uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Jun Hu

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.C.S. (Master of Computer Science)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Secure Collaboration Environments in a Service Oriented Architecture

TITRE DE LA THÈSE / TITLE OF THESIS

Liam Peyton

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

Abdulmotaleb El Saddik

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Daniel Amyot

Tony White

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

# Secure Collaboration Environments in a Service Oriented Architecture

**Jun Hu**

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

in partial fulfillment of the requirements

for the degree of Master of Computer Science

Ottawa-Carleton Institute for Computer Science

School of Information Technology and Engineering

University of Ottawa

*Your file Votre référence*
*ISBN: 978-0-494-18426-4*
*Our file Notre référence*
*ISBN: 978-0-494-18426-4*

# Abstract

Reliable, secure and verifiable data sharing and exchange over the Internet has become a very important need, especially in the face of privacy regulations that have been enacted by governments. Sensitive data must be well managed when there is collaboration between individuals and organizations. This thesis investigates some collaboration models and systems; identifies several security and privacy issues in sharing sensitive data within a collaborative system; and proposes a Web-based and Web service-oriented approach to enable collaboration over the Internet while protecting and enabling the sharing of sensitive data. The proposed solution takes advantage of both service oriented architecture (SOA) and Web service, combining multiple access control strategies to share and protect sensitive data within a secure, flexible and scalable collaborative environment.

The proposed framework is illustrated in the context of a collaborative online medical consultation system with sensitive data sharing. The collaborative online medical consultation system is developed with SOA, which integrated several collaboration Web services and security Web services.

# Acknowledgements

My sincere gratitude goes to my supervisors Dr. Liam Peyton and Dr. Abdulmotaleb El Saddik for their patience, guidance and encouragement throughout my thesis research. I would also like to thank Dr. Daniel Amyot and Dr. Tony White, for their valuable comments and suggestions in writing this thesis.

I owe a lot of thanks to my family for their encouragement and support throughout my graduate studies. The thesis would not have been possible without their help. Special thanks go to NSERC, who gave me Canada Graduate Scholarship. Finally I would like to thank all my friends who helped me either with suggestions or with positive affirmation.

# Table of Contents

# Table of Figures

# List of Tables

# Glossary of Acronyms

**ACL**: Access Control Lists. ACLs are lists that define access rights.

**BPEL**: Business Process Execution Language. BPEL is an XML-based language for the specification of business processes as an orchestration of web services.

**CAD**: Computer Aided Design. It is an automated system for the design, drafting, and display of graphically oriented information.

**CSCW**: Computer-Supported Cooperative Work. CSCW addresses how collaborative activities can be supported by means of computer systems.

**CoMed**: Cooperation on Medicine. This is a desktop conferencing application, which allows interactive real-time cooperation among several medical experts.

**DRM**: Digital rights management. It is a technology used to protect digital products from copyright infringement.

**JASMINE**: Java Application Sharing in Multi-user INteractive Environment. It enables users to share Java applications or applets in real time.

**HTML** : HyperText Markup Language. It is used for creating World Wide Web pages.

**MVC**: Model-View-Controller. MVC is a software architecture that separates an application's data model, user interface, and control logic into three distinct components.

**OASIS**: Organization for the Advancement of Structured Information Standards. OASIS is a global consortium that drives the development of e-business and web service standards.

**OWL-S**: Ontology Web Language for Services. OWL-S is an ontology that enables automatic service discovery, invocation, composition and execution monitoring.

**P2P**: Peer to Peer. P2P architecture allows hardware or software to function on a network without the need for central servers.

**PCASSO**: Patient-Centered Access to Secure Systems Online. PCASSO represents a prototype for providing secure remote viewing of electronic health information for patients and healthcare providers using the Internet.

**PKI**: Public-Key Infrastructure. PKI is a secure method for exchanging information. It uses a public/private key, to encrypt IDs, documents, or messages.

**PIPEDA**: Personal Information Protection and Electronic Documents Act. PIPEDA is a Canadian law to protect personal information.

**RBAC**: Role-Based Access Control. In RBAC, the permissions are assigned to roles.

**SOA**: Service-Oriented Architecture. SOA is a distributed systems architecture in which functionality is provided and encapsulated in services.

**SOAP**: Simple Object Access Protocol. SOAP is an XML-based protocol that defines a vocabulary for electronic message exchange.

**SSL**: Secure Socket Layer. SSL is a commonly used protocol for managing the security of a message transmission over the Internet.

**TMAC**: Team-based Access Control. It is an approach to applying role-based access control in collaborative environments.

**UDDI**: Universal Description, Discovery, and Integration. It facilitates describing and discovering Web services through the registration of service information.

**VPN**: Virtual Private Network. VPN is a private data network that makes use of the public telecommunication infrastructure.

**XACML**: EXtensible Access Control Markup Language. It is a standard to protect content in enterprise data exchange.

**XML**: EXtensible Markup Language. It is based on tags and designed to easily represent data in a very portable manner

**WebOnCOLL**: Web On Collaboration. It is a web-based medical collaboration environment designed in the context of the regional healthcare network of Crete.

**WSDL**: Web Services Description Language. It is an XML format for describing Web services, including who operates it, where it is located, and how it is accessed.

**WS-BPEL**: Web Services Business Process Execution Language. It is an XML grammar defining and standardizing structures necessary for web services orchestration.

# Chapter 1 Introduction

Collaboration environments allow enterprises to realize a number of competitive advantages by using their existing computers and network infrastructure for group and personal collaboration. In an open collaborative environment, users may come from different organizations and participate from a diverse set of locations. They may collaboratively require heterogeneous access of dynamic and static resources. Secure access to sensitive and confidential resources on the Internet has emerged as a major challenge for businesses. An example application area for collaboration is telemedicine. Collaboration support for telemedicine would allow a family doctor in a remote community to consult with a specialist from a big city hospital for a particular patient, without the patient or the doctor or the specialist having to physically travel from one location to another.

Organizations have developed identity and access management strategies that define local security domains to provide a specific set of users with access to a defined set of resources. But to support collaboration environments, this needs to be expanded to dynamically provide access to remote autonomous business units and business partners while ensuring that sensitive data is recognized as such, protected, and only shared with appropriate safeguards in place.

Balancing collaboration with security and privacy is difficult since the interaction in collaborative systems is to make resources available to all who need them, whereas information security and privacy attempts to ensure they are only available to those with proper authorization. Moreover, in a collaborative environment where unpredictable users interact with the system in unexpected manners, protection of information and resources entails addressing several requirements not raised by a traditional single-user environment.

There are several concerns about sharing sensitive data in collaborative environments. The typical security and privacy requirements for a collaborative environment include authentication, authorization, data confidentiality, integrity and non-repudiation.

A mechanism is needed for shared access to resources, especially for extending or refusing permissions to collaborators. Some of the existing solutions can be adopted, but they are still not complete enough due to the complex nature of collaboration systems. Access controls, firewalls, virtual private networks (VPNs), public-key infrastructure (PKI) and secure socket layer (SSL) provide authentication and secure transmission, but they do not control information use. For the purposes of this thesis, we will assume that all suitable mechanisms are in place for providing authentication and secure transmission and focus on information use.

Digital rights management (DRM) solutions control the use of information, but do not enable collaboration. Sensitive information must be protected, tracked, monitored and ultimately controlled. It is a significant challenge, which has not been solved in industry, to ensure the security and privacy of sensitive information while enabling collaboration that dynamically shares sensitive information.

## 1.1 Motivation and Objectives

Collaboration support for telemedicine is an application area of particular interest to our research. Collaborative online medical consultation systems integrate computer-based collaboration tools to achieve synchronous or asynchronous interaction among doctors. A rich set of resources is shared among participants: documents, database access and URLs that are from different domains or organizations over Internet. Security and privacy concerns raise the issue of how to share and control information to provide the right mixes of capabilities for organizations or collaborators to share sensitive and valuable information with confidence. Especially, how to extend permission to new collaborators dynamically when a user try to share information with a collaborator, when that collaborator has not yet been granted access to the information.

There are several collaborative telemedicine applications already in the market, but most of them are limited to intra-hospital communication through LANs. The bigger challenge is that the system includes not only collaboration within a hospital but also extends collaboration to remote experts and health-care providers outside the hospital. The rapid worldwide Internet and Web deployment is the enabler of a new generation of e-healthcare applications, but the provision of a security architecture that can ensure the privacy and security of sensitive healthcare data is still an open question.

Besides addressing issues around information theft, privacy legislation is the primary driver for organizations to improve content security and their means of collaboration. In the beginning of 2004, the government enacted Canada's Personal Information Protection and Electronic Documents Act (PIPEDA) [PIPEDA2000], which included mandates for information privacy and penalties for improper storage or access to confidential information. While increased security and privacy is a requirement, it should not undermine collaboration, which is one of the essential elements for achieving business objectives.

The objectives of this thesis are to enhance the understanding of these issues, then present an approach that enforces security and privacy with the support of collaboration. Specially, the following questions and issues are addressed in this thesis:

- What system architecture can enable collaboration over the Internet with flexibility, security and scalability?
- What access control mechanism can be flexible enough for different contexts, especially for extending or refusing permissions to collaborators?
- How to make collaborators' activities accountable?

## 1.2 Thesis Contributions

This thesis presents a framework for defining and evaluating a secure collaboration environment in a service-oriented architecture (SOA). More specifically, the thesis makes the following achievements:

1. Categorizes the requirements of a secure collaborative environment which must be met by any solution for sharing sensitive data;

2. Designs a framework to enable collaboration and sharing of sensitive data over the Internet with flexibility, security and scalability in a service-oriented architecture;

3. Defines flexible control of the collaborators' interactions with local or remote contexts that have ownership of sensitive data by allowing authorization to be evaluated by a Web service;

4. Develops a collaboration control model for sharing sensitive data by combining team-, role- and policy- based access control models for dynamic access to resources and support for delegated self-administration in collaborative environments;

5. Evaluates and illustrates the framework with a collaborative online medical consultation system case study.

6. Develops a prototype for online collaborative medical consultation system. The programming contributions include:

    - Development of Collaboration Manager to manage collaboration session

    - Development of Chat Web service, Shared Whiteboard Web service, FileSharing Web service, Auditing Web service and Authorization Web service to support Collaboration Manager.

    - Development of Access control mechanism to control data use; including XACML policies, dynamic role change and delegation of authority in a collaboration session

## 1.3 Scopes and Limitation of the Thesis

Collaborative environments are the result of interactions of various disciplines. A security and privacy system for collaboration has to be based on different requirements and take

into account specific limitations of collaboration. The following assumptions are made in this thesis:

- The collaborators have already been authenticated in some valid way when they interact with the system. The communication channels over which the collaboration environment transmits information have been secured in typical industry fashion (SSL, VPN, etc.) Our thesis does not discuss or address how to deal with malicious attacks on the system or attempts to break into the system;
- The network used in the system is fully reliable and ideal. Our thesis does not attempt to deal with network unreliability or other quality of service issues;
- If the shared data is sensitive, the owners of the data have given their consent to access the data. In particular, in the case of personal information, limited consent has been provided by the persons in question to use the data and share the data with others. (But there is a requirement on collaborators and the collaboration environments, to comply with the use limitations defined in that consent, and establish an audit trail to demonstrate compliance). Our thesis does not discuss how to get consent from the owners. A collaborative approval process for accessing sensitive data [Peyton2005] has been proposed to solve this problem.

The service or component used in the collaboration environment can be deployed within an organization or across organizations. Among the many issues in a secure collaborative environment, this thesis addresses the following aspects:

- Architecture of the collaborative system;
- Access control for sharing sensitive data;

## 1.4 Organization of Thesis

Chapter 2 outlines background information and summarizes related work on collaboration environments.

Chapter 3 describes the methodology used in this thesis.

Chapter 4 proposes a Web-based and Web service-oriented approach that addresses issues of collaboration, security and privacy in sharing sensitive data within a collaborative environment.

Chapter 5 presents a case study in healthcare to illustrate how to apply the approach to a collaborative environment with shared sensitive data.

Chapter 6 evaluates the proposed solution and analyzes the result.

Chapter 7 summarizes the conclusions and suggestions for future study.

# Chapter 2 Background and Related Works

This chapter explains the basic concept of a secure collaborative environment. As well, related technologies used in this thesis are described. Finally, we discuss some related works relevant to this thesis.

## 2.1 Secure Collaborative Environments

A secure collaboration environment actually includes two concepts. One is "secure environment"; the other is "collaborative environment". This section briefly explains collaborative environment, security and privacy.

### 2.1.1 Concept of "Collaborative Environment"

This thesis defines a "collaborative environment" as a system that supports the interaction among a number of people to achieve a single goal or set of goals. Collaborative applications may be synchronous or asynchronous. They may be designed for cooperative or competitive groups of users; they may be designed around replicated or distributed architectures. The important defining feature of these applications is that they are multi-user and support the process of collaboration among their participants. Example systems include a wide range of applications such as instance messaging, audio/video conferencing, collaborative document sharing/editing, distance learning, workflow management systems, and so on.

The Web environment offers significant benefits to computer supported collaborative work (CSCW). Beside its global scope, high availability and rich content, it owns multi-platform clients, and ability to host executable. All these properties make the Web a powerful platform for building collaborative applications. In fact, many of the new industrial groupware systems such as Lotus SameTime [SameTime2005], Live Meeting (PlaceWare) [LiveMeeting2005] are Web based. As the Web-based collaborative systems gain industrial acceptance, they will have to undergo evolution to make them secure and integrate them with existing Internet security infrastructure.

Collaboration over Internet is the clear wave of the future, provided that such collaboration is reliable, dependable, and authentic. The Internet provides a powerful, standardized, worldwide, ubiquitous communications mechanism whose benefits are impossible to ignore [Weaver2004].

### 2.1.2 Security and Privacy

Security is defined as something that provides "safety, freedom from danger or anxiety" in the Oxford dictionary. When applied to a computer system, security protects resources by preventing access from an unauthorized entity. Security can be implemented at many levels ranging from low-level hardware to high-level software.

Security issues in collaborative environments mainly include authentication, authorization, data confidentiality and integrity and non-repudiation. Authentication is the process of determining whether someone or something is who or what it is declared to be. Authorization is the process of determining which permissions a person or system is supposed to have. It implies that a given user, either authenticated or not, has the permission to perform certain operations on certain resources. Data confidentiality is a set of services that provide protection against unauthorized disclosure. Privacy and protection of sensitive information over insecure public networks means assuring content confidentiality and message flow confidentiality. Data integrity implies that that the data is protected from unauthorized, unanticipated, or unintentional modification. The level of security in any system is equal to the security of the weakest link in the system. As a result, all security services are of equal importance.

For term "privacy", Marc Langheinrich divides it into four categories: territorial privacy, communication privacy, bodily privacy, and information privacy [Langheinrich2001]. Our work focuses on protecting information privacy, defined as the ability to control one's own information, by incorporating the notion of privacy into access control mechanisms.

Private information can be further divided into two categories: static information and dynamic information. Static information is the information that does not change very

often and does not require any deduction to understand the information. Credit card numbers and social security numbers are examples of static information. Dynamic information is information that changes often and requires some forms of analysis. Internet users' behavioral profiles created by monitoring their activities in the website represent an example of dynamic information. Some personal information is very sensitive, such as a person's medical records.

In the beginning of 2004, the government enacted the Privacy Act to regulate government processing of information and the Personal Information Protection and Electronic Documents Act (PIPEDA) to regulate all other processing of information. In an era in which technology increasingly facilitates the circulation and exchange of information, PIPEDA establishes "the rules to govern the collection, use and disclosure of personal information" [PIPEDA2000]. Organizations must establish policies and implement procedures to ensure the protection of privacy and compliance with the requirements of the Act. Related to these legislations, the ITR approach [Peyton2004], the HP Framework [Mont2003], and Project Liberty [Liberty2003] all address aspects of privacy protection.

### 2.1.3 Relationship Between Collaboration and Security

Collaborative systems are potentially chaotic environments [Edwards1996] where multiple users create opportunities for collaboration and provide rich and unexpected interactions between them. All collaborative systems contain information and resources with different degrees of sensitivity. The applications deployed in such systems must create, manipulate, and provide access to a variety of protected information and resources.

Until now, there was a gap between the need for collaboration and the need to protect sensitive information [Price2004]. Figure 2.1 indicates the need for secure collaboration increases with the value and sensitivity of content [Price2004].

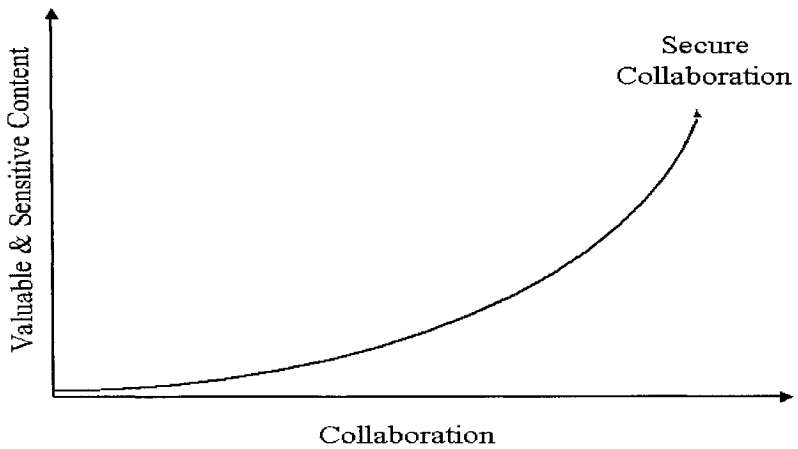


**Figure 2-1** the Need for Secure Collaboration Increases with the Value and Sensitivity of Content [Price2004]

## 2.2 Related Technologies

### 2.2.1 XML and XACML

XML [XML2004] is EXtensible Markup Language for web applications. It is based on tags and designed to easily represent data in a very portable manner.

The key message of XML is data portability. The key benefits of XML can be summarized as the following:

- It is independent of vendor and platform;
- It is easy to model data at any level of complexity;
- It is extremely extensible: defining new tags as needed;
- It is easy to validate data to check for structural correctness;
- It is independent of the media used to publish content in multiple formats:

XACML [XACML2004] is an OASIS standard that describes both a policy language and an access control decision request/response language, both encoded in XML. The policy language is used to describe general access control requirements. The request/response language form a query to ask whether or not a given action should be allowed, and interpret the result. The response always includes an answer about whether the request should be allowed using one of four values: Permit, Deny, Indeterminate or Not Applicable.

The XACML language effectively protects content from unauthorized use in enterprise data exchanges. It is written in XML and designed to expand base in global enterprise environments. It allows creating and deploying authorization policies to match its mix of assets and business use-cases, and then plug in additional policies as the business and its standards evolve. [XACML2004]

The typical setup is that someone wants to take some action on a resource. They will make a request to whatever actually protects that resource which is called a Policy Enforcement Point (PEP). The PEP will form a request based on the requester's attributes, the resource in question, the action, and other information pertaining to the request. The PEP will then send this request to a Policy Decision Point (PDP), which will look at the request, find some policy that applies to the request, and come up with a decision about whether access should be granted. That decision is returned to the PEP, which can then allow or deny access to the requester. [XACML2004]

The key benefits of XACML can be summarized as the following:

- It is standard and powerful. The standard language supports a wide variety of data types, functions, and rules about combining the results of different policies.
- It is generic. It can be used in any environment rather than providing access control for a particular environment or a specific kind of resource.
- It is distributed. A policy can be written which in turn refers to other policies kept in arbitrary locations.

### 2.2.2 Web Service

According to the W3C Web services Architecture working group [WSArch2003], a Web service is a software system identified by a URI [RFC2396], whose public interfaces and bindings are defined and described using XML. It is designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). A key element is the capability to publish, find and bind Web Services through public or private registries. The major advantage is that applications can be accessed using the following standard Web protocols:

- eXtensible Markup Language (XML) [XML2004] -- a text markup language for interchange of structured data. The reference character set is Unicode, which enables worldwide data exchange.
- Simple Object Access Protocol (SOAP) [SOAP2003] -- an XML-based protocol that defines a vocabulary for electronic message exchange. SOAP is an envelope containing a header and a body (the message).
- Web Services Description Language (WSDL) [WSDL2001] -- an XML format for describing Web services as end points that act on messages containing either documents or procedure calls. It describes the service, including who operates it, where it is located, and how it is accessed.
- Universal Description, Discovery, and Integration (UDDI) [UDDI2004] -- facilitates describing and discovering Web services through the registration of service information. The description is expressed in WSDL.

Web Services play an important role in providing an interface between end user applications and the underlying technologies. Being the base for distributed decentralized systems and growing for its flexibility, reusability and extensibility, Web services replace closed, centralized systems increasingly.

### 2.2.3 Service-Oriented Architecture (SOA)

A service-oriented architecture (SOA) defines the services of which the system is composed, describes the interactions that occur among the services to realize certain behavior, and maps the services into one or more implementations in specific technologies [WS2004]. The term SOA is a specific type of distributed system in which the agents are "services". A service is a software agent that performs some well-defined operation and can be invoked outside of the context of a larger application [WSArch2003]. It is coined to describe the overall approach of building loosely coupled distributed systems with minimal shared understanding among system components. "Services" have a network-addressable interface and communicate via standard protocols and data formats. They may be Web services or not.

SOA represents a way to achieve a vision of seamlessly composing and interoperating between services. It is particularly applicable when multiple applications running on varied technologies and platforms need to communicate with each other. Conceptually, SOA is comprised of three roles, namely, Provider, Directory (Registry), and Consumer.

The interactions between these roles involve (Figure 2.2):

- Services providers define service descriptions for services and publish them to directory;
- A consumer uses a directory and service descriptions for services they are interested in using;
- With the service description available, the consumer sends a service request to a specific provider.

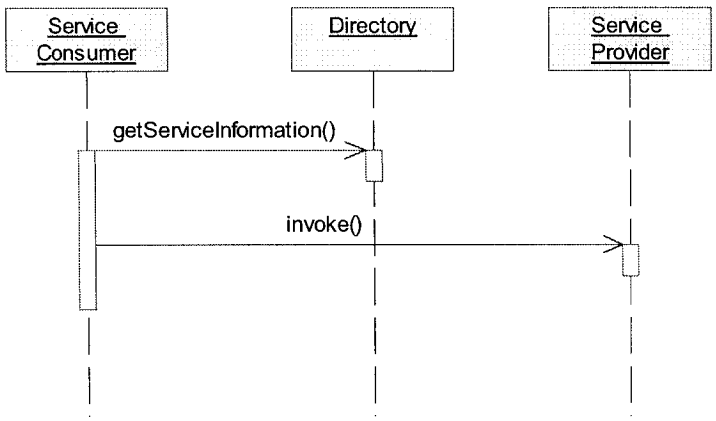


**Figure 2-2** Service Oriented Concepts: Interaction Diagram

A service-oriented architecture can facilitate the reuse of business services. It focuses on the building of more loosely coupled applications that can be assembled from existing to provide new business functionality.

A key focal point of a SOA is that aggregating components can be exposed as services so they can be accessed in a ubiquitous fashion. These services reside in a common runtime infrastructure — the application server. This allows business policies to more closely represent business process workflow. Summarizing these observations from a business perspective, SOA is a way of architecting an enterprise to provide services to either end-user applications or other services through published interfaces in a plug-and-play manner [FAMIS2005]. Services offer a great way to expose discrete business functionality and provide an excellent way to develop applications that support business policy and processes.

### 2.2.4 Web Services Composition: WS-BPEL and OWL-S

Web services composition is the task of combining and linking existing web services to create new web processes in order to add value to the collection of services. Two example approaches are:

- Industry solution such as WS-BPEL [WSBPEL2005]
- Semantic web solution such as OWL-S [OWLS2004]

WS-BPEL stands for Web Services Business Process Execution Language. It is an XML grammar defining and standardizing structures necessary for web services orchestration. In WS-BPEL everything is a service. Composition is based on pre-modeled workflow. Activities in WS-BPEL refer message exchange or intermediate result transformation. Process is the composition result. A process consists of a set of activities.

OWL-S stands for Ontology Web Language for Services. It is an ontology that enables automatic service discovery, invocation, composition and execution monitoring. Composition is based on pre- and post-conditions.

The overall structure of the ontology has three main parts:

- Service Profile – advertising and discovering services
- Process Model – giving detailed description of the service in which it is modeled as a process
- Service Grounding – binding level information of how a client can access the service, via messages.

The following table (Table 2-1) is a comparison of WS-BPEL and OWL-S [Maigre2006].

**Table 2-1 Comparison of WS-BPEL and OWL-S [Maigre2006]**

| WS-BPEL | OWL-S |
|---|---|
| Industry driven | Academically driven |
| Lacks semantics | Majority of services are described in WSDL |
| Runtime engine | Planner |

| | |
|---|---|
| Has good control over workflow at design time. Not too flexible at runtime | More flexible at runtime |
| Inputs and outputs of the service described in WSDL | Pre- and post-conditions |
| Set of choices is pre-determined | Choices are based on goals |

### 2.2.5 Web Service Security

Microsoft and IBM defined several security specifications for Web services from different aspects: WS-Policy [Apshankar2002], WS-Trust [WSTrust2004], WS-Privacy [Apshankar2002], WS-Secure Conversation [Apshankar2002], WS-Federation [WSFederation 2003], WS-Authorization [Apshankar2002]. They are built on top of the WS-Security [WSSecurity2002]. WS-Security is a message security model that provides the basis for the other security specifications.

Nevertheless, these specifications have not found their way to handle specifications of composite web services. In their paper [Charfi2005], Anis Charfi and Mira Mezini present a framework for securing BPEL compositions using WS-Security and WS-Policy. The main components of their framework are the process container implemented by a set of aspects in AO4BPEL, the security service and the deployment descriptor. AO4BPEL is an aspect-oriented extension to BPEL; they also introduce the notion of policy-based process deployment to check the compatibility of the security policies of the composition and its partners at deployment time.

The following table (Table 2-2) is summary of problems that above works considered and addressed:

**Table 2-2 Addressed Problems of Related Works for Web Service Security**

| Related works for Web service security | Addressed Problems |
|---|---|
| WS-Security, WS-Policy,<br>WS-Privacy, WS-Authorization<br>WS-Federation, WS-Trust<br>WS-Secure Conversation | 1. Message level security.<br>2. Defined to understand these threats:<br>• The SOAP message could be modified or read by hackers.<br>• A hacker could send messages to a service to carry on the processing<br>• Service theft. E.g., an unauthorized user could get a subscription based web service for free. |
| Using Aspects for Security Engineering of Web Service Compositions | 1. The specification of interactions among web services within a composition<br>2. Security issues in the context of BPEL activities. Solution provides support for signing, encrypting, preventing repudiation, and adding authentication credentials to an <invoke>, <reply>, or <receive> activity. |

## 2.3 Access Control Models for Collaboration

Although generic access control has been studied extensively in non-collaborative domains, collaborative environment has new requirements for access control. In this section, we discuss some existing access control models for collaborative environments. As part of this discussion, we present an overview of the advantages and weaknesses for each model.

### 2.3.1 Access Matrix Model and Access Control Lists (ACL)

The access matrix is a conceptual model that specifies the rights that each subject possesses for each object. The access matrix [Lampson1971] provides a useful

framework for describing resource protection in operating systems. This model defines three kinds of access control entities:

a) Protected objects: the entities or resources which can be accessed;

b) Subjects: the active entities who access objects;

c) Access rights: associate the subject with the protected objects by specifying the operations that subjects are allowed to perform on objects.

An access matrix A is used to define the protection state with rows representing subjects and columns representing objects. For example, A[aSubject, aObject] denotes the access rights a subject aSubject has over an object aObject. The access-checking rule of the model states that a request by subject aSubject for accessing object aObject is granted only if A[aSubject, aObject] contains the requisite right.

An example of an access matrix is shown in Figure 2.3 (left) [Tolone2005], where the rights R and W denote read and write, respectively. This matrix specifies that, for example, John is the owner of File 3 and can read and write that file, but has no access to File 2. Since John owns File 1, he can give Alice the R right and Bob the R and W rights, as shown in Figure 2.3 (right). John can later revoke one or more of these rights at his discretion.

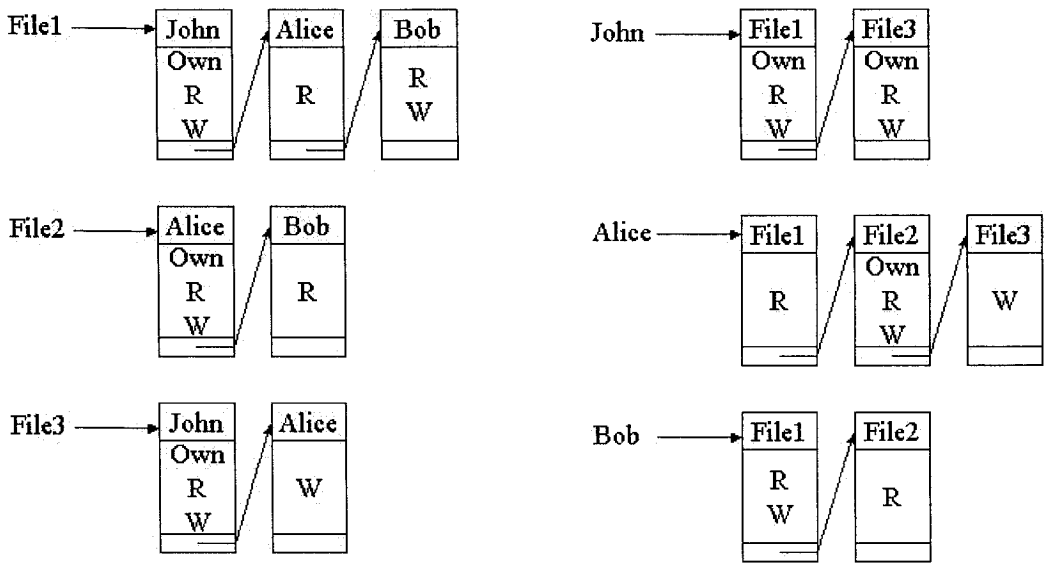


**Figure 2-3** Access control list (left) and Capability list (right) [Tolone2005]

Access Control Lists (ACL) is a common approach to implementing the access matrix. They are the lists describing the access rights an entity has on an object resource. The file system permissions mechanism used in the UNIX operating system is a typical example.

Grove Outline Editor [Ellis1991] and RTCAL [Greif1986] are examples of collaborative systems using ACLs. The Intermezzo collaborative framework [Edwards 1996] use roles in conjunction with ACLs to specify access models where data storage and replication, authentication and authorization, and awareness service. The access control model proposed by the SUITE model proposed by Shen and Dewan [Shen1992] supports multiple roles for users and identifies a set of collaborative. Sikkel [Sikkel1997] defined an access control system for the BSCW system. This model avoids the need for conflict resolution rules by introducing negative group membership rather than negative rights on access.

The advantages of ACL can be summarized as follows:

- Easy to understand and review access to an object
- Better control over propagation and revocation of rights

The weakness of ACL can be summarized as follows:

- It has no delegation mechanism. [Bonatti2004]
- It lacks a generic security mechanism so it cannot be extended with new conditions and restrictions without the need to rewrite applications. [Bonatti2004]
- It lacks the ability to support dynamic changes of access rights. [Tolone2005]
- It does not use any contexts. It does not account for the situation where access rights may be related to content, attribute of resources, or other contextual information, such as in a collaborative or organizational workflow setup. [Tolone2005]
- It has weaknesses in its size and its lack of contexts. Since ACLs use no hierarchy, the size of access rules scales with respect to the multiplication between the number of users and resources.
- It does not distinguish between permission assignment and activation.

### 2.3.2 Role-Based Access Control (RBAC)

According to the American Heritage Dictionary of the English Language, role is a "character or part played by a performer"; or "the characteristic and expected social behavior of an individual"; or "a function or position". In this thesis, a role is a set of connected behaviours, rights and obligation as conceptualized by actors in a special domain. Domain is important for defining a role. It can be treated as an attribute administratively assigned to elements in order to group instances of the same class. A role can be simple like “doctor in the General Hospital”; it can have more attributes like “doctor in the General Hospital in 2006”.

Role-based access control is presented in Sandhu et al.’s seminal paper [Sandhu1996], where the main RBAC components, including users, roles, permissions, and sessions, are systematically addressed. The essence of RBAC [Sandhu1996] is that permissions are

assigned to roles rather than to individual users. Roles are created for various job functions, and users are assigned to roles based on their qualifications and responsibilities. This way, the task of specifying user authorization is divided into two logically independent parts as illustrated in Figure 2.4 [Tolone2005]:

- One part to assign users to roles;
- One part to assign access rights for objects to roles

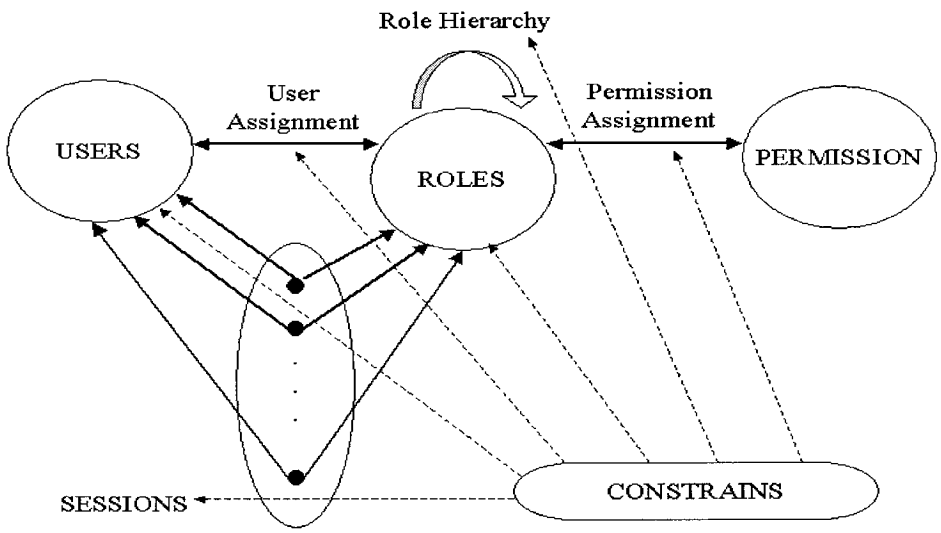


**Figure 2-4** Role-based Access Control [Tolone2005]

The early version of RBAC model [Sandhu1996] supports the notion of role activation within sessions, where session is a concept that is bound to a single user and allows the user to activate the permissions of a subset of roles to which he/she belongs. From a policy perspective, the capability within RBAC to impose constraints on user membership by assigning users to roles provides a powerful means of enforcing conflict of interest and cardinality rules for roles as they uniquely apply to a collaborative environment [Ahn2000].

Most early systems used the notion of roles to define access control groups, but they concentrated on individuals working together, for example Grief and Sarin's MPCAL system [Greif1986], Brothers's ICICLE [Brothers1990], Shen and Dewan's SUITE [Shen1992]. Jaeger and Prakash [Jaeger1996] have suggested the requirements of an RBAC system for implementing a DAC model. Such a model proposes to enable users and their applications to control access at runtime.

However RBAC model has not discussed the specification of the constraints, which are an important aspect of role-based access control and a powerful mechanism.

Some approaches have also been proposed to applying the RBAC model to applications distributed over the Internet. Kang [Kang2001] addresses how to separate inter-organizational workflow from concrete organizational-level security enforcement; however, it does not give much consideration to integrating context data with their models. Bonatti and Samarati [Bonatti2000] propose an approach for regulating service access and information disclosure on the Web. Taylor and Murty [Taylor2003] and Joshi et al. [Joshi2001] describe a security model for federated systems.

The advantages of RBAC can be summarized as follows:

- It is very effective by associating permissions with roles rather than users.
- It is reassigned easily from one role to another without modifying the underlying access structure.
- It is more scaleable than user-based security specifications
- It reduces greatly the cost and administrative overhead associated with fine-grained security administration at the level of individual users, objects, or permissions. [Tolone2005]
- It is popular for traditional and collaborative systems.

The weaknesses of RBAC are as follows:

- It is difficult to automatically coordinate effective collaboration session that require use of resources across the enterprise since each organization manages their own access rights to resources
- It does not support coalition access control since it does not provide an abstraction to capture a set of collaborative users or delegations for a collaborative environment.
- It is not enough in encompassing the overall context associated with any collaborative activity. [Tolone2005]
- It lacks flexibility and responsiveness to the environment since it does not well supported changes of the roles and users when traditional RBAC determines the set of roles in use as well as the role membership early in the lifetime of a session. [Tolone2005]
- It is insufficient to have role permissions based on object types for collaborative environments since traditional RBAC lacks the ability to specify a fine-grained control on individual users in certain roles and on individual object instances. [Tolone2005]

### 2.3.3 Team-Based Access Control (TMAC)

Roshan Thomas introduced the notion of Team-based Access Control (TMAC) as an approach to applying role-based access control in collaborative environments [Thomas1997]. A "team" is a notion as an abstraction that encapsulates a collection of users in specific roles in a specific task. The TMAC model defines two important aspects of the collaboration context:

- User context;
- Object context.

Figure 2.5 shows these components in TMAC. User context provides a way of identifying specific users playing a role on a team at any given moment, and object context identifies specific objects required for collaboration purposes.

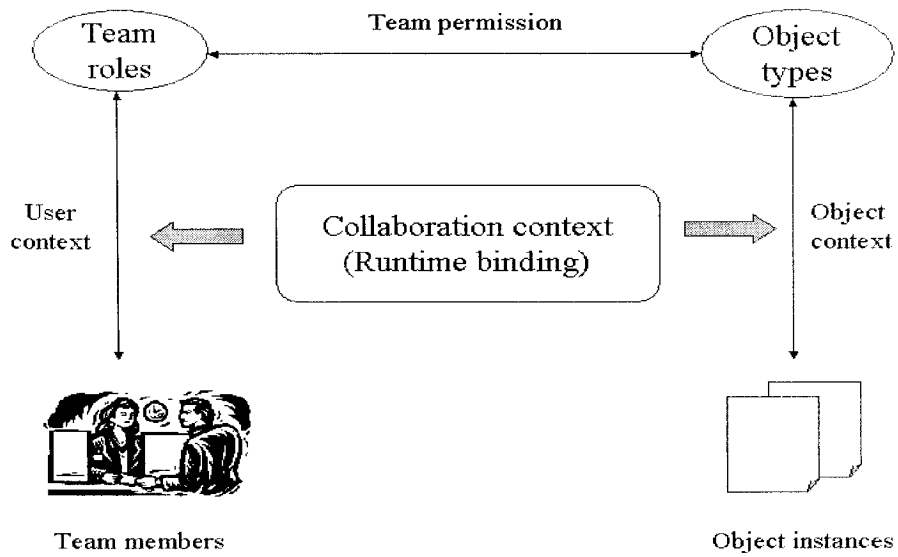


**Figure 2-5** Team-based Access Control [Thomas1997]

Context-based TMAC [CTMAC2001] is an extension to TMAC. It integrates RBAC and TMAC by incorporating context as an entity in the architecture. C-TMAC seeks to include contextual information other than user and object contexts such as time, place, and so forth. The C-TMAC model allows the use of general contextual information and gives TMAC the capability to model a rich set of security policies. Wang suggested the use of both Team- and Role-Based Access Control for Hypermedia environments [Wang1999].

TMAC allows us to formulate a security model that dovetails the team-based nature of access and work in collaborative settings. TMAC has the following advantages:

- It is able to offer the administrative and modeling advantages of RBAC. [Tolone2005]

- It provides fine-grained control over permission activation to individual users and objects. Considered as an active model of access control by distinguishing permission assignment from context-based, run-time permission activation.
- It provides just-in-time permissions
- It supports to a higher degree the principle of least privilege in comparison to passive security models.

TMAC has very unique features to support contextual information and the dynamic nature of team-based environments. However, there are several weaknesses to this model [Tolone2005]:

- It has not yet been fully developed, and it is not clear how to incorporate the team concept into a general RBAC framework. [Tolone2005]
- It lacks the self-administration of assignment relations between entities.
- It needs to reflect the multidimensional definition of rich collaborative contexts such as organizational entities, workflow tasks, groupware's environmental components, and so on. [Tolone2005]

### 2.3.4 Context-Aware Access Control

Dey, Abowd & Salber [Dey2001] define context as: "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves." Covington has extended RBAC with a well-developed notion of roles to capture security relevant context of the environment [Covington2001]. By introducing environment roles, a uniform access control framework can be used to secure context-aware applications. These roles are activated based on environment conditions at the time of request.

Context-based security has already been applied in various settings [Hulsebosch2004][ Hulsebosch2005][Bardram2003]. Compared to traditional access control systems, it has the following advantages:

- It offers access control mechanisms of various strengths that can adapt to user, group and environmental situational contexts.
- It allows for multiple points of access (e.g. location, temperature and velocity) instead of a single one for conventional systems (identity).

But it is also complexity that should be dealt with in an appropriate way while it offers flexibility.

### 2.3.5 Summary of Access Control Models

The following table (Table 2-3) is a comparison of access control models. We compare the advantages and weakness of these modes.

**Table 2-3 Comparison of Access Control Models**

| | ACL | RBAC | TMAC | Context-aware |
|---|---|---|---|---|
| **Advantages** | Easy to understand | Effectively, easily reassign the role; | Provide just-in-time permissions; | Multiple points of access |
| **Weakness** | lack the ability to support dynamic changes of access rights | no abstraction to capture a set of collaborative users | not fully developed, lack the self-administration | complexity |

## 2.4 Related Works

This section will describe some collaboration systems that will be used in comparison with our approach to collaboration and security.

### 2.4.1 JASMINE

JASMINE (Java Application Sharing in Multi-user INteractive Environment) [ElSaddik2001], developed jointly by the University of Ottawa and the Darmstadt University of Technology, enables users to share Java applications or applets in real time. These applications or applets can be pre-loaded or brought to the session live.

JASMINE is based on a Java Events Delegation Model to provide a standard mechanism for a source component to generate an event and sent it to a set of listeners. It also allows sending events to an adapter. This adapter works as an event listener for the source and as the source for listeners. This mechanism makes event handling much easier and flexible since event handling is critical in an event-sharing collaborative system. The idea behind JASMINE is to catch, distribute and reconstruct all the events happening to the GUI of an application or applet on the client side. As the result, Java applications or applets can be shared transparently. This type of collaboration allows users to interact in real time [ElSaddik2001].

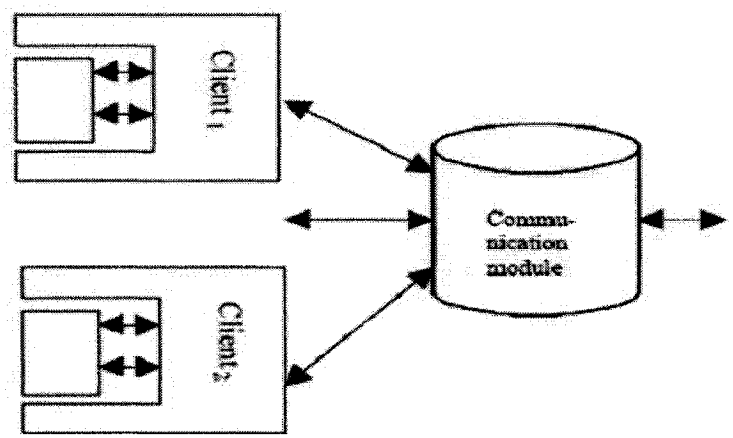


**Figure 2-6** Concept of JASMINE [ElSaddik2000]

Figure 2.6 shows the concept of JASMINE. The framework listens to all the events occurring on the GUI of the application and distributed these events to all other participants and then reconstructs events there. JASMINE uses the event-sharing

paradigm to implement the shared window system. Therefore, it comes with advantages and disadvantages of this approach. The advantage is:

- Be easy to implemented;
- Have short corresponding time;
- Consume low network bandwidth.

However, it will not function when it shares non-deterministic applications [ElSaddik2000].

JASMINE uses a client-server approach. The server is responsible for receiving events and distributing them to other users in the session. The communication module is implemented on top of TCP because the reliability is critical for a collaborative system. The advantage of a client-server approach is:

- To make the system structure clearer and simpler,

The disadvantage of a client-server approach is that:

- The scalability becomes a major problem.

As the number of connected users increases, the workload of the server increases in a non-linear way. This will definitely cause a bottleneck on the server and reduce the performance of the whole system. Therefore, JASMINE is suitable for small to medium size group collaborative sessions [ElSaddik2000].

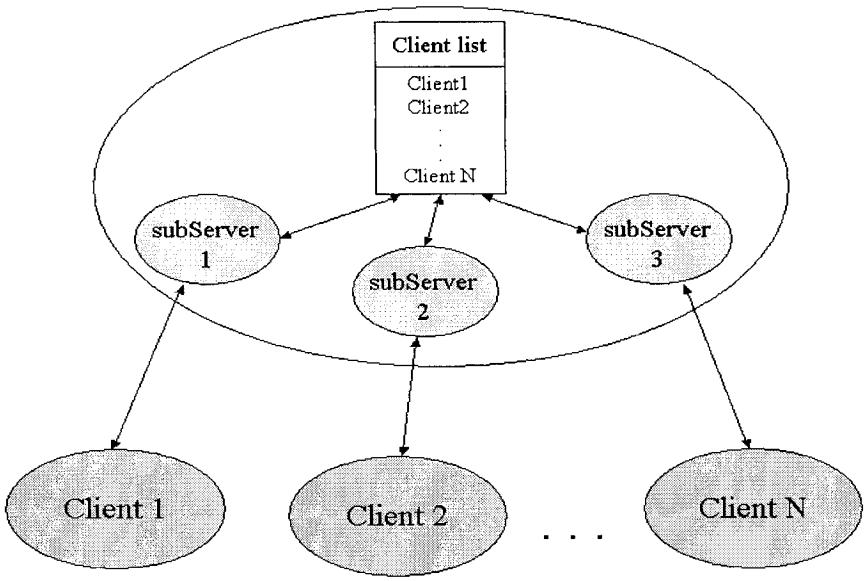


**Figure 2-7** JASMINE Server-client Architecture [ElSaddik2000]

JASMINE uses a multithreaded server as shown in the figure 2.7. When a client connects to the server, the main server starts a sub-server for this client and it handles only incoming messages from its own client. When the sub-server receives the update messages from its client, it sends these messages to all other clients in the session. Only one client can interact with the shared application at a time and other threads are simply waiting for their turns. This will not consume too many resources. Besides handling incoming events and messages, the server also provides other services, such as the session moderation and floor control [ElSaddik2000].

The JASMINE client is responsible for capturing events, sending events to the server, receiving events from the server and reconstructing events locally.

The system also provides basic moderation control and enables diverse views of the same visualization in a moderation session, in which the moderator can see more than others [ElSaddik2001]. The moderator is usually the person who calls for a collaborative session and starts the server. In JASMINE we have two types of sessions: moderated, and non-

moderated. Specifying a login name and a password for the moderator can start the server.

### 2.4.2 CoMed

CoMed [CoMed1996] [CoMed2000] is a desktop conferencing application, which allows interactive real-time cooperation among several medical experts. The application provides a shared workspace for the display, discussion and annotation of multiple radiological images. These documents are multicasted to conference participants prior to the consultation session. An agent based architecture and techniques for caching, data compression and screening of events allow its successful operation over heterogeneous, relatively slow networks [CoMed1996].

The basic features of the system include:

- Real-time consultation on medical images;
- Expandable and scaleable architecture;
- Object oriented design based on distributed autonomous cooperating agents;
- Shared and private workspace for the display, discussion and annotation of multiple radiological images;
- Data replication, and event compression and screening of events to allow successful operation over heterogeneous, relatively slow networks;
- Variable number of participants within the same session;
- Multiple telepointers, image browser for previewing conference material;
- Bulletin board for real-time text based communication;
- Friendly, and customizable user-interface;
- Mechanisms for integration with other information systems;

CoMed is based on a hybrid architecture, which is mainly distributed, but also incorporates a central module [CoMed2000]. The main body of the consultation session is supported by distributed replicated applications, while the central module is only used to provide sophisticated management and authentication facilities. The basic architectural (Figure 2.8) components are:

- The conference management agent;
- The mediator;
- The user application.

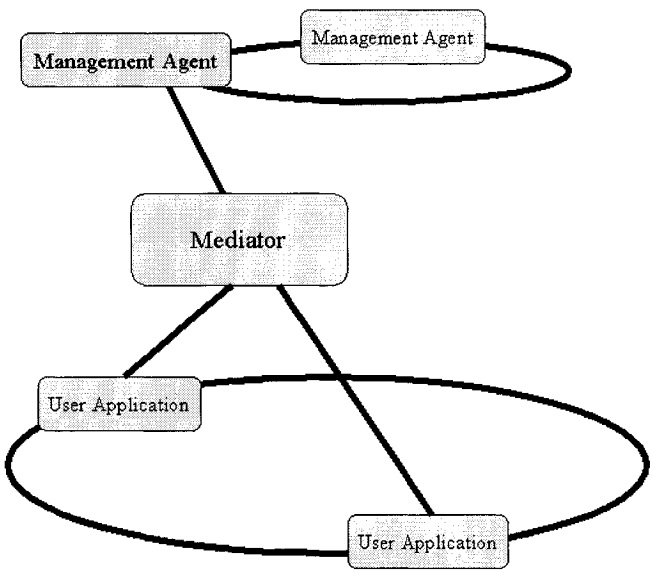


**Figure 2-8** System Architecture for CoMed

The main responsibility of the conference management agent is:

- Creation and management of conference sessions,
- User management and authentication, enforcement of access restrictions,
- Scheduling of pre-conference and post-conference material.

The main task of the mediator agent is:

- Routing management related messages between the management agent and the user applications running on the local workstation.
- Responsible for the temporary storage of conference material transmitted prior to a consultation session.

The basic functionality is provided to the user is to connect to the system, including starting a new conference, joining an existing conference, importing conference material to the system, annotating shared conference material, exchanging comments, etc. Continuously reporting on the conference status provides group awareness.

CoMed should be readily implemented on various operating systems and over heterogeneous networks posed certain implementation constraints. Inter-process communication is based on the TCP/IP network protocol. The application supports the synchronous use of multiple telepointers that appear in a different color, unique for each conference participant.

### 2.4.3 WebOnCOLL

WebOnCOLL [WebOnCOLL1997] [WebOnCOLL2002], a web-based medical collaboration environment, has been designed in the context of the regional healthcare network of Crete. WebOnCOLL employs the infrastructure of regional healthcare networks to provide integrated services for virtual workspaces, annotations, e-mail, and on-line collaboration.

Virtual workspaces support collaborative concepts like personal web pages, bulletin boards, discussion lists, shared workspaces, and medical case folders. Annotations provide a natural way for people to interact with multimedia content. E-mail is one of the most popular forms of communication. On-line collaboration satisfies the need for a more direct form of communication.

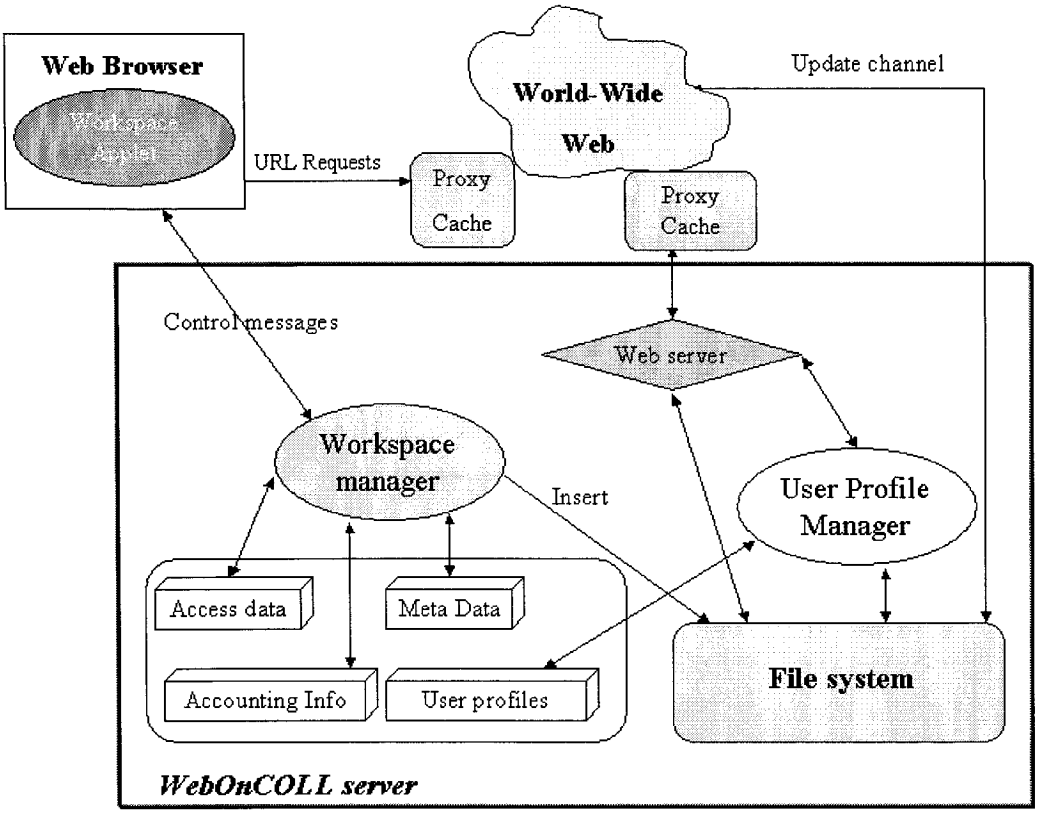


**Figure 2-9** Architecture of a WebOnCOLL Server [WebOnCOLL1997]

A basic requirement of the WebOnCOLL architecture is conformance to the so-called "Open Workspace" paradigm. According to the Open Workspace paradigm, the computer should be integrated into the workspace, rather than isolating the user from the workspace. To attain this goal, the WebOnCOLL architecture is centered on the concept of virtual workspaces and user profiles. Virtual workspaces provide the collaboration framework, while user profiles customize it to the needs and tasks of the current user.

The architecture of a WebOnCOLL server appears in Figure 2.9. The basic components of the architecture are the web server, the workspace manager, the user profile manager, and the information repositories, which reside on a database management system (DBMS) and the file system. The workspace manager controls virtual workspaces and provides notification and awareness information to its clients. The user profile manager

maintains user profiles and personalized information channels and customizes the collaboration environment to user.

### 2.4.4 Patient-Centered Access to Secure Systems Online (PCASSO)

Patient-Centered Access to Secure Systems Online [PCASSO2003] was a research project funded from the National Library of Medicine under the Health Information Infrastructure program, USA. It represents a prototype for providing secure remote viewing of electronic health information for patients and healthcare providers using the Internet. The project focused on developing and deploying technical controls for protecting electronic personal health information against two types of threat:

- Threats from unauthorized users breaking into the server, snooping into the data packets passing through the Internet, capturing information from user's personal computer using a Trojan Horse, and launching denial-of-service attacks
- Threats from authorized users browsing information to which they are not authorized.

The project sought to establish true "end-to-end" information security, including high assurance server, multifactor authentication, advanced transmission security, and protection for and from insecure clients. Within those constraints, the PCASSO system encompasses five aspects to securing the system itself, and its data:

- Role Based Access Control
- Multi-factor Authentication
- Transmission Confidentiality
- Protection from Client Machine
- High Assurance Server

PCASSO associates every user with one or more "roles", which determine what information you can see, and what actions you can perform. PCASSO utilizes its trusted

operating system to assign and enforce data labeling for every piece of patient information it stores. A user can only see information associated with their role in the system, and cannot see any other information they are not explicitly authorized to see.

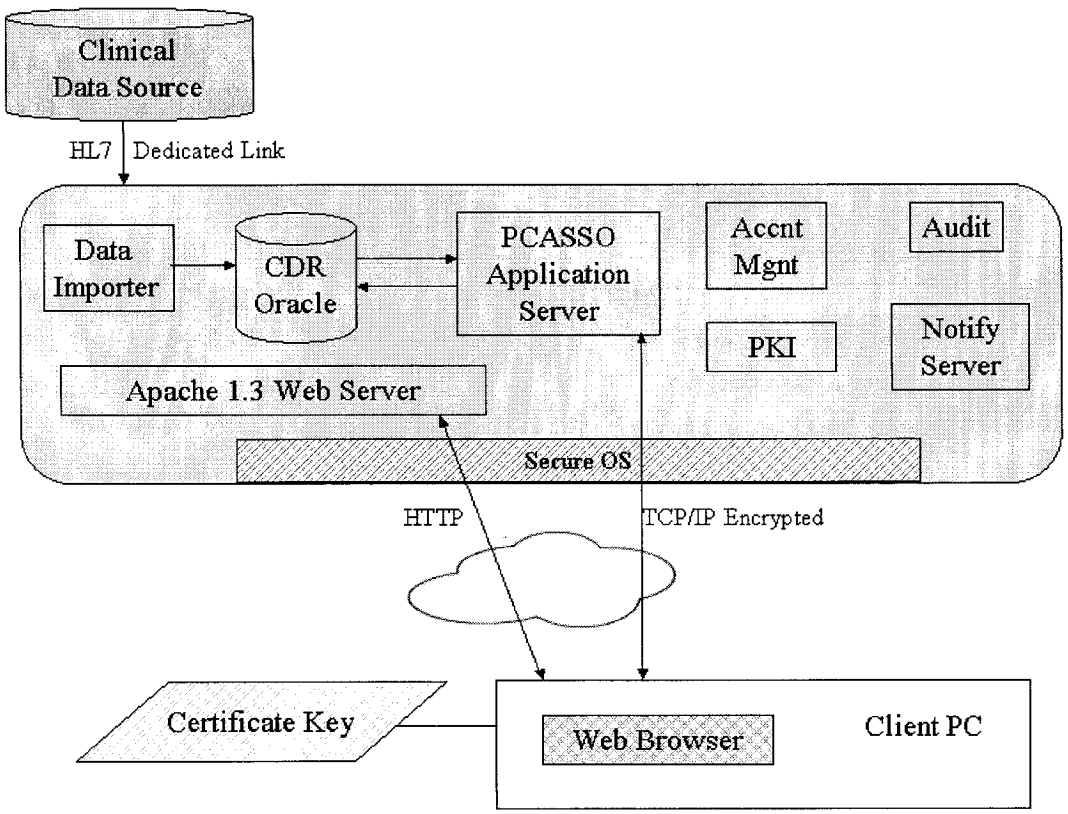


**Figure 2-10** Architecture for PCASSO [PCASSO2003]

The above diagram Figure 2.10 is an overview of the PCASSO architecture which shows both the commercially purchased elements as well as those which were developed specifically for this effort. The Data Importer and Labeler upload the clinical data received from UCSD's interface engine to the Clinical Data Repository (CDR). During this process, the data in the clinical messages is labeled to determine who will have access to the data. The CDR was built using the Label-Based Access Control of Oracle 8i. Access to the CDR is limited to stored procedures. No one may directly access the data in the database.

### 2.4.5 Using Web Services Implementing Collaborative Design for CAD Systems

Pan used web services to support collaborative Computer Aided Design (CAD) systems [Pan2004] over the Internet in an environment that enabled designers to work together naturally and was easy to maintain. It could enable the designers all over the world work together naturally on a common design task and reduce difficulties greatly in every stage of its operation.

The framework adopts the loosely coupled open architecture of web services. Services are encapsulated into web services. In the architecture, described as Figure 2.11, the client side uses browser as its working environment and the services are distributed on the Internet. The working process and message transport is through Internet. When a designer sends out a connection request, this request will be first sent to the CAD System Collaboration Server (CSCS), which will then create a special agent for this client. According to the client request, the CSCS will choose which collaborative mode to use. In the CSCS, there deployed several collaborative services, each service realizes one collaborative strategy. In addition, the services in CSCS could connect to the CSCW database to retrieve its information. Based on the information retrieved, the services could determine whether the operation needs to be passed to other servers for computing and could decide which service in which web server will be utilized. The overall architecture is illustrated in Figure 2.11.

Two crucial technologies also used in the framework:

- The services division method has the following benefits:
    - Guarantee high feasibility in upgrading the CAD software.
    - Low cost in deploying and maintaining web services.
- The distribution services are used to
    - Deploy policy to balance the web servers' load
    - Enable high speed of services response

- Fully utilize the performance of all the web servers.

However, there are still some work needs to be done for integrating the collaborative CAD systems with enterprise business process, and more precise and dynamic in the services load balance [Pan2004].

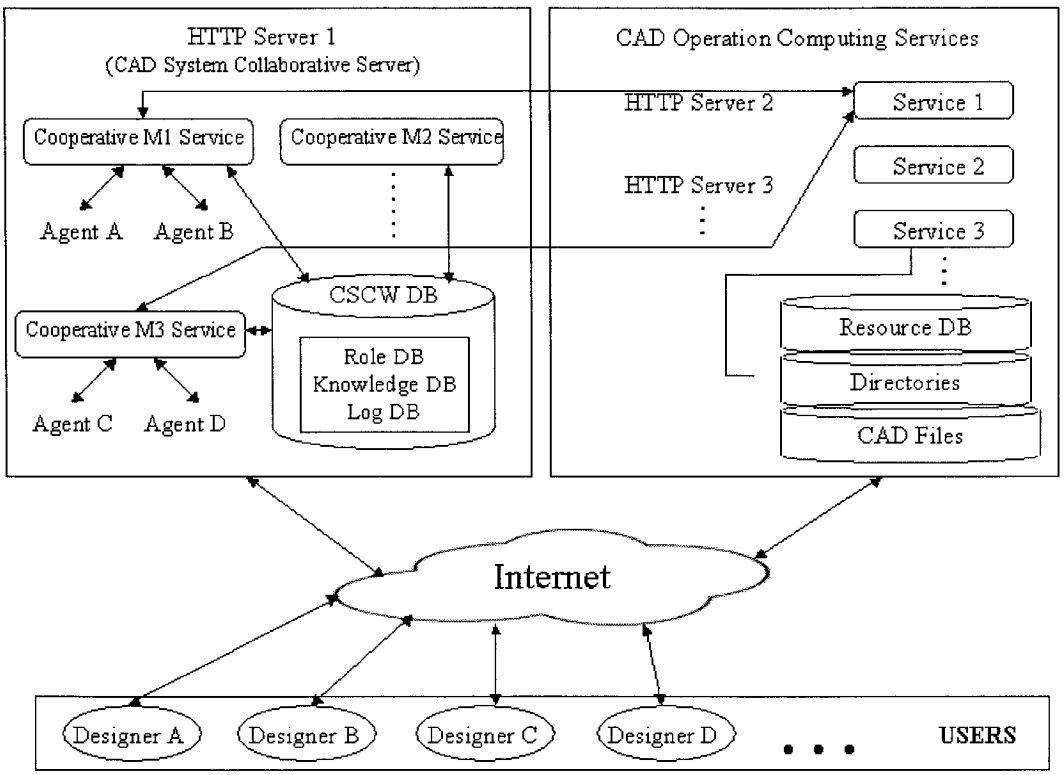


**Figure 2-11** Overall Architecture of the Collaborative CAD Framework [Pan2004]

### 2.4.6 Groove Virtual Office

Groove Virtual Office [Groove2006] is software that allows teams of people to work together over a network as if they were in the same physical location. Groove Networks relies on P2P networks. In P2P networks, the client devices on the edge of the network

communicate directly with one another, thus eliminating the need for the expensive architecture required for centralized servers [Groove2002].

The Groove architecture was conceived specifically to support a decentralized work style. It relies on the power of the desktop to put application capabilities directly in the hands of users. It uses servers as brokers to negotiate the connection between users who are not currently on the network, who are separated by firewalls, or who are connected in bandwidth-constrained locales.

Groove Virtual Office is an application that sits directly on a user's PC, not on a Web site or centralized server. How does Groove Work? As said in [Groove2002], "when users download Groove to their desktop, Groove creates a secure shared space for a group. With Groove, one can use a variety of tools to facilitate group collaboration including text chat, voice chat, file sharing, picture viewing, web browsing, sketchpad, forms, and calendars. Changes made by any team members are instantly and securely transmitted to all group members using XML protocols".

For security, Groove automatically encrypts the user's account, all shared spaces and their contents locally on the user's device, using strong, 192-bit encryption. Users create a pass phrase to encrypt their account and shared spaces. Groove's security design reconciles two somewhat opposing factors: complete end user control and the need for airtight security in and around the content and activities of a shared space.

Groove Virtual Office Project Edition leverages all of Groove's core capabilities to offer the following benefits:

- Dynamic interaction: team members "see" each other online when active.
- Share files.
- Manage meeting and projects.
- Track data and processes.

### 2.4.7 Summary of Related Works

The following table (Table 2-4) is the summary and comparison of related works we described before. We compare their main characteristics and weakness.

**Table 2-4** Comparison of Related Works

| | JASMIN | Web Service CAD | CoMed | WebOnCOLL | Groove | PCASSO |
|---|---|---|---|---|---|---|
| **Technology Focus** | Multimedia Collaboration | Collaboration | Multimedia Teleconsultation | Multimedia Teleconsultation | Collaboration | Medical info Security |
| **Architecture** | Client/Server | Web service | Hybrid, CORBA | Web-based | P2P, desktop | Web-based |
| **Access Control Model** | No | No | Role-based | Role-based | End user control | Role-based |
| **Other Features** | Event-sharing paradigm | Distribution services deployed policy | Basic security | "Open Workspace" paradigm;<br>Basic security | Encrypt account and shared spaces | Multi-factor Authentication |
| **Weakness** | 1.Lack of scalability<br>2. No security consideration | No security consideration | No access control in collaboration session | No access control in collaboration session | 1. Download software to the desktop;<br>2. No B2B interoperate | Not support for a richer collaboration |

## 2.5 Summary

First this chapter explains the basic concept of a secure collaborative environment. Then, it describes related technologies such as XML, XACML, SOA and Web service. After that, it summarizes the problems addressed by existing solutions for Web service security and compares several access control models. Finally it discusses some other collaboration systems that use a different approach. In the next chapter, we will discuss the requirements for sharing sensitive data in a collaboration environment.

# Chapter 3 Requirements for Securing Sensitive Data in Collaboration Environments

This chapter categorizes the requirements for securing sensitive data in collaboration environments. An analysis of requirements is fundamental to the methodology that this thesis takes to achieve the desired solution. A pretty straightforward methodology is used in this thesis, which takes the following steps:

1. Deciding and narrowing the research area.
2. Choosing a typical collaboration scenario to understand the situation to be considered.
3. Identifying, summarizing, and categorizing the general requirements for secure collaboration environment.
4. Investigating some related works and analyzing their advantages and disadvantages.
5. Analyzing to see if these existing solutions work for the chosen scenario and identifying the problems to be solved or finding the weakness to be improved.
6. Researching on new technologies that could be relevant to the problem.
7. Proposing the new solution to improve the functionality or solve the problems
8. Using a case study to build a prototype for testing and validating
9. Evaluating the proposed solution

The aim of a collaborative environment is to provide collaborators with a set of mechanisms and tools for manipulating, accessing and sharing information using meaningful tools supporting their views and ideas. To analyze the requirements better, a scenario in the context of health care is used to reach the goal.

## 3.1 Collaborative Online Medical Consultation System Scenario

Collaborative online medical consultation sessions, among healthcare professionals, may compensate for the lack of experienced or specialized personnel at remote sites, at the site of an accident, or at primary care facilities. They are intended to address emergencies or to evaluate the severity of a case. Furthermore, as far as community care is concerned, collaboration among user groups that share the same medical condition may provide comfort through the sharing of useful information and experience.

In the meantime, medical data privacy and security requires flexible, extensible context-aware access control and dynamic policy enforcement. With on-demand authentication, users are authenticated according to their task-specific situations. Extensible context-aware access control enables administrators to specify more precise and fine-grain authorization polices for any application. Dynamic authorization enforcement makes authorization decisions based upon runtime parameters rather than simply the role of the user.

Consider the following situation (see Figure 3.1): a general physician at the remote healthcare center conducts a clinical examination of the patient, records diagnostic reports and orders some relevant laboratory exams. All information is stored in the local electronic health record system. Sometimes, the case is difficult, and thus it will need a second opinion diagnosis from other expert specialists who work in different organizations, available for giving such a service (step1). In this case, this physician would like to create a collaboration group (step 2) to consult a cardiology specialist working at the Cardiology Research Institute and an oncology specialist working at the general hospital. Some materials have to be collected (step 3) to be shared between the referring and consultant specialists. It is necessary to have an abstract of the clinical record (step 3) with appropriate images, e.g., radiographs or skin photographs, or have obtained reference links stored in a database. Sharing and collaboration (step 4) may occur synchronously or asynchronously. In the synchronous way, some real time collaboration tool is used and an appointment should be made with a date and time agreed by both parties while an asynchronous way is a way similar to usual mailing or

message board. Finally, when the second opinion diagnosis is obtained, it should be inserted into the electronic patient record (step 5).

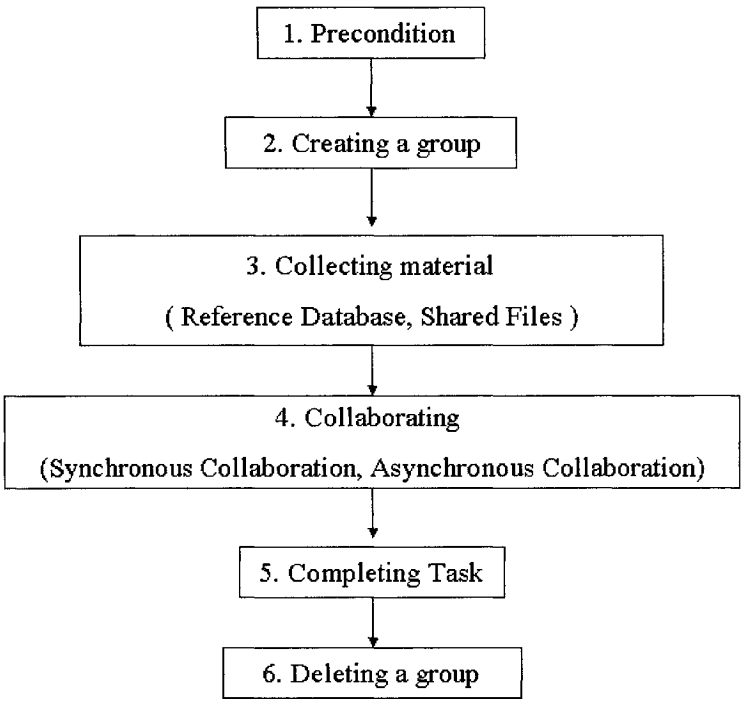


**Figure 3-1** Collaborative Online Medical Consultation Scenario

To facilitate the collaborative environment in this scenario, several functionalities and privacy and security features need to be provided. The requirements for security, privacy and collaboration will be investigated and discussed in the following sections. These requirements are based on the nature of interactions between users and their interactions with the applications, services and data.

## 3.2 General Requirements of a Collaborative Environment

- As a computer-supported cooperative work (CSCW), a collaborative environment should satisfy the specifications of functional and technical requirements for CSCW. Reinhard defined and described criteria and requirements for CSCW systems [Reinhard1994]. Interaction, coordination, distribution, flexibility,

visualization and data hiding were necessary criteria for a collaborative environment. Stiemerling recognized tailorability as an important feature in a CSCW system [Stiemerling1999]. Simone and Schmidt suggest that a CSCW system should provide awareness [Simone1998]. In short, the general requirements for a collaborative environment are listed as follows: Interaction: the system should allow collaborative work among users by proceeding in asynchronous or synchronous mode.

- Coordination: the system should provide a means of communication within the group of users.
- Distribution: the system should enable people to interact from remote places.
- Visualization: the system should visualize public data collaboratively in the paradigm of what-you-see-is-what-I-see.
- Flexibility: the system should be flexible in application field, functional fields and in technical field.
- Awareness: the system should provide users a way to be aware of what is happening in the environment.
- Tailorability: the system can compose modules from a set of predefined components and can add new functionalities to the system.
- Usability: the system should have a user-friendly interface to let the user play any content at will.

Among them, flexibility and tailorability are especially important in a collaboration environment when sharing and protecting sensitive data.

## 3.3 Security and Privacy Requirements

In a collaborative environment, collaborators may be sharing services or information that is restricted based on organizational policies, government regulations, or laws. One

example would be, when someone is attempting to gain access to information of a sensitive nature, like a medical consultation in which the collaborators share sensitive data of the patients. In most countries, personal information is protected by law, and individual organizations have their own policies and rules as well. It is essentially important to protect personal data. Access can be expanded as needed, but privacy, once violated, can seldom be repaired.

On the other hand, access rights are dynamic and frequently changed in a collaboration environment. For example, if User1 has right to read File1 and share it with User2, but User1 is not allowed to see it. Therefore, access facility should be able to extend or refuse access right dynamically.

In the following, some basic security and privacy requirements for distributed collaborative environment, complementing requirements are presented by Greif [Greif1986], Ellis [Ellis1991], Thomas [Thomas1997], Edwards [Edwards1996] and Bullock [Bullock1999].

- Authentication of the users is necessary so that collaboration participants can determine whom they are interacting with and possibly have access to the information about other group member.
- Authorization mechanisms are necessary to determine access rights on the ways in which the availability of sources is managed when sharing.
    - Access control should be generic and enable access rights to be configured and controlled while sharing
    - Access control should allow users to take multiple roles simultaneously and change these roles dynamically during different phases of collaboration.
    - Access control should allow dynamic establishment of security policies between a pair of collaboration parties which means it must be application-instance specific and change policies runtime depending on the environment or collaboration dynamics.

- Securing shared data exchange in communication channel. Privacy and integrity should be ensured when documents flow among different parties.
- Auditing mechanism is necessary to prove to auditors and regulators that its secure-collaboration policy is working.
    - Maintain a history of the activities performed by each collaborator in the system.

## 3.4 Summary of Related Works and Technologies

In the previous chapter, several related technologies and collaboration systems have been described. None of them has met all the requirements described above.

For collaboration, all of the related works are good at functionality and performance. They can efficiently achieve interaction among users. Except Groove is scalable, they are weak in flexibility and scalability from the point of view of architecture.

For privacy and security, Groove adopts user control model to allow users to control their own files; but this security model is not enough to support sharing sensitive data that belongs to an organization. Most of other systems assume that as long as one is authenticated, one can see everything in a collaboration session, let alone dynamically change the access policies or the user roles. They are weak in access control during collaboration.

This thesis aims to provide a flexible architecture with a strong access control notion.

In the next chapter, we will illustrate service oriented architect support for collaboration environment.

# Chapter 4 Service Oriented Architecture support for Collaborative Environments

In this chapter, a Web-based and Web service-oriented architecture support for secure collaborative environment is presented. The combination of distributed, layered architecture technologies, together with XML, can support the creation of a collaboration environment with secure access to a diverse set of services. In addition, it can be efficiently integrated with the existing Internet security infrastructures such as firewall and SSL. In all cases, end users need only access Internet.

## 4.1 Initiative for Proposed Solution

The proposed architecture is an application architecture based on the philosophy of reusing common services for more flexible, scalable, maintainable software. Applications have a lot of commonalties between them and reductions in design and development can be achieved with preparation and advanced planning. In essence, applications contain a collection of services, which in turn contain collections of components, which in turn contain collections of Objects – the atomic units of applications. Therefore, by abstracting the common functionality within each of these resources, many applications can take advantage of code reuse. Furthermore, when problems are identified, they only need to be fixed in one place without breaking down others.

The combination of distributed, n-tier architecture technologies, together with SOA, can support the creation of collaboration services and secure access to a diverse set of services in a single integrated environment. Through its loosely coupled, coarse-grained, standards-based approach, SOA avoids the drawbacks of traditional methods and changes the way companies can integrate information. Web services provide an industry standard mechanism for making remote function calls over a single port, the same one that is used for web data transfer. Their role in n-tier applications is as a middle-layer component, likely to be accessed by many different systems to provide the same business functionality.

## 4.2 SOA Support for Secure Collaboration Environment

The collaboration environment is an application implemented on the top of an organization's service oriented architecture with special services to both protect and enable the sharing of sensitive data belonging to that organization or to other partner organizations.

From a design perspective, the proposed architecture uses a service-oriented architecture (SOA) to support secure collaboration environments. SOA is based on a layered architectural concept in which each layer is loosely coupled and is able to function independently from the layers around it. A key focal point of a SOA is aggregating components, or internal back-office functionality, that can be exposed as services so they can be accessed in a ubiquitous fashion. These services reside in a common runtime infrastructure and allow core business policies to more closely represent business process workflow and not be tied to application silos. [Versata2004]

From an implementation perspective, the proposed architecture defines a collection of common services, which can be used or built into applications. A service provides a high-level, well-defined function and is generally implemented as a coarse-grained, discoverable software entity that exists as a single instance and interacts with applications and other services through a loosely coupled, message-based communication model.

Figure 4.1 is the overall architecture support for a collaboration environment for sharing sensitive data. One collaborator can dynamically grant permission to share sensitive data with a collaborator if they have the right to grant access. The design of the system assumes that collaborators have already signed agreements that they will protect the sensitivity of any data shared. Most privacy legislation including PIPEDA allows the sharing of sensitive data with other authorized users in the context of shared work to provide a service. The users, or collaborators, use Web browser as the working space. They interact with each other by a Collaboration Manager. The Collaboration Manager uses collaboration services and security services to realize its functions. These services could be distributed over the Internet. The working process and message transport are through Internet. They are illustrated as follows.

## 1. Web browser

A Web browser, such as Netscape Navigator or Microsoft Internet Explorer, is the most common tool running on client side. It provides users with a graphical interface to view and navigate the collaboration Web pages.. The users use Web browsers as their working environment.

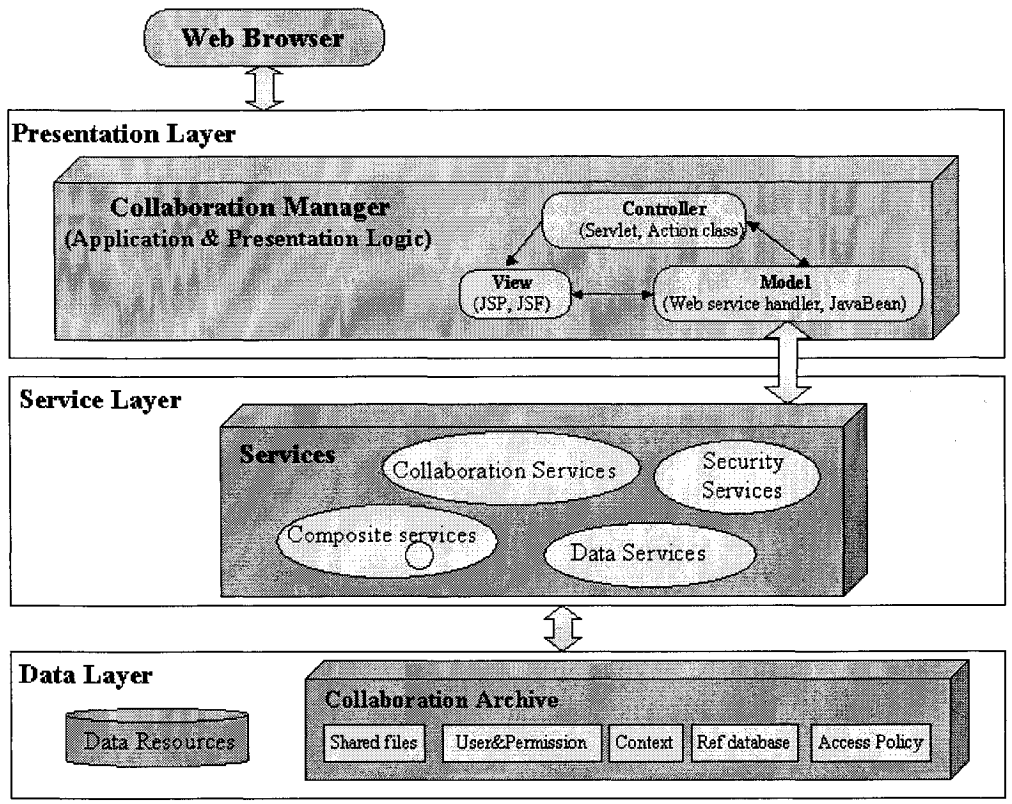


**Figure 4-1** Overall Architecture for Secure Collaboration Environment

## 2. Presentation Layer

The presentation layer is very important from a user perspective. This layer implements Collaboration Manager, which is the important collaboration logic.

1) Functions of Collaboration Manager

Collaboration Manager is the application logic required to perform tasks related to the collaboration. The Collaboration Manager runs on the Collaboration Server and deals with important issues such as information organization and availability, session management and awareness. It supports user cooperation by providing secure and transparent access to a shared data collection; maintains access control information and a list of the currently connected users.

According to the request of the user, the Collaboration Manager decides which services need to invoke to complete the task. When a collaboration client opens a connection to the Collaboration Manager and provides authentication information, i.e. workgroup, user and password. The Collaboration Manager uses an authentication service to verify that the user has indeed authority to access the shared workgroup and records the connection in the event log by invoking Audit/log Web service. Then, the Collaboration Manager adds the user to the list of users that are currently connected to the requested workgroup. For the next activity, if the user types instance message in the chat box, the Collaboration Manager invokes Chat Web service to realize chat function. If the user requests to view a specific document, the Collaboration Manager checks Authorization Web service if the user has authority to access this data. Upon the permission, this document is displayed. The Collaboration Manager uses a Web service by its Web service proxy. Finally, when the user exits, the collaboration manager removes the user from the user list.

2) Design of Collaboration Manager

An important step to design a system is to choose a suitable architecture. The Model View Controller (MVC) architecture is a powerful design pattern [MVC2002] that recurs in most object-oriented applications. The goal of MVC is to separate the application object (model) from the way it is represented to the user (view) from the way in which the user interacts and controls it (controller).

MVC is particularly well suited for interactive Web applications, where a Web user interacts with a Web site, with multiple iterations of screen page displays and multiple round-trips of requesting and displaying data. Since the nature of collaboration, we use the Model-View-Control (MVC) as design pattern.

**Controller** – The controller translates interactions with the view into actions to be performed by the model. In our collaboration application, user interactions are GET and POST HTTP requests. The actions performed by the model include invoking Web services or changing the state of the model. Based on the user interactions and the outcome of the model actions, the controller responds by selecting an appropriate view. Each panel that makes up the collaboration User Interface has its own servlet that acts as the controller and is in charge of the request processing and the creation of any beans or objects used by the panel, as well as deciding, depending on the user's actions, which JSP page to forward the request to. [MVC2002]

**Model** – The model represents enterprise resources and data, as well as the collaboration rules that govern access to resources and data. The Collaboration Manager is a client to services that provide resources and data. Within the Collaboration Manager, the Model is responsible for invoking and interacting with the Web Services in response to requests from both the Controller and the View. Classes that make up the model are called handler classes. The handler classes invoke the Web Services using generated proxy classes, which are bound to the host and application providing the Web Services.

**View** –The view renders the contents of a model. It accesses desired data through the model and specifies how that data should be presented. The view is responsible to maintain consistency in its presentation when the model changes. There are two models to allow view to achieve consistency. One is a push model, where the view registers itself with the model for change notifications; the other one is a pull model, where the view is responsible for calling the model when it needs to retrieve the most current data. [MVC2002]. In our case, the view uses a pull model and is defined by a single .JSP file that defines the User Interface for the collaboration environment. Each "panel" of the User Interface is defined by a JSP include file that defines how the Panel looks, and how it interacts with services to draw that look. Actions

### 3. Service Layer

These services are set of common services for use in the Collaboration Manager. They encapsulate a specific set of functions, which can be easily integrated into an application.

Collaboration and security are two kinds of important services. Especially, Web service is chosen to implement these services.

Why a Web Services Approach?

As promoted by the World Wide Web Consortium, Web services are now seen as the preferential way to link applications both within and without an organization in a loosely-coupled, language-neutral, platform-independent way.

- Web services mitigate the application integration crisis. They are platform- and language-independent. They communicate using XML and Web protocols, which are pervasive, work both internally and across the Internet, and support heterogeneous interoperability.

- Web services reduce complexity and simplify the process of making applications talk to each other. They're easy to work with.

- Web services make component reuse and enables application "plug-and-play". These service hubs aggregate individual application capabilities into coarsely grained business services that are exposed for use by multiple systems.

- When implemented over HTTP on a standard HTTP port, web services can supports the cross-firewall environment in which most collaboration takes place without having to change any of the filter rules. This can be an advantage of Web services as the internal application "glue". It provides flexibility for the future

- Developing a service on a higher level so that the Web Services currently being used can benefit from the gained security, such as SSL.

The greatest challenge of building a service-oriented application is creating an interface with the right level of abstraction. Authorization Web service and Auditing/logging Web service are implemented for providing access control management during collaboration. Chat Web service, shared document Web service and shared whiteboard Web service are developed for sharing data and idea among the users. These services will be outlined in future sections.

**4. Data Layer**

This layer provides persistent data storage and management. It can store application data such as audit trail records; or it can be a resource for a Web service to retrieve. It also can be used to store commonly used collaboration logic procedures to reduce the network traffic associated with repeated operations.

The Collaboration Archive is an important component. This is where the application's data is stored. The data stored here are data required for the collaboration application to perform its functions. The Collaboration Archive maintains the list of available workgroups, their contents, as well as access data associated with recent connections. The shared folder or a reference database includes documents and data references shared in the course of collaboration session and management data regarding events that occurred. The users, data access policy and their permission are also archived here. They can be modified and updated during the collaboration. Authorized users may retrieve data to get information relevant to specific types of episodes.

## 4.3 Collaboration with Web Service

The primary benefit of SOA is the ability to compose services from other less complex services. It is also attractive for its tailorability, flexibility, reusability and extensibility.

The proposed secure collaboration environment is based on SOA using Web services. Services can be distributed over multiple servers from any geographical location in a transparent manner for users. In this architecture, the user uses graphic user interface as its working environment and connect the collaboration Web services by a collaboration manager and Web service proxy. The working process and message transport is through Internet.

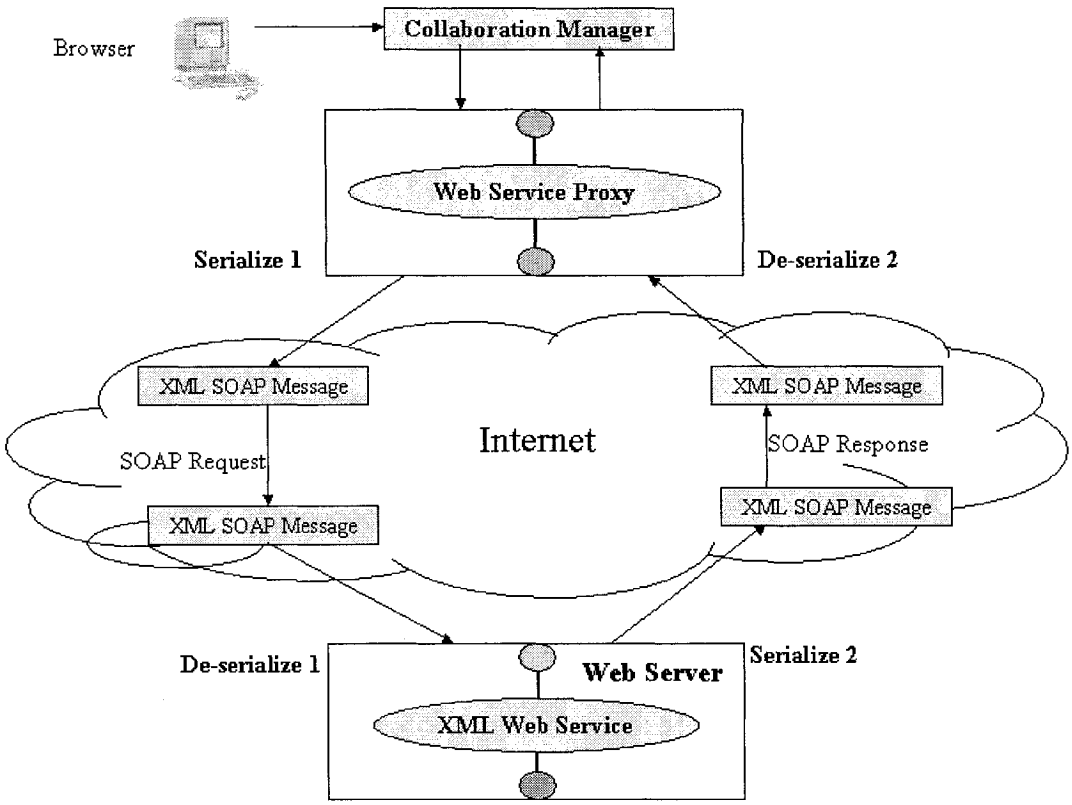


**Figure 4-2**Communication Between Web Service Client and Web Service

Figure 4.2 shows how a Web service client communicates with Web service. A Web service uses the Simple Object Access Protocol (SOAP) to encode its invocation parameters and results over HTTP. A client proxy encapsulates the SOAP client programming APIs and the details of writing, sending, receiving, and parsing SOAP XML documents.

Moreover, SOAP makes use of the secure HTTP protocol or SSL for better data security during transmission. SSL is a security protocol that provides more secure mechanism; it is built on top of the TCP/IP protocol suite and is used to authenticate and encrypt data

For the online collaboration environment, the Collaboration Manager can be considered as a Web service client. In the same way as Figure 4.2, it communicates with different Web services by Web service proxy. When a user operation is performed, this operation

is parsed and translated to request message and sent to Collaboration Manager. With Web service proxy, it will use the proper Web services to do computing task and send the result to the user. The collaboration services include Chat Web service, shared whiteboard Web service, and shared document Web service.

## 4.4 Access Control and Auditing Framework

Thinking about security-as-a-service within a service-oriented architecture, this section describes the benefits of a SOA based approach to security infrastructure components in a collaboration environment.

Security as a Service occurs when an application does not internalize or locally host security functionality. Actually this logically is the model that has been adopted by the International Standards Organization (ISO) model of security as implemented through a "Policy Decision Point" (PDP) and a "Policy Enforcement Point" (PEP) (ISO10181) [ISO1996].

When a security is treated as a service, the access control decision and enforcement functionality is not embedded within an application. Instead, the application provides enforcement functionality for an external access control decision but relies on a service to acquire the decision [Hinton2005]. This split of enforcement and decision point allows for multiple enforcement points to re-use the same decision point functionality. This in turn promotes component re-use as well as the consistent application of access control decisions across an environment.

Authentication, authorization and auditing are important security aspects and they are implemented as Web services in the proposed collaboration environment. When any user or system wants to access the data, they have to be authenticated, then evaluated by authorization Web service. Upon access, an Auditing Web service records the activities to auditing databases or files.

Figure 4.3 shows the access control and auditing framework. The key point here is that by allowing access rule to be evaluated by Web services, users can collaboration with different kind of context, no matter locally or remotely.

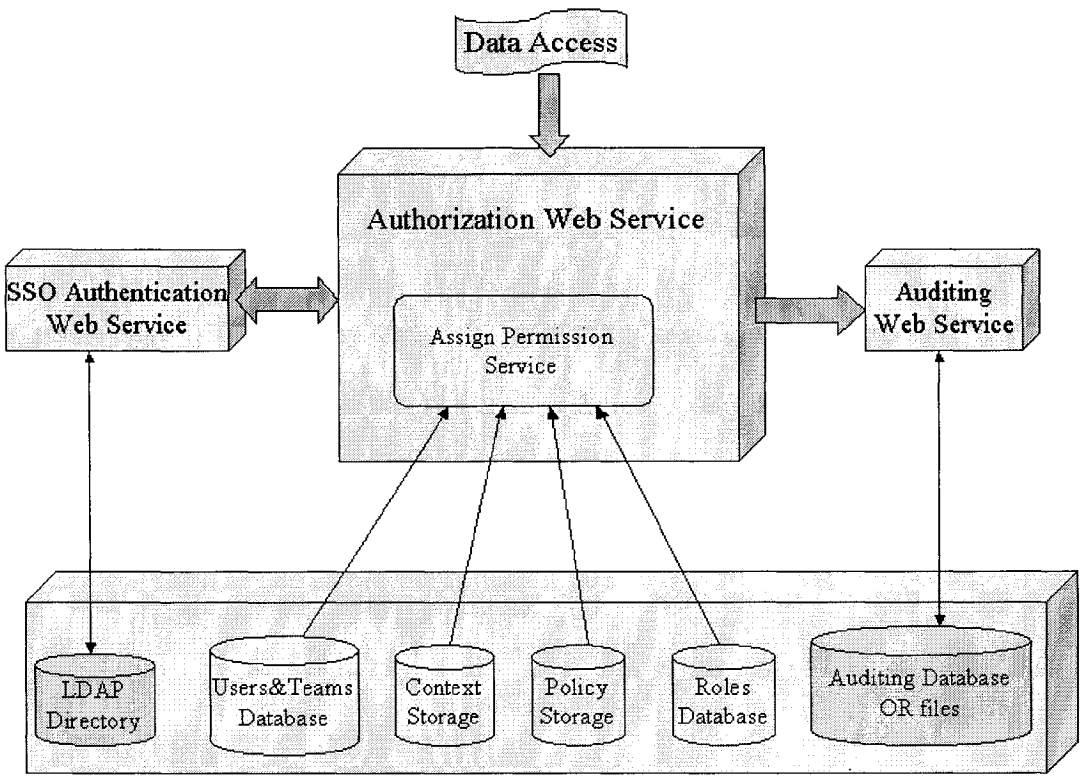


**Figure 4-3** Access Control and Auditing Framework

When an authenticated user want to access certain service or data, the authorization Web service consults related XML-based authorization policies and context to determine what permissions are to be given to a particular user when attempting to touch a protected sources. The authorization Web service returns an "access permitted" or "access denied" decision based upon the user's identity, role, context and permission assigned to the user. Upon the data being accessed, the Auditing Web service records the action in the auditing database or files.

Security at the transport level uses the inbuilt security features of HTTP or SSL. The Web services use this type of security and have been deployed behind a Web Site's firewall. Using SSL, the channel over which two parties communicate can be kept confidential – data is encrypted by the sender and decrypted by the recipient.

In the following sections, we will discuss access control in detail.

### 4.4.1 Terminology

Various research and engineering approaches related to access control or policy management show some variance in terminology. To assure a common unambiguous terminology throughout the rest of this thesis we define the following terms.

**Rule**: A logic construct of a rule like:

*on <event(s)> [ priority <priority> ] [ if <condition> ] do <action(s)>*

A rule is being evaluated at discrete points in time given by the event(s). In XACML, each rule is composed of a condition, an effect, and a target.

**Condition**: An expression or statement that is evaluated to a Boolean value each time the rule is triggered. If the condition contains variables that specify elements, it is evaluated for each combination of those elements.

**Effect**: the intended consequence of the satisfied rule. It can either take the value Permit or Deny.

**Target**: in XACML, as in the case of a policy, helps in determining whether or not a rule is relevant for a request.

**Policy**: A set of rules, usually concerned with a common domain

**Role**: An attribute administratively assigned to elements in order to group instances of the same class. This allows rules to operate on a subset of elements that are assigned to a specific role.

**Action**: An operation that is executed each time the condition has been evaluated to true. Attributes of the event that triggered the execution and variables specifying the elements that lead to the matching condition can be referred within the action.

**PolicySet**: A container that holds other Policies or PolicySets. This is a concept in XACML.

### 4.4.2 Dynamic Role and Dynamic Role-based Access Control

With role-based access controls, access rights are grouped by role name. This approach offers significant advantages because of scalability. Each user is assigned one or more roles, and each role is assigned one or more permissions that can be given to users in that role.

We define our dynamic roles using the follow criteria:

1. The membership in a particular role is determined by a predicate function.

   For example, a group named "*Team 1*" is discussing the case of a patient. We have a role named "*Doctor in Team 1*", a user named "*Mike*". We determined "*Mike*" has a role "*Doctor in Team 1*" if "*Mike*" is family doctor of this patient and "*Mike*" is present in "*Team 1*".

2. The membership in the role is determined at runtime when requests for access are made.

   For example, when "*Mike*" tries to access the medical record of a patient, we will determine what is his role at this moment using the criteria 1 as above.

3. The potential membership can vary from time to time during the lifetime of a session.

   For example, "*Mike*" is a "*Doctor in Team 1*" when he joins "*Team 1*"; he could be a role "*Admin in Team 1*".

4. The roles can be described in terms of their attributes.

   For example, a role can be described as (Admin, team 1, June 2006).

5. The roles of a user in collaboration may include the roles for this group and the roles for an object, such a file or a set of data.

   For example, Mike owns two roles "*Doctor in Team 1*" and *"Normal User of File 1"* in the same time.

6. Define access policies to map different roles

   We use XACML to define access policies. There are two policy types:

   - Role Assignment Policy: a policy that contains rules to assign roles to a subject
   - Permission Policy: a policy that contains all the permissions associated with a role

### 4.4.3 Team-based Access Control

Although Role-based access control is powerful, it is not enough for a collaboration environment. Team-based access is applied in our collaboration environment. Recognizing the importance of context information associated with collaborative tasks, the collaboration activities can be best accomplished through organized teams.

A team consists of the following elements:

- A team name, such as "*Team 1*";
- A set of team members, such as the user *{"Mike", "Rachel", ...}*
- A set of team roles, such as {*doctor, super, admin, delegator ...* };
- A set of object types, such as {*xml file, image*};
- A set of team permissions, such as {allow to view file1, ...};
- A collaboration context including a user context and an object context.

To enable access control on the team as a whole, we have the following primitives:

- User_assign(user, team): assign a user to a team
- Team_activate(team): bind the team permission to the team members and the object they need.

During the collaboration, each access request is evaluated to provide just-in-time permission.

### 4.4.4 XACML and Policy-based Access Control

Policy-based access control enforces policies based on the results of the endpoint analysis to control what resources users can access and what they can do once they are granted access (e.g., create, download, copy, or save).

Access decisions are made in real time by static and dynamic policies or rules. Policies may include:

- Who can define new activities or instantiate an activity;
- During the lifetime of an activity, who can change various policies and enforce additional constraints on shared objects;
- Who can access a specific file that is provided by a specific user.

XACML provides a standard architecture for making authorization decisions. [XACML2004]

1. Access requests are filtered by the Policy Enforcement Point (PEP).
2. The PEP forms a formal XACML request using context from the access and any additional, necessary context from the Policy Information Point (PIP).
3. The XACML request is forwarded to the Policy Decision Point (PDP).
4. The PDP gathers authorization policies that apply to this request.
5. The PDP gathers additional attributes needed to compare the request to the policies.
6. A comparison is made by the PDP and the decision is returned as an XACML response to the Policy Enforcement Point (PEP).
7. The PEP permits or denies access.

The Figure 4.6 illustrates how Policy Decision Point determines whether Mike is permitted to read a patient's record. This <Request> matches the "Permission to create patient's record" <Rule> in Doctor Permission.

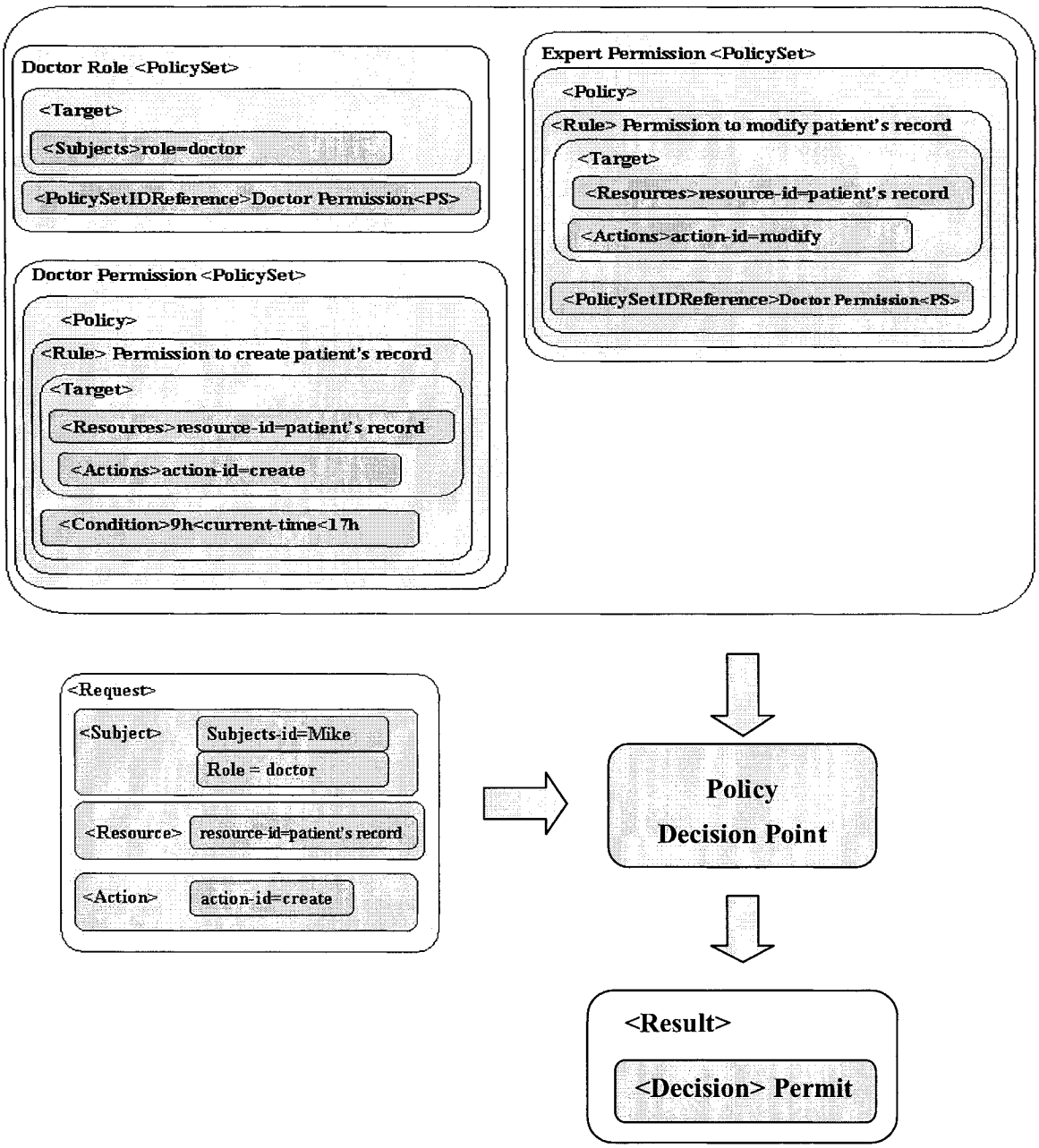


**Figure 4-4** Policy Decision Point Determines Whether Mike is Permitted to Create a Patient's Record

Because a Policy or PolicySet may contain multiple policies or Rules, each of which may evaluate to different access control decisions, XACML needs some way of reconciling the decisions each makes. This is done through a collection of Combining Algorithms. Each algorithm represents a different way of combining multiple decisions into a single decision. There are Policy Combining Algorithms that are used by PolicySet, and Rule Combining Algorithms that are used by Policy.

There are two basic ways to combine policies:

- Deny Overrides Algorithm

  No matter what, if any evaluation returns "Deny", or no evaluation permits, then the final result is also "Deny".

- Permit Overrides Algorithm

  In the entire set of rules in the policy, if any rule evaluates to "Permit", then the result of the rule combination should be "Permit". If any rule evaluates to "Deny" and all other rules evaluate to "NotApplicable", then the policy should evaluate to "Deny".

These Combining Algorithms and other Combining Algorithms are used to build up increasingly complex policies, and they allow XACML policies to be distributed and decentralized.

### 4.4.5 Delegation of Authority

One of the advantages of our system is to give some specific collaborators rights to perform many self-administration tasks such as changing the roles, or configuring the policies. For example, Super User can manage and configure the access policies and the user roles with such privilege at a real time. Also the owner of a file can grant read and write rights to other users during the collaboration.

One of the most important features of policy management is delegation. Delegation is simply one entity giving a portion of its authority to another. An entity can be either an

actor (such as a physician) or software (such as a web service). Examples of how delegation may occur in the online medical consultation system include:

- A physician may delegate his right to view a patient record to one of his medical students who discuss a case of that patient.
- A mammographer can delegate her rights to read a patient's mammograms to an image analysis web service to aid in diagnosis.

To control how a delegator is allowed to delegate a role, some conditions must be fulfilled in order for a delegator to be able to delegate the permission. Also the delegator should have the delegation role with this permission assigned. That is, every delegatable permission in a role has a delegation condition; and every delegatable permission in a role has a delegate enabling condition. A delegation role can only be a subset of the permissions within an existing role.

Similar to other core permission assignment, permission delegation must be accomplished by role assignment.

## 4.5 Authorization Web Service

Authorization is a process undertaken by applications to determine whether a subject can undertake a particular action with a specified resource. This service supports that process by providing an interface by which applications can ask for access control decisions.

Collaboration access control decisions can be handled using the authorization Web service. With it, knowledge and control are separated to allow for dynamic policy creation and modification with no code recompilation, or system downtime. Consequently, deployment of new and updated policies is made easier since new executables do not need to be installed. The proposed framework separates out the function of access control and makes this as a distributed service, which performs authorization on behalf of the resources in the system. In collaboration environment, the policy and the role can be changed frequently and at any time. So the authorization

should evaluate the access request with different policy and different context dynamically.

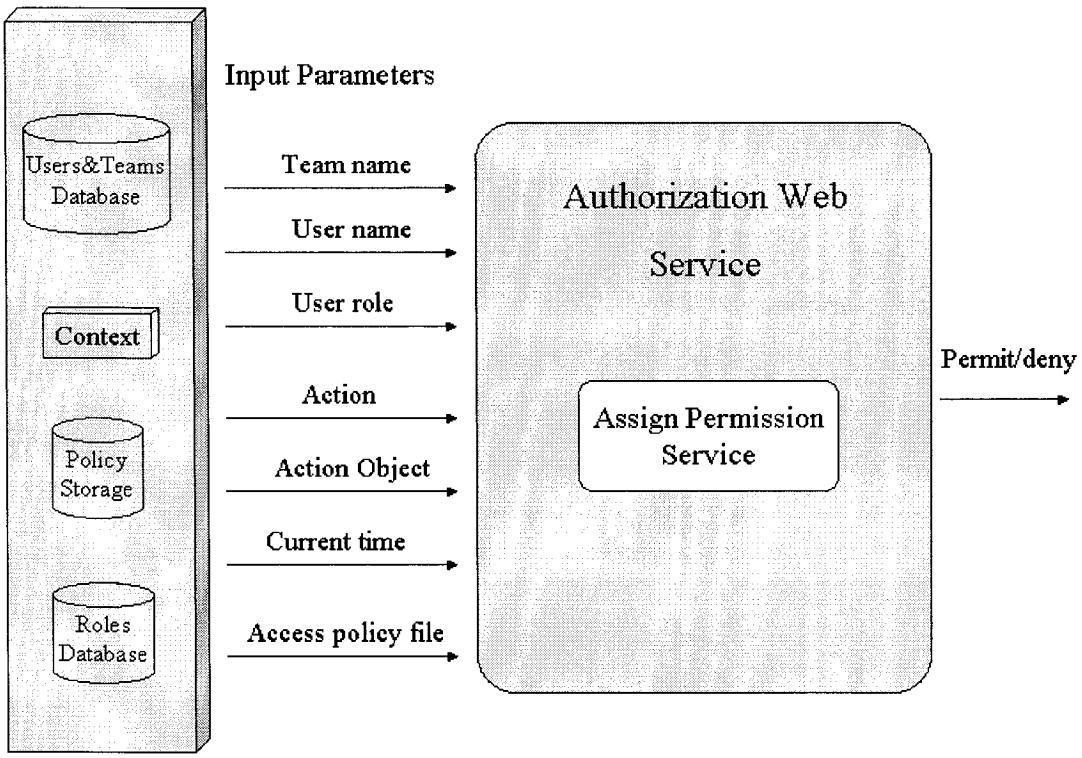


**Figure 4-5** Authorization Web Service

Generally authorization service is comprised of two main parts. One is decision engine and other is knowledge base. Knowledge base can be context, constrains and policies or rules. Decision engine will make decisions about which rules to fire and when and how. It can be integrated into the applications using its API to fire rules on demand. Therefore, the application can take advantages of dynamic business rule development and modification. The proposed solution will use a Web service to implement decision engine, namely Authorization Web Service.

Hooking into decision point is the API of the authorization Web service. These API allows other applications or services to build the Authorization service functionality into their process space and control when Authorization Web service is run.

To evaluate an access request, the Authorization Web service needs the following seven parameters as input:

- Team name
- XML Access Policy
- User name
- Role name in the team
- Action
- Action Object
- Current time

All the values of these parameters will automatically be evaluated in the current context when a registered user takes some action in a system.

After these input parameters are passed to the Authorization Web service, it forms a XACML request, gathers the XACML policy and any additional information required to compare the request to the policies. A permit or deny decision is output.

## 4.6 Auditing Web Service

Auditing, logging and exception/error handling are fundamental to all application. They are services, which provide a level of security by monitoring, and capturing application and data access, messaging request and user events. Auditing is the process whereby information about operating requests and the outcome of those requests are collected, stored, and distributed for the purposes of non-repudiation. In other words, auditing provides an electronic trail of computer activity. If the intrusion into the system already happened, the mechanisms should be in place to allow discovery of the break-in. Auditing can be used to enable such detection. Since logs and audits are typically generated to indicate activity within the fault, performance, security and accounting realms, the Log/Audit play an important role in managing applications.

In the proposed solution the auditing service maintains a log of the most important events in the system. Logging and auditing messages will be directed to the file system for performance reasons. Further more, a Log/Audit Adapter Agent will be used to adapt log messages from applications (stored in files), and inserted in a standard format into a database for analysis. An interpretive mechanism is required within the adapter to correctly associate an application's logs with this standard format.

Analysis of the information contained in the log can give insight into activities of the users and can help in detection of the potential intrusion. Secure auditing provides the mechanism to track how information in the repository is created, modified and used. This is an essential enabler for forensic analysis, which is used to determine how and by whom policy controls were circumvented.

Auditing is a monitoring and tracking service that records activity within the system for later analysis. In a collaboration environment, events to be audited will include application specific messaging information. Some of these might include the requesting application name, time/data stamp of the event, the messaging requests being made, repositories being accessed, source and destination.

Each entry in the log contains following data:

- Operation type
- Target resource
- User identity
- Timestamp

As above information is input to the Auditing Web service, a record is written to the database or files.

## 4.7 Summary

This chapter describes a service oriented architecture support for secure collaboration environment. It has the following characteristics:

- Web based

- Service oriented architecture support
- MVC design pattern
- Collaboration as Web service
- Security as Web service
- Combining multiple access control model

In the next chapter, we will describe a case study including scenario description, design and implement considerations.

# Chapter 5 Case Study

This chapter presents a case study, where a SOA supported architecture described in the previous section is used to build a secure collaborative online consultation environment. First, a collaborative online medical consultation scenario is introduced; then the system that was implemented to validate our thesis is described. Unified Modeling Language (UML) is used to describe the design of our prototype system.

## 5.1 Collaborative Online Medical Consultation Scenario

"Telemedicine" is defined as the provision of health care consultations and other services, with the help of telecommunication technologies [Muirhead2000]. In Telemedicine, collaborate online-medical consultation sessions, among healthcare professionals, may compensate for the lack of experienced or specialized personnel at remote sites, at the site of an accident, or at primary care facilities. They are intended to address emergencies or to evaluate the severity of a case. Furthermore, as far as community care is concerned, collaboration among user groups that share the same chronic medical condition may provide comfort through the sharing of useful information and experience.

In the meantime, medical data privacy and security requires flexible, extensible context-aware access control and dynamic policy enforcement. With on-demand authentication, users are authenticated according to their task-specific situations. Extensible context-aware access control enables administrators to specify more precise and fine-grain authorization polices for any application. Dynamic authorization enforcement makes authorization decisions based upon runtime parameters rather than simply the role of the user.

### 5.1.1 Scenario Description

A near real time collaboration scenario, which is derived from the scenario introduced in Chapter 3 (please see Figure 3.1), is further analyzed in order to identify some of the relevant issues and test the proposed approach. This scenario happens when a general

physician at the remote healthcare center is consulting a cardiology specialist working at the Cardiology Research Institute and an oncology specialist working at the general hospital with a real time collaboration interface (step 4 in Figure 3.1). This collaboration interface allows them to communicate and exchange ideas with each other; view patient documents and x-ray images in the paradigm of what-you-see-is-what-I-see. In a word, the collaboration interface should integrate several collaboration tools and enable their interaction.

At the same time, multiple participant organizations and individuals are recognized in this scenario:

- A general physician at the remote healthcare center;
- A cardiology specialist at the Cardiology Research Institute;
- A oncology specialist at the general hospital;
- The information systems of the remote healthcare center;
- The information systems of the laboratories.

These collaborators work in different organizations, and different organizations have different data access policies, therefore the collaborators have different privileges to access the patient data. To achieve the paradigm of what-you-see-is-what-I-see and view the data together, a mechanism of authority delegation should be provided to extend permissions to other collaborators.

Altogether, the collaboration consultation environment included the following collaboration tools and management interface:

- Chat message window
- Shared Whiteboard window
- Document sharing windows
- Session management window

Of particular concern from the point of view of sharing sensitive data and ensuring security and privacy are the Document Sharing Windows and the shared Whiteboard window. In particular, when a document and images are shared, the Authorization Web Service is used to check permission to ensure security and privacy.

### 5.1.2 Assumptions for the Prototype

In order to facilitate above scenario and evaluate the approach proposed in this thesis, a prototype has been developed along with a testing environment.

For simplicity, the following assumptions and preconditions are made before designing the system:

1) The collaborators have already been authenticated in some valid way when they interact with the system. The prototype does not implement the authentication service;

2) The patients have been given their consent to access to their private data. The prototype does not implement the process to get the consent. A collaborative approval process for enabling access sensitive data is described in [Peyton2005].

3) Shared data in collaborative environment is in a predefined standardized format. The Shared Whiteboard window in the prototype system only can display an image with its name ending with gif, bmp or jpg. The Document Sharing window can only work with browser-enabled formats.

### 5.1.3 UML

UML is the standard language for modeling software-intensive systems [Booch1999]. There have been many efforts made to model web applications at multiple levels by using UML. The following sections model the above scenario and our framework in UML.

- Use cases are used to describe the functional requirements of a system. A use case is a description of set of sequence of actions that a system performs [Booch1999]. It yields an observable result to a particular actor.
- Package and class diagrams show the static view of a system.
- Sequence diagrams illustrate the dynamic view of a system, that is, how objects work together to accomplish the requested functionality.
- Component diagrams describe the organization of the physical components in a system.
- Deployment diagrams depict the physical resources in a system including nodes, components, and connections.

## 5.2 Prototype System Requirement Considerations

### 5.2.1 Use Case Diagram

To better analyze and capture the potential requirement of the system, uses cases are provided to convey how the system should interact with the end user (collaborators) or another system to achieve collaboration or other task. Figure 5.1 is the use case diagram for real time collaboration environment.

Since we assume the collaborators are authenticated, the collaborators in our system are the actors named Authenticated Users. They can be divided into two categories. One is Normal User; the other is Super User. All Authenticated Users can chat, upload files, browser the shared documents, display images and draw diagrams in the shared whiteboard. The system should check access permission to allow or deny the collaborators' actions (except Chat). Upon the user's action, an audit trail will be created for tracking and analyzing in the future.

The Super user is capable to do more. The Super user for a group can create and delete this group. It can grant or revoke action rights for this group. Its actions will be recorded after being authorized.


**Figure 5-1** Use Case Diagram for Real Time Collaboration environment

### 5.2.2 User Interface Considerations

The participants interact with each other in a collaboration interface (Figure 5.2) that includes the following collaboration tools to allow successful consultation:

- Chat message window

  Collaboration participants can chat with each other. With the help of Collaboration Manager, the Chat Interface works with Chat Web Service and Auditing Web Service to realize chat functionality and record all messages.

- Shared Whiteboard window

  Collaboration participants can hand-draw some information and show images such as x-ray and figures in the shared whiteboard window, which are hard to describe through text. According to the participants' requests, which are passed through Shared Whiteboard Interface, Collaboration Manager chooses which Web service to use. Shared Whiteboard Web Service helps to realize draw and display functionalities. Authorization Web Service checks the permission of drawing and viewing data. Auditing Web Service record actions of the users.

- Document sharing windows

  Collaboration participants can view the same document. Shared Document Web service helps to realize viewing document functionality. Authorization Web service checks the permission of viewing data or files. Auditing Web Service record actions of the users. Collaboration Manager ties them together.

- Session management window

  In this window, there are several buttons. When press "User List" button, a user list in current group will be displayed in a new window; when press "Update file for sharing", a interface is popped up to allow collaborators to upload files from local to the server; when press "privacy Report", a privacy auditing report shows

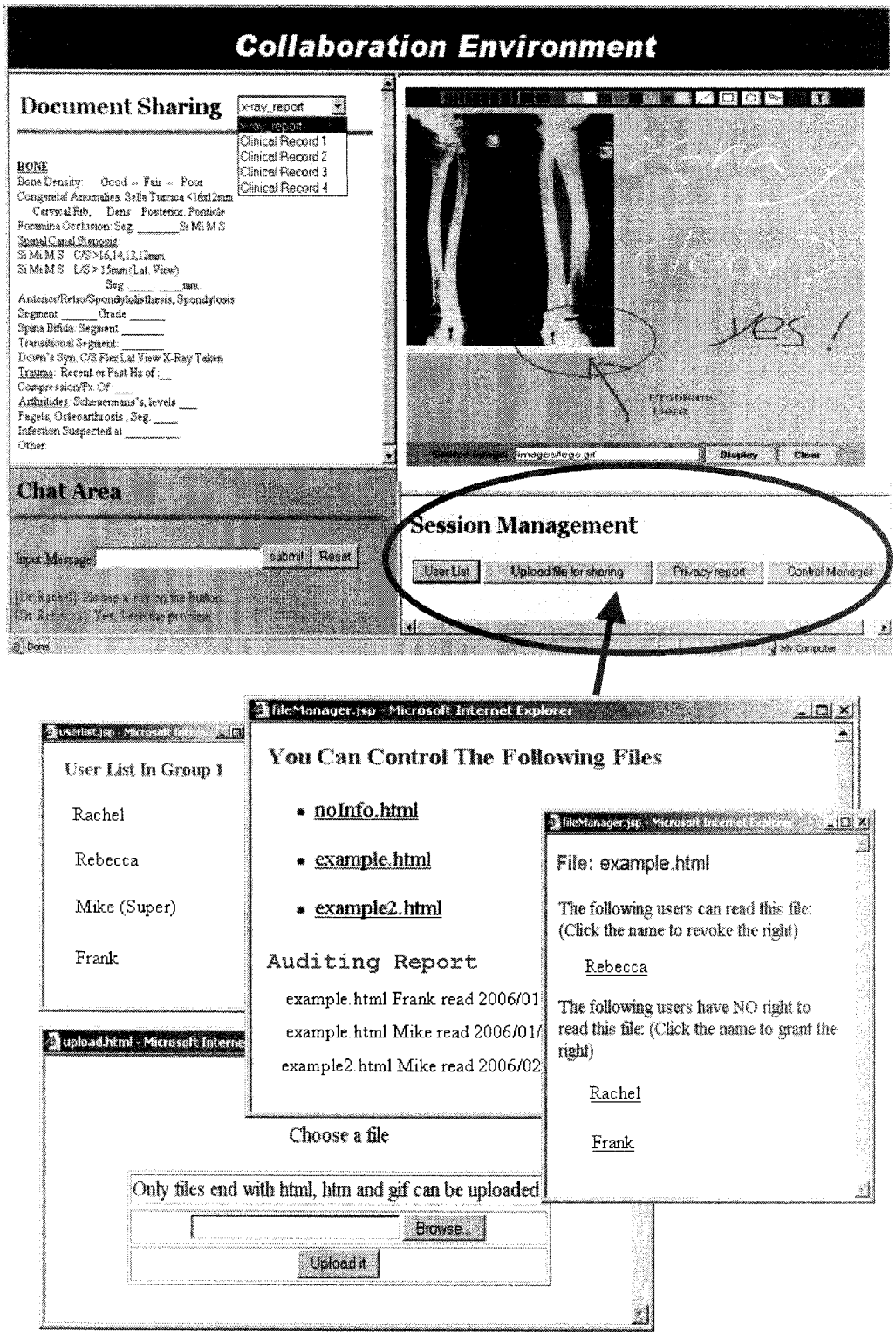


Figure 5-2 User Interface Page for real time collaboration

who, when and how the files/documents controlled by you are shared; and when press "Control Manager", the collaborators can manage and configure data access policy dynamically during collaboration.

Of particular concern from the point of view of sharing sensitive data and ensuring security and privacy are the Document Sharing Windows and the shared Whiteboard window. When a document and images are shared, the Authorization Web Service is used to check permission to ensure security and privacy.

### 5.2.3 Web Application Server and Development Tools

An application server is needed to run our application and host Web services. There are many application servers in the market. JBOSS and WebSphere are two of most popular application servers. While JBOSS has the advantage of open source and open standard, WebSphere is a commercial product that has own API, built-in security, and GUI configuration, GUI setup for administration. We choose WebSphere application server to run our prototype because IBM WebSphere studio is a powerful development tool for Java, J2EE, EJB, JSP, Servlet, XML, UML, and Web Services. Especially it has lots of wizards for Web Service generation, with built-in WebSphere application server that speeds up debugging and testing Web Services and Collaboration Manager. So we can develop our prototype in a short time.

## 5.3 Design Considerations for Collaboration Manager

### 5.3.1 Design Pattern: MVC

Figure 5.3 illustrations how MVC is applied in our collaboration application. When it receives a request from a client, it processes the request in the following manner:

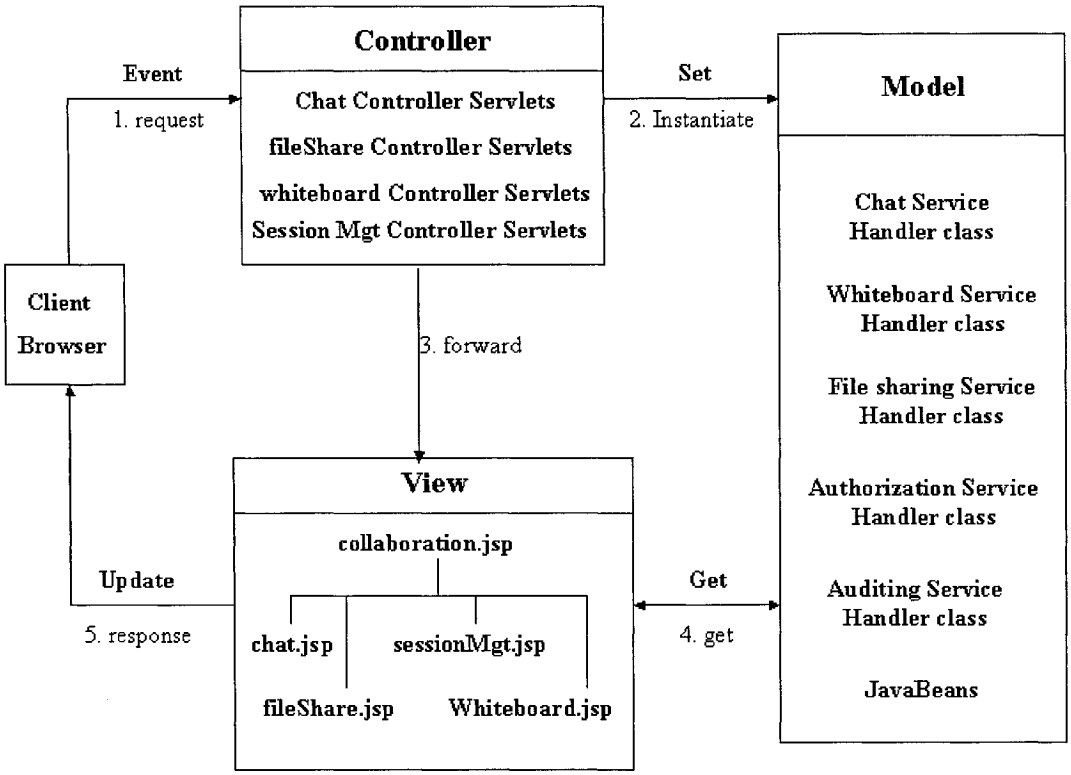


**Figure 5-3** MVC in Collaboration Environment

1. The client sends a request, which is handled by the controller.

2. The controller determines the action specified by the request, and looks up the class for the action. The controller creates an instance of the class and invokes a method on that instance.

3. The instance processes the request. It forwards the request to a JSP page.

4. The JSP page gets an instance of appropriate class for the action and invokes the method to perform the action.

5. The JSP page then extracts the data that the method returned and sends a response to the client

### 5.3.2 Component Diagram and Implementing a Component

A component diagram provides a physical view of the system. Its main purpose is to show the structural relationships between the components of a system.

Figure 5.4 is a component diagram. The prototype collaboration application includes four components. They are components Chat, FileSharing, Whiteboard and CollaborationMgt. The component CollaborationMgt is dependent on other three components. We call these three components Collaboration Components.

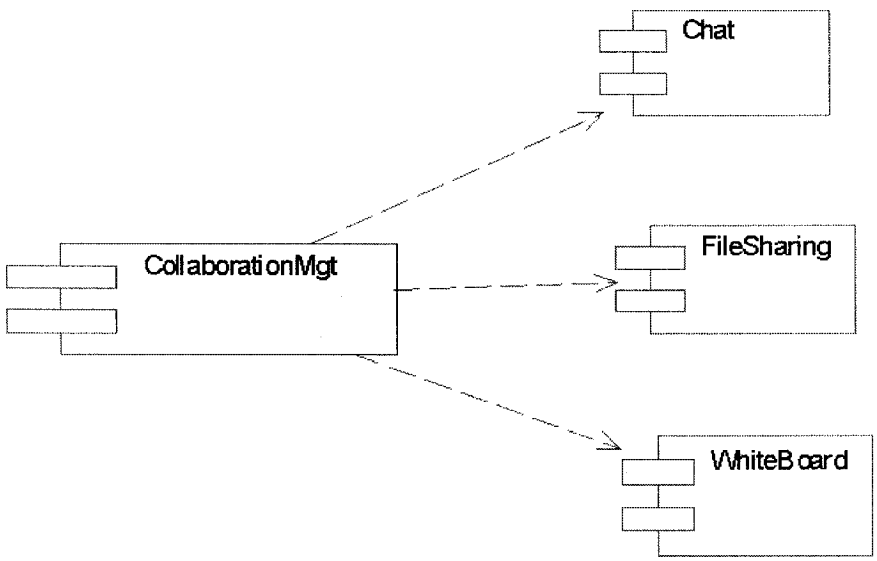


**Figure 5-4** Component Diagram for a Collaboration Environment

**Building a component**

A component is a physical building block of the system. There are four kinds of classes in each Collaboration Component. They are JSP, Action servlet, Security Web service handler and Collaboration Web service handler class.

Figure 5.5-5.7 depicts a design for the Collaboration Components.

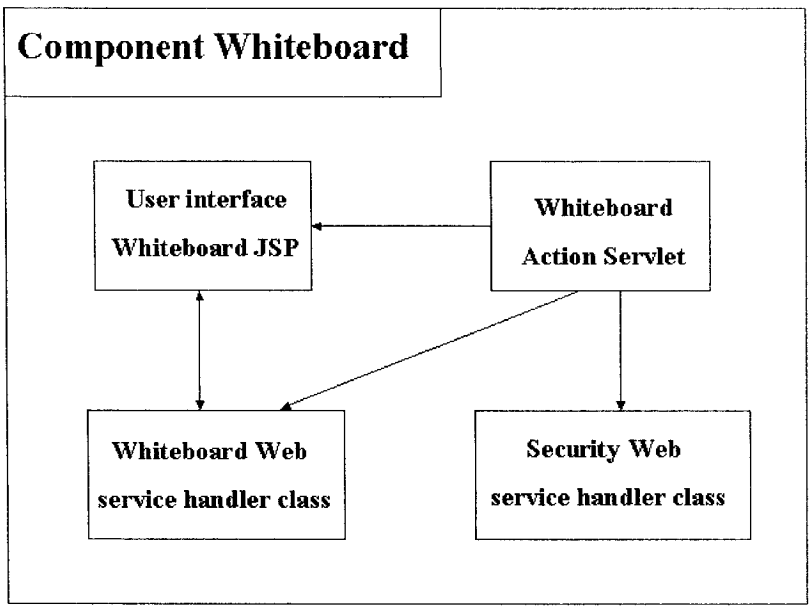


**Figure 5-5** Component Whiteboard

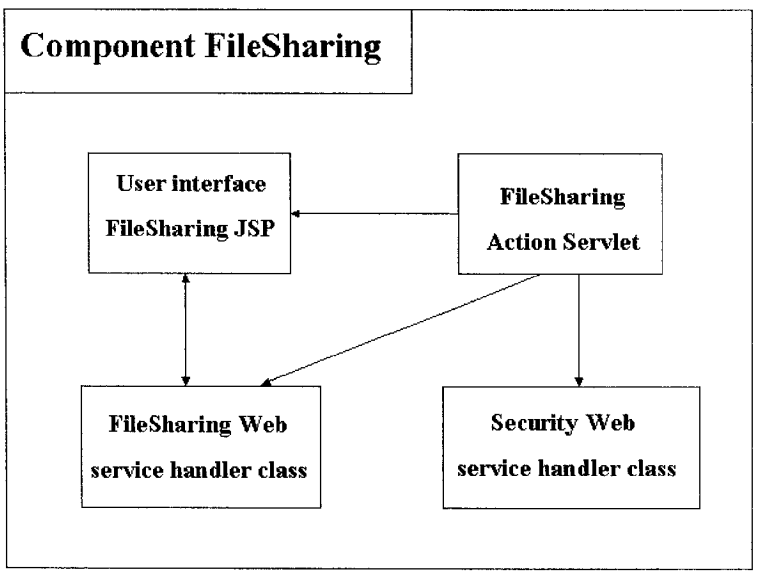


**Figure 5-6** Component FileSharing

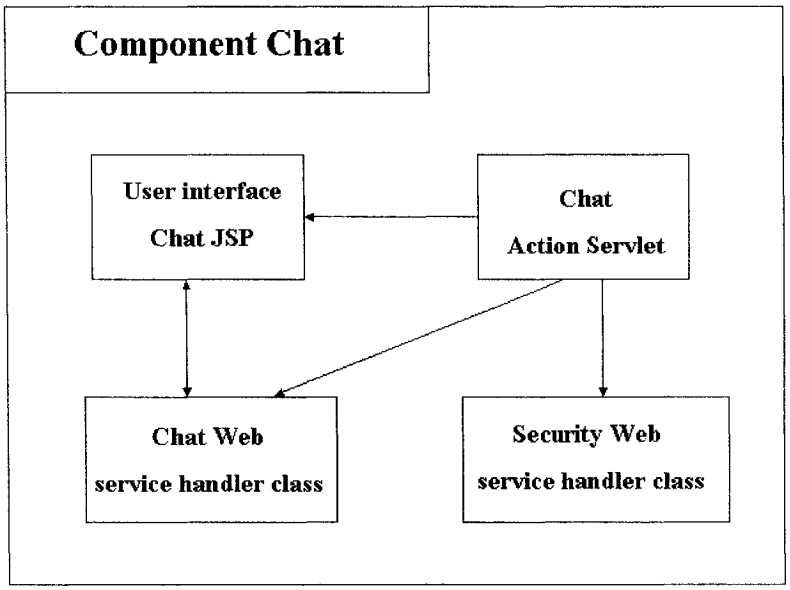


**Figure 5-7** Component Chat

## Integrating a new component

It is not difficult to build a new Collaboration Component and plug it into the collaboration system. For example, if we need a collaborative editing function, we just need to build a new component and plug into the system:

First, building a new component, we need the following steps:

1. Script an editing.jsp file and an action servlet to get and response the user's request;
2. Discover a Web service that satisfies our need and write a service handler class to connect the Web service and allow the action servlet to call;
3. If authorization and auditing are need, we should use an existing security Web service handler class or write a new one to check authority or do audit.

Second, to plug into the collaboration environment, just include editing.jsp file into collaboration.jsp.

## 5.4 Design and Implementation Collaborations for Web Services

The greatest challenge of building a service-oriented application is creating an interface with the right level of abstraction. Among all services, collaboration Web services such as Chat, FileSharing and SharedWhiteboard have common characteristics. We discuss them together.

### 5.4.1. Use Case Diagram for Web services

When collaborators want to post a message, view a file or draw a diagram, their requests are sent to the Collaboration Manager. The Collaboration Manager will invoke proper collaboration Web Service. This Web Service should be able to post a message, display a file or draw a diagram after checking permission. Besides this, collaboration Web Services should have other functions to help realize collaboration functionality. The use case diagram (Figure 5.8) shows the functionalities of Web services and the relationships among them.

1. Web services have to provide the six main functions to allow its client to use.

- Register user

  Allow the Collaboration Manager to register a collaborator to join a specific group. Upon the collaborator join a group, an audit trail is created.

- Remove user

  Allow the Collaboration Manager to remove a collaborator from a specific group. Upon the collaborator leave a group, an audit trail is created.

- Get user list

  Allow the Collaboration Manager to get a user list for a specific group.

- Get current object name

- Allow the Collaboration Manager to know which object is shared in a specific group

- Take a collaboration action

  Allow the Collaboration Manager to help a collaborator to take a collaborative action such as post a message, display a file or draw a diagram in a specific group. Special concern to the actions to a sensitive data, checking the permission is necessary before doing any actions. Upon the taking any action, an audit trail is created.

- Synchronize the contents

  Allow the Collaboration Manager to help collaborators to synchronize the contents with their group's contents. A more detail explanation will be described in the next session. Special concern to the access to sensitive data, checking the permission is necessary before access the sensitive contents. Upon the access to the data, an audit trail is created.

2. Authorization Web Service should check permission for the collaborators to access sensitive data or take some actions.

3. Auditing Web Service should create audit trail when the collaborators join or leave a group, and when a collaborator is grand to access sensitive data or take some actions.

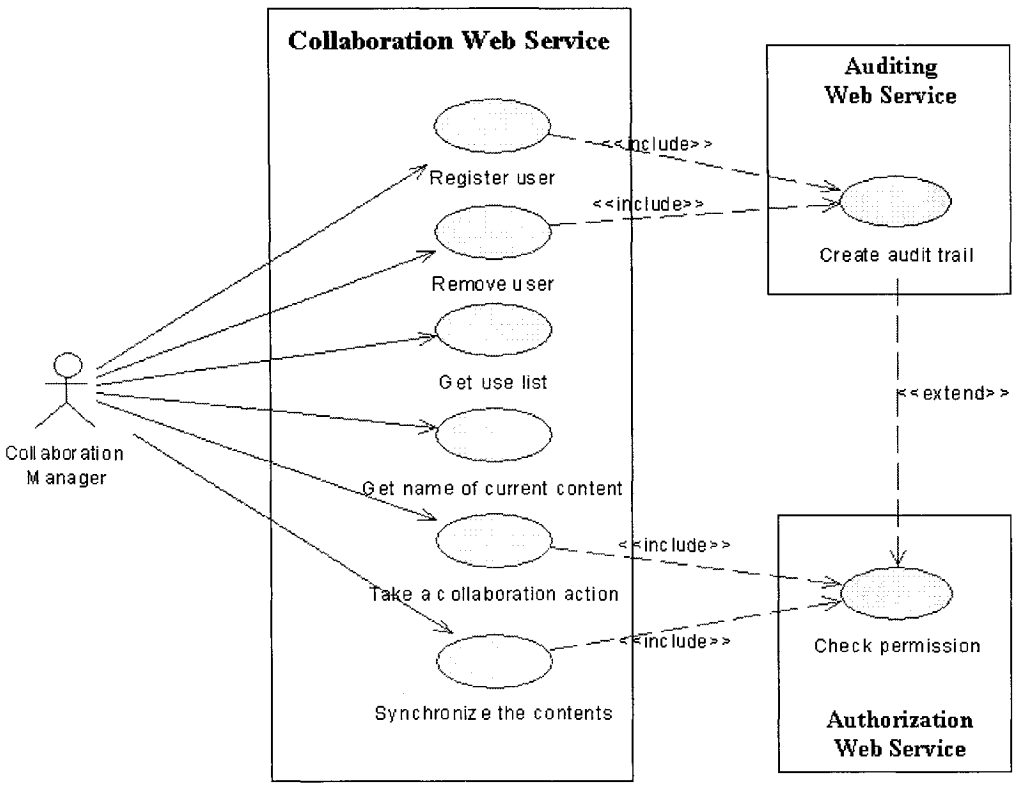


**Figure 5-8** Use Case Diagram for Web Services

### 5.4.2 Design and Implementation Considerations for Authorization Web Service

Authorization Web Service should check permission before the collaborators access sensitive data or take some actions. Some considerations are described in the following:

## 1. Dynamic Roles and Access Control Policy

This thesis takes all advantages of role-based, team-based, policy-based, and context aware access control models, leveraging with Web service, to provide a flexible, dynamic access control for sensitive data.

Users' roles are decided when the users register to the system but can be changed during collaboration. For example, in the scenario mentioned in the beginning of this chapter, the general physician, the cardiology specialist and the oncology specialist are all referred as *Doctor*s as their roles when they are registered at the medical portal. However, when the general physician creates a group named Team#1, his role becomes *Super User* in Team#1. After he invites the cardiology specialist and the oncology specialist to join, their roles become *Normal User* in Team#1. During collaboration, the general physician can modify the roles of group member such as modify the cardiology specialist's role from *Normal User* to *Super User*. Since different roles have different privilege, these doctors can perform different actions when they are in different roles. Moreover, if Team#1 exists only in a predefined time, these privileges can only be used in the term of validity. That means that the context has to be cached when evaluating an action request.

The prototype also defines the user's roles for files. These roles decide who can modify and view these files. The following is an example of definitions of roles and their policies in a team:

- Roles for a team: "Super User" and "Normal User"
    - Only "Super User" can create and delete a group
- Role for a file: "Owner", "User" and "Non-User"
    - "Owner" is the one who uploads the file before or during collaboration. It can read and write its files.
    - "User" of a file can read this file, while "Non-User" cannot.
    - "Owner" of a file can give grant or revoke right to its file.

All roles defined above can be dynamically changed during collaboration.

As for access control policy, the prototype system uses XACML. An access control policy can simply states "who can do what to what". The "who" is a subject, the "what" is an action, and the other "what" is a resource. In prototype system, terms the *subject* is a user or a process; the *action* could be a read, a write, a creation, a modification, or a deletion; and the *resource* could be a file, a url, a database record, or an API method. Examples of XACML policies used in the prototype can be seen in Appendix.

**2. Delegation of Authority**

One of the advantages of our system is to give some specific collaborators rights to perform many self-administration tasks such as changing the roles, or configuring the policies. One of the important questions we have to answer in this thesis is:

How to delegate the authority in a collaboration session? That is how does collaborator dynamically give permission to other collaborator who has no right?

A situation occurs in our scenario: the general physician wants to share a patient's record with oncology specialist who has not currently the right to read this file.

How to make it? First we have to have the following conditions:

- the general physician is allow to access this file;
- the general physician has a role "Delegator";
- the general physician has a agreement with oncology specialist, which builds a trust between then

Second we should have an interface to allow the general physician to realize this function. To do it, we design a button "Control Manager" in Session Management window of collaboration environment user interface (Figure 5.1). After the collaborator press this button, a link "Policy Management" leads the collaborator to a list of the files he can control. After selecting a file, he can see who can read this file and who cannot. At this page, he can grant or invoke a read right to other collaborators just click their name.

The following is a screen shot for dynamically configuring access rights in a collaboration session:

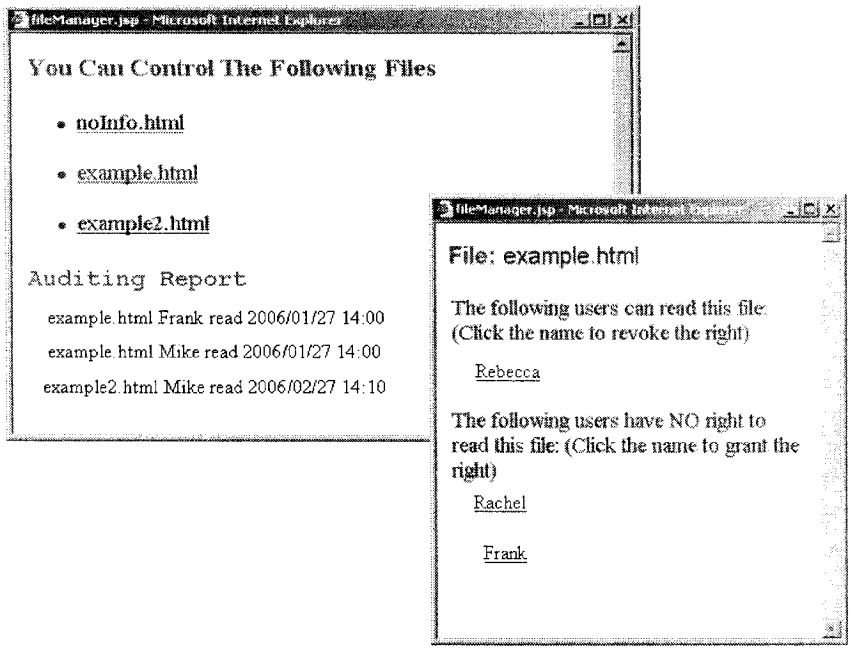


Figure 5-9 Screen Shot for Configuring Access Rights in a Collaboration Session

### 5.4.3 Design and Implementation Considerations for Auditing Web Service

All important activities such as access to data or taking collaboration actions, need to be logged in a file or a database for auditing. The following list is the summary of the content to be recorded for the prototype system:

- Chat: user name, messages, time
- Shared documents: document name, users, action (read/write), time
- Share whiteboard: image name, users, action (read/write), time
- Upload file: file name, user name, time
- Create group: group name, user, time
- Delete group: group name, user, time

### 5.4.4 Deployment Diagram for Prototype System

For simplicity, the prototype system is deployed as Figure 5.10. All components including Web services are deployed in the same Web server. Considering that the number of users in a medical consultation system is small, mostly only a few doctors work in a group; this kind of deployment is good enough.

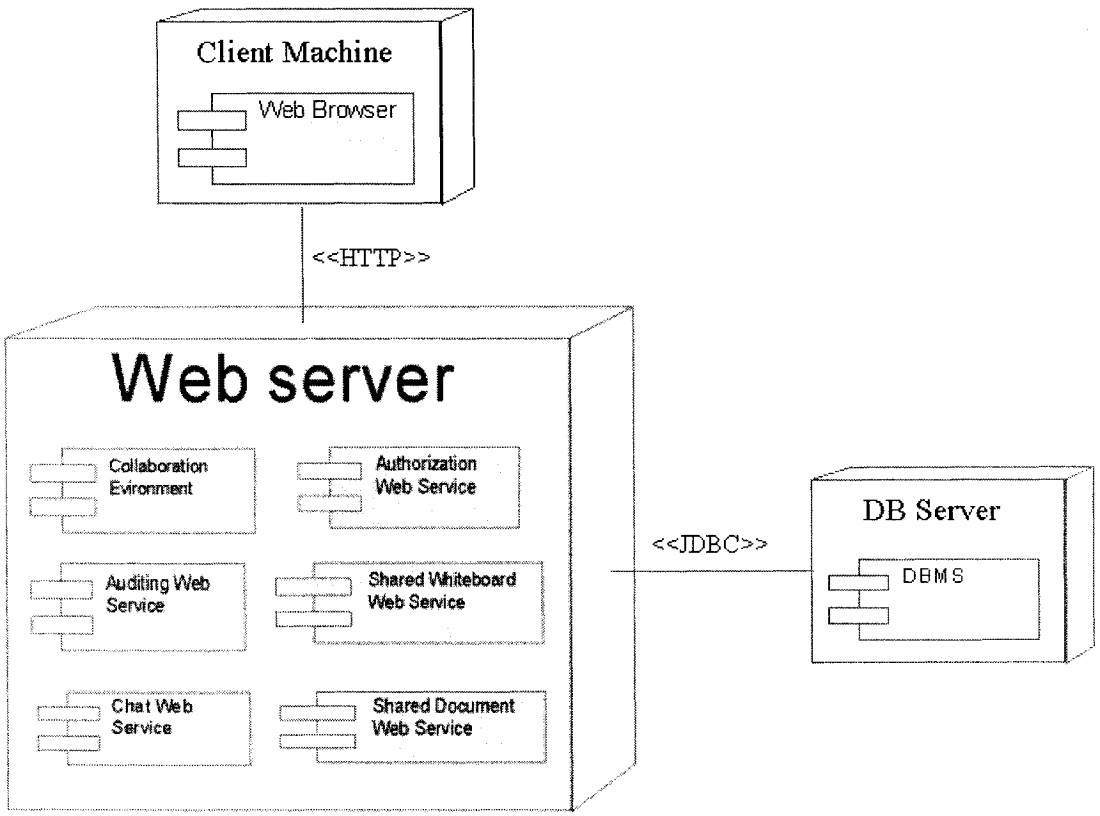


**Figure 5-10** Deployment Diagram for Prototype System

If considering a large-scale situation, an alternate deployment solution is showed as Figure 5.11. To balance server workload, each Web service is deployed in a separate

Web server. Therefore some complex computing tasks can be executed in a separately server; consequently it improves performance and scalability.

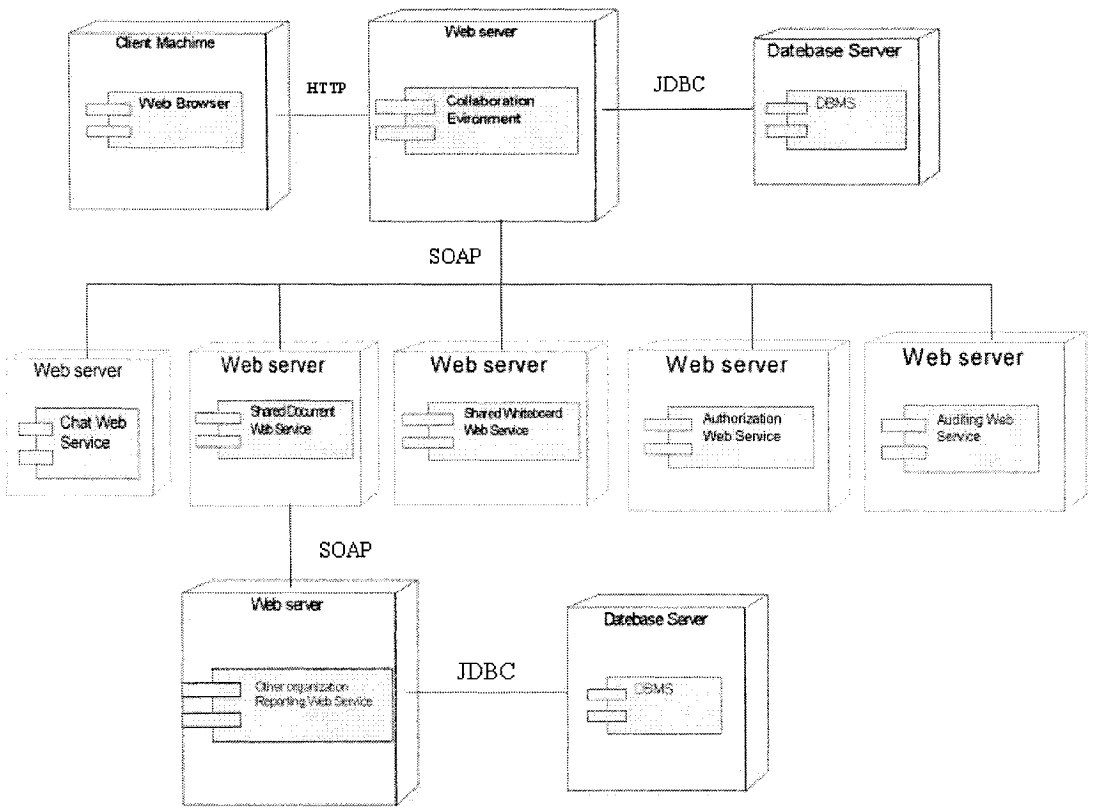


**Figure 5-11** Alternate Deployment Solution for Collaboration System

## 5.5 Summary

This chapter presents a case study, where a collaborative online consultation environment for sharing sensitive data is built on service oriented architecture. Some important issues of implement are addressed. In the next chapter, we will evaluate our system by comparing with other related works.

# Chapter 6 Evaluation and Results

In this chapter, our collaborative online medical consultation prototype system is evaluated in terms of several key characteristics that are important to collaboration environments. Our main contribution is a new architecture for collaboration environments that provides more flexible and comprehensive security. We evaluate some key characteristics of any architecture such as scalability and performance. As well, we evaluate some security requirements of a collaboration environment. But we also analyze our system in terms of interaction, coordination, awareness, distribution, flexibility, tailorability, visualization, and usability, which we described in Chapter 3.2, to see if there is a benefit (or cost) in these areas as well.

Then we compare our collaborative online medical consultation system with some of the main collaboration systems described in Chapter 2 Background and Related Work.

## 6.1 Architecture Analysis

The main character of the proposed architecture is Web-based and Web service-oriented N-tier architecture. It takes both advantages of Web service and N-tier architecture.

### 6.1.1 SOA vs. P2P

Peer-to-peer (P2P) is a communications model in which each party has the same capabilities and either party can initiate a communication session [P2P2006].

Peer-to-peer computing based architecture allows for decentralized application design, moving from centralized server models to a distributed model where each peer, independent of software and hardware platforms, can benefit and profit from being connected to millions of other peers. In such architectures, clients and servers have a lateral relationship rather than the traditional vertical relationship, giving the whole peer group tremendous processing power and storage space.

The features of a P2P system:

- User interfaces are outside of a Web browser.
- A P2P application is able to locate other peers in the network.
- A P2P application should be able to communicate with them using messages. [Samtani2002]
- A P2P application should be able to receive and provide information, such as content. [Samtani2002]
- The system supports cross network protocols like SOAP or XML.

There are quite a few common features of P2P and Web Services technologies.

- Both define a computing topology and set of communication protocols.
- Both aim to become a common stack for publishing and discovery across networks. [Samtani2002]
- Both are designed to enable loosely coupled systems often employing layered stacks and referencing best-of-breed protocols. [Schneider2001]
- Both have a heavy emphasis on distributed computing and using XML as a means to describe information. [Schneider2001]

Table 6.1 shows different features of P2P and Web Services technologies.

**Table 6-1 Different Features of P2P and Web Services**

| | P2P computing | Web Services |
|---|---|---|
| **Based model** | Decentralized model | Centralized model |
| **Tech focus** | Supplying processing power, content, or applications to peers in a distributed manner | Standardizing messaging formats and communication protocols |
| **Other features** | Robust, storage, efficiency | Interoperable, location transparency, standards |

**Why is SOA suitable for our scenario?**

An important goal in peer-to-peer networks is that all clients provide resources, including bandwidth, storage space, and computing power. In our scenario, the goal of a collaboration application is to provide an environment to allow the collaborators to share sensitive data, most of which is located in the databases in the organization or in the partner organizations. This data is delivered as Web services along with access policies. In a collaboration environment, the organizations that own the data only need to provide data by Web services to the collaborators. For the collaborators, the basic requirements are to share sensitive data, but not intend to exchange the data. Although there is a need for collaborators to share data located in their machines, it is not a main goal in our scenario.

A better solution is to use SOA in P2P networks or use P2P in a SOA architecture. This could be a challenge for future work.

### 6.1.2 N-tier Architecture vs. Client/Server Two-tier Architecture

In the n-tier model, the application components are grouped according to the type and level of the business functionality that they carry out. All of the components that encapsulate elements of business logic would be grouped together and separated from components concerned with user interaction or database communication. These

groupings would partition the application into different layers, each of which provides different kinds of services to the layers above them. [Davis2005]

N-tier architecture separates components into logical tiers and provides more flexibility in the way the application functions. For an individual application, whether or not there are other systems with sharing services, the main benefits are in the following areas. [Davis2005]

- Thin clients – being able to run the user interface on less-powerful machines
- Reduced maintenance – only making changes to the required components
- Deployment ease – being able to roll out changes more easily
- Pooling – providing shared resources which can be re-used on-demand
- Re-use – using existing components and tiers to provide services to new applications
- Support flexibility – meeting new requirements in a cost efficient manner, consequently improve availability, reliability, scalability

Client-server system is two-tier system that is less flexible comparing with N-tier system. But it is more manageable from a management perspective, so it is still playing a significant part in business [Davis2005]. Client-server systems typically have a fairly thick client, and may have business logic in the client with the server providing only database services.

### 6.1.3 Web Service-based SOA vs. Component-based Technologies

Traditionally, distributed application called for component-object technology such as the Microsoft Distributed Component Object Model (DCOM), the Object Management Group's Common Object Request Broker Architecture (CORBA), or Sun's Remote Method Invocation (RMI). These technologies provided reliable, scalable architecture to meet the growing needs of applications. [MSDN2002]

Though these component-based technologies work very well in an intranet environment, attempting to use them over the Internet presents two significant problems [MSDN2002]. First, the technologies do not interoperate. While they all dealt with objects, they disagreed over the details. Second, and more important, their focus on RPC-style (Remote Procedure Call) communication typically led to tightly coupled systems built around the explicit invocations of object methods.

Service Oriented Architecture (SOA) is the current alternate approach to design of distributed computing architectures in which software resources, viewed as unique, self-contained Services, are integrated through standards-based connections.

Through its loosely coupled, coarse-grained, standards-based approach, SOA avoids the drawbacks of traditional methods and changes the way companies can integrate information. Specific advantages include:

- Less Complexity: XML abstraction offers simplified and elegant data aggregation.
- Lower Costs: Use of standards, such as Web Services and XML, eliminate expensive customization and modeling.
- Flexible and Responsive: Quick response to constantly changing business requirements without impacting data accessibility.
- Better Distributed IT Environments: Intelligent link between business needs and technology.

Web services adapt the traditional Web programming models from all sorts of applications, not just browser-based ones. They exchange SOAP messages using HTTP and other Internet protocols. Because Web services rely on industry standards, including HTTP, XML, SOAP and WSDL, to expose application functionality on the Internet, they are independent of programming language, platforms and devices [MSDN2002].

In a collaborative environment, Web services provide the highest levels of interoperability with full support for WSDL and SOAP over HTTP.

## 6.2 Scalability Analysis

Scalability is the ability to economically support the required quality of service as the load increases. Generally, several techniques can be used to scale the system [SCALE2005]:

- Scale hardware vertically
    - Increasing capacity by upgrading hardware.
- Scale hardware horizontally
    - Increasing capacity by adding servers.
- Optimize system architecture
    - Improving server efficiency by dedicating servers to operations with similar workload factors.

From the architecture point of view, the proposed approach uses different Web services for different functionality. These Web services can be deployed in the local server or remote servers according to the workload of the server side. Multiple servers can achieve scalability of the system. Consequently increase reliability, capacity and flexibility.

The more scalable the system is the more capacity it can support. This must be traded-off against the complexity & manageability costs.

## 6.3 Performance Analysis and Test

This section analyzes the factors that affect the performance and provides details on the performance test.

### 6.3.1 Factors that Affect Performance

Performance of cross-process communication in distributed applications depends on a number of factors. Table 6-2 outlines the workload factors identified for the response time.

**Table 6-2 Workload Factors Identified for the Response Time**

| Workload Factors | Description |
|---|---|
| Hardware and Bandwidth | The hardware and the bandwidth used for passing messages between components are an important factor. They depends on devise and network quality. |
| Transport Channel | Channels including TCP and HTTP used to transport messages between applications across remote boundaries impose various amounts of overheads. In terms of raw performance, TCP sockets are more efficient than HTTP. |
| Serialization | The serialized stream can be encoded using XML, SOAP, or a compact binary representation. The SOAP Formatter has some additional overheads of generating encoded SOAP messages. |
| Message Size | Different size of messages will consume different XML parsing time, Encryption/Decryption time. So they will have different response time. |
| Message (Command) Complexity | There are different message or command complexities, e.g. a simple display html file vs. a command to draw an image. This will affect the response time. |
| HTTPS | HTTPS will do encryption and decryption to message passed through it. This process takes time according to the size and complexity of the message. It will then influence the response time. |

| | |
|---|---|
| Access Control Check | A certain overhead will always be involved in performing access control checks but this overhead should be kept at a reasonable level. |
| Logging | It takes time to do logging. If turn on or off the logging, the logging status will affect the response time. |
| Location of SOAP Server | SOAP server may locate in local or remote machine. This will affect message-passing time and then influence the response time. |
| Machine of Server | The response time could be different in Windows machine or UNIX machine. |

### 6.3.2 Test Scenarios

Latency is one of the key performance indicators. An experiment is conducted to test latency—measured as response time. As showed in the following tables (Table 6-3 to Table 6-7), the test scenarios are to simulate two users' interaction and collaboration. The scenarios are incorporated into a test script to test the system's performance. The experiment has been done for several times and average values are calculated based on data collected from all the experiments. The result (response time) is the duration to complete all steps described in "Scenario Description" in each use case table.

**Table 6-3 Use Case 1 – Two Users Chat**

| | |
|---|---|
| **Goal** | User1 types a message, and both User1 and User2 can see it |
| **Precondition** | 1. User1 and User2 are authenticated and in the same group<br>2. User1 and User2 are in "Collaboration Interface" Page |
| **Post Condition** | User1 and User2 see the same message |

| | |
|---|---|
| **Scenario Descriptions** | 1. User1 posts a message |
| | 2. Same message is displayed in User2's browser |

From the above table, the request/response time need to account two actions for this use case.

**Table 6-4 Use Case 2 – Two Users Share Files**

| | |
|---|---|
| **Goal** | User1 shares File1 with User2. |
| **Precondition** | 1. User1 and User2 are authenticated and in the same group<br>2. User1 and User2 are in "Collaboration Interface" Page |
| **Post Condition** | User1 and User2 see the same file (File1) |
| **Scenario Descriptions** | 1. User1 clicks the link to share File1 |
| | 2. User1 see File1 displayed in "Collaboration Interface" Page |
| | 3. The system checks if User2 is authorized to see File1 |
| | 4. Access to File1 is permitted for User2 |
| | 5. User2 see File1 displayed in "Collaboration Interface" Page |

From the above table, the request/response time need to account five actions for this use case.

**Table 6-5 Use Case 3 – Two Users Share Drawing**

| | |
|---|---|
| **Goal** | User1 draws a red circle in the whiteboard, and both User1 and User2 can see it in their own browser |
| **Precondition** | 1. User1 and User2 are authenticated and in the same group<br>2. User1 and User2 are in "Collaboration Interface" Page |
| **Post Condition** | User1 and User2 see the same circle in their own browser |
| **Scenario Descriptions** | 1. User1 draws a red circle in his whiteboard |
| | 2. Same circle is displayed in User2's whiteboard |

From the above table, the request/response time need to account two actions for this use case.

**Table 6-6 Use Case 4 – Two Users Share Typing in Whiteboard**

| | |
|---|---|
| **Goal** | User1 types a sentence in the whiteboard, and both User1 and User2 can see it in their own browser |
| **Precondition** | 1. User1 and User2 are authenticated and in the same group<br>2. User1 and User2 are in "Collaboration Interface" Page |
| **Post Condition** | User1 and User2 see the same sentence in their own whiteboard |
| **Scenario Descriptions** | 1. User1 types a sentence in his whiteboard |
| | 2. Same sentence is displayed in User2's whiteboard |

From the above table, the request/response time need to account two actions for this use case.

**Table 6-7 Use Case 5 – Two Users Share Image in Whiteboard**

| | |
|---|---|
| **Goal** | User1 shares Image1 with User2. |
| **Precondition** | 1. User1 and User2 are authenticated and in the same group<br>2. User1 and User2 are in "Collaboration Interface" Page |
| **Post Condition** | User1 and User2 see the same image (Image1) in their own whiteboard |
| **Scenario Descriptions** | 1. User1 clicks the link to share Image1 |
| | 2. User1 see File1 displayed in his whiteboard |
| | 3. The system checks if User2 is authorized to see Image1 |
| | 4. Access to Image1 is permitted for User2 |
| | 5. User2 see Image1 displayed in his whiteboard |

From the above table, the request/response time need to account five actions for this use case.

### 6.3.3 Machine Configuration

The following tables provide a brief summary of the test bed configuration used to perform the tests.

**Table 6-8 Client Machine Configuration**

| Client | Processor (CPU) | Processor (CPU) speed | Operation System | Memory | Hard Disk |
|---|---|---|---|---|---|
| **User1** | Pentium M | 1.6 GHz | Windows XP | 1 GB | 35 GB |
| **User2** | Pentium 4 | 1 GHz | Window 2000 | 512 MB | 20 GB |

**Table 6-9 Web Application server configuration**

| Web Application Server | Processor (CPU) | Processor (CPU) speed | Operation System | Memory | Hard Disk |
|---|---|---|---|---|---|
| **WebSphere v5.1 Testing Env.** | Pentium M | 1.6 GHz | Windows XP | 1 GB | 35 GB |

**Table 6-10 Performance Test results**

| Iteration | Use Case 1 (ms) | Use Case 2 (ms) | Use Case 3 (ms) | Use Case 4 (ms) | Use Case 5 (ms) |
|---|---|---|---|---|---|
| 1st run | 554 | 548 | 1112 | 567 | 2300 |
| 2nd run | 335 | 755 | 889 | 456 | 2234 |
| 3rd run | 901 | 632 | 997 | 890 | 1454 |
| 4th run | 878 | 1090 | 1340 | 779 | 2345 |
| 5th run | 342 | 623 | 777 | 234 | 1432 |
| 6th run | 852 | 954 | 987 | 654 | 1875 |
| Average | 644 | 898 | 1017 | 597 | 1940 |

**Result analysis:**

The overhead is reasonable and the performance is acceptable. The difference between Chat and Type is explained by the size of the message typed. The difference between ShareFile and ShareImage is the difference between the size of an X-ray image and a simple HTML file.

## 6.4 Security Analysis

The main characteristic of security feature for the proposed system is that security functions are actually Web services.

### 6.4.1 Security Role of Web Service in Communication Between Tiers

Since Web services rely on HTTP, they can easily integrate with the standard Internet security infrastructure [MSDN2002]. We can leverages the security features available with Web server to provide strong support for standard HTTP authentication schemes

including Basic, Digest, digital certificates. One advantage of using the available HTTP authentication schemes is that no code change is required in a Web service [MSDN2002]. Web server performs authentication before the Web services are called. In addition, SSL can be used to ensure private communication over the wire. Moreover, when implemented over HTTP on a standard HTTP port, web services can supports the cross-firewall environment in which most collaboration takes place without having to change any of the filter rules.

While the standard transport-level techniques to secure Web services are quite effective, some XML Web Services security specifications, example like the Security Language (WS-Security), can be used to build on the extensibility of SOAP messages to offer message-level security capabilities.

### 6.4.2 Authorization Web Service with Dynamic Access Control

Considering a security as a service, the access control decision and enforcement functionality are separated. The application provides enforcement functionality for an external access control decision. This split of enforcement and decision point has advantages for multiple enforcement points to re-use the same decision point functionality. This in turn promotes component re-use as well as the consistent application of access control decisions across an environment. Security ensures that information is neither modified nor disclosed except in accordance with the security policy. In a collaboration environment, the roles of participants can be changed dynamically and access policy can be modified frequently. Access data request have to be evaluated according to different policies.

The high interoperation of Web service allows Authorization Web service to be used in different environments, independent from programming languages, platforms and devices. The consumer application only needs to collect run time values of input parameters and pass them to the interface of Authorization Web service. A deny or permit decision will be output afterward.

Role-based access control is very powerful for his flexibility and manageability. However, it is not enough in a collaborative environment. It does not support coalition access control since it does not provide an abstraction to capture a set of collaborative users or delegations for a collaborative environment. So it is difficult to automatically coordinate effective collaboration session that requires use of resources across the enterprise.

The approach that this thesis proposes takes all advantages of role-based, team-based, policy-based, and context aware access control models, leveraging with Web service and providing a flexible, dynamic access control for sensitive data.

In summary, supposing the user has been correctly authenticated through an effective mechanism, our approach can achieve:

- Privacy – Data is disclosed to authorized entities in authorized ways
- Auditability – The system maintains logs of actions taken for later analysis
- Authority – The user can perform only allowed activities
- Integrity – Data can only be modified in allowed ways

### 6.4.3 Relationship with Existing Web Service Security Specifications

In section 2.2.5, we mentioned a set of Web service security specifications [Apshankar2002]. They are WS-Policy, WS-Trust, WS-Privacy, WS-Secure Conversation, WS-Federation, WS-Authorization, and Web service composition security [Charfi2005]. They are built on top of the WS-Security [WSSecurity2002] and secure Web services from different aspects. One important characteristic is that they are built on top of transport level security such as SSL, and provide for message level security capabilities for Web Services and Web service composition.

On the other side, the objective of this thesis is to provide an environment in application level to control information use and enable sharing the sensitive data. This system is built

on the top of message level security. It can solve the problems that cannot be solved in transport level and message level. These problems relate to sharing sensitive data and control information use in a collaboration environment. Especially, it deals with issue related to collaboration and how to extend permission to new collaborators dynamically when a user try to share information with a collaborator, when that collaborator has not yet been granted access to the information.

Therefore, the solution presented in this thesis is built on top of existing Web service security specifications. It is a complement to other security solutions.

## 6.5 Interaction, Coordination, Distribution, Awareness and Visualization

Interaction and Coordination mean that the system allows the group of users to work collaboratively in asynchronous or synchronous mode and provides a means of communication within the users. Awareness and Visualization should provide users a way to be aware of what is happening in the environment and visualize public data collaboratively in the paradigm of what-you-see-is-what-I-see. Distribution should enable people to interact from remote places.

Our prototype uses Chat, Shared Whiteboard and Shared Document as collaboration tools for interaction and coordination among the users from remote places. Under the control of the Authorization Web Service, the users can share the data or files in the paradigm of what-you-see-is-what-I-see. The users also can check the user list and shared file list to be aware of who is in their group and who can view their files. They can access the system remotely over the Internet.

## 6.6 Flexibility, Tailorability and Usability

Tailorability is the ability to change the architecture to meet new requirements in a cost-efficient manner. A tailorable system should be more maintainable in the face of changes to the environment and/or to the application itself.

The proposed architecture is flexible enough to allow enterprises to mix and match components to assemble and compose their new business. It keeps the domain-specific components dependent of each other, allowing technologist to upgrade each system without breaking the other functionality.

The prototype demonstrated this ability in a few different ways. First, in a multi-layer architecture, user interface, functional process logic, data storage and data access are developed as independent modules on separate platforms. Modifying one layer does not affect other layers. Secondly, Web service approach also helps to achieve flexibility. The prototype uses different Web services to realize different functionalities. Maintaining one Web service does not break others. Lastly expandability is supported by the architecture in much the same way it supports flexibility – the system can be expanded by simply adding a new component.

Usability is the quality of a system that makes it easy to learn, easy to use and encourages the user to regard the system as a positive help in getting the task done. A user-friendly interface and document will let the user read or play any content at will. Ensure that the system can be used remotely over the Internet, not just with a LAN.

As showed in Figure 5.2, a full functional graphic user interface is displayed after the users are authenticated and joined to a collaboration group. The different collaboration areas include chat area, shared document area, shared whiteboard area and session management area work fine when testing.

## 6.7 Comparison with Related Works

Finally, we create a comparison chart after analyzing the proposed approach and related works describe in Chapter 2. Different aspects are compared to show that the proposed solution is improved solution for a collaboration environment.

**Table 6-11 Comparison Our Solution with Related Works**

| | Proposed framework | JASMIN | Web Service CAD | CoMed | WebOnCOLL | Groove | PCASSO |
|---|---|---|---|---|---|---|---|
| **Technology Focus** | Collaboration Security | Multimedia Collaboration | Collaboration | Multimedia Teleconsultation | Multimedia Teleconsultation | Collaboration | Medical info Security |
| **Architecture** | SOA support | Client/Server | Web service | Hybrid | Open Workspace | P2P, desktop | Web-based |
| **Security Focus** | Access control audit | -- | -- | -- | -- | Encrypt account and shared spaces | End to end security |
| **Access Control Model** | Role-, team-, policy-based | -- | -- | -- | -- | End user control | Role-based |
| **Synchronous Collaboration** | √ | √ | √ | √ | √ | √ | -- |
| **Asynchronous Collaboration** | √ | -- | -- | √ | √ | √ | √ |
| **Flexibility** | √ | -- | √ | √ | -- | | -- |
| **Usability** | √ | √ | √ | √ | √ | √ | √ |
| **Scalability** | √ | -- | √ | √ | -- | √ | -- |
| **Performance** | Acceptable | Faster | Acceptable | Faster | Faster | Faster | Faster |

From the above chart, we can see the differences between related works and our system. JASMIN and Web service CAD are close to our system in terms of rich collaboration but

they do not have security information at all. PCASSO is close to our system in terms of security, but we have additional support for security and a richer collaboration. Several systems such as JASMIN, CoMed and WebOnCOLL have better performance than ours but a price is worth paying for better security as long as our performance is acceptable and most importantly scalable. Groove is most scalable and end users control their own data, but it is not suitable for our scenario where users share the sensitive data belonging to an organization.

## 6.8 Summary

This chapter evaluates our collaborative online medical consultation prototype system in terms of several key characteristics that are important to collaboration environments: flexibility, security, scalability, performance, interaction, coordination, awareness, distribution, flexibility, tailorability, visualization, and usability, which we described in Chapter 3.2. We also compare our collaborative online medical consultation system with some of the main collaboration systems described in Chapter 2 Background and Related Work. We conclude that our system provides more flexible and comprehensive security. In the next chapter, conclusion and future work will be presented.

# Chapter 7 Conclusions and Future Work

## 7.1 Conclusions

The aim of this thesis is to create a collaboration environment that provides collaboration and security. In particular, it's focused on issues that need to be improved or had not yet been adequately addressed by other systems, such as architecture and access control during collaboration.

Firstly, this thesis categorizes and investigates the requirements of a secure collaborative environment. Identifying secure and collaborative requirements enables to clearly identify the problem, consequently finding the solution to solve the problem. It turns out that the set of requirements we define for architecture and security for collaboration environments is a useful checklist. They make it possible to compare collaboration environments in a systematic fashion and provide insights into the nature of the different approaches.

Secondly, this thesis designs and demonstrates a framework to enable collaboration over Internet with flexibility and security in a Web-based, service-oriented architecture. And with our validation, we also see that scalability and performance are not compromised. In fact they are better scalable. With its loosely coupled, coarse-grained, standards-based approach, SOA provides the specific advantages of less complexity, lower costs, flexible and intelligent link among collaboration, security and technology.

Thirdly, this thesis proposes flexible control of the collaborators' interactions with local or remote contexts by allowing authorization to be evaluated by a Web service. In this approach, the knowledge is separated from control for authorization. The code recompilation or system downtime will not be triggered by modifying the roles of the users dynamically during collaboration or by modifying its policies. In addition, a Web service links applications both within and without an organization in a loosely coupled, language-neutral and platform-independent way. It enables controlling access in different situations. Moreover, the thesis mainly demonstrate that the approach makes the system

with flexibility but it still has secure access to sensitive data over the Internet, and such flexibility does not exist in other systems.

As well, this thesis develops a collaboration control model by combining team-, role- and policy- based access control models for access to resources dynamically and it's capable of self-administration in collaborative environments. By dynamic access control, each user is assigned one or more roles for its group and objects, and each role is assigned one or more permissions that can be given to users in that role. In the mean time, these roles and access policies can be modified and configured during collaboration, which enables self-administration capable.

At last, this thesis uses a collaborative online medical consultation system as case study to evaluate and illustrate the framework. We evaluate our system in terms of several key characteristics such as security, flexibility, scalability and performance, which are important to collaboration environments. We also analyze some of the general requirements: interaction, coordination, awareness, distribution, flexibility, tailor ability, visualization, and usability, and check if there is a benefit or cost in these areas as well. Our results demonstrate that the proposed framework is secure, flexible and scalable, and meets all other requirements for collaboration environments.

## 7.2 Future Work

Some future work can be done based on the results of this thesis.

This thesis has built a secure collaboration system in E-health. One of the next works could be done by integrating such secure collaborative system into the enterprise business process and extending it to the similar business scenarios. More concerns should be considered when it's integrated into the enterprise business process. For example, how to separate the accessible data from the patient's record for the collaborators' sharing? What is the format standard of the accessible data for collaborator's easy access? In addition to the collaboration scenario among doctors, similar scenario of collaboration between a doctor and a patient can be extended.

This thesis develops a collaboration control model by combining team-, role- and policy-based access control models for access to resources dynamically and self-administration capable in collaborative environments, and also it's been implemented successfully in our prototype. In the next step, it can be systematized and formulized to be a well-defined security and privacy model for collaboration environment.

In the case study of collaborative online consultation system, we implement several collaboration tools: chat, shared whiteboard and shared document window. To make the collaboration system more powerful, we can integrate other collaboration tools such as Co-editor and video conference.

Comparing with other approaches, our solution has a weakness in performance. Improving performance while taking an advantage of flexibility of Web services could be the next challenge.

# Reference

[Ahn2000] G. Ahn andR. Sandhu, "Role-based authorization constraints specification", ACM Transaction on Information and System Security, Volume 3, Issue 4. Nov.2000.

[Apshankar2002] K. Apshankar, "WS-Security: security for web services", available at www.webservicesarchitect.com/content/articles/apshankar04print.asp, last retrieved: Oct 18, 2005.

[Bardram2003] J. Bardram, R. E. Kjær, and M. Ø. Pedersen, "Context-Aware User Authentication - Supporting Proximity-Based Login in Pervasive Computing", Ubicomp 2003: 107-123.

[Bonatti2000] P. Bonatti and P. Samarati, "Regulating Service Access and Information Release on the Web", Proceedings of the 7th ACM Conference on Computer and Communications Security, November 2000, Athens, Greece.

[Bonatti2004] P. A. Bonatti, N. Shahmehri, C. Duma, D. Olmedilla, W. Nejdl, M. Baldoni, C. Baroglio, A. Martelli, V. Patti, P. Coraggio, G. Antoniou, J. Peer, N. E. Fuchs, "Rule-based Policy Specification: State of the Art and Future Work", Project deliverable D1, Working Group I2, EU NoE REWERSE, September 2004.

[Booch1999] G. Booch, J. Rumbaugh and I. Jacobson, "The unified modeling language user guide", Addison-Wesley, 1999.

[Brothers1990] L. Brothers, V. Sembugamoorthy, and M. Muller, "Icicle: Groupware for code inspection", ACM Conference on Computer-Supported Cooperative Work. Los Angeles, CA., 1990, P169–181.

[Bullock1999] A. Bullock AND S. Benford, "An access control framework for multi-user collaborative environments", ACM GROUP 99, Phoenix, AZ, 1999.

[Charfi2005] A. Charfi1 and M. Mezini, "Using Aspects for Security Engineering of Web Service Compositions", 2005 IEEE International Conference on Web Services, July 2005, Orlando, FL, USA. IEEE Computer Society 2005, pp. 59-66.

[CoMed1996] M. Zikos et al., "CoMed: Cooperation in Medicine", EuroPACS '96 Proceedings, Heraklion, Greece, 1996, pp.88-92.

[CoMed2000] M. Y. Sung, M. S. Kim, E. J. Kim, J. H. Yoo and M. W. Sung, "CoMed: a real-timecollaborative medicine system", International Journal of Medical Informatics, Issue 57,2000, pp 117–126.

[Covington2001] M. Covington, W. long, S. srinivasan, A. dey, M. ahamad, and G. D. Abowd, "Securing context-aware applications using environment roles", ACM Symposium on Access Control Model and Technology. Chantilly, VA., pages 10-12, 2001.

[CTMAC2001] C.K. Georgiadis, I. mavridis, G. pangalos, and R. thomas, "Flexible team-based access control using contexts", ACM Symposium on Access Control Model and Technology. Chantilly, VA. May 2001, pages 21-27.

[Davis2005] T. Davis, "N-tier Architecture", http://www.tessella.com/Literature/ Supplements/PDF/n-tier.pdf

[Dey2001] A.K. Dey, G.D. Abowd, and D. Salber, "A Conceptual Framework And A Toolkit For Supporting The Rapid Prototyping Of Context-Aware Applications", Human-Computer Interaction, 16, 97-166, 2001.

[Edwards1996] W. K. Edwards, "Policies and Roles in Collaborative Applications", ACM Conference on Computer-Supported Cooperative Work, page 11-20, Boston, MA. November 16-20, 1996.

[Ellis1991] l. C. A. Ellis, S. J. Gibbs, and G. L. Rein, "Groupware: Some issues and experiences", CACM, 34, 1, pp. 38-58, January 1991.

[ElSaddik2001] A. El Saddik, "Interactive Multimedia Learning – Shared Reusable Visualization-based Modules", Springer, Berlin, 2001.

[ElSaddik2000] A. El Saddik, S. Shirmohammadi, N. D. Georganas and R. Steinmetz, "JASMINE: Java Application Sharing in Multiuser Interactive Environment", IDMS'2000, October 2000.

[FAMIS2005] FAMIS Software, Inc., "Introduction to Service-Oriented Architecture", 2005, www.famis.com/downloads/ FAMIS_SOArchitecture.pdf, last retrieved: Jan 16, 2006.

[Greif1986] I. Greif, and S. Satin , "Data Sharing in Group Work", CSCW'86, pp. 175-183, Austin, Texas, USA, 1986.

[Groove2002] Groove Networks: Making P2P a Reality, http://mba.tuck.dartmouth.edu/pdf/

2002-6-0008.pdf, last retrieved: June 28, 2006

[Groove2006] Groove, http://www.groove.net, last retrieved: June 10, 2006

[Hinton2005] H. Hinton , M. Hondo, B. Hutchison, "Security Patterns within a Service-Oriented Architecture", http://searchwebservices. techtarget.com/search WebServices/ downloads/SecuritySOA_(2).pdf

[Hulsebosch2004] B. Hulsebosch, A. Salden, and Bargh, M, "Context-Based Service Access for Train Travelers", In Proceedings of the 2nd European Symposium on Ambient Intelligence (EUSAI), Markopoulos et al. (Eds.), LNCS 3295, 84-87, Eindhoven, the Netherlands, 2004.

[Hulsebosch2005] R. J. Hulsebosch, A. H. Salden, M. S. Bargh, P. W. G. Ebben, and J. Reitsma, "Applications: Context Sensitive Access Control", Proceedings of the tenth ACM symposium on Access control models and technologies, June 2005

[ISO1996] ISO/IEC 10181-3 1996(E), "Security Frameworks for Open Systems: Access control frame-work", http://www.iso.org, last retrieved: Oct 18, 2005.

[Jaeger1996] T. Jaeger, and A. prakash, 1996. "Requirements of role-based access control for collaborative systems", ACM Role-based Access Control Workshop. Gaithersburg, MD. 53–64.

[Joshi2001] J. B. D. Joshi, Walid G. Aref, Arif Ghafoor and Eugene H. Spafford, "Security Models for Web-Based Applications," Communications of the ACM, Vol. 44, No. 2, February 2001, pp. 38-72.

[Kang2001] M. H. Kang, S. S. Park, and J. N. Froscher, 2001. "Access control mechanisms for inter-organizational workflow". In ACM Symposium on Access Control Model and Technology. Chantilly, VA.

[Lampson1971] B. Lampson, , "Protection", 5th Princeton Symposium on Information Science and Systems. 437–443. Reprinted in ACM Operat. Syst. Rev. 8,1, 18–24, 1974.

[Langheinrich2001] M. Langheinrich, "Privacy by Design - Principles of Privacy-Aware Ubiquitous Systems". In In Proceedings of Ubicomp, pages 273–291, 2001.

[Liberty 2003] Liberty Alliance Project, www.projectliberty.org, 2003

[LiveMeeting2005] Microsoft, "Collaborate in real time with Microsoft office Live Meeting", http://www.microsoft.com/office/uc/livemeeting/default.mspx

[Maigre2006] R. Maigre, "Web services composition with WS-BPEL and OWL-S", http://cs.ioc.ee/~tarmo/tsem05/maigre0902-slides.pdf, last retrieved: June 12, 2006

[Mont2003] M. Mont, S. Pearson, and P. Bramhall, "Towards Accountable Management of Identity and Privacy: Sticky Policies and Enforceable Tracing Services", 8th European

Symposium on Research in Computer Security, Norway, 2003. http://www.hpl.hp.com/techreports/2003/HPL-2003-49.pdf

[MSDN2002] "ASP.NET Web Services or .NET Remoting: How to Choose", http://msdn.microsoft.com /library/default.asp?url=/library/en-us/dnbda/html/bdadotnetarch16.asp, last retrieve: June 10, 2006.

[Muirhead2000] G. Muirhead "An update on telemedicine". Patient Care 2000; 34:96-109

[MVC2002] Model-View-Controller, http://java.sun.com/blueprints/patterns/MVC.html, last retrieved June 12, 2006.

[P2P2006] Peer-to-peer, http://whatis.techtarget.com/, last retrieved June 12, 2006.

[Pan2004] Y Pan, D Xu, C Chen, and Y Zhang, "Using Web services implementing collaborative design for CAD systems", Services Computing, 2004. (SCC 2004). Proceedings, 2004 IEEE International Conference on 15-18 Sept. 2004 Page(s):475 – 478.

[PCASSO2003] "Patient-Centered Access to Secure Systems Online (PCASSO)", http://www.himss.org/content/files/CPRIToolkit/version4/pdf/4.9.3.pdf, last retrieved: Oct 18, 2005.

[Peyton2004] L. Peyton, and M. Nozin, "Tracking Privacy Compliance in B2B Networks", The proceedings of the Sixth International Conference on Electronic Commerce, 2004

[Peyton2005] L. Peyton, J. Hu, H. Liu, M. Al-Saleh, and A. El Saddik, "A Collaborative Approval Process for Accessing Sensitive Data ", accepted by International Journal of Computer Applications in Technology (to appear), 2005.

[PIPEDA2000] "The Personal Information Protection and Electronic Documents Act" , Department of Justice, Canada, 2000. http://e-com.ic.gc.ca/ epic/internet/inecic-ceac.nsf/ vwGeneratedInterE/h_gv00045e.html, last retrieved: Oct 18, 2005.

[Price2004] E. Price. "An Overview of Secure Collaboration", Software Business EXECUTIVE REPORT, March 2004.

[Reinhard1994] W. Reinhard, J. Schweitzer and G. Volksen, "CSCW Tools: Concepts and Architectures", IEEE Computer, Volume 27, Issue 5, May 1994 Page(s):28 – 36.

[RFC2396] "Uniform Resource Identifiers (URI): Generic Syntax", Internet RFC 2396, 1998. http://www.ietf.org/rfc/rfc2396.txt, last retrieved: Oct 18, 2005.

[SameTime2005] IBM, "IBM Lotus Sametime", www.lotus.com/products/product3.nsf/wdocs/ homepage, last retrieved: Jan 26, 2006

[Samtani2002] G. Samtani and D. Sadhwani, "Web Services and Peer-to-Peer Computing" http://www.webservicesarchitect.com/content/articles/samtani05.asp, last retrieved: June 10, 2006

[Sandhu1996] R. S. Sandhu,E. J. Coyne, H. J. Feinstein, and C. E. Youman, "Role-based access control models". 1996. IEEE Computer 29, 2 (Feb.), 38–47.

[SCALE2005] Microsoft, "Commerce Server 2000 Resource Kit: -- Ch. 5, Planning for Scalability", www.microsoft.com/technet/prodtechnol/comm/comm2000/reskit/csrkch5.mspx, last retrieved: Oct 18, 2005

[Schneider2001] J. Schneider, Convergence of Peer and Web Services, http://www.openp2p.com/

pub/a/p2p/2001/07/20/convergence.html, last retrieved: June 10, 2006

[Shen1992] H. Shen, P. Anddewan,. "Access control for collaborative environments". In ACM Conference on Computer-Supported Cooperative Work. 1992.

[Sikkel1997] K. Sikkel,. "A group-based authorization model for cooperative systems". In ACM Conference on Computer-Supported Cooperative Work.345–360. 1997

[Simone1998] C. Simone, and K. Schmidt, 1998, "Taking the distributed nature of cooperative work seriously ", Proceedings of 6th Euromicro Workshop on Parallel and Distributed Processing, Madrid, Spain, IEEE Press, pp. 295-301.

[Stiemerling1999] O. StiemerlingR. Hinken, A. B. Cremers, "Distributed Component-Based Tailorability for CSCW Applications, in: Proceedings of the ISADS '99", IEEE Press, Tokyo, Mar. 20-23, 1999, pp. 345-352.

[SOAP2003] W3C, "SOAP Version 1.2", http://www.w3.org/TR/soap12, last retrieved: Oct 18, 2005.[Taylor2003] K. Taylor and J. Murty, "Implementing Role Based Access Control for Federated Information Systems on The Web," Proceedings of the Australasian Information Security Workshop, January 2003, Adelaide, Australia.

[Thomas1997] R. K. Thomas, "Team-based Access Control (TMAC): A Primitive for Applying Role-based Access Controls in Collaborative Environments", November 1997, Proceedings of the second ACM workshop on Role-based access control

[Tolone2005] W. Tolone, G. Ahn, and T. Pai, "Access Control in Collaborative Systems", ACM Computing Surveys, Vol. 37, No. 1, March 2005, pp. 29–41.

[UDDI2004] OASIS, "UDDI Version 3.0.2", http://uddi.org/pubs/uddi_v3.htm, last retrieved: Oct 18, 2005

[Versata2004] Whitepapers, "Understanding service-oriented architecture",

http://www.versata.com/documents/wp-SOA20041015.pdf, last retrieved: Oct 18, 2005.

[Wang1999] W WANG,. "Team-And-Role-Based Organizational Context And Access Control For Cooperative Hypermedia Environments". In ACM Hypertext. 1999.

[Weaver2004] A. C. Weaver, "Enforcing Distributed Data Security via Web Services ".

[WebOnCOLL1997] C.E., Chronaki, Katehakis, D.G., Zabulis, X.C., Tsiknakis, M., and Orphanoudakis, S.C., "WebOnCOLL: Medical collaboration in regional healthcare networks",

IEEE Transactions on Information Technology in Biomedicine Dec 1(4): 257-69

[WebOnCOLL2002] M. Tsiknakis, P. Lelis, C. Chronaki, "HL7 CDA in HYGEIAnet", the Regional Health Information Network of Crete, HL7 International CDA Conference October 7-9, 2002.

[WS2004] W3C Working Group, "Web Services Architecture", Note 11 February 2004, http://www.w3.org/TR/ws-arch, last retrieved: Oct 18, 2005

[WSArch2003] W3C working group, "Web Services Architecture", August 2003. http://www.w3.org/TR/2003/WD-ws-arch-20030808, last retrieved: Oct 18, 2005

[WSBPEL2005] OASIS, "Web Services Business Process Execution Language Version 2.0", September 2005, http://www.oasis-open.org/committees/download.php/14616/wsbpel-specification-draft.htm

[WSDL2001] "Web Services Description Language (WSDL) 1.1", http://www.w3.org/TR/wsdl, last retrieved: Oct 18, 2005

[WSSecurity2002] IBM, Microsoft. Security in a Web Services World: A Proposed Architecture and Roadmap, April, 2002, last retrieved: June 10, 2006

[WSSecurity2003]] OASIS, Web Services Security: SOAP Message Security. WSS TC Working Draft, August, 2003, last retrieved: June 10, 2006

[WSTrust2004] BEA, et. Al., Web Services Trust Language (WSTrust), May, 2004, last retrieved: June 10, 2006

[WSFederation2003] IBM, et Al, Web Services Federation Language (WSFederation), July, 2003, last retrieved: June 10, 2006

[XACML2004] "Sun's XACML Implementation", http://sunxacml.sourceforge.net/, 2004 , last retrieved: Oct 18, 2005

[XML2004] "Extensible Markup Language (XML) 1.0 (Third dition)", 2004, http://www.w3.org/TR/2004/REC-xml-20040204/, last retrieved: Oct 18, 2005

# Appendix: XACML Policy Examples in Prototype

## Permission Policy set for superUser in team 1

```
<?xml version="1.0" encoding="UTF-8"?>

<PolicySet
        xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
        http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd"
        PolicySetId="thesis:superUserPermission:team1"
        PolicyCombiningAlgId=
                "urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides">

 <Description>
        Permission Policy set for superUser in Team 1
 </Description>

 <Target>
  <Subjects>
   <Subject>
    <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue  DataType="http://www.w3.org/2001/XMLSchema#string">
                superUserTeam1</AttributeValue>
     <SubjectAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
                DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </SubjectMatch>
    </Subject>
  </Subjects>
  <Resources>
      <AnyResource/>
```

```
    </Resources>
     <Actions>
          <AnyAction/>
      </Actions>
  </Target>

<PolicyIdReference> thesis: team1:files </PolicyIdReference>

<Policy
     PolicyId="Thesis:group:team1"
     RuleCombiningAlgId=
                    "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">

   <Description>
          Policy applies to requests on a group named "team 1"
   </Description>

   <Target>
       <Subjects>
            <AnySubject/>
       </Subjects>
       <Resources>
            <Resource>
                <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue DataType=
                        "http://www.w3.org/2001/XMLSchema#string">team 1</AttributeValue>
                  <ResourceAttributeDesignator
                        DataType="http://www.w3.org/2001/XMLSchema#string"
                        AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
              </ResourceMatch>
          </Resource>
       </Resources>
       <Actions>
            <AnyAction/>
       </Actions>
   </Target>
```

```
<Rule RuleId="thesis:team1:create" Effect="Permit">

<Description>
        A super user may create a group named "team 1" during a predefined period
</Description>

 <Target>
   <Subjects>
    <Subject>
      <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType=
               "http://www.w3.org/2001/XMLSchema#string">superUserTeam1</AttributeValue>
          <SubjectAttributeDesignator
               AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
               DataType="http://www.w3.org/2001/XMLSchema#string"/>
       </SubjectMatch>
    </Subject>
  </Subjects>
  <Resources>
    <Resource>
     <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
         <AttributeValue DataType=
                    "http://www.w3.org/2001/XMLSchema#string">team 1</AttributeValue>
         <ResourceAttributeDesignator
                    DataType="http://www.w3.org/2001/XMLSchema#string"
                    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
     </ResourceMatch>
     </Resource>
   </Resources>
   <Actions>
    <Action>
      <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType=
                  "http://www.w3.org/2001/XMLSchema#string">create</AttributeValue>
          <ActionAttributeDesignator
                  DataType="http://www.w3.org/2001/XMLSchema#string"
                  AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>
```

```
      </ActionMatch>
    </Action>
  </Actions>
</Target>

<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-greater-than">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">
      <EnvironmentAttributeDesignator
          DataType="http://www.w3.org/2001/XMLSchema#date"
          AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"/>
    </Apply>
    <AttributeValue DataType=
          "http://www.w3.org/2001/XMLSchema#date">2006-01-01</AttributeValue>
  </Apply>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-less-than">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">
      <EnvironmentAttributeDesignator
          DataType="http://www.w3.org/2001/XMLSchema#date"
          AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"/>
    </Apply>
    <AttributeValue DataType=
          "http://www.w3.org/2001/XMLSchema#date">2006-02-01</AttributeValue>
    </Apply>
  </Condition>
</Rule>

<Rule RuleId="thesis:team1:delete" Effect="Permit">
  <Description>
      A super user may delete a group named "team 1"
  </Description>
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType=
              "http://www.w3.org/2001/XMLSchema#string">superUserTeam1</AttributeValue>
```

```
            <SubjectAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
                DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <Resource>
       <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType=
                    "http://www.w3.org/2001/XMLSchema#string">team 1</AttributeValue>
            <ResourceAttributeDesignator
                    DataType="http://www.w3.org/2001/XMLSchema#string"
                    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
        </ResourceMatch>
       </Resource>
     </Resources>
     <Actions>
      <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
         <AttributeValue DataType=
                "http://www.w3.org/2001/XMLSchema#string">delete</AttributeValue>
         <ActionAttributeDesignator
                DataType="http://www.w3.org/2001/XMLSchema#string"
                AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>
        </ActionMatch>
       </Action>
     </Actions>
    </Target>
  </Rule>

  <Rule RuleId="Thesis: finalRule" Effect="Deny"/>

</Policy>
</PolicySet>
```

## Access Policy for files located in /collaborationApp/team1/*

```
<?xml version="1.0" encoding="UTF-8"?>

<Policy
        xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
        http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd"
        PolicyId=" thesis: team1:files"
        RuleCombiningAlgId=
                "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">

  <Description>
        Access Policy for files located in /collaborationApp/team1/
  </Description>

  <Target>
   <Subjects>
     <AnySubject/>
   </Subjects>
   <Resources>
     <Resource>
      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                  /collaborationApp/team1/*</AttributeValue>
           <ResourceAttributeDesignator
                  DataType="http://www.w3.org/2001/XMLSchema#string"
                  AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
       </ResourceMatch>
     </Resource>
   </Resources>
   <Actions>
     <AnyAction/>
   </Actions>
  </Target>
```

```
<Rule RuleId="thesis:team1:files:read:1" Effect="Permit">
 <Description>
       A normalUser in team 1 may read a file in the directory match as
       /collaborationApp/team1/*
 </Description>
 <Target>
  <Subjects>
   <Subject>
     <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                 normalUserTeam1</AttributeValue>
           <SubjectAttributeDesignator
                 AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
     </SubjectMatch>
   </Subject>
 </Subjects>
 <Resources>
   <Resource>
    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                 /collaborationApp/team1/*</AttributeValue>
      <ResourceAttributeDesignator
                 DataType="http://www.w3.org/2001/XMLSchema#string"
                 AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
     </ResourceMatch>
   </Resource>
</Resources>
<Actions>
   <Action>
     <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue DataType=
             "http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
        <ActionAttributeDesignator
             DataType="http://www.w3.org/2001/XMLSchema#string"
             AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>
```

```
      </ActionMatch>
    </Action>
  </Actions>
 </Target>
</Rule>

<Rule RuleId="thesis:team1:files:read:2" Effect="Permit">
 <Description>
       A superUser in team 1 may read a file in the directory match as
       /collaborationApp/team1/*
 </Description>
 <Target>
  <Subjects>
   <Subject>
     <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                    superUserTeam1</AttributeValue>
           <SubjectAttributeDesignator
                    AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
                    DataType="http://www.w3.org/2001/XMLSchema#string"/>
     </SubjectMatch>
   </Subject>
  </Subjects>
  <Resources>
   <Resource>
     <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                    /collaborationApp/team1/*</AttributeValue>
           <ResourceAttributeDesignator
                    DataType="http://www.w3.org/2001/XMLSchema#string"
                    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
     </ResourceMatch>
   </Resource>
 </Resources>
 <Actions>
   <Action>
     <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
```

```
            <AttributeValue DataType=
                        "http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
            <ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
                        AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>
        </ActionMatch>
       </Action>
     </Actions>
    </Target>
  </Rule>

  <Rule RuleId="Thesis: finalRule" Effect="Deny"/>

</Policy>
```