessions and constraints. Furthermore, the proposed framework relies on traditional ACLs for access control. However, ACL is unable to model a variety of security policies and security concepts in a heterogenous Web environment.

#### *5.2.5.3 Using Ontologies for Policy Representation*

Semantic discrepancies pose serious challenges to access control of Web services as they prohibit information sharing among different services. A number of papers have proposed formal policy representation frameworks to resolve semantic discrepancies between security policies in a Web environment. Some important requirements for a policy representation framework are identified in [CLEM05].

- A policy framework needs to provide unambiguous description of the terms and concepts, and the meaning of a policy written in this language is independent of its particular implementation.
- A policy framework needs to be flexible and extensible to allow new types of policies to be added.
- A policy framework should provide interoperability among different domains.

XML-based policy frameworks were adopted by many people to specify security relevant information in distributed environments, since they provide uniform, vendor-neutral representations of enterprise data, and allow information sharing across heterogeneous systems [BHAT05]. Two important XML specification languages are the Security Assertions Markup

Language (SAML) and the XML Access Control Language (XACML). SAML is mainly designed for authentication purposes, whereas XACML is intended for Web-based authorization. XACML provides access control specification, supporting assignment of privileges according to user credentials and context information. However, "XML can only provide surface syntax for contextual information, and imposes no semantic constraints on the meaning of the content" [OWLa]. SAML and XACML are unable to resolve semantic discrepancies between different security services. Ontologies have been used for knowledge representation in many areas of the computer science. Recently, a number of attempts were made to use ontology as a formal specification tool for Web service policies, such as KAoS (a policy and domain service framework) [USZO03], Rei (a policy specification language, "Rei" is a Japanese word meaning universal or essence) [KAGA03], and Semantic Web Rule Language (SWRL) [SWRL04]. These policy specification frameworks have provided specification, management, conflict resolution and enforcement of deontic-logic-based policies for services, agents, tasks and even concepts of rights, prohibitions, obligations, and dispensations. They have provided vocabularies for describing security policies using semantic languages. However, none of the above policy representation frameworks provided comprehensive semantic for describing application-level access control and access control models.

### 5.2.6 Summary of Existing Work

As an integral part of the Semantic Web, ontologies and ontology languages provide semantic representation and reasoning ability. A number of security frameworks have recently started to use ontologies for describing security concepts, security policies and security mechanisms related

to the Semantic Web. Qin and Atluri [QIN03] provided a conceptual model for the Semantic Web. This model was based on the subject-action-object paradigm and employed a Semantic Access Control Language (SACL). The subject-operation-object paradigm, discussed in chapter 3, has some drawbacks in terms of security management, and it cannot support complex security policies. A number of papers have provided ontological descriptions of security concepts and security mechanisms to support semantic Web services. Security ontologies were developed to represent well-known security concepts like credentials and Access Control Lists (ACL), and they were integrated with DAML-S ontology for Web services to provide security brokering between agents and services [DENK03, AGAR04a]. Such integration have provided the reasoning ability for access control decision making, and allowed policies to be specified in a unified framework. Both papers focused on communication-level security services. Ontology has also been used in formal policy representation frameworks, such as KAoS [USZO03], Rei [KAGA03], and SWRL [SWRL04]. These policy specification frameworks provided vocabularies for describing security policies using semantic languages.

The above work has provided ontological descriptions for security concepts, security policies and security mechanisms that are related to the Semantic Web. However, the security mechanisms provided in these papers are either for lower level security services or are inadequate to model complex access control scenarios at the application level. Currently, there is no comprehensive model that provides application-level access control for semantic Web services.

## 5.3 Ontology-based RBAC for Semantic Web Services (OB-ERBAC)

This section presents an Ontology-based Extended Role-based Access Control model for Semantic Web Services (OB-ERBAC). This model provides ontology-based semantic descriptions for ERBAC models in order to bridge the gap between security services and the Semantic Web and facilitate automatic reasoning during access control.

### 5.3.1 Motivation

The development of the Semantic Web has attracted much attention, since it offers common machine-understandable semantics for describing Web content, and allows data to be shared and automatically processed by different services. Ontology plays an important role in terms of providing formal syntax and semantics for Web service descriptions. However, existing access control models are not suitable for Semantic Web services, since they do not provide compatible security frameworks for semantic Web resources. For example, XML-based access control models for Web services, like X-RBAC [BHAT03a, BHAT03b] and X-GRBAC [BHAT04b], provide limited semantics for describing the security concepts and mechanisms involved in Web service access control. Qin and Atluri [QIN03] argued that when applied to the Semantic Web, existing XML-based access control models should be elevated into a higher layer of the Semantic Web architecture, such as the Resource Description Framework (RDF) and ontologies, since access control for the Semantic Web is dependent on the semantic relationships among concepts. These semantic relationships are represented by ontologies. Therefore, it is necessary to develop an ontology-based access control model that provides semantic descriptions for security concepts, security policies and security mechanisms in order to comply with ontology-based Semantic Web

services. This framework will also provide automatic reasoning ability to query and discover security-related information. It is envisaged that when an ontology-based access control model is integrated with an ontology-based Semantic Web service, the unified framework will be able to provide highly automated access control.

### 5.3.2 Security Ontologies for the Extended RBAC Model

#### *5.3.2.1 Analysis of the Extended RBAC Model*

In this chapter, we will apply ontology techniques to the Extended Role-based Access Control (ERBAC) model proposed in chapter 4 for Semantic Web services. ERBAC has several important advantages that make it a suitable access control model for Semantic Web services. As shown in chapter 4, ERBAC is capable of handling dynamic and anonymous users and reducing security management tasks. ERBAC also supports a variety of access control policies and provides fine-grained access control for Web service applications such as controlling parameters of the user request. However, in order to develop a semantically compatible access control model for Semantic Web services, it is necessary to provide ontological representations for the concepts and relations involved in the ERBAC model.

The basic concepts involved in ERBAC are user capability/user request, role, permission, session, constraint, user-role assignment and permission-role assignment. Each of these concepts can be represented by a separate ontology, and the aggregation of all these ontologies form the complete ontology-based ERBAC model. Ontologies can be classified into different types such as process ontology, role-based ontology and service ontology, each serving different purposes as

explained in section 5.2.2. The core of the ERBAC model is user-role assignment that ultimately determines access control outcomes. The user-role assignment can be represented by a process ontology which is described in terms of input, output, precondition and result. Another important relationship, permission-role assignment, defines which roles have what privileges. It differs to user-role assignment, as the latter is dynamic and are depend on a variety of conditions, whereas permission-role assignment is predefined by system administrators, and used to provide access control policies for the user-role assignment process. Thus, permission-role assignment can be represented by a policy ontology.

#### *5.3.2.2 Vocabulary and Symbols Used in the Model*

Since OWL and OWL-S provide a vocabulary to describe profiles and processes for Web services, some of their vocabulary can also be used for our ontology-based access control model. Each component of the ERBAC model, such as user and role, can be represented by the term *Class* which is an OWL term. These security components (classes) are linked together by using the term *ObjectProperty* (OWL term). The attributes of the security components are described as *DatatypePropery* (OWL term). All properties will have *domain* (OWL term) and *range* (OWL term): the domain specifies the owner of the property, whereas the range specifies the target of the property. The term *subClassOf* (OWL term) is used to specify that a security component, a class, is a subclass of another security component (class), hence it inherits all the properties of the latter. We define a set of symbols to represent the above mentioned terms and relations as depicted in Figure 5.2.

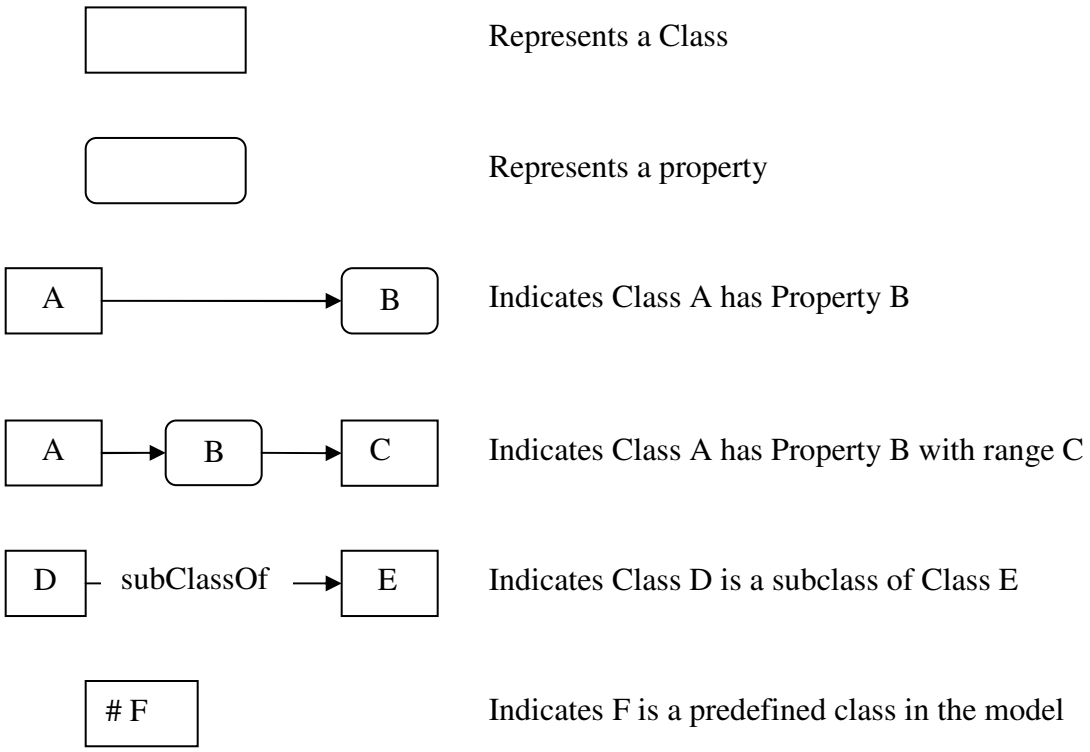



*Figure 5.2 Symbols Used in the Security Ontologies*

A predefined class means this class has been defined by other ontologies in the model. By using OWL and OWL-S vocabulary, our ontology-based ERBAC model has been made compatible with the Semantic Web.

#### *5.3.2.3 Policy Ontology and Process Ontology*

Three different types of security ontologies are defined in the specification of the ERBAC ontology: general security ontology, access control policy ontology and access control process ontology. General security ontology describes the features and functionality of a security-related

object such as user, role, session and permission. Access control policy ontology and access control process ontology have more complex properties. Their unique features are described below.

- Access Control Policy Ontology

In general, access control policy can be seen as a subclass of a policy ontology. A policy ontology defines a vocabulary for representing security and privacy policies and a description logic based mechanism for reasoning about the defined policies [CHEN04]. An access control policy may be described as a security subject being permitted or denied access to a particular security object with a particular operation. The proposed access control policy ontology is influenced by the existing policy ontologies like Rei [KAGA03] and SOUPA [CHEN04]. It can be described with a tuple of four elements (security subject, operation, security object, sign) as shown in Figure 5.3.

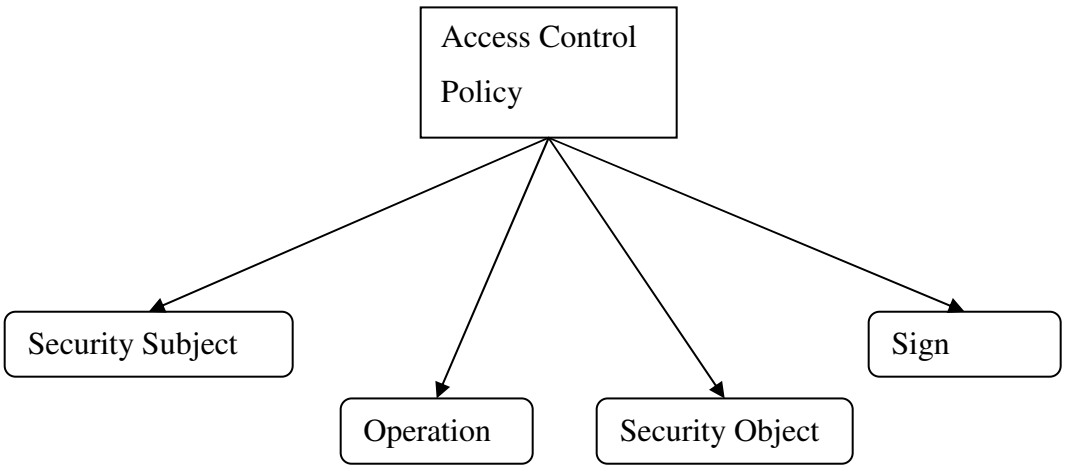



*Figure 5.3 Access Control Policy Ontology*

In the ERBAC model, we only consider whether or not a user can access a particular Web service or a particular A-function, so the operation will always be "has access to", and the sign is always positive since a closed world policy is adopted. Therefore, these two elements do not need to be explicitly specified; However, in most of the other access control models, operations (or access type, access mode) are specified, such as "read", "write" or "modify; and signs can be either positive or negative for positive authorizations and negative authorizations respectively. In order to provide a generic access control policy ontology, we will include operations and signs.

- Access Control Process Ontology

An access control process can be seen as a subclass of the process ontology. OWL-S provides vocabulary for a process ontology using its process model. Hence, the access control policy ontology shown in Figure 5.4 will be described in OWL-S vocabularies.

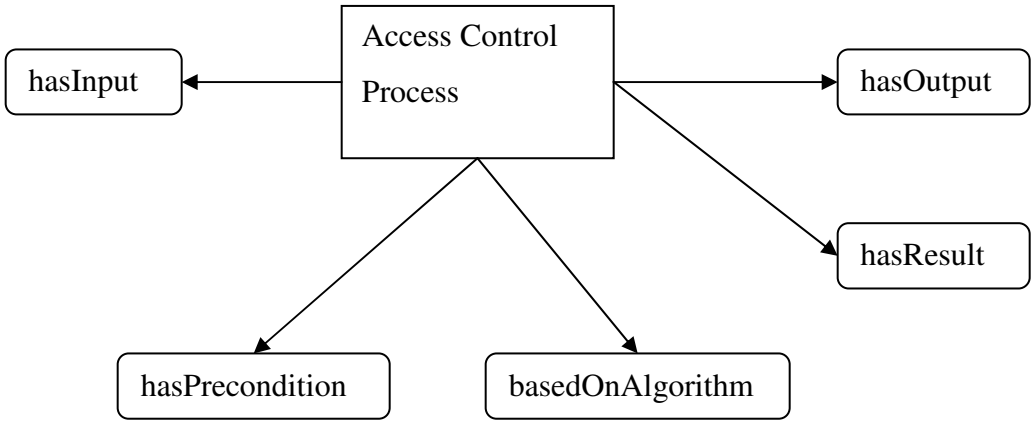



*Figure 5.4 Access Control Process Ontology*

OWL-S describes a process model with the following properties: hasInput, hasOutput, hasPrecondition and hasResult. On top of the OWL-S properties, the access control process ontology will have a unique property called "basedOnAlgorithm" with range "access control algorithm". The access control algorithms specify how access control is carried out, and they are described with logic expressions like "if-then-else" and "while" loops. These algorithms take the inputs and preconditions and produce output and results. The specification of these properties varies from one access control model to the other. However, the result of the access control process will always be either access granted or access denied.

### 5.3.3 Component Ontologies for the Extended RBAC Model

In this section, we will provide separate ontologies for each component in the ERBAC model based on the definition given in section 4.3.2.1 and the ontologies provided in section 5.3.2.

- Security-focused Web service Ontology and A-function Ontology

The description of Web service ontology will be based on the definition provided in the ERBAC model. In the ERBAC model, a Web service is considered as a security object assigned to roles and accessed by users. Therefore, the proposed Web service ontology will only involve security-related aspects. A Web service contains a set of properties that distinguishes one Web service from others such as service name, service type and Authorization-function (A-function). Each Web service consists of a set of A-functions to carry out designated tasks. A security-focused Web service ontology is depicted as in Figure 5.5.

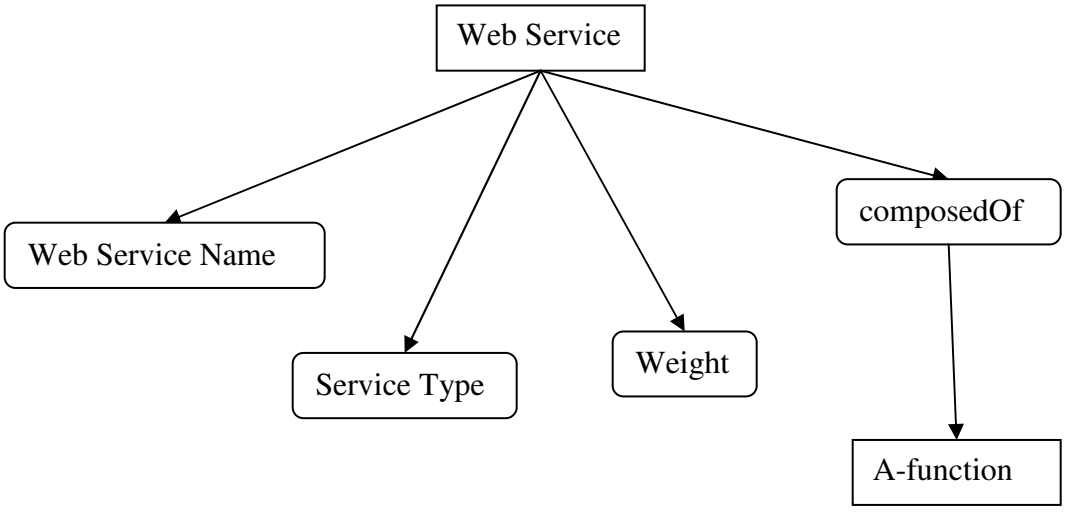
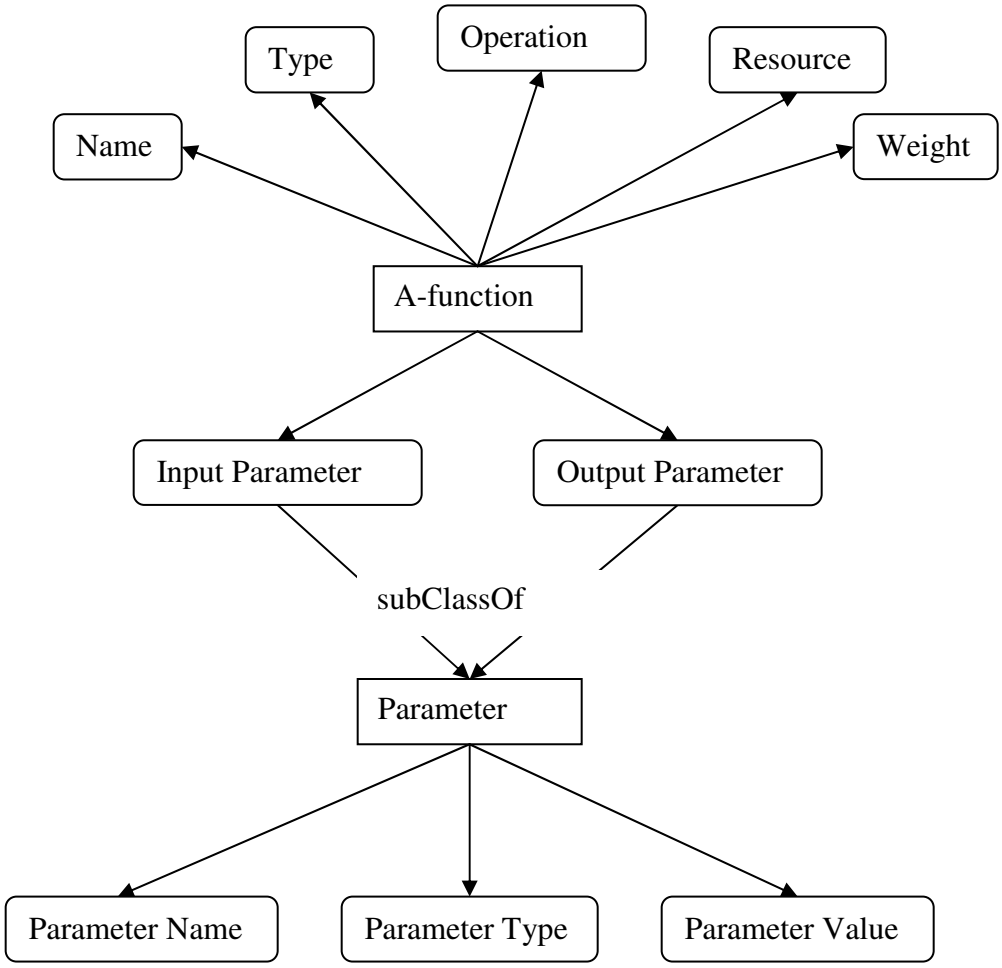



*Figure 5.5 Web Service Ontology*

An Authorization-function (A-function) is a function exposed by an application interface to the outside world in order to execute certain user requests. An A-function is regarded as the basic unit of access control, and it is also a type of security object. An A-function is a part of a Web service, but it can also be represented as a separate ontology because of its importance.

An A-function has the following properties: name, type, operation, resource, weight, input parameter and output parameter as shown in Figure 5.6. A-function type specifies the types of A-functions in terms of the classification method. For instance, an A-function type may be determined by the purpose of an A-function. Each A-function performs one or more operations on back-end resources, therefore can be represented by operation and resource. An A-function has a set of input parameters and a set of output parameters for taking user input and outputting the result of execution.




Each A-function is assigned a weight indicating its security importance. "Weight" can be derived in different ways. The conventional way is based on the importance of the Authorization-functions. In general, an authorization-function updating or modifying the content of databases or files will be given the highest security protection, and hence will have more weight than an authorization-function reading from or writing to databases or files. However, there is no single way of defining "weights". It always depends on the actual application. Input parameter and

output parameter can be described by separate classes, and they are both subclasses of class *parameter*. Class parameter has the following attributes: parameter name, parameter type and parameter value. Input parameters and output parameters inherit all three attributes.

*Figure 5.6 Authorization-Function Ontology*

- User Capability Ontology and User Request Ontology

In ERBAC model, user-role assignment is no longer performed between user and role. Instead, roles are assigned to user capability or user request dynamically. User capability indicates what security objects can be accessed by a particular user. It can be described in the form of subject-operation-object. We will describe user capability as an access control policy ontology, which describes an access control policy using a tuple of four elements (Security subject, Operation, Security object, Sign) (Figure 5.7). User capability is provided by a user credential, so it may be described as a property of user credential. User credentials may also have some other properties related to network-level security services such as PKI certificate and certificate issuer [DENK03, AGAR04a]. However, we are focusing on application-level access control, low-level security services and security concepts will not be discussed here.

A User request is made towards a particular A-function. It has the properties “requester”, “target” and “input”. The requester can be represented by a credential ID, since each user is represented by a credential; the target refers to the requested A-function that will use the input parameters (Figure 5.8).

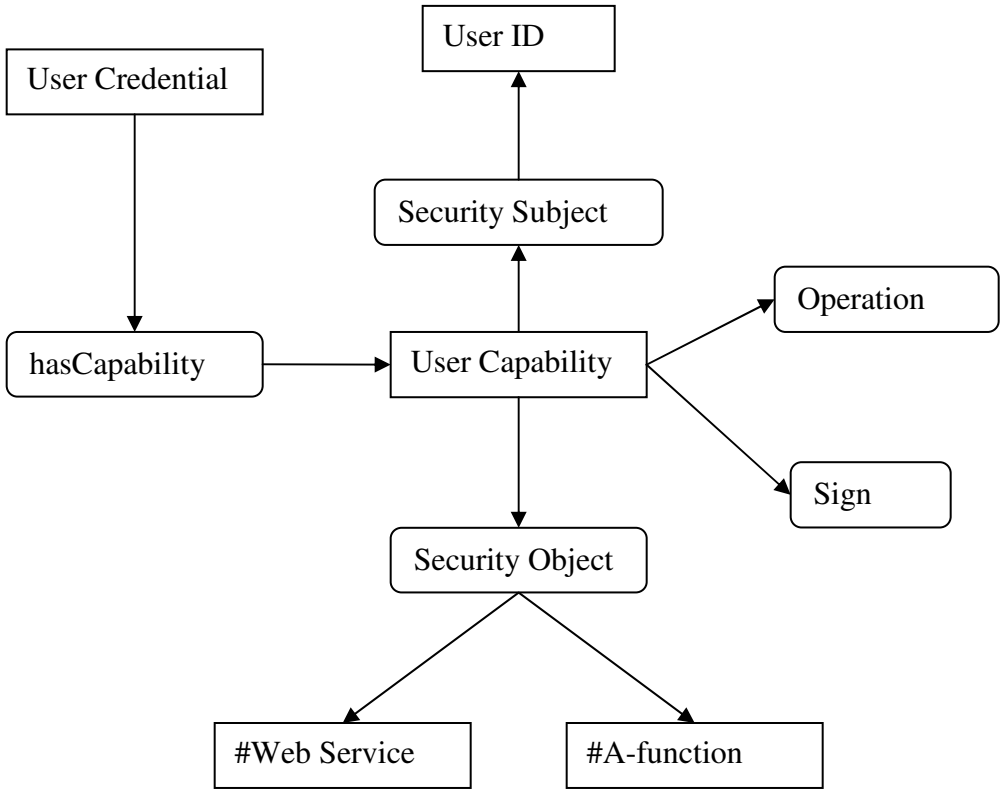



*Figure 5.7 User Capability Ontology*

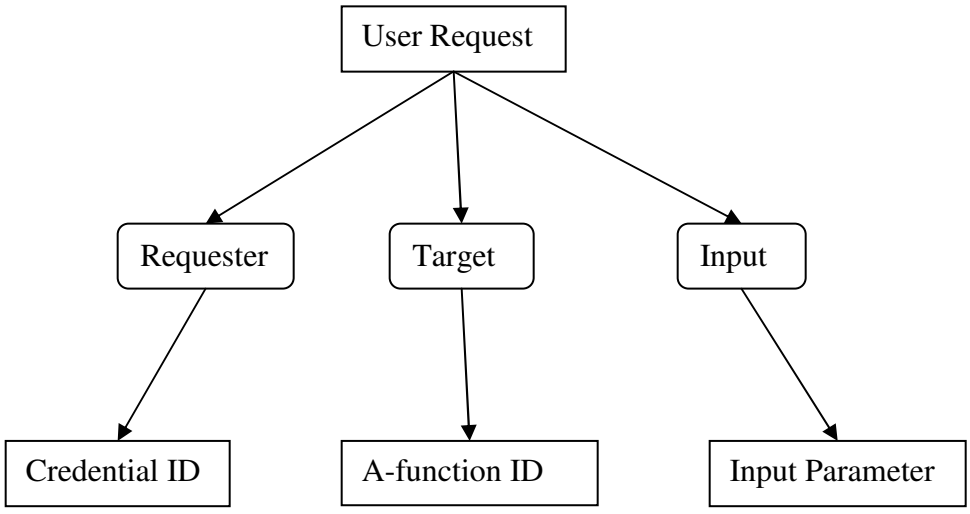



*Figure 5.8 User Request Ontology*

- Role Ontology

Roles represent the privileges of users within an organization. They bring together users with permissions in the ERBAC model. A role is assigned with property "role name", "role type", "weight", "senior role" and "junior role" (Figure 5.9). Role type can be either capability role or request role, depending on whether the role is mapped to a user capability or a user request. The weight of the role is the total weight of permissions assigned to this role. A role is associated with users and permissions via user-role assignment and permission-role assignment respectively. Role inheritance is an important relation in the ERBAC model. In the role ontology, it is specified by using the property senior role and junior role. If one role is specified as the senior role of another then it will inherit all properties of the junior role, whereas if one role is specified as the junior role of another role, then all its properties will be inherited by the senior role.

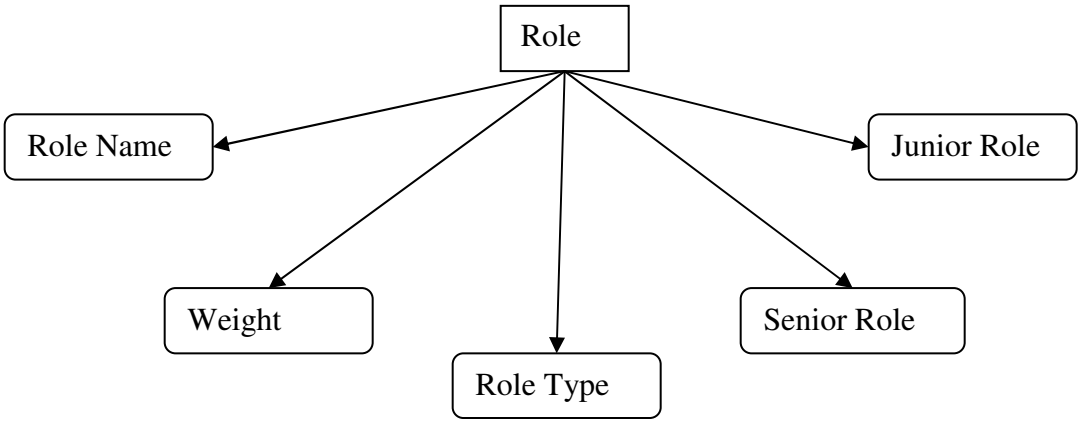



*Figure 5.9 Role Ontology*

- Permission-role Assignment

Permission-role assignment specifies which role has the permission to access what resources. In fact, it provides predefined access control policies. Therefore, permission-role assignment can be represented by an access control policy ontology. It defines access control policies using a tuple of four elements (security subject, security object, operation, sign) as shown in Figure 5.10.

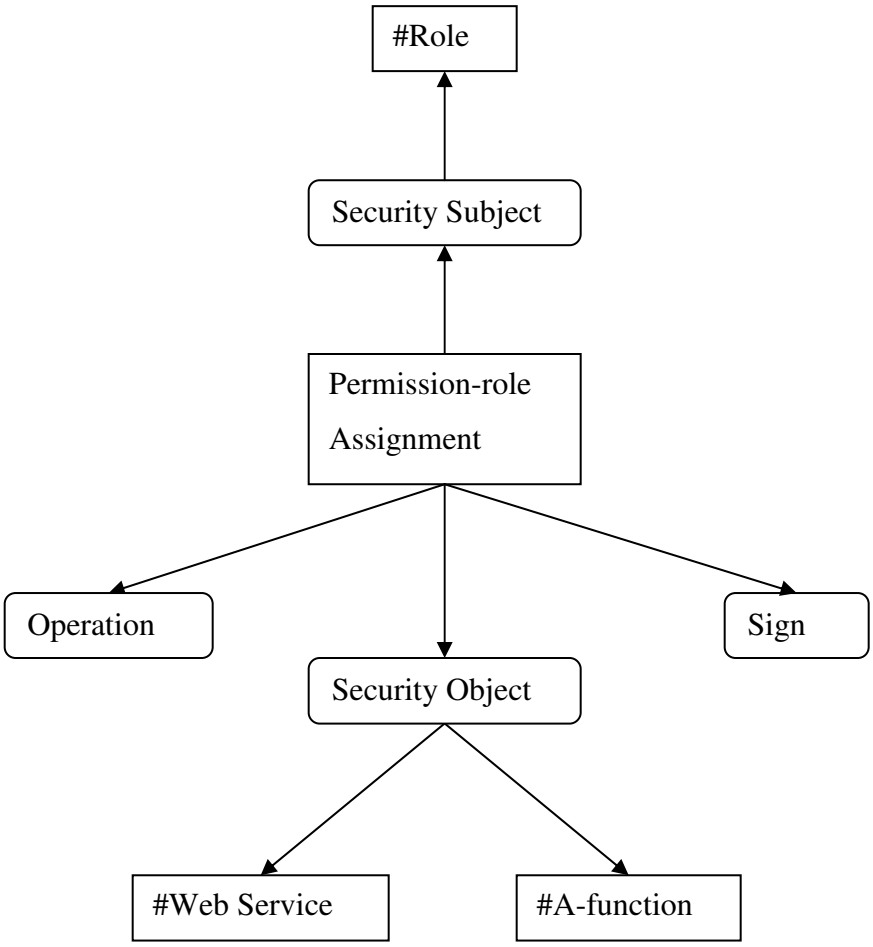



*Figure 5.10 Permission-Role Assignment Ontology*

The security subject will be a role. Operation is always "hasAccessTo" in the ERBAC model. A sign can be either positive or negative depending on what security policy is used. Security object can be any type of protected resource, such as a Web service or an A-function.

- Constraint Ontology

Constraints are conditions that must be satisfied when authorizations are granted. They are used to describe different security policies. A constraint has a number of properties including constraint name, constraint type, constraint target and constraint purpose as shown in Figure 5.11.

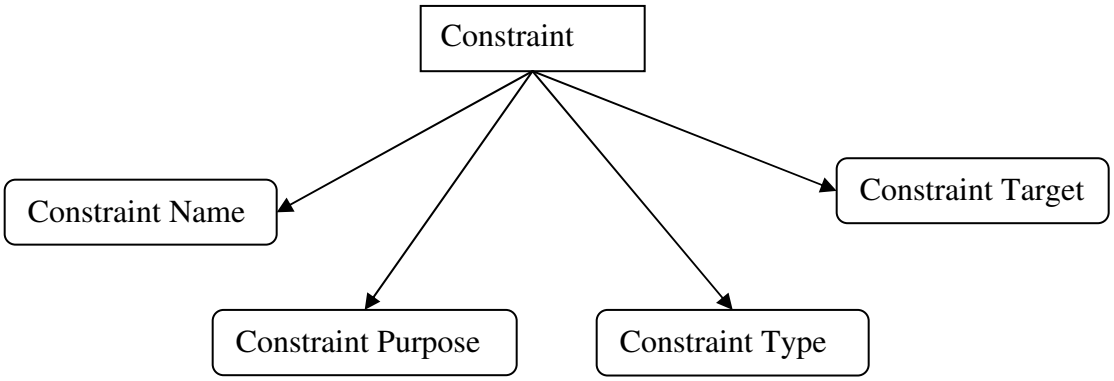



*Figure 5.11 Constraint Ontology*

Constraint type can be A-function attribute constraint, environmental constraint or history-dependent constraint as explained in section 3.3.3.1. Constraint target indicates what the constraint is imposed on. For example, constraints may be imposed on user-role assignment. Constraint purpose tells what the constraint is used for, such as precondition or obligation.

Preconditions must be satisfied before any action is taken, such as granting an access right to a user. Obligations are rules that must be followed in an action, they constrain the way a request is executed. Constraints can be imposed on any concepts in the RBAC model. There are currently many different types of constraints, such as separation of duty (SoD), cardinality, time and location. Each of them has a unique structure. Developing different types of constraint ontologies can be important future research work.

- Session Ontology

A Session refers to the set of events that occurs from when a user connects to an application/system to when that user disconnects from it. A session ontology represents the profile of a particular session.

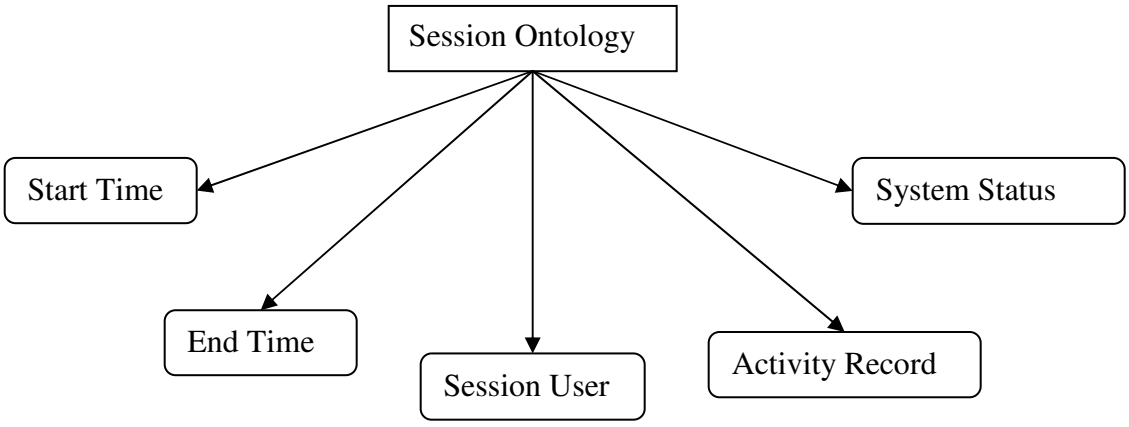



*Figure 5.12 Session Ontology*

Session profiles record all events that happened in each session, and they can be described by a number of parameters. These parameters are often dynamic, as they change with time. The session ontology represents all these parameters as properties. Session properties can vary from one system to another. The common properties are start time, end time, location, activity record and system status (Figure 5.12). A session ontology is linked with user ontology, role ontology, Web service ontology, A-function ontology and constraint ontology. It provides a variety of security conditions for supporting dynamic role assignment.

- User-role Assignment Ontology

User-role assignment (or simply role assignment) is the most important part of the ERBAC model, since it is used as the access control decision-making engine in role-based security systems. The system architecture is depicted in Figure 5.16. User-role assignment matches user capabilities or user requests with appropriate roles, and decides whether or not an access is granted. According to the ERBAC model, the user-ole assignment, as discussed in section 4.3.3, relies on three essential factors: user capability/user request, permission-role assignment and constraint. User-role assignment is also dependent on the role assignment algorithms. We provided two role assignment algorithms in section 4.3. These two algorithms are fed with the above-mentioned three essential factors of role assignment to produce a matching role as output. They are described in the user-role assignment ontology.

In this model, user-role assignment will be represented by an access control process ontology. The access control process can be seen as a subclass of process. A process ontology defines input, output, constraints, relationships, terms, and sequencing information relevant to a particular

business process or a set of processes. OWL-S provides a vocabulary for describing service profiles and service processes. We will use some of the vocabulary provided in OWL-S to describe the user-role assignment process (Figure 5.13).

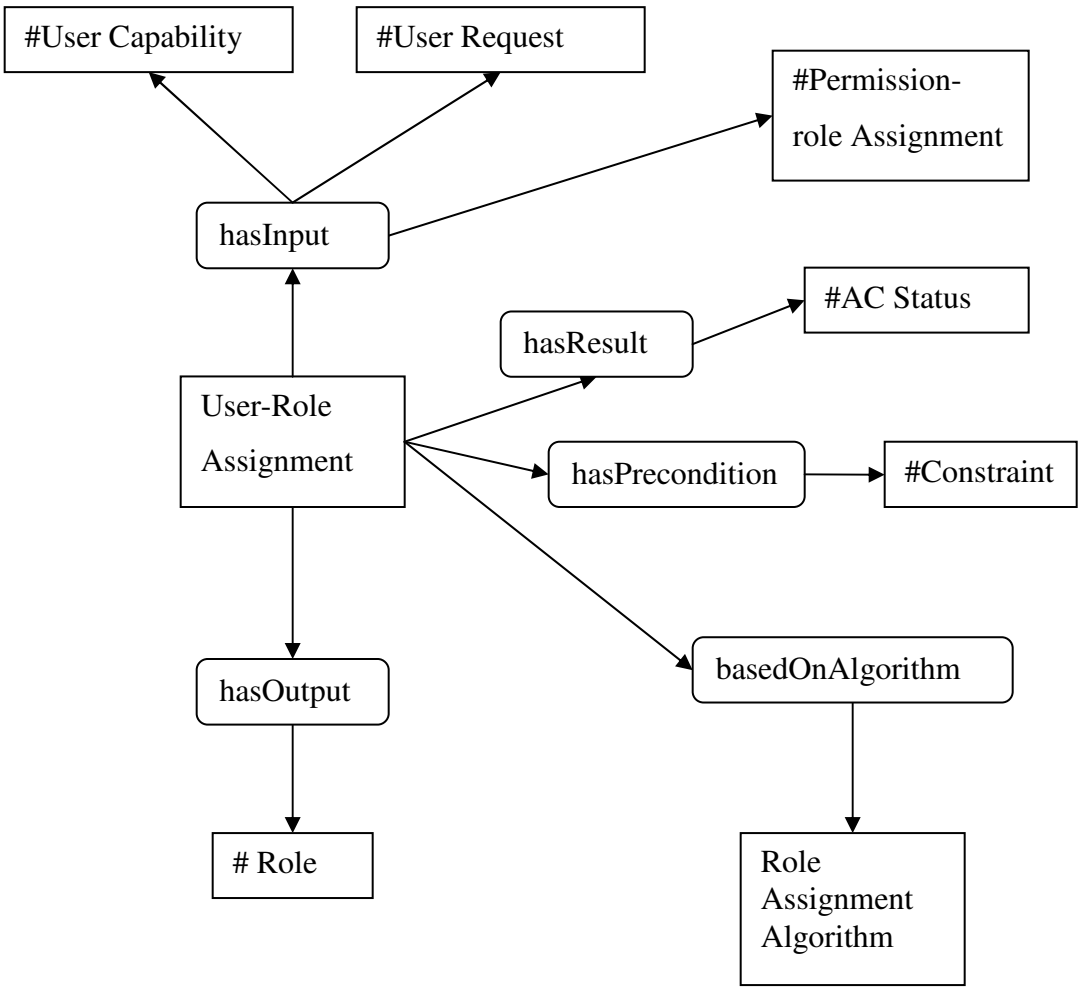



*Figure 5.13 User-Role Assignment Ontology*

According to the definition provided in section 4.3.2, user capability/user request and permission-role assignment are described as the input to the process, and constraints imposed on the user-role assignment are described as the preconditions of the process. The process of deriving an access control decision relies on the role assignment algorithms. We describe it by using a new property called "basedOnAlgorithm" that has range "role assignment algorithm". The algorithms will now take user capability, user request and constraints as inputs, to produce a matching role; they are referred as Algorithm 1 and Algorithm 2 in section 4.3.3. Since OWL-S provides vocabulary for describing logic expressions, we will describe role assignment with an OWL vocabulary. The output of the process is a matching role for each user capability/user request. The result will be an access control decision, either access granted or access denied.

### 5.3.4 Ontology-based Web Service Access Control

#### *5.3.4.1 Full ERBAC Ontology and its Integration with Web Service Ontology*

The complete ERBAC ontology shown in Figure 5.14 is formed by aggregating all component ontologies. These separate security ontologies are linked together by defining one ontology as the property of another. Once the relationship between different components of the model is formalized with ontologies, security reasoning becomes possible. We will be able to query and discover the required security information. Having all required security conditions available, the complete ERBAC ontology is able to generate access control decisions automatically.

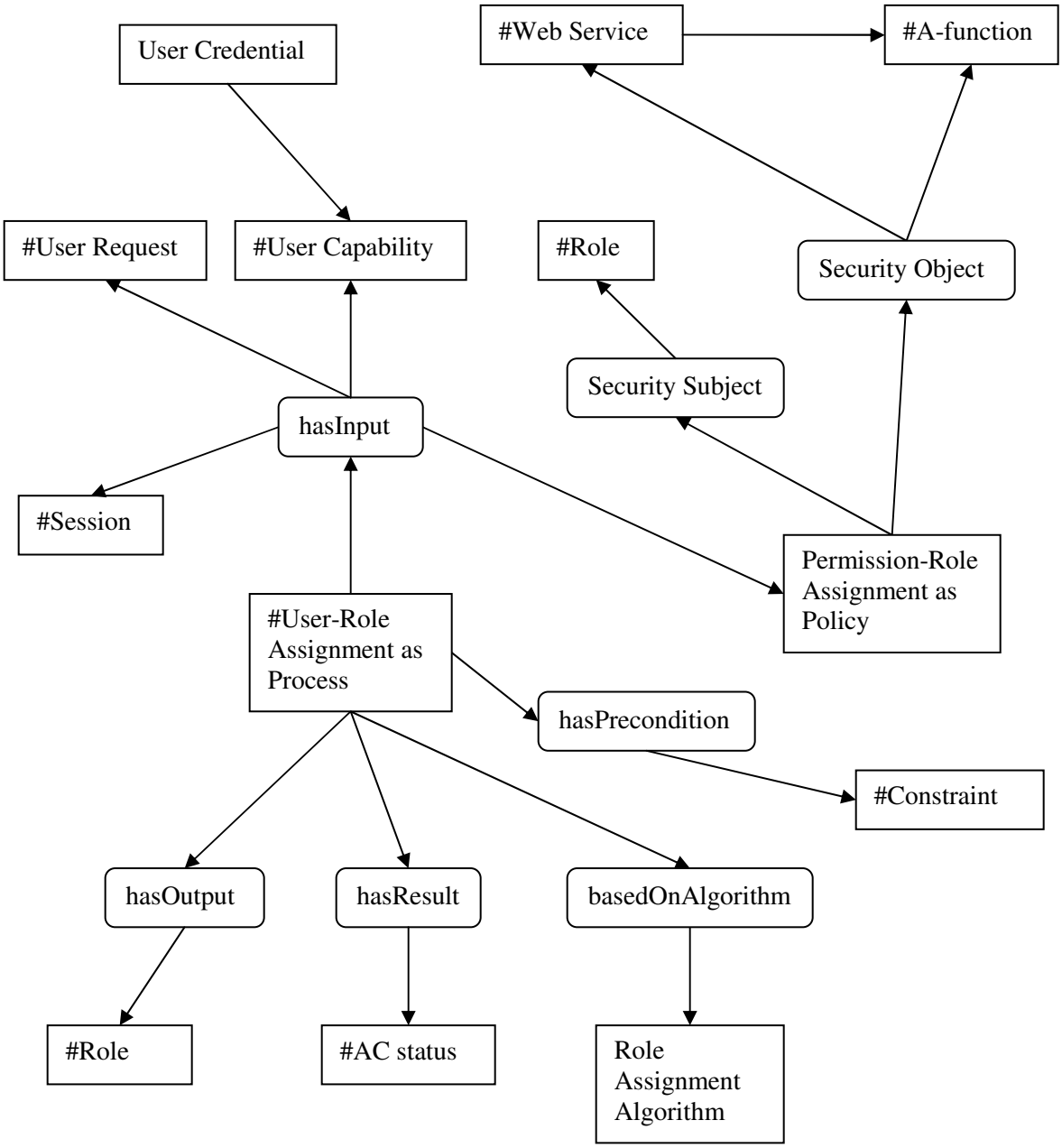



*Figure 5.14 The Complete ERBAC Ontology*

The security ontologies have to be linked with the Web service ontology in order to provide advanced reasoning ability for access control. The entire ERBAC ontology will then serve as the

precondition of the Web service ontology defined in OWL-S [OWL-S], since it must be satisfied for a Web service request to be processed. In OWL-S, the property "precondition" is defined to have range "condition". We will provide a subclass "access control condition" to represent this type of ontology so that security ontologies can be integrated with Web service ontologies as shown in Figure 5.15.

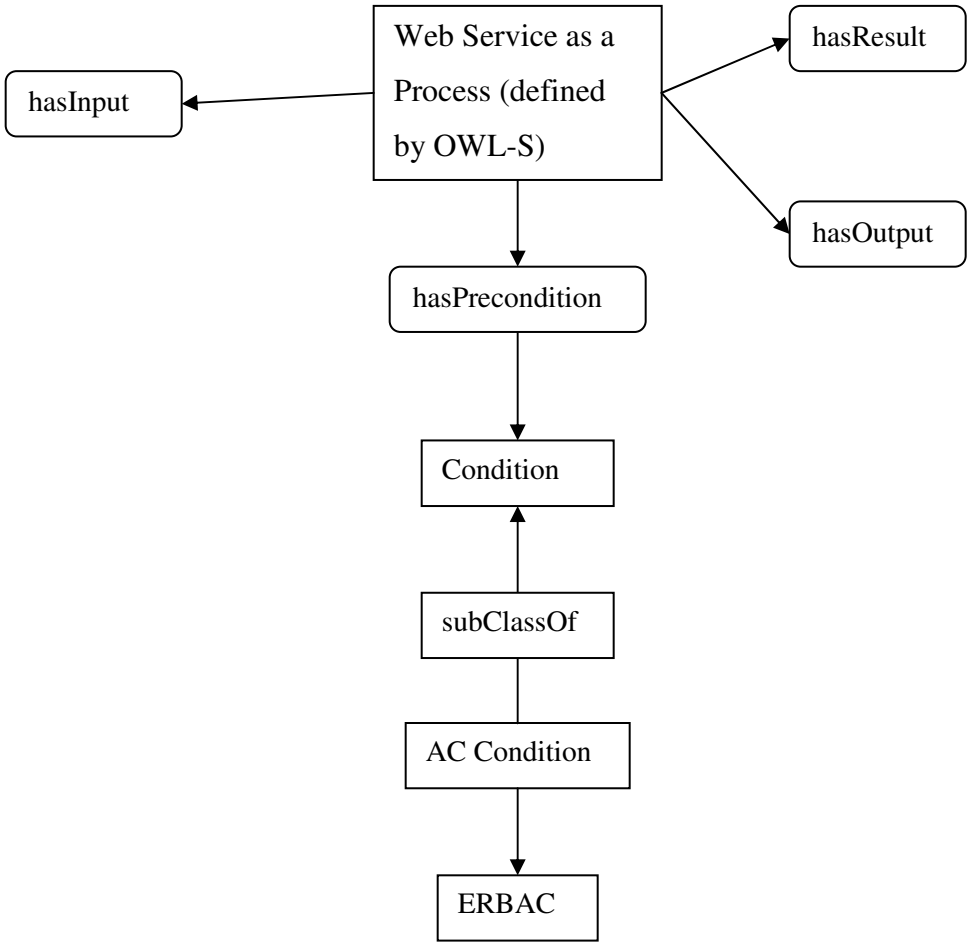



*Figure 5.15 The Integration of ERBAC Ontology and Web Service Ontology*

### *5.3.4.2 An Ontology-based Access Control System*

In this section, we will establish an access control system that provides security mechanisms for Semantic Web services. These security mechanisms implement the proposed ontology-based ERBAC model and provide access control at the application level. The main components of the system are described as the following:

- Web service tier: this layer contains access control mechanisms and security data repositories.
- Web service requester: the agent (user) who makes the request to access a particular Web service or an A-function of a Web service. Web service requesters are represented by their credentials (user capability).
- Session Manager: it records the session history and session related parameters. It also provides session information related to the semantic policy repository.
- Ontology-based Access Decision Engine: this module makes access decisions based on the ontology-based ERBAC model (Figure 5.14) and the policies stored in a semantic policy repository.
- Access Enforcement Module: this module executes the result produced by the access control decision engine. If a request is granted, the access enforcement engine will fetch the user requested data from the back-end resource repository.
- Access Control Policy Base: stores security policies represented in ontology-based semantic language.
- Back-end Resource Repository: it can be a collection of data repositories.

This system architecture is depicted in Figure 5.16.

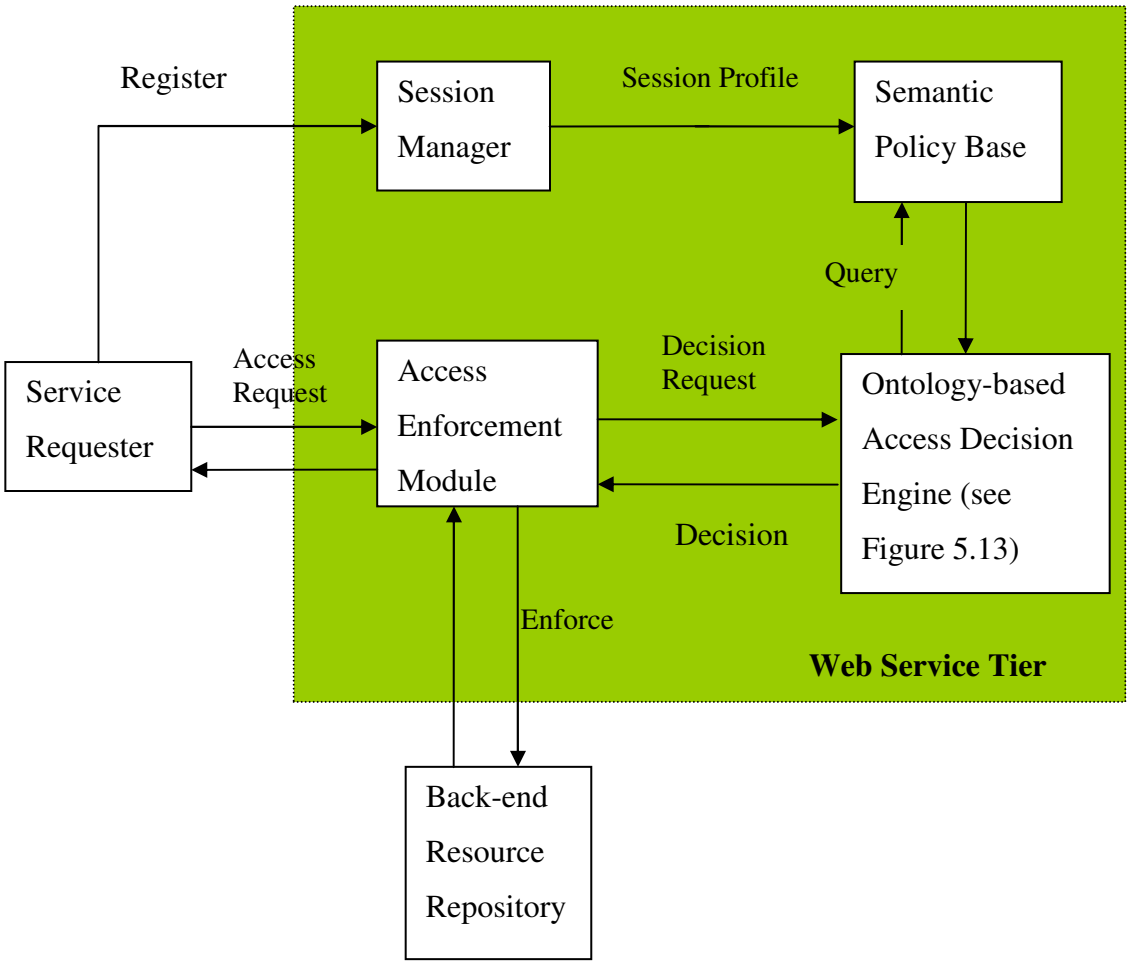



*Figure 5.16 An Access Control System Architecture for Semantic Web Services*

The dynamic role assignment is performed by the Ontology-Based Access Decision Engine (OBADE). OBADE implements the ERBAC model that uses the security information stored in

the semantic policy base (SPB). The result of an access control decision is presented to the enforcement module that allows or denies the user access to Semantic Web services.

## 5.4 Implementation

### 5.4.1 A Web-based Bookstore Management System

In this section, we present a Web-based bookstore management system used to test the proposed model. This system provides two Web services: Customer Management and Book Order. Customer management provides users with functions to manage customer records. A customer record stores customer ID, customer name, address, telephone number and credit limit. Each Web service consists of a number of authorization-functions (A-functions). Web service "customer management" consists of the following A-functions:

- Customer searchCustomerByID(int cust_id)
- Customer[] searchCustomerByName(String cust_name)
- int insertCustomer(Customer newCust)
- String requestCreditUpdate(Customer cust, int newCreditLimit, int pid)
- CustomerCredit[] getAllCreditUpdateRequest()
- String approveCreditUpdate(int pid)
- CustomerCredit[] getAllApprovedUpdateRequest()
- String updateCreditLimit(CustomerCredit cust)

Each of these A-functions is used to perform a particular user request. They are the basic unit of access control in the proposed model.

### 5.4.2 System Architecture

In this section, we will introduce the basic components of the bookstore system. The system is implemented by the following client/server model.

- User Interface (UI) – UI provides client side support by interacting with users
- Web Server – provides Web services to users of the system. It contains access control mechanisms for a Web service.
- Composite (CS) Server – integrates functions from different applications. It encapsulates business logic and exchanges data with a number of database servers.
- Database (DB) server – provides access to different database resources.
- Database (DB) – stores data and resources.
- Access Control Enforcement Module (ACE) Module – presented in Figure 5.16, is implemented by a list of A-functions specified in section 5.4.1. These A-functions are the only channels to access back-end resources.
- Access Control Decision (ACD) Engine – provides access control mechanisms that use dynamic role assignment. This module implements dynamic role assignment as described in Figure 5.13.
- Ontology Interpreter – interpret ontology-based resources
- Access Control (AC) Policy Base– stores semantic security information specified in OWL

- Session Manager – records session related parameters, such as session start time, session end time and activities conducted in the current session. These parameters may be used for time-related or history-dependent access control.

Figure 5.17 presents the bookstore system that is derived from the system architecture presented in Figure 5.16.

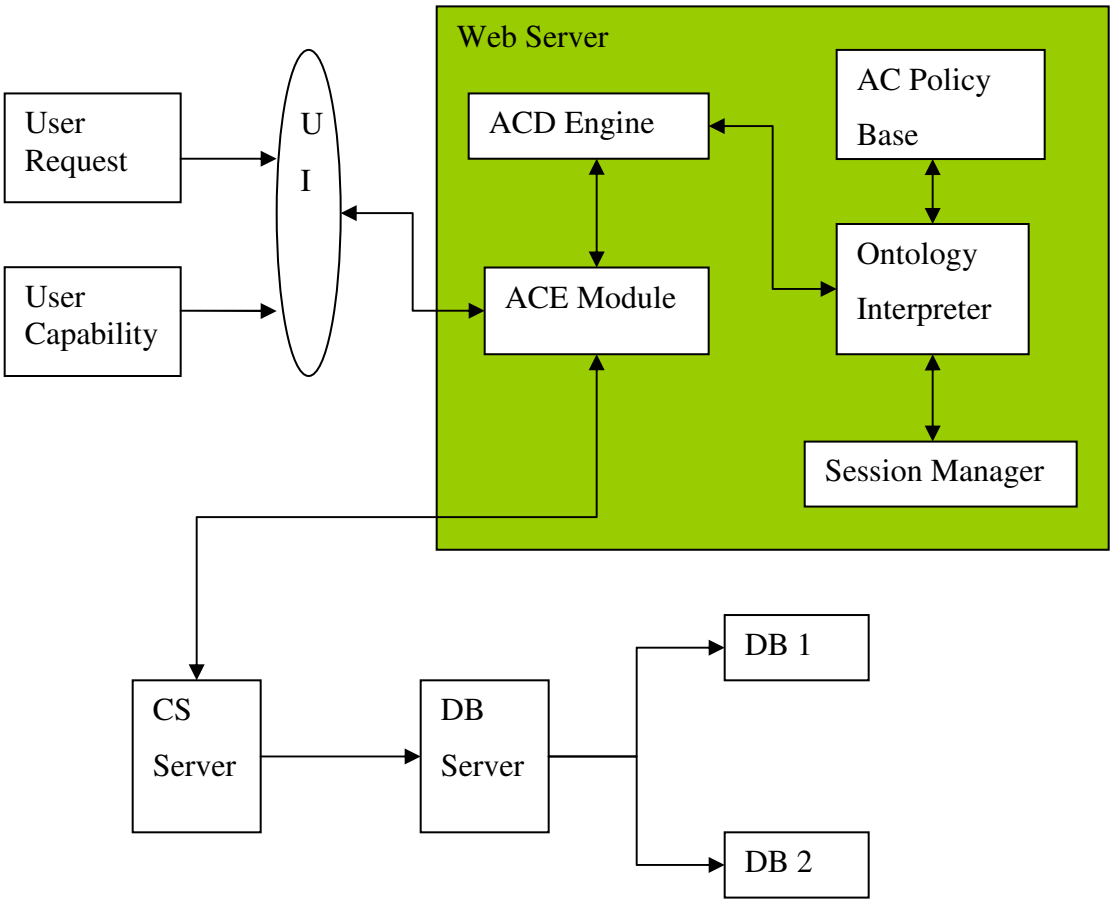



*Figure 5.17 A Bookstore Access Control System*

Except for database and access control policy base, all the above-mentioned components are implemented in Java (JBuilder 9). The Web server is developed using the J2EE application server package. Databases are implemented by Oracle 9 including a Customer Information table. Access control policy base consists of several OWL files, such as Constraint.RDF, PR_Assignment.RDF, Web_Service.RDF. Session manager records the user activities in the current session in OWL files like Auditing.RDF. Communication between Web server, CS server and DB server is achieved via Remote Procedure Invocation (RMI).

### 5.4.3 Examples of Ontology-based Access Control

In this section, we will show how the proposed ontology-based model can be used to control a credit approval process. The operation involves two clerks and an accountant. The process starts when a clerk makes a request to update the credit limit of a particular customer. An accountant, upon receiving the request, will need to approve the credit update request in order to continue this process. Once the credit update request is approved, a different clerk will finalize the process by updating the credit limit of the customer in the system.

The access control process is based on the role assignment ontology illustrated in Figure 5.13. Role assignment ontology is implemented by the Access Control Decision (ACD) Engine shown in Figure 5.17. The inputs of the role assignment ontology include the user capability ontology, the user request ontology and the permission-role assignment ontology. Role assignment ontology is also restricted by the constraint ontology as the precondition. In the implemented system, these ontologies are specified by the files User_Capability.RDF, PR_Assignment.RDF, constraint.RDF respectively. They are implemented in OWL and are stored in an access control policy base. User

requests are user inputs to the system. Hence, they will be shown through several screen shots. The core of the role assignment ontology is the role assignment algorithm. Role assignment algorithms make access control decisions according to various inputs and preconditions. We will use the two role assignment algorithms specified in chapter 4 (section 4.3.3.3 and section 4.3.3.4) for this system. They are used to map user capability and user requests to corresponding roles and to make access control decisions. The outputs of the role assignment ontology are roles that allow users to fulfil their requests. The result of the role assignment ontology is either authorization granted or authorization denied.

In the access control process, constraint specifies conditions that must be satisfied before or during dynamic role assignment. To ensure there is no conflict of interest and prevent potential frauds, we will impose constraints "separation of duty" and "workflow" on the access control process. The "separation of duty" constraint makes A-functions "requestCreditUpdate", "approveCreditUpdate" and "updateCreditLimit" mutually exclusive of each other. The "workflow" constraint assigns an order number to each step of the update credit limit process which indicates the process that needs to be executed in exact orders. An example of Constraint.RDF is shown below.

Constraint.RDF (part)

```
<con:ME_Function ID = "ME1">
    <con:hasMEFunction rdf:resource = "requestCreditUpdate"/>
    <con:hasMEFunction rdf:resource = "approveCreditUpdate"/>
    <con:hasMEFunction rdf:resource = "updateCreditLimit"/>
</con:ME_Function>
<con:Work_Flow ID = "WF1">
```

```
      <con:hasWFFunction rdf:resource = "requestCreditUpdate">
            <con:Order>1</con:Order>
      </con:hasWFFunction>
      <con:hasWFFunction rdf:resource = "approveCreditUpdate">
            <con:Order>2</con:Order>
      </con:hasWFFunction>
      <con:hasWFFunction rdf:resource = "updateCreditLimit">
            <con:Order>3</con:Order>
      </con:hasWFFunction>
  </con:Work_Flow>
```

Another important access control factor is the permission-role assignment relationship which is predefined by system administrators. File PR_Assignment.RDF specifies what A-functions are accessible to each role.

PR_Assignment.RDF (part)

```
      <rd:PR_Assignment rdf:ID = "1">
            <rd:SecuritySubject rdf:resource = "System_User">
            <rd:SecurityObject rdf:resource= "searchBookByID"/>
      </rd:PR_Assignment>
      .......
      <rd:PR_Assignment rdf:ID = "8">
            <rd:SecuritySubject rdf:resource = "Employee">
            <rd:SecurityObject rdf:resource= "searchCustomerByID"/>
      </rd:PR_Assignment>
      .......
      <rd:PR_Assignment rdf:ID = "10">
            <rd:SecuritySubject rdf:resource = "Accountant">
            <rd:SecurityObject rdf:resource= "approveCreditUpdate"/>
      </rd:PR_Assignment>
      .......
      <rd:PR_Assignment rdf:ID = "14">
            <rd:SecuritySubject rdf:resource = "Clerk">
```

```
            <rd:SecurityObject rdf:resource= "requestCreditUpdate"/>
      </rd:PR_Assignment>
      <rd:PR_Assignment rdf:ID = "15">
            <rd:SecuritySubject rdf:resource = "Clerk">
            <rd:SecurityObject rdf:resource= "updateCreditLimit"/>
      </rd:PR_Assignment>
      .......
```

For example, a user Wei logs into the system and carries the following user capability:

User_Capability.RDF (part)

```
      <User_Capability rdf:ID = "CI3">
            <SecuritySubject rdf:resource = "Wei"/>
            <SecurityObject rdf:resource = "searchCustomerByID"/>
            <SecurityObject rdf:resource= "searchCustomerByName"/>
            <SecurityObject rdf:resource= "insertCustomer"/>
            <SecurityObject rdf:resource= "requestCreditUpdate"/>
            <SecurityObject rdf:resource= "updateCreditLimit"/>
      </User_Capability >
```

Wei's user capability is mapped to the role "clerk" according to the algorithm user capability to role assignment (Algorithm 1 in section 4.3.3.3). Role "clerk" is a capability role which indicates that all privileges can be exercised by the user in this session. This will give Wei the privileges to make a credit update request and to update the approved credit request. A personalised user interface is displayed according to the user capability (Figure 5.18). Only A-functions that are accessible to the user will be activated on the Graphic User Interface (GUI).

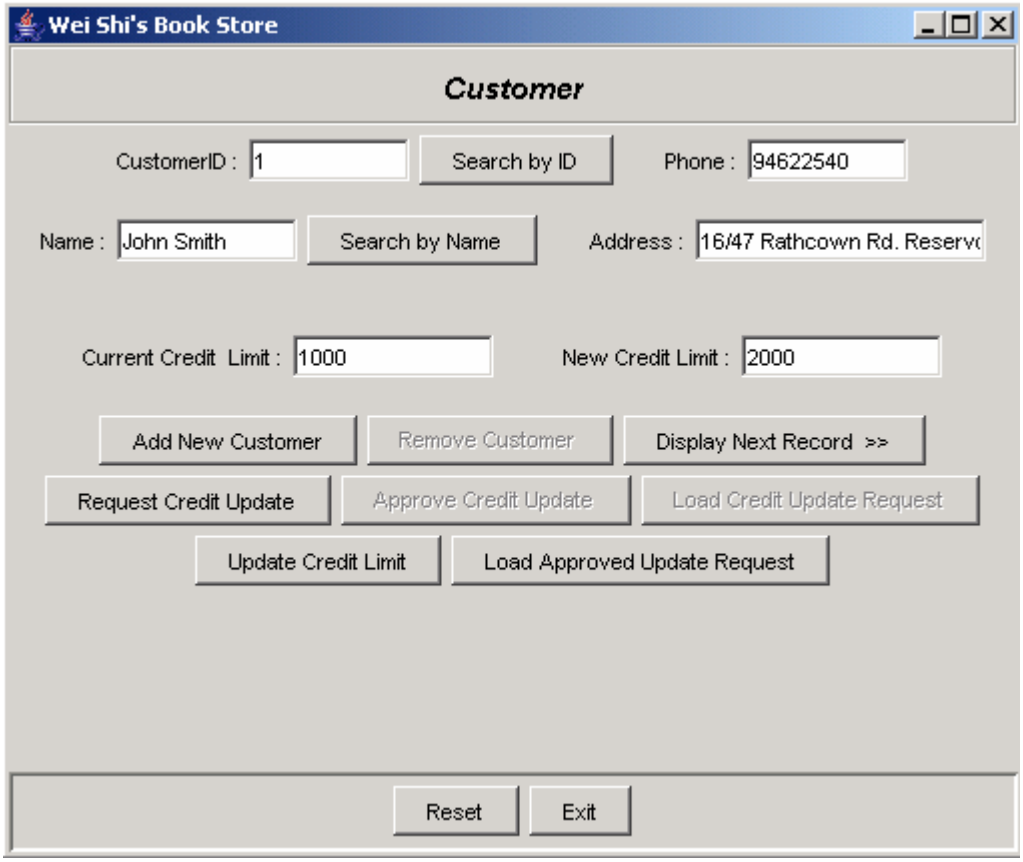



*Figure 5.18 Personalized Web Service 1*

The proposed model follows the strict least privileged principle in access control. Hence, each request made by the user will be assigned to a request role (Algorithm 2 in section 4.3.2.4) that has just enough privilege to access the requested A-function. For example, when the A-function searchCustomerByID() is accessed, the request role for this particular request will be the role "employee" according to PR_Assignment.RDF. When Wei makes a request to update the credit limit of a customer "John Smith", for example from 1000 to 2000 (refer to Figure 5.18), it is assigned to the role "clerk" since it is the least privileged role to access A-function requestCreditUpdate() (refer to PR_Assignment.RDF). If Wei then tries to update the credit limit

himself, he will get the message "Authorization Denied" due to the mutual exclusion defined in the ME_Function node of Constraint.RDF. The Auditing.RDF file is a part of the activity record controlled by the session manager. It records all A-functions executed in a particular session. The Auditing.RDF consists of access records. Each of these access records stores the A-function accessed (FNAME); the role used to access the particular A-function (RNAME), the process ID (PID) indicates an A-function can be a part of a process, and the session identification number (SID).

Auditing.RDF

```
<AUDIT_TRAIL>
      .......
      <ACCESS_RECORD ID = "2">
            <FNAME rdf:resource = "requestCreditUpdate"/>
            <RNAME rdf:resource = "Clerk"/>
            <PID rdf:resource ="1"/>
            <SID rdf:resource ="25"/>
      <ACCESS_RECORD/>
      .......
</AUDIT_TRAIL>
```

The file updateCreditLimit.RDF is designed to record the status of the update credit limit process. It is part of the activity records controlled by the session manager. User requests of different A-functions will have different properties. According to user request ontology depicted in Figure 5.8, the user request has three properties: credential ID (CID), A-function ID (FID) and input parameters (Customer_ID, current_credit and new credit) of the A-function.

updateCreditLimit.RDF

```
<CreditUpdate ID ="1">
      <CID rdf:resource = "Wei"/>
      <FID rdf:resource = "updateCreditLimit"/>
      <CUSTOMER_ID rdf:resource = "John Smith"/>
      <CURRENT_CREDIT rdf:resource ="1000"/>
      <NEW_CREDIT rdf:resource ="2000"/>
      <STATUS rdf:resource ="requested"/>
      <PID rdf:resource ="1"/>
      <SID rdf:resource = "25"/>
</CreditUpdate>
```

The second step of the update credit limit process is to approve the update request. For example, another user, Peter logs into the system carrying the following user capability.

```
<User_Capability rdf:ID = "CI1">
      <SecuritySubject rdf:resource = "Peter"/>
      <SecurityObject rdf:resource= "searchCustomerByID"/>
      <SecurityObject rdf:resource= "searchCustomerByName"/>
      <SecurityObject rdf:resource= "approveCreditUpdate"/>
</User_Capability>
```

This user capability is mapped to role accountant. A different instance of the Web service (see Figure 5.19) is activated according to Peter's user capability. The role "accountant" has the privilege to load the credit update requests and approve it if the claim is valid. The Auditing.RDF and updateCreditLimit.RDF record the activity in this session. They are similar to the ones presented above, but this time, the "STATUS" attribute in updateCreditLimit.RDF is changed to "approved".

Once the credit update request is approved, another user Jim, who has the same user capability as Wei, is mapped to the role "clerk". Jim can then load the approved request and execute the credit update function. The "STATUS" attribute in updateCreditLimit.RDF is changed to "updated", indicating that the credit update process has concluded.

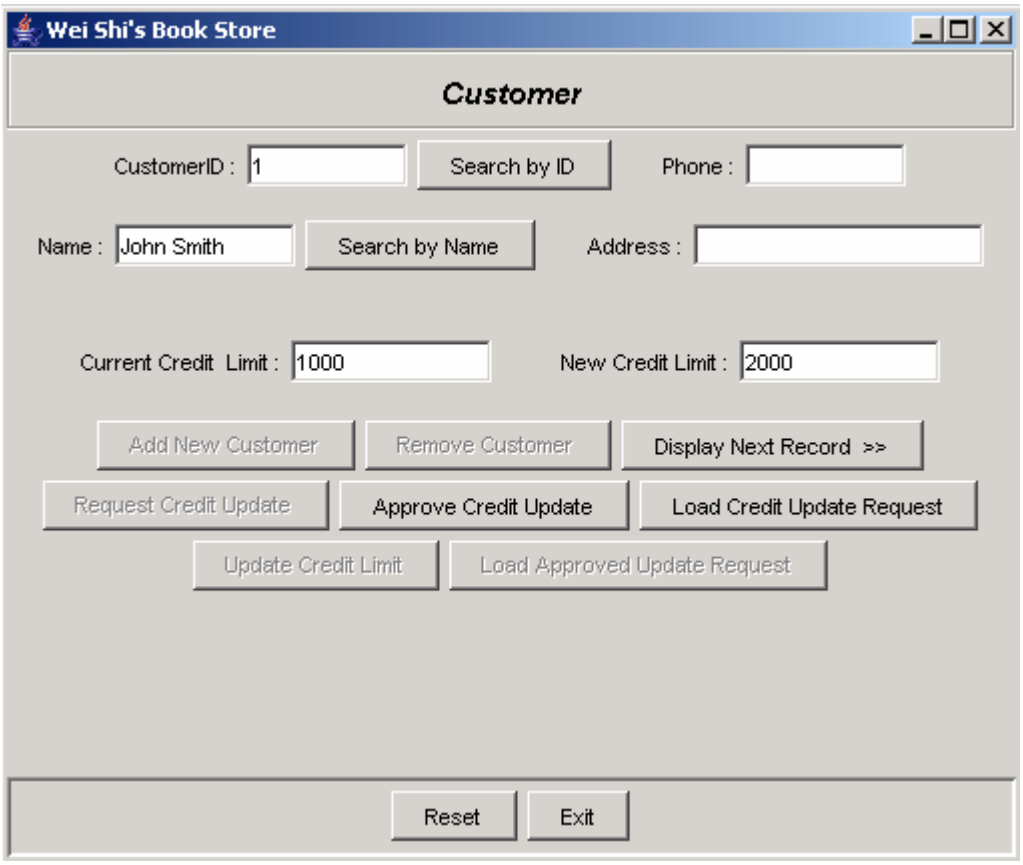



*Figure 5.19 Personalized Web Service 2*

In the whole process, the workflow constraint ensures all the steps are followed in the exact order defined in the Work_Flow node of Constraint.RDF. The clerk who updated the credit limit has to be a different person to the clerk who requested the credit update according to

ME_Function node of Constraint.RDF. This example illustrates how the ontology-based model can be used to provide access control for Web-based systems at the application level.

### 5.4.4 System, Language and Performance

Our bookstore system was tested on three PCs all running Windows 2000. Each computer had 512M RAM and an AMD Athlon XP 2600 processor. The three computers formed a TCP/IP based distributed system. Installation of the system took about 200KB hard disk space for each computer. Since the proposed model is ontology-based, it can be used on any platform that has an ontology interpreter. This makes it a suitable model for large-scale enterprise systems consisting of heterogeneous applications. In the bookstore system, the model was implemented in the Java language to make use of Java's enhanced security features, such as cryptography, public key infrastructure, secure communication, authentication, and access control, to complement the proposed model. Java Remote Method Invocation (RMI) and Java Messaging Service (JMS) were used to exchange data between different terminals. System performance was expected to be affected, as an additional step, namely data conversion between ontology language and XML (as described in 5.4) was carried out by an ontology interpreter during access control. However, in the tests, this effect proved to be minimal. The result indicates that the difference between running an operation with and without ontology interpreters is negligible (less than 0.01 second). However, when the proposed model is implemented in a large-scale enterprise system, the effects on system performance may be more noticeable due to the complexity and number of participating systems. Therefore, it is important to find a balance between security and system performance when implementing the proposed ontology-based model.

## 5.5 Discussion and Future Work

This chapter presented an ontology-based application-level access control model (OB-ERBAC) for Semantic Web services. In this model, ontology techniques were applied to Extended RBAC described in chapter 4 to provide interoperable and unambiguous descriptions for security services, security mechanisms and security policies that are in line with the Semantic Web. These security-relevant ontologies were aggregated and then integrated with Web service ontology to provide automatic reasoning ability for access control decision making. Since the OB-ERBAC model is based on ontology and uses vocabulary from OWL and OWL-S, it can be easily mapped to ontology languages such as OWL. OWL and OWL-S are platform-independent and extensible semantic languages that allow the model to be implemented, extended and modified by different security services. The implementation of the bookstore access control system (section 5.4) proved that the proposed model is capable of supporting a variety of access control policies, such as SoD. However, an ontology interpreter is required to process OWL specifications. In order to utilize the full reasoning ability of ontologies, the ontology interpreter has to understand not only the semantics of all ontologies but the interrelationships between all ontologies. It should be noted that as the ontology network continues to expand, the ontology interpreter will become increasingly difficult to implement. Furthermore, ontology specifications are designed for computer processing, it is difficult for human users to interpret. Its complex semantics make debugging and modification more difficult than the data stored in databases or in XML files.

The proposed model is only the first step towards a complete access control ontology. One important aspect that has not been addressed in detail is the constraint ontologies. Since there are many types of constraints, this research has not yet provided ontologies for different types of constraints. The ontology-based approach is able to derive access control decisions automatically based on user capabilities provided in credentials and on the security requirements of the Web services. However, this model has not provided automatic reasoning ability for additional security information required by Web services; for example, if a user credential has not provided adequate security information for the security requirements of a Web service, then the system will not be able to automatically query and discover the missing information.

In the future, we may provide ontology descriptions for different types of constraints, and this may utilize the existing ontologies for time, person and policy. Furthermore, it is necessary to develop an ontology-based access control language that provides formal representations for the concepts and the mechanisms involved in the OB-ERBAC model. By connecting the role assignment process ontology to other security ontologies, such as credential ontologies specified in [DENK03, AGAR04a], the access control engine will be able to search and discover additional security information required by Web services. By using ontology techniques, it will be possible to compose credentials that satisfy the security requirements of Web services in an automatic way.

## 5.6 Summary and Conclusion

This chapter presented an ontology-based access control model for semantic Web services. An ontology-based specification framework was developed to provide semantic descriptions for

the Extended RBAC using a range of security ontologies. These security ontologies can be used to specify access control policies and the role assignment process. The access control policy ontology was based on the subject-operation-object paradigm that described Web service and A-function as the security objects, and users and roles as the security subjects. The role assignment ontology is the core component for realizing dynamic access control. It used new vocabulary and vocabulary from OWL-S to describe dynamic role assignment. User capability/user request, session, and permission-role assignment are described as the input to the process; constraints are described as preconditions of the process; the output of the process will be a least privileged role. These security ontologies were aggregated and integrated with Web service ontology to provide automatic reasoning ability for Web service access control. Finally, a semantic access control system was developed that includes security mechanisms to process semantic security information and provides ontology-based access control for Web services. A bookstore management system was implemented to test the proposed OB-ERBAC model. The result shows that the OB-ERBAC can provide a unified access control framework for Web-based systems. However, the implementation of an ontology-based access control framework is more complex than the existing ones, since it needs to provide sophisticated mechanisms to interpret and utilize heterogeneous ontologies such as policy ontologies and process ontologies.

# Chapter 6

# Summary and Conclusion

## 6.1 Summary

### 6.1.1 A Review of Research Objectives

The main objective of this thesis is to provide application-level access control for enterprise-wide systems and Web services (Web-based enterprise systems). This includes providing efficient and expressive, dynamic and fine-grained access control models to address subject heterogeneity and object heterogeneity in a distributed environment, and establishing a formal specification framework to represent security concepts in an unambiguous form.

### 6.1.2 Achievements and Contributions

In chapter 3, we developed an authorization-function-based access control (FB-RBAC) model for enterprise-wide systems. Chapter 4 presented an extension of the A-function access control model (ERBAC) aimed for Web services. ERBAC inherits all advantages of FB-RBAC.

Additionally, it utilizes credentials, environmental factors and predefined access control policies to achieve dynamic role assignment. Chapter 5 proposed an Ontology-based ERBAC model (OB-ERBAC) for the semantic Web services. Ontology is applied to the ERBAC model to provide semantic representations of access control policies and mechanisms. The main contributions of this thesis can be summarized as follows:

- Authorization-function-based (A-function) access control

An A-function is regarded as the basic access control unit in all models presented in this thesis. It provides a new access control paradigm for enterprise systems at the application level and complements various lower level security infrastructures. The use of A-function simplifies security management, and enables efficient access control for generic enterprise applications. A-function-based access control also supports a variety of access control policies through the use of constraints, such as controlling the parameters of the A-function. This allows the proposed model to express various access control scenarios, and provides content-based and fine-grained access control for enterprise systems.

- Dynamic Role Assignment

Dynamic role assignment is the core of the ERBAC model and of the OB-ERBAC model. It assigns roles to users dynamically according to user capabilities provided at the time of the request, a variety of environmental factors and a set of predefined access control policies. Dynamic role assignment is designed for distributed environments where there are large numbers of subjects and objects. The main contributions of dynamic role assignment is that it provides

credential-based access control that does not rely on user identities, and it allows access control decision making to be performed dynamically thus capturing changing environmental factors.

- Ontology-based access control representation framework

The ontology-based access control representation framework aims to support Semantic Web services. Security ontologies are developed to specify concepts and terms involved in access control. This model can be easily mapped to the OWL access control specification language. The main contribution of the ontology-based framework is that it provides common access control framework that is inline with the Semantic Web. The use of ontology provides reasoning ability for access control decision making, and allows access control information to be searched, queried and discovered automatically.

### 6.1.3 Future Work

This research has provided a foundation for access control frameworks in a Web-based environment. Using ontology to represent security-related concepts is a new approach towards access control automation and policy specification. In the future, we may:

- investigate different types of constraints and how they can be used to describe different access control scenarios,
- investigate various types of security information carried by user credentials, and how they can be used for dynamic access control decision making,
- ensure consistency between application-level access control service and lower level access control service, and

- provide ontology-based descriptions for a variety of constraints and a more detailed user credentials.

The constraints discussed in the proposed model may be classified according to their nature, such as time constraints, location constraints, and environmental constraints. They can be sub-divided into more fine-grained constraints. Each of these constraints may be described by a set of attributes. The next step can then be to formalize these constraints using ontologies, so they can be utilized in the proposed ontology-based model.

A user credential can carry more security information than just user capability. For example, it may contain the location where user identity can be verified, or a variety of user attributes. A user credential can be viewed as a security object that contains smaller security objects. Each of these objects can be described by a variety of predefined attributes. These security objects should be formally described by ontologies and used in the proposed access control model.

An enterprise security framework provides enterprise-level access control on top of independent local security services. It is crucial that the entire enterprise system is presented in a uniform and consistent manner. While authorization autonomy of the local system is maintained, enterprise security policies may contradict back-end security policies, in which cases conflict resolution is required. Although researchers have studied this topic previously for distributed systems, none of them has taken an ontological approach in a Web service environment. To make the proposed model more comprehensive, this issue should be addressed accordingly.

Each of the constraints and user credential objects can be formally described by an ontology. These ontologies may be developed individually and connected to the access control process ontology when needed. As a result, security constraints and security objects carried by user

credentials can be easily added to or removed from the network of access control ontologies. More importantly, the constraint ontologies and credential ontologies also contain links to other ontologies. After adding some reasoning ability to the proposed model, it will be able to locate and fetch security information automatically during access control.

## 6.2 Conclusion

This thesis presents an authorization-function-based access control (FB-RBAC) model for enterprise systems. The model extends the well-known RBAC96 [SAND96] model and adds some unique features, like Authorization-function-based access control and constraint-based fine-grained access control, to the original model. The basic unit of access control has changed from data items in the traditional access control models to Authorization-functions (A-functions) in FB-RBAC, since an A-function is a more efficient notion to represent permissions in authorization specifications. The concept of constraint brought up in RBAC96 is also extended to describe a variety of access control policies in which some are used to provide fine-grained access control. FB-RBAC is independent of any system architecture or implementation platform. It can be applied to any type of enterprise system and is designed to provide access control at the application level. This model should be implemented at the middle-tier of an enterprise system to complement the existing security services at the back-end. FB-RBAC provides an overall security protection to integrate enterprise systems without interfering with existing security infrastructure in the participating applications.

FB-RBAC is then extended to support Web service applications (enterprise applications using Web services). Web services can also be used as communication channels for data exchange across corporate boundaries. However, to address the problem of user heterogeneity and dynamicity in the Web environment, some new features like credential-based access control and dynamic role assignment were added to FB-RBAC to form Extended RBAC (ERBAC) model. Credential-based access control is different from identity-based control employed by most of the traditional models using credentials, provided by trusted third parties and secured by encryption techniques like Public-key Infrastructure (PKI). In ERBAC, role assignment was made dynamic according to user credentials, environmental conditions and a set of access control policies. All of these aspects change frequently over time, hence making access control more flexible and dynamic. On the other hand, dynamic role assignment also made security management more simple and efficient since it avoids the complexity of manually assigning huge number of access rights to a large number of subjects. The ERBAC model has filled a gap in the area of Web service security as most of the Web security services were aimed at network level and designed to ensure communication security, there were no security services at the application level.

The proposed ERBAC model has been further extended to support Semantic Web services. The Semantic Web was designed to be machine-interpretable and specified by ontologies. This thesis applied ontology techniques to the ERBAC model to form an ontology-based ERBAC model. This model provides ontological specifications for all features of the ERBAC model. Since the Semantic Web is also based on ontologies, this specification framework would allow security services to be integrated with the Semantic Web by joining up security ontologies with ontologies designed for Semantic Web services such OWL-S. In this unified framework, it will

be possible to query and discover access control information required for semantic Web security from a network of ontologies. This is an important requirement for realizing automatic access control decision making. However, both research in semantic Web and in ontology-based security services are still at their early stages. The proposed model is only the first step towards a complete ontology-based access control model. It is envisaged that the full version of the model will provide significant contributions to the research of access control.

# Bibliography

[ACEV97] Acevedo, M. T., Fillingham, D. and Nicolettos, J. L., "Enterprise Security Applications of Partition Rule Based Access Control (PRBAC)", *In Proceedings of the 6th Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises*, 1997, pp. 285-292.

[AGAR04a] S. Agarwal, B. Sprick, S. Wortmann: "Credential Based Access Control for Semantic Web Services", *In Proceedings of 2004 American Association for Artificial Intelligence Spring Symposium Series*, Stanford, California, USA, March 2004.

[AGAR04b] Agarwal, S. and Sprick, B., "Access Control for Semantic Web Services", *In Proceedings of the International Conference on Web Services (ICWS) 2004*, San Diego, USA, July 2004.

[AHN00] Ahn, G. J. and Sandhu, R., "Role-Based Authorization Constraints Specification", *In Proceedings of ACM Transactions on Information and System Security*, Vol. 3, No. 4, November 2000, pp. 207-226.

[ALKA02] Al-Kahtani, M. and Sandhu, R., "A model for Attribute-based User-Role Assignment", *In Proceedings of the 18th Annual Computer Security Applications Conference*, Las Vegas, December 2002.

[AMOR94] Amoroso, E., "Fundamentals of Computer Security Technology", Published by Prentice Hall, 1994.

[ANDE04] Anderson, A. as Editor, "XACML Profile for Role-based Access Control", Committee Draft 01, http://docs.oasis-open.org/xacml/cd-xacml-rbac-profile-01.pdf, 13 February, 2004, last visited on August 2006.

[BARK97] Barkley, J. F., Cincotta, A. V., Ferraiolo, D. F., Gavrilla, S. and Kuhn, D. R., "Role-based Access Control for the World Wide Web", Technical Report, National Institute of Standards and Technology Gaithersburg, Maryland, 1997.

[BARK03] Barker, S. and Stuckey, P., "Flexible Access Control Policy Specification with Constraint Logic Programming", *In Proceedings of ACM Transactions on Information and System Security*, Vol. 6, No. 4, November 2003, pp. 501-546.

[BARR03] Douglas, K. B., "Web Services and Service-Oriented Architectures", published by Morgan Kaufmann, April 2003.

[BEZN98] Beznosov, K., Deng, Y., Blakley, B., Burt, C. and Barkley, J., "A Resource Access Decision Service for CORBA-based Distributed Systems", *In Proceedings of the 15th Annual Computer Security Applications Conference*, 1998.

[BERT99] Bertino, E. and Samarati, P., "A Flexible Authorization Mechanism for Relational Data Management Systems", *In Proceedings of ACM Transactions on Information Systems (TOIS)*, Vol. 17, No. 2, April 1999, pp. 101-140.

[BERT01a] Bertino, E., Castano, S. and Ferrari, E., "On specifying security policies for web documents with an XML-based language", *In Proceedings of the sixth ACM symposium on Access control models and technologies*, Chantilly, Virginia, United States, 2001, pp. 57-65.

[BERT01b] Bertino, E., Castano, S. and Ferrari, E., "Securing XML Documents with Author-X", *In Proceedings of IEEE Internet Computing*, Vol. 5, No. 3, May 2001, pp. 21-31.

[BERT01c] Bertino, E., Bonatti, P. A. and Ferrari, E., "TRBAC: A temporal role-based access control model", *In Proceedings of ACM Transactions on Information and System Security (TISSEC)*, Vol. 4, No. 3, August 2001, pp. 191-233.

[BERT05] Bertino, E. and Ghafoor, A., "A Generalized Temporal Role-Based Access Control Model", *In Proceedings of IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No. 1, January 2005, pp. 4-23.

[BIDA98] Bidan, C., and Issarny, V., "Dealing with Multi-policy Security in Large Open Distributed Systems", *In Proceedings of 5th European Symposium on Research in Computer Security* (ESORICS 98), 1998.

[BISH05] Bishop, M., "Introduction to Computer Security", Published by Addison-Wesley, 2005, pp. 27-35.

[BHAT03a] Bhatti, R., Joshi, J. B. D., Bertino, E., and Ghafoor, A., "Access Control in Dynamic XML-Based Web Services with X-RBAC", *In proceedings of the First International Conference on Web Services ( ICWS)*, June 2003.

[BHAT03b] Bhatti, R., "X-GTRBAC: An XML-based Policy Specification Framework and Architecture for Enterprise-Wide Access Control", *Master Thesis*, Purdue University, May 2003.

[BHAT04a] Bhatti, R., Joshi, J. B. D., Bertino, E., and Ghafoor, A., "XML-Based Specification for Web Services Document Security", *In Proceedings of IEEE Computer*, Vol. 37, No. 4, April 2004, pp. 41-49.

[BHAT04b] Bhatti, R., Bertino, E., and Ghafoor, A., "A Trust-based Context-Aware Access Control Model for Web Services", *Distributed and Parallel Databases Archive*, Vol. 18, No. 1, July 2005, pp. 83-105.

[BHAT05] Bhatti, R., Ghafoor, A., Bertino, E. and Joshi, J. B. D., "X-GTRBAC: an XML-based policy specification framework and architecture for enterprise-wide access control", *In Proceedings of ACM Transactions on Information and System Security (TISSEC)*, Vol. 8, No. 2, May 2005.

[BIDA98] Bidan, C., and Issarny, V., "Dealing with Multi-policy Security in Large Open Distributed Systems", *In Proceedings of 5th European Symposium on Research in Computer Security (ESORICS 98)*, 1998.

[BLAZ00] Blaze, M., Feigenbaum, J., Joannidis, J. and Keromytis, A.D., "The Role of Trust Management in Distributed System Security", *Chapters in "Secure Internet*

*Programming: Security Issues for Mobile and Distributed Objects"*, Springer-Verlag Inc., New York, NY, USA, 2000.

[BPEL03] "Business Process Execution Language for Web Services", Version 1.1, 31 March 2003, http://xml.coverpages.org/WS-BPELv11-20030331.pdf, last visited on December, 2005.

[BREW89] Brewer, D.F. and Nash, M.J., "The Chinese Wall Security Policy", *In Proceedings of IEEE Symposium on Security and Privacy*, 1989, p. 206.

[CAST94] Castano, S., Fugini, M. G., Martella, G., and Samarati, P., "Database Security", ACM Press, ADDISON-WESLEY, 1994, pp. 39-45.

[CHAN03] Chandramouli, R., "Specification and Validation of Enterprise Access Control Data for Conformance to Model and Policy Constraints", *In Proceedings of 7th World Multi-conference on Systemics, Cybernetics and Informatics (SCI)*, 2003.

[CHEN04] Chen, H., Perich, F., Finin, T. and Joshi, A., "SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications", *In Proceedings of International Conference on Mobile and Ubiquitous Systems: Networking and Services*, August 2004.

[CLAR87] Clark, D. and Wilson, D., "A Comparison of Commercial and Military Computer Security Policies", *In Proceedings of IEEE Symposium on Security and Privacy,* 1987, pp.184-194.

[CLEM05] Clemente, F. J. G., Perez, G. M., Blaya, J. A. B. and Skarmeta, A. F. G., "Representing Security Policies in Web Information Systems", *In Proceedings of 14th International World Wide Web Conference*, Chiba, Japan, May 2005.

[COET03] Coetzee, M. and Eloff, J. H. P., "Virtual Enterprise Access Control Requirements", ACM International Conference Proceeding Series; Vol. 47, *In Proceedings of the 2003 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement through Technology*, 2003, pp. 285-294.

[COVI00] Covington, M. J., Moyer, M. J., and Ahamad, M., "Generalized Role-Based Access Control for Securing Future Applications", *In Proceedings of the National Information Systems Security Conference (NISSC)*, October 2000.

[DAML-S] "DAML Services", http://www.daml.org/services/owl-s/, last visited on October 2005.

[DENK03] Denker, G., Kagal, L., Finin, T., Sycara, K. and Paoucci, M., "Security for DAML Web Services: Annotation and Matchmaking", *In Proceedings of 2nd International Semantic Web Conference (ISWC2003)*, Sanibel Island, Florida, USA, September 2003.

[EDEL94] Edelstein, H., "Unraveling Client/Server Architecture", *In Proceedings of Database Management System (DBMS 7), Issue 5,* May 1994.

[ERL04] Erl, T., "Service-Oriented Architecture – A Field Guide to Integrating XML and Web Services", Published by Pearson Education, Inc., 2004, pp. 21, 31-36, 50-52.

[ESSM99] Essmayr W., Wagner R. R., Kapsammer E., Tjoa A. M., "Meta-Data for Enterprise-Wide Security Administration", *Technical Report, Austria FWF Project Number P12314-TEC*, 1999.

[EVER00] Evered, M., "A Two-level Architecture for Semantic Protection of Persistent Distribute Objects", *In Proceedings of International Conference on Software Methods and Tools (SMT'00)*, Wollongong, Australia, 2000, p. 149.

[EVER02] Evered, M., "Opsis: A Distributed Object Architecture Based on Bracket Capabilities", *In Proceedings of 40th International Conference on Technology of Objected-Oriented Languages and Systems (TOOLS Pacific 2002)*, Sydney Australia, 2002.

[FERR92] Ferraiolo, D., and Kuhn, R., "Role-based Access Control", *In Proceedings of 15th NIST-NCSC National Computer Security Conference,* 1992.

[FERR93] Ferraiolo, D., Gilbert, D. and Lynch, N., "An Examination of federal and commercial access control policy needs", *In Proceedings of 16th NIST-NCSC National Computer Security Conference*, 1993.

[FRAN03] Frankel, D. S., "Model Driven Architecture: Applying MDA to Enterprise Computing", Published by Wiley Publishing, Inc., Indianapolis, Indiana, 2003, pp. 9-25.

[GRIM04] Grimm, S., Lamparter, S., Abecker, A., Agarwal, S. and Eberhart, A., "Ontology Based Specification of Web Service Policies", *In Proceedings of Semantic Web Services and Dynamic Networks (INFORMATIK),* September 2004.

[GRUB] Gruber, T. R., "What is an Ontology?" http://www-ksl.stanford.edu/kst/what-is-an-ontology.html, last visited on December 2005.

[HALE99] Hale, J., Galiasso, P., Papa, M., and Shenoi, S., "Security Policy Coordination for Heterogeneous Information Systems", *In Proceedings of Annual Computer Security Applications Conference, Phoenix,* Arizona, USA, 1999, pp. 219-228.

[HARR76] Harrrison, M. A., Ruzzo, M. L., and Ullman, J. D., "Protection in operating systems", *In Proceedings of Communication of ACM*, August 1976, pp. 461-471.

[HERZ00] Herzberg, A., Mass, Y. and Mihaeli, J., "Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers", *In Proceedings of 2000 IEEE Symposium on Security and Privacy*, 2000.

[IDRS94] Idris, N.B., Gray, W. A. and Qutashat, M. A., "Integration of Secrecy Features in a Federated Database Environment", *In Proceedings of Database Security VII,* 1994.

[INDR04] Indrakanti, S., Varadharajan, V. and Hitchens, M., "Authorization Service for Web Services and its Implementation", *In Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, 2004.

[JAEG99] Jaeger, T., "On the Increasing Importance of Constraints", *In: Proceedings of the 4th ACM Workshop on Role-Based Access Control,* New York, 1999.

[JOSH01] Joshi, J. B. D., Aref, W. G., Ghafoor, A. and Spafford, E. H., "Security Models for Web-based Applications", *In Proceedings Communications of the ACM*, Vol. 44, No. 2, February 2001.

[KAGA03] Kagal, L., Finin, T., and Johshi, A., "A Policy Langauge for Pervasive Computing Environment." *Policy 2003: Workshop on Policies for Distributed Systems and Networks*, Springer-Verlag, 2003.

[LIU04] Liu, P. and Chen, Z., "An Extended RBAC Model for Web Services in Business Process", *In proceedings of IEEE International Conference on E-Commerce Technology for Dynamic E-Business (CEC-East'04)*, 2004.

[LOSA02] Losavio, F., Ortega, D. and Perez, M., "Modeling EAI", *In Proceedings of Computer Science Society 12th International Conference of the Chilean*, 2002.

[MCGO03] McGovern, J., Ambler, S. W., Stevens, M. E., Linn, J., Sharan, V. and Jo, E. K., "A Practical Guide to Enterprise Architecture", by Pearson Education, Inc., Publishing as Prentice Hall Professional Technical Reference, 2004.

[NYAN95] Nyanchama, M. and Osborn, S., "Access Rights Administration in Role-based Security Systems", *In Bishop, J., Morgernsterm, M. and Landwehr, C., editors, Database Security VIII: Status and Prospects*, 1995.

[OWLa] "OWL Web Ontology Language Overview", http://www.w3.org/TR/owl-features/, last visited on January 2006.

[OWLb] "OWL Web Ontology Language Guide", http://www.w3.org/TR/owl-guide/, last visited on January 2006.

[OWL-S] "OWL-S: Semantic Markup for Web Services", http://www.daml.org/services/owl-s/1.1/overview/, last visited on January 2006.

[PAYT00] Payton, J., Gamble, R., Kimsen, and S.,Davis, L. "The Opportunity for Formal Models of Integration", *In Proceedings of 2nd International Conference on Information Reuse and Integration*, Honolulu, Hawaii, 2000.

[PERK03] Perks, C. and Beveridge, T., "Guide to Enterprise IT Architecture", Springer-Verlag New York, Inc., 2003, pp. 12-20.

[QIN03] Qin, L. and Atluri, V., "Concept-level Access Control for the Semantic Web", *In Proceedings of ACM Workshop on XML Security*, Fairfax, VA, USA, October 2003.

[RABI91] Rabitti, F., Bertino, E., Kim, W. and Woelk, D., "A Model of Authorization for Next-Generation Database Systems", *In Proceedings of ACM Transactions on Database Systems*, Vol. 16, No. 1, March 1991, pp. 88-131.

[RICH04] Richardson, C. W., Avondolio, D., Vitale, J., Len, P. and Smith, K. T., "Portal Development with Open Source Tools", Published by Wiley Publishing, Inc., Indianapolis, 2004.

[SAND] Sandpiper Software Technical Report, http://www.sandsoft.com/technology.html, last visited on December 2005.

[SAND94] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E., "Role-Based Access Control: A Multi-Dimensional View", *In Proceedings of 10th Annual Computer Security Applications Conference*, 1994.

[SAND96] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E., "Role Based access control models", *In Proceedings of IEEE Computer*, vol. 2, February 1996.

[SEMA] "Semantic Web", http://www.w3.org/2001/sw/, last visited on January 2006.

[SWRL04] "SWRL: A Semantic Web Rule Language Combining OWL and RuleML", *version 6*, http://www.w3.org/Submission/SWRL/, last visited on December 2005.

[SWSA] "SWSA Use Cases", http://www.daml.org/services/use-cases/architecture/, last visited on October 2005.

[THOM93] Thomas, R. K. and Sandhu, R. S., "Towards a Task-based paradigm for Flexible and Adaptable Access Control in Distributed Applications", *In Proceedings of the 1992-1993 Workshop on New Security Paradigms*, Little Compton, Rhode Island, United States, 1993, pp. 138-142.

[USZO03] Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., et al. "KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction, and Enforcement." *Policy 2003: Workshop on Policies for Distributed Systems and Networks*, Springer-Verlag, 2003.

[VIME96] Vimercati, S. D. C., and Samarati, P., "Access Control in Federated System", *ACM New Security Paradigm Workshop 96*, 1996.

[WANG04] Wang, X. F. and Li, Y. J., "Formal definition and implementation of business-oriented SoD access control policy", *In Proceedings of Information Management & Computer Security*, Vol. 12, No. 5, May 2004, pp. 379-388.

[WSAC] "Web Service Activity", http://www.w3.org/2002/ws/, last visited on January 2006.

[WSPO] "Web Services Policy Framework (WS-Policy)", http://www-128.ibm.com/developerworks/library/specification/ws-polfram/, last visited on January 2006.

[YANG03] Yang, C. G. and Zhang, C. N., "An XML-based administration method on role-based access control in the enterprise environment", *In Proceedings of Information Management & Computer Security*, Vol.11, No. 5, 2003, pp. 249-257.

[YAO01] Yao, W., Moody, K. and Bacon, J., "A Model of OASIS Role-based Access Control and its Support for Active Security", *In Proceedings of ACM Transactions on Information and System Security (TISSEC),* Vol. 5, No. 4 November 2002, pp. 492-540.

[ZHON01] Zhong, Y., Bhargava, B. and Mahoui, M., "Trustworthiness Based Authorization on WWW", *In Proceedings of IEEE Workshop on Security in Distributed Data warehousing*, New Oleans, USA, October 2001.