Implementing Web-based e-Health Portal Systems

Shuo Lu, Yuan Hong and Qian Liu and Lingyu Wang and Rachida Dssouli

Department of Computer Science and CIISE, Concordia University

Abstract

As an emerging form of enabling technology, Web-based e-Health portals provide patients easier accesses to their healthcare information and services. We design and implement such an e-Health portal which can integrate many backend medical services effectively. A major challenge in designing such a system is to meet critical security requirements, such as the confidentiality of patient data, the integrity of diagnosis results, and the availability of healthcare services. In this thesis I address the issue from the access control perspective. More specifically, I first propose a two-tier approach to access control for e-Health portals. The approach supplements existing Role Based Access Control (RBAC) capabilities with a rule-based access control module based on the classical Flexible Authorization Framework (FAF) model. I study conflict resolution and interaction between the two modules. I also address authentication for real-time services provided by remote service providers.

1 Introduction

1.1 Background

"e-Health is an emerging field in the intersection of medical informatics, public health and business, referring to health services and information delivered or enhanced through the Internet and related technologies" [1].

The healthcare industry is undergoing fundamental changes. Examples of such changes include a shift from hospital-centric services to a more ambulatory system (with homecare, day care clinics, and so on) and the treatment of chronic diseases that actively involves the patient himself/herself [2]. The emergence of Web-based e-Health portals is a natural result of such changes because such portals provide patients and healthcare professionals easy accesses to information no matter where they are. According to a recent survey, most patients say they are very interested in and capable of accessing healthcare information and services via a Web-based portal system [1].

1.2 Motivation

The design of e-Health portal is, however, particularly challenging due to its unique functionality and security requirements. First, a traditional design of portal systems will encounter difficulties in integrating heterogeneous e-Health services implemented with different technologies. The complexity of such integration will make it difficult to extend an existing system with new services. Second, a general purpose Web-based portal usually cannot meet the security requirements of an e-Health portal system because the consequence of a security breach is far more serious in the latter. For example, an inappropriate disclosure of patient data will lead to privacy breaches and legal issues, whereas an improper modification to diagnosis results or a denial of critical healthcare service may threaten a patient's health or even his/her life.

We address the above issues through the design and implementation of a secure Web-based e-Health portal. To meet the functional requirements, we adopt a service-oriented approach to the design of our portal. We then tackle various security issues involved in such a design. More specifically, we outline our solutions for authentication and authorization of users for local and remote services in different operating

modes, for trust management between patients and doctors using PKI and biometrics, and for preserving patients' privacy through preference negotiation and database technology. We also discuss implementation issues of the proposed portal system.

1.3 Contributions

My contributions concentrate on the system architecture design and access control of e-Health portal systems. In the architecture design, I adopt Service-Oriented Architecture-based three-tier architecture, which includes portal server, portlet container, and service container. In the design of access control, I design a two-tier access control mechanism, which combines traditional role-based access control with rule-based access control [3][4][5][6].

2 Related Work

This section describes the background related to our e-Health portals. Related work of e-Health systems and access control model are also illustrated.

2.1 Service-Oriented Architecture

Service-Oriented Architecture (SOA) provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations [7]. Many enterprise architects believe that SOA can help businesses respond more quickly and cost-effectively to changing market conditions [8].

SOA is a software system architecture, which enables various applications' components (exposed as services) to communicate with each others via well defined programming interfaces. Those interfaces are hardware platform, software platform and programming language-independent. The most popular SOA implementation has been using the Web services protocol stack. Such services are generally described by platform-independent XML documents, i.e. the Web Services Description Language (WSDL) which will be illustrated in the next section. The services communicate with each other in messages defined in XML since service consumers and providers may reside in heterogeneous environments, and they may know nothing about each other. Services are generally published to a registry by service providers, and consumers can discover and invoke the services regardless of their underlying implementations.

Besides Web Service, there are many other protocols that can be used to implement the SOA. Such as DCOM (Distributed Component Object Model, which is Microsoft specific), RMI(Remote Method Invocation) and JINI (which are Java specific), and OMG's CORBA (Common Object Request Broker Architecture, which is platform and language independent). CORBA has been successfully applied in various areas ranging from telecommunications, e-commerce, to healthcare etc. However, the biggest challenge faced by CORBA is that it is hard to find a unique RPC middleware to support all the programming languages and all the platforms at a reasonable price [9]. CORBA only supports for UNIX and Windows platform, and Visual Basic does not provide any support for CORBA and therefore limits its usage.

2.2 Web Services

Web services provide a standard means of interoperating between different software applications running on a variety of platforms and/or frameworks [10]. Web services are applications that expose their business logic, data and processes through programmatic interface. Unlike traditional Client/Server models, Web services generally use HTTP as the underlying communication protocol which allows messages to go through most of the firewalls (most of the firewalls' setting allow the access to HTTP port). Web services do not provide users with GUI (Graphic User Interface). However, developers can add Web services into their Web page or applications and offer users with GUI which contains the functionalities of the Web services.

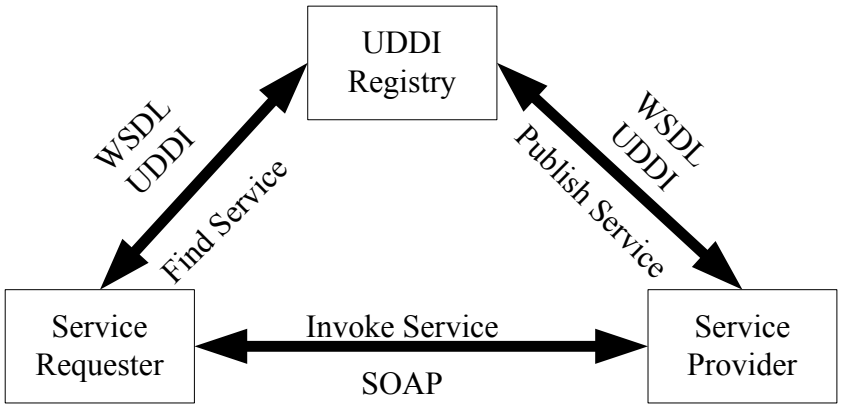

Figure 1: Web Services Architecture

The major advantages of implementing an SOA using Web services are that Web services are very simple and programming language and platform-independent. Moreover, there are many extended Web services specifications ranging from transactions, business process, messaging, transport, security, metadata to performance, which are on their way toward standardization. These provide enterprise-level integration in a standard way for services to interoperate with each other without any incompatibility and at the same time ensures the QoS and other requirements. As the three core specifications of the Web services protocol stack, SOAP, WSDL and UDDI form the initial specifications for Web services. Figure 1 describes the architecture of Web services. In the following, each of them will be introduced in details.

2.2.1 SOAP

SOAP is a protocol for exchanging XML-based messages between peers over computer networks. It provides a standard way to access Web services. SOAP is the foundation of the Web services protocol stack and provides the basis upon which other abstract layers are built. It generally uses the HTTP/HTTPS (other protocols like SMTP and XMPP can also be used) as its underlying protocol. The major advantage of using SOAP over other distributed protocols like GIOP/IIOP or DCOM is that SOAP with HTTP works well with network firewalls, whereas other protocols are normally blocked by firewalls.

SOAP 1.1 was proposed to W3C in May 2000 [10]. Currently, W3C is working on SOAP 1.2. A SOAP message is an XML document which contains the following XML elements: Envelop, Header, Body and Fault. The SOAP Envelop element is the root element of a SOAP message. It is used to identify that the XML document is a SOAP message and encapsulates all other XML elements. The SOAP Header element is optional and may contain application specific information like security features. The SOAP Body element is required and contains call and response information. The SOAP Fault element is optional and it represents error information when processing the message. Each Fault element must contain a faultCode element and a faultString element. The former one represents the code of the error and the latter one is used to provide the detailed information of the error.

Figure 2 depicts how SOAP messages exchanged between SOAP client and server. A SOAP client (also termed as service requester) is an application that creates a SOAP message containing the information needed to invoke remote methods in heterogeneous environments. It could be a traditional program, Web services or any other server-based applications (such as servlet, portlet etc.). A SOAP server (also termed as service provider) is a program generally resides in the Web server that listens, distributes and interprets the received SOAP messages. In Figure 2, the SOAP client sends a request (a SOAP message) to the SOAP server via HTTP/HTTPs. After receiving this request, the SOAP server sends it to the SOAP processor which resides in the server side and acts as an evaluator to validate this request against the XML schema. If it is valid, the SOAP processor invokes the Web service. Then the Web service processes this request and

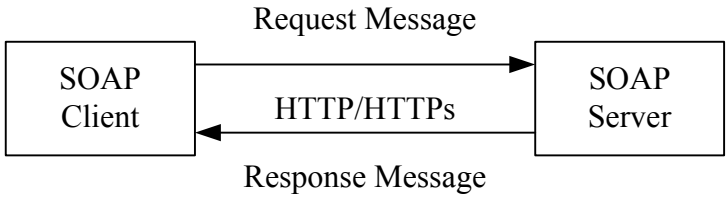

Figure 2: Exchanging Message with SOAP

creates a response. After that, the SOAP processor wraps this response with SOAP message format and sends the response SOAP message back to the SOAP client over HTTP/HTTPs. Like the SOAP server, the SOAP processor in the SOAP client will parse and validate this response message and present the result to the SOAP client user interface.

2.2.2 WSDL

The WSDL (Web Services Description Language) is a specification recommended by W3C [10]. The current draft version is 2.0. It is an XML-based language used for describing Web services. Each WSDL document contains five major elements which describe three aspects of Web services. The types, message and portType elements are used to describe what tasks the service provides. The types element defines the data type used by Web services (Generally, for maximum platform independent purpose, WSDL uses XML Schema syntax to define data types). The message element defines the data elements within each operation. These data elements can be compared to the parameters of a function call in traditional programming languages. The portType element describes the operations that a Web service can perform. It can be compared to a function in traditional programming languages. The binding element defines the message format and underlying transport protocol details for each port and the service element represents the location of the service. The main purpose of the WSDL is to provide human readable and machine interpretable information about the service.

2.2.3 UDDI

UDDI (Universal Description, Discovery and Integration), sponsored by OASIS [7], is a platform-independent, XML-based registry, which enables Web services implementations to be published and discovered either within or between enterprises. A UDDI business registration contains three components: White Pages (describes basic information about the service provider, such as address, contact, and other identifiers), Yellow Pages (describes services by industry categorizations, service type or geography information) and Green Pages (describes technical information about services, such as interfaces and URL locations etc.).

UDDI eases enterprises to create a platform-independent and open architecture for quickly discovering and publishing businesses and services via Internet. Enterprises can describe and publish their services to the UDDI registry. Once an enterprise finds a potential business partner, they can quickly and easily begin trading. As shown in Figure 1, the service provider may publish its WSDL document in the UDDI registry and the service requester may retrieve it by searching the UDDI registry according to the keywords described in the above pages. When the service requester obtains the WSDL document for a service, it may invoke it according to the elements described in the WSDL document.

2.3 Portal

With the expanding of enterprise and business, it is necessary to have a centralized application that can integrate various applications and systems within the same place to share the resources. It also provides various users with a single point to access all of the applications over the Internet. All these can be achieved through portals. A portal lets users view each application or Web page in its own window, called a portlet,

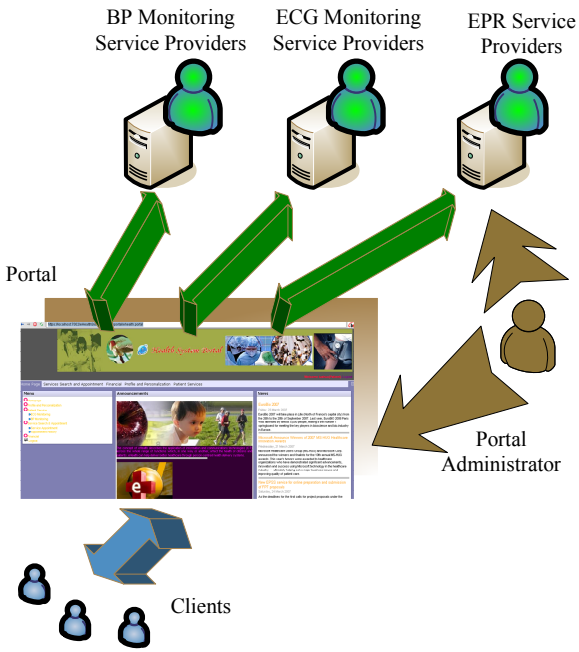

Figure 3: A Simple Portal Platform

and a single browser window can contain one or multiple portlets. It is a collection of resources in an enterprise application that can be displayed in customizable, personalized, and audience-specific views called desktops [11].

As the basic unit of portal, portlets are pluggable user interface components that are managed and displayed in a portal. Portlets produce markup fragments and then these fragments are aggregated into a portal page. Typically, a portal page contains a collection of portlet windows, where each of them displays a portlet. Thus a portlet resembles a Web-based application (such as a JSP, HTML page, Java Page Flow or Web services etc.) that is hosted in local or remote portal servers. The Java Portlet Specification (JSR168) is a standard that enables interoperability for portlets between different portals. This specification defines a set of APIs for interaction between the portlet container and the portlet addressing the areas of personalization, presentation and security [12]. Apache Pluto [13] is a reference implementation of JSR168. Also, there are many other vendors (such as BEA, IBM, SUN, Oracle etc.) provide commercial implementations of the standard-compatible portlet container.

As shown in Figure 3, it is a simple portal platform in which the portal integrates multiple backend medical systems. The interfaces of these systems are represented in a single browser window to clients. Administrators resided in the portal assemble, configure and manage the portal via portal administration tools.

Some characteristics and benefits of portal solution are listed as follow:

- Intelligent resources integration: Integration of various enterprise applications, services and processes together is key to developing an environment that fully supports business processes.
- Rendering content on different devices: By using portal technology, content can be delivered to and displayed on multiple devices (such as PC, PDAs, smart phones etc.).
- Personalization: The ability to serve dynamic response to the user based on personal profiles.
- Rapid, easy management of Web content: Portal administrator may easily modify, add or remove content from the portal.

- Single Sign-on: Portal is a place where a user can authenticate once and gain access to the resources of multiple backend systems.
- Federation: This feature enables a portal to include remotely distributed resources in a standard way. It also reduces the cost for maintenance, testing and deployment and increases the reuse of portal components and interoperability.

2.4 Flexible Authorization Framework

Many access control policies have been proposed and are widely applied in information systems. However, in practice, only a specific policy (such as closed policy or open policy) can be applied in a system, which cannot satisfy increasing access control requirements. Thus, Flexible Authorization Framework (FAF) [14] is proposed to allows multiple access control policies to be applied within a single system. FAF allows positive, negative and hybrid (both positive and negative) policies to be specified to allow or deny an access in a powerful, declarative and flexible way. It also employs meta-policy to solve the conflict resolution problem when hybrid policy is specified.

FAF employs four tiers of policies to manage access control [15]. The first stage includes the description of subject and object hierarchies, and a set of authorizations according to the protection requirements. In this stage, conflict problems may occur since both positive and negative authorizations may be derived for executing an action on a given object, thus a subject could be authorized and denied to perform the action at the same time. To solve this problem, in the second stage conflict resolution policies are enforced. However, it is possible that access is neither authorized nor denied. Thus, the third stage employs the decision policies to make final decisions. Finally, integrity rules are used to identify and remove errors that may arise in authorization specification.

Based on this framework, security administrators may use the Authorization Specification Language (ASL), a stratified first-order logic language, to specify access control rules according to the protection requirements. ASL syntax includes a set of stratified predicates. The level of these predicates is corresponding to the stages stated above.

2.5 Kerberos

Kerberos is a network authentication protocol that is based on the Needham-Schroeder protocol. It is designed to provide strong authentication for client/server applications by using symmetric key cryptography [16]. The core of the Kerberos is the Key Distribution Center (KDC), a trusted third party which contains two logical parts, i.e. an Authentication Server (AS) and a Ticket Granting Server (TGS). AS maintains a database to store the credentials of users and issues users with Ticket-Granting Ticket (TGT) upon successful authentication. TGS is used to issue users with Client-to-Server ticket upon receiving valid TGT. Kerberos also enables mutual authentication (i.e. both the user and the server verify each other's identity). A free implementation of this protocol is available from the Massachusetts Institute of Technology [16]. Figure 4 is a simplified description of Kerberos Protocol. It contains three phases.

Phase A: The Authentication Service Exchange

1. The client sends a plaintext which contains the identities of the client and the server to the AS.

2. The AS checks the identity of the client. If it is valid, the AS sends back the client two messages: the client/TGS session key (shared between the client and TGS) encrypted using the user secret key and the TGT (which contains the client ID, client IP address, ticket validity period, and the client/TGS session key) encrypted using the secret key of the TGS. The user can get the session key by using his/her secret key to decrypt the first message.

Phase B: The Ticket-Granting Service Exchange

3. When requesting services, the client sends two messages to the TGS: the first one contains the TGT and the ID of the requested service and the second one is an Authenticator (which contains the client identity and a timestamp), which is encrypted by the client/TGS session key.

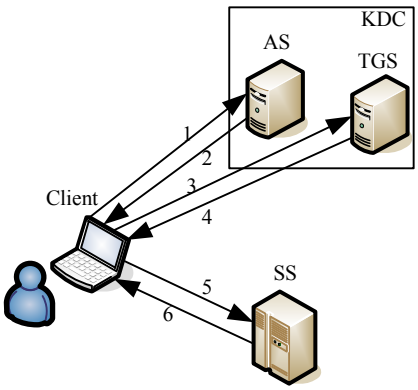

Figure 4: Simplified Description of Kerberos Protocol

4. Upon receiving the above two messages, the TGS decrypts the Authenticator using the client/TGS session key and sends the following two messages to the client: client-to-server ticket (which includes the client identity, client ip address, validity period and client/server session key) encrypted using the service's secret key and client/server session key encrypted with the client/TGS session key.

Phase C: The Client/Server Authentication Exchange

5. In order to use the service, the client sends the following two messages to the Service Server (SS): the first one is the client-to-server ticket which is encrypted using service's secret key and the second one is an Authenticator (which contains the client identity and a timestamp) encrypted using client/server session key.

6. Upon receiving the client-to-server ticket, the SS decrypts it using its secret key and increases the timestamp within the Authenticator by 1 and then encrypts it using the client/server session key and reply to the client.

Finally, the client decrypts the reply using the client/server session key and checks the value of the timestamp. If it is increased by 1, the client can begin to send the service request to the server.

2.6 Telemedicine and e-Health Systems

The increasing demand for e-Health services has led to many research efforts [17]. Different e-Health or other healthcare-related systems have been designed and implemented. Some of them are used in special areas, such as trauma [18], cardiology [19], neurosurgery [20], pathology treatment [21] etc. Some are used with special purposes, such as emergency [22] [23], aeronautic cure [24], marine purpose [25], patient monitoring [26] etc. With the advantages of wireless technologies, there are also many wireless-based e-Health systems (i.e. m-Health system) emerging [27]. For complete surveys of m-Health, refer to [28] [29] [30]. The common feature of these systems is that they only provide limited or special services to users or as the complement of the e-Health industry. Our goal is not to implement an e-Health system used in a special area or for some purposes. Instead, we concentrate on proposing a framework for general e-Health systems, which can integrate most of the existing healthcare applications, modules or other healthcare related systems.

Many architectures have been proposed for telemedicine and e-Health system. In this section, we give a brief review.

HISA (Healthcare Information System Architecture) [31] is a European pre-standard for medical information systems that deals with the architecture of medicine information system. Even though this architecture has no direct relationship to the field of e-Health, we still can benefit from it. It adopts three-layer architecture and uses middleware (CORBA and DCOM) technologies to develop distributed healthcare information systems to enable different health centers to interoperate with each other on the basis of information consistency.

Later, SAMTA [32] developed an open scalable architecture for multimedia telemedicine, applications which allows designers to efficiently use the network bandwidth. It classifies services into medical services, technical services and physical services according to different levels of complexity. This separation of services enables the technical details behind the medical services to be hidden from users. Among these levels of services, medical services are those offered to the user directly. Each service corresponds to one or more operations in the real world. Technical Services do not have direct relationship with the medical field. It only concentrates on network transmission, local services, security, compression and human interaction. Physical Services refer to the structures, protocols and services used by the technical services. They are close to the underlying hardware and act as an interface between the application and the environments (such as hardware and software platforms).

Authors in [33] proposed a Telemedicine System Interoperability Architecture which contains two levels of interoperability. The first level concentrates on how to compose stations within a telemedicine system and how to deliver the functionality within the station. This level is based on three sets of interfaces: station-to-device, station-to-station and station internal interfaces. The second level concentrates on how different stations can discover each other within a system and then begin transactions. The goal of this level is to allow independent systems to locate (discovery), negotiate (QoS parameters) and interoperate each other. Each of these levels provides a number of features to support different aspects of interoperability.

[34] proposed a distributed framework of Web-based telemedicine system which addresses two types of servers, i.e. Web servers and data servers. This framework is based on CORBA technique.

[35] proposed a SOA-based e-Health services architecture. It consists of six main components for defining interactions among different layers. In this architecture, the consumers include the hospital, medical staff, or other e-Health enterprise applications. Support function layer is used to help consumers to discover, deploy, and invoke health services and infrastructures. The control system layer enables the high availability, reliability, fidelity, and QoS. The goal of this layer is to employ different algorithms to achieve the service duplication, fault tolerance, and load balancing. Infrastructures and services layer is a container of health services. The security system provides the access control and other security functionalities. And the management system concentrates on controlling the data flow from one layer to another.

Moreover, [36] proposed the general access control requirements of Health Information System (HIS). However, it is based on the small component-based HIS. There are also many other research effort have been conducted with regards to the access control requirements for the healthcare system [37]. Most of them concentrate on the access control of a special subsystem, especially the Electronic Patient Record (EPR) system. However, the analysis of the access control from our research is from the perspective of the e-Health system integration.

2.7 Access Control Models

This subsection will focus on related work on access control models that can be applied to e-Health portals.

The access control matrix (ACM) [38] is an early access control model which can be represented by a triple <subject, object, right>. The object is an entity (such as file, process etc.) that to be protected and the subject is the entity (such as user, process etc.) who wishes to access the objects. The right specifies the operations that a subject is allowed to perform on the desired object. Access control lists (ACLs) is a common way to implement the ACM. Each object is associated with an ACL to indicate the access right that each subject is authorized to perform action on the object.

In discretionary access control (DAC), accesses are assigned by the owner of objects. DAC also allows a subject to pass its access permissions to another subject. Upon acquiring the permission either directly or indirectly, this subject is authorized to access these objects.

Unlike DAC, Mandatory access control (MAC) is an access control model where the permission of access is determined by the system, instead of the owner. In MAC, accesses are based on security clearances and levels (sensitivity labeling or security labeling). If the sensitivity level associated with the subject is equal or higher than the desired object, the subject is allowed to access the object. MAC is generally used in multilevel systems such as military information system.

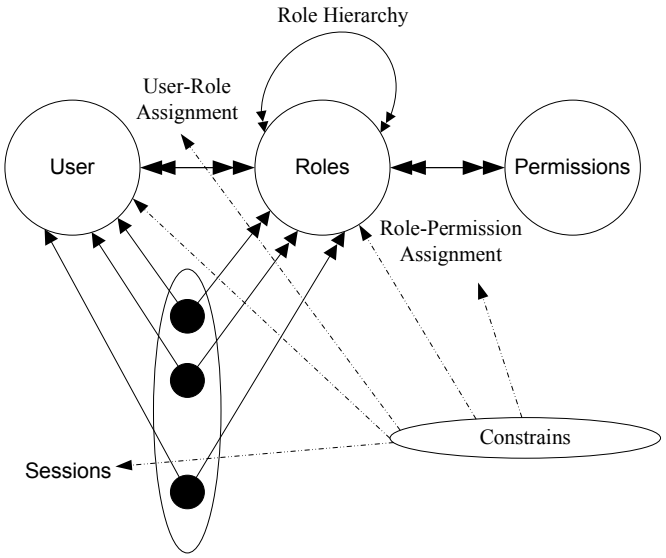

Figure 5: Role Based Access Control Model

Sandhu et al. proposed the RBAC model in 1996 [39]. The RBAC is an access control model where permissions are assigned to roles rather than directly assigned to users (refer to Figure 5). Roles are defined according to different job titles or functions, such as Medical Staff, Patient, and Administrator etc. Users are assigned with roles based on their responsibilities. The authorization of RBAC is split into two independent phases, i.e. user-role assignment and role-permission assignment. Sandhu also specifies four conceptual models derived from RBAC. RBAC-0 is the basic conceptual model which contains users, roles, permissions and sessions. A user can establish a session to activate a set of the roles to which the user is assigned. RBAC-1 encloses RBAC-0 and introduces the concept of role hierarchy. Role hierarchy uses partial order to represent the inheritance between roles which can reflect the structure of an organization in the reality. It also increases the reusability of permissions by which senior roles can inherit permissions from a junior role. RBAC-2 extends RBAC-0 with constraints, which add restrictions when assigning users or permissions to roles, or in the activation of roles in sessions. Finally, RBAC-3 combines all features provided by RBAC-1 and RBAC-2.

Besides the access control models stated above, grouping permissions can also be achieved by using rule based access control. Generally, a rule consists of two parts: Left-Hand Side (LHS) and Right-Hand Side (RHS) [40]. The LHS represents a set of conditions and the RHS stands for consequences. When those conditions on the LHS are true, actions (conclusions) on the RHS can be executed. Here is an example of rule:

IF LHS (Conditions) THEN RHS (Conclusions)

Thus, rule based access control model allows subjects to access various resources based on predefined rules. Rule engine is the core component of the model which is used to interpret rules at runtime. After loading the rule set from the rule repository, the rule engine will analyze conditions within rules and infer consequences. If consequences are positive, all actions will be executed. On the other hand, executions will be denied. Compared with RBAC, rules are less intuitive than roles and general non-technical administrators are incapable to specify rules.

3 Architecture Design

3.1 Functional Requirements

The design of e-Health portal aims to meet a collection of functional requirements as follows. The first three requirements are in the point of view of users of the portal, whereas the last four are about portal administrators or service providers.

- User Personalization. Users of an e-Health portal can create and save a personalized page including only the content they would like to access. For example, a patient may prefer seeing only the newsfeed in cardiology.
- Content Aggregation. Users of a portal can access related services on a single page, regardless how many service providers are involved or how different those services are implemented. Navigation elements should be provided such that users can easily switch to a different page when necessary.
- Ease of Use. This requirement is particularly relevant to e-Health systems where many users are seniors or have limited knowledge of computer technology and even the installation of client-side software may go beyond their capability.
- Backend Customization. Administrators of a portal can customize the source of services provided by the portal using a content management system, and such modifications should be transparent to normal users.
- Interoperability. The portal must be able to seamlessly integrate heterogeneous medical services implemented on different platforms and with different technologies. Such implementation details should be transparent to users of the portal.
- Extensibility. The integration of new services should imply minimal impact on the normal operation of other services provided by the same portal. Downtime should be minimized because the continuous availability of medical services may have direct impact on patients' health or life. Moreover, service providers should be able to independently produce services and seamlessly plug them into the portal.
- Support of Different Service Modes. A medical service can run in real time, automation or store-and-forward mode, and the portal should support all of above. Real time mode service is a kind of service that different participators can interact with each other at the same time. Generally, video and audio transmissions are involved in such kind of service. Automation mode means the medical data users submitted is processed and analyzed according to some sophistical algorithms automatically and the result will be returned to users. Store-and-forward mode means medical data is sent to a station where it is kept and examined at a later time by a medical staff.

3.2 System Architecture Design

We adopt the Service-Oriented Architecture (SOA) for the e-Health portal. Traditional designs of software systems usually have difficulties in meeting the aforementioned functional requirements. The interoperability among heterogeneous components requires a complicated integration process, which then implies unacceptable downtime and efforts for the integration. In contrast, the SOA exposes resources of each software component as standard-conforming services that can be accessed without understanding the underlying implementation details.

The main components of our design are e-Health portals interconnected via the Web Services for Remote Portlets (WSRP) protocol. The portals constitute an e-Health portals federation (in the reality, portal federation can also include systems with WSRP-enabled components). Each e-Health portal is a Web-based application that provides users a unified interface to all the services provided by these medical organizations. The portal allows its users to personalize desired content through creating customized Web page, namely, portal interface. Each portal interface also plays the role of a content aggregator by including a collection

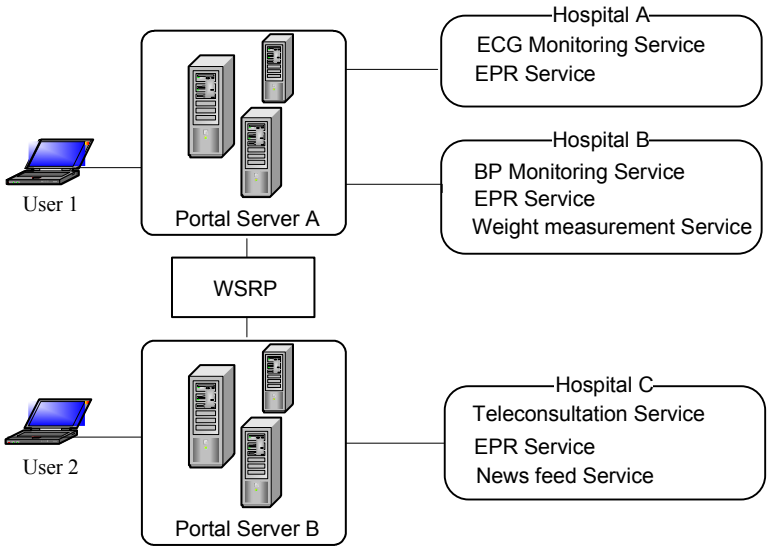

Figure 6: An Example Deployment of e-Health Portals Federation

of related services. Navigation elements inside each portal interface allow users to easily navigate among different collections of services just like surfing the Web. In our design, all client-side processing is supported either by the built-in functionalities of a standard Web browser or through applets/activeX controls that are automatically downloaded and executed in the browser. Users are not required to possess sophisticated computer skills in order to use services provided by the portal. Figure 6 is a simple deployment of such federation, which contains two e-Health portals. Portal A integrates medical services provided by Hospital A and B, and Portal B contains three services provided by Hospital C.

More details of the proposed architecture are depicted in Figure 7. End users interact through browser with the portal server via HTTP or HTTPs. The portal server consists of a portal engine and a portlet container. The main responsibility of the portal engine is to aggregate content from different sources and to serve these content to multiple devices. The portal engine also performs the role based access control [3] and redirects users to appropriate portal interfaces.

A portlet container provides a runtime environment for portlets implemented according to the Portlet API. In this container, portlets can be instantiated, used and finally destroyed. The separation of portal interfaces from portlets allows portal administrators to easily customize the source of services using a content management system.

Interoperability and extensibility are both inherent to the architecture because most of the backend services (run in automation or store-and-forward mode, e.g. the BP monitoring service is running in automation mode) can be integrated in a Web services manner. Service provider (backend services) and service requester (portlet) can publish and find desired medical services via UDDI, respectively. Once a new medical service is integrated into the system, it can be described by a WSDL file and this file will be sent to the UDDI. After discovering the binding information of the services via UDDI, service requester will communicate with backend services through standard SOAP regardless of what kinds of platforms are used in each side.

To access a service, a user connects to the portal server via a standard Web browser. According to the user's personalized settings and the entitlement, a portal interface is displayed with a collection of portlets inside it. When the user clicks on a button encapsulated in a portlet, the corresponding action of the service will be performed at backend service providers (which may be governed by a remote portal). As an exception, services run in real-time mode (such as the ECG Monitoring service and Teleconsultation service) are allowed to bypass the portal since they usually demand better performance than what SOAP can provide. Thus, an applet or activeX control downloaded to the browser will perform the required actions on behalf of the users. From the above descriptions, the proposed architecture clearly meets all requirements stated in Section 3.1.

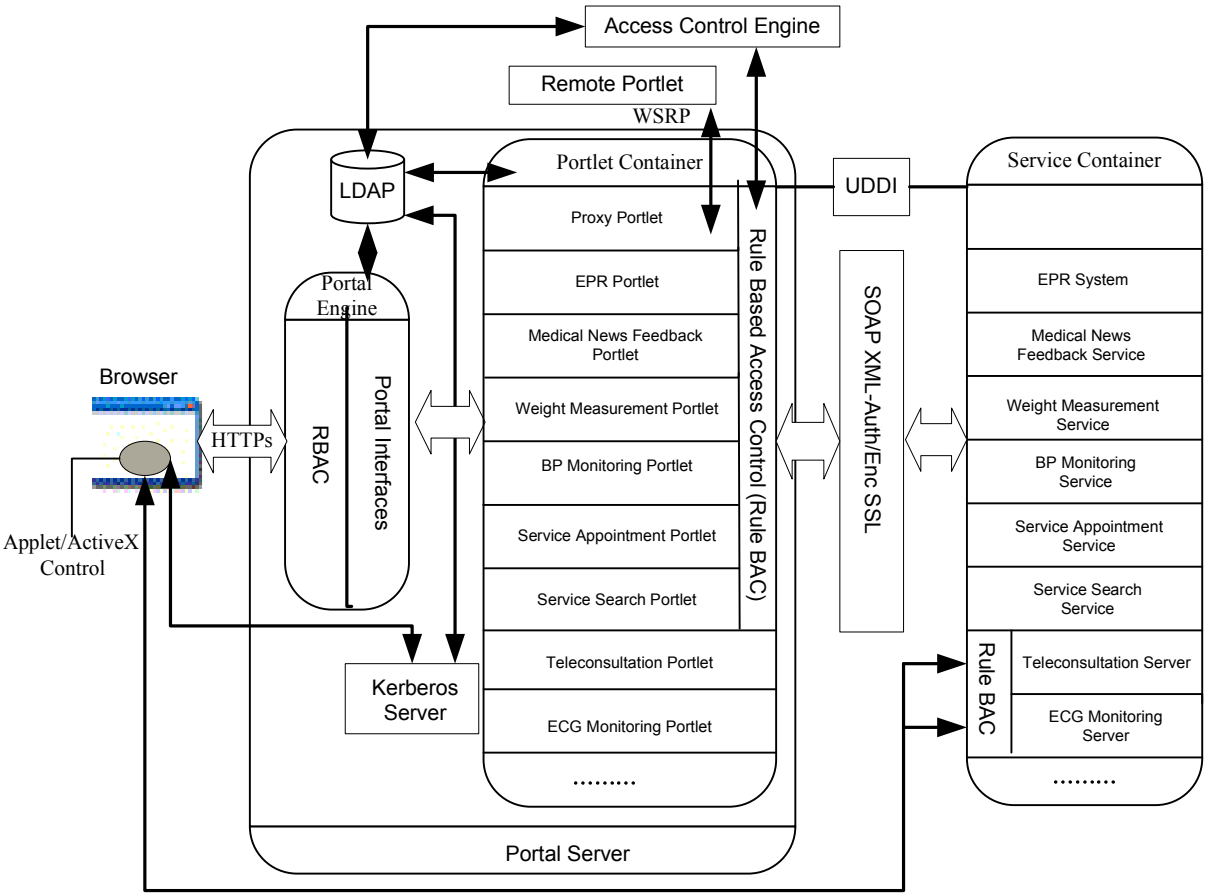

Figure 7: A Service-Oriented Architecture for e-Health Portal

Figure 8: Patient Side ECG Monitoring Interface

There are also some other components in the architecture. Whose functionalities and features will be described in following chapters.

4 Implementation

This section describes the implementation details of a prototype of the e-Health system portal. The system is designed to integrate existing medical systems, application, and services. We also illustrate the design and implementation of the access control engine and some medical services.

4.1 System Deployment Platform

The overall system architecture is described in chapter 3. Recall that in Figure 6, our prototype system contains two portal servers, that is, portal A and portal B. Portal A integrates services provided by hospital A and B, and portal B integrates services provided by hospital C. All hospitals provide the EPR service pointing to a centralized EPR system.

Our portal server utilizes the BEA Weblogic Portal 8.1, which provides enterprise portal infrastructure for streamlined portal development. This framework includes a graphical environment for developing portals, as well as browser-based assembly tools for business experts. It also simplifies the production and management of customized portals, allowing us to leverage a shared service environment to incorporate changes with minimal complexity and efforts [11]. Figure 8 shows the homepage of the patient portal interface.

4.2 Medical Services

Our research group has designed and implemented ECG Telemonitoring service, Blood Pressure (BP) Monitoring service, Teleconference service, and EPR system. I was responsible for the implementation of the ECG Telemonitoring service and BP Monitoring service.

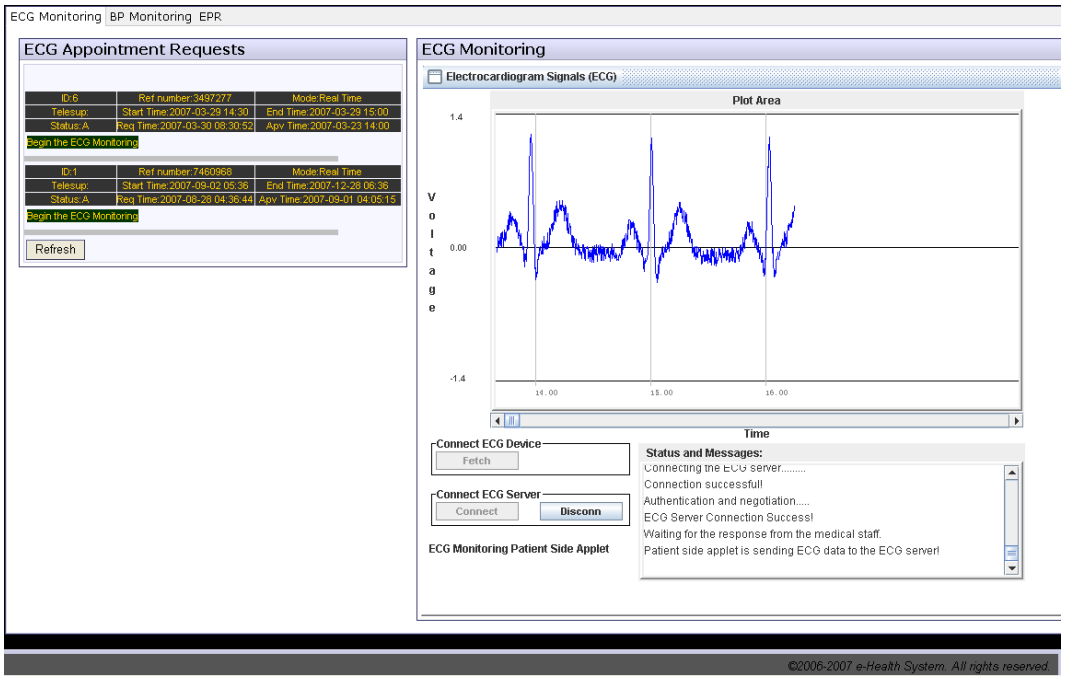

Figure 9: Patient Side ECG Monitoring Interface

4.2.1 ECG Monitoring Service

In our implementation, the ECG monitoring service is in real-time mode. For the automated mode, sophisticated algorithms will be needed to analyze the ECG signals, which is out of the scope of this research. The ECG monitoring service contains three components: a patient-side applet, a medical staff-side applet, and an ECG monitoring server. This patient-side applet is based on the toolkit ECGSYN [41], which is a realistic ECG waveform generator in the PhysioNet toolkits. It can generate a synthesized ECG signal with user-settable mean heart rate, number of beats, etc. PhysioNet is an Internet resource for biomedical research and development sponsored by the NIH's National Center for Research Resources. In real world, the ECG signals should be gathered from the external medical devices which are wired (such as USB, RS-232 etc) or wireless (WiFi, Bluetooth, IR etc.), and are connected to a computer. For demonstration purposes, we use a ECG signal generator to simulate this procedure. We also simplify the implementation of the toolkit and add our own features, such as signal transmission, Kerberos functions, etc. The ECG monitoring server is a java-based application that acts as a repeater to forward signals to the medical staff side and stores the ECG signal data into a database. Figure 9 and Figure 10 show the snapshots of the patient-side and medical staff-side ECG monitoring interface respectively. When a patient wants to use the ECG monitoring service, s/he needs to simply click a desired appointment in the ECG appointment request panel, and the applet will be downloaded and run in the patient's local machine. The patient can then perform the monitoring via the applet interface.

4.2.2 BP Monitoring Service

The architecture of the BP monitoring service is shown in Figure 11. The BP monitoring service is implemented in Java. It exposes its functionality as Web services. In the portal server side, the BP monitoring portlet acts as a service requester on behalf of the user. Unlike traditional servlet or JSP, portlet is a container of servlet, JSP or Java Page Flow (JPF). In our implementation, the JPF, a special Java file that uses a JPF file extension, the nerve center of the portlet, is employed to separate the user interface code from nav-

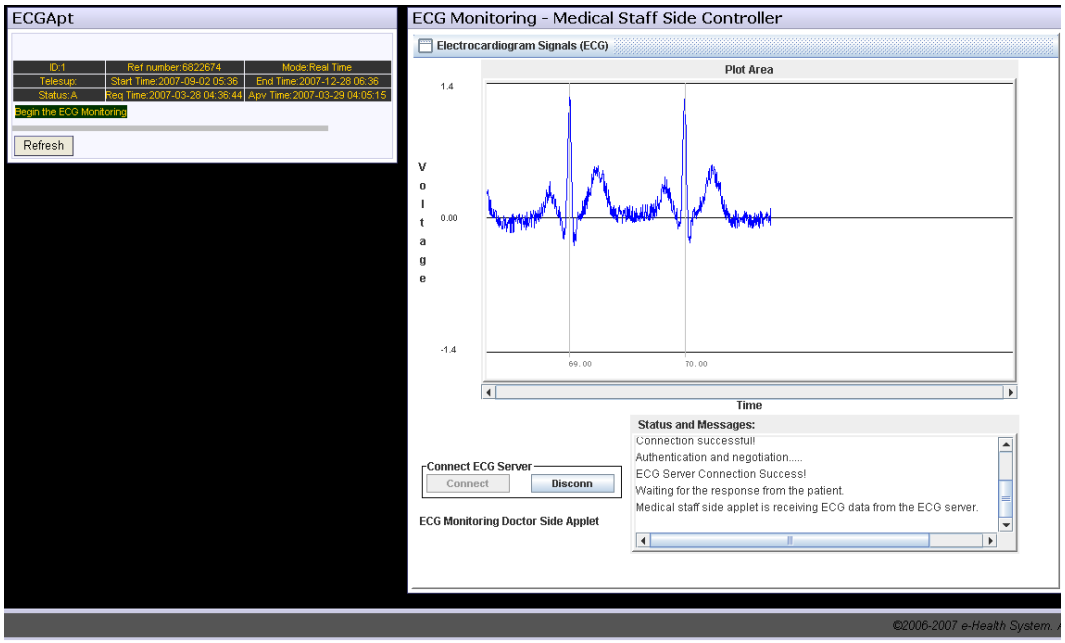

Figure 10: Medical Staff Side ECG Monitoring Interface

igational control and business logic. User interface codes are placed in a set of JSP files, which render users BP values input forms, diagnostic result, or any error information. Navigational control is also implemented in a page flow's single controller file. It utilizes a special JSP tag to invoke an action with a hyperlink. When the link on a JSP file is clicked, the page flow runtime detects the action and runs the navigational action method and controls user navigation between those JSP pages. Business logic can be implemented either in the page controller file, Java controls, or proxies called from the JPF file. In the implementation, business logic is implemented as a stand alone service and is wrapped as a Web service. The service proxy located in the JPF helps the portlet to locate and communicate with this service. The service is also for demonstration purposes. The medical criteria data is collected according to the US National Library of Medicine [42] and American Medical Association Report [43].

4.3 Two-Tier Access Control

The access control engine is a dedicated service that performs rule-based access control for users' requests. It contains three components, as shown in Figure 12. The core of the engine is an evaluator, which acts as a reasoning system to validate the rule set. This component is written in Prolog and runs in the Prolog server (as shown in Figure 13). Prolog is a simple but powerful programming language developed at the University of Marseille [44]. It is also a practical tool for logic programming. By using Prolog, we can write clear, readable, concise, and error-free programs. There are many Prolog products on the market, which includes open source (such as SWI-Prolog [45], YAP-Prolog [46]) and commercial ones (such as SICStus Prolog [47]). As a leading product, SICStus Prolog is a state-of-the-art, ISO standard compliant Prolog development system [47]. It is efficient and robust for handling large amounts of data and large applications. It supports bi-directional interfaces to C & C++, .NET and Java (in the current implementation, SICStus 4.0.1, the callback is not supported for .NET and Java). It also supports multiple platforms (such as Windows, Linux, Mac etc.). Thus, the SICStus Prolog is used to implement the core of the engine.

Another important component of the engine is the predicates API, which implements all required user-specific predicates used in our system. It is also responsible for collecting data, such as Current Server Time, Server Load, Appointment Information, User Profile, etc., to instantiate the variables in each predicate. This

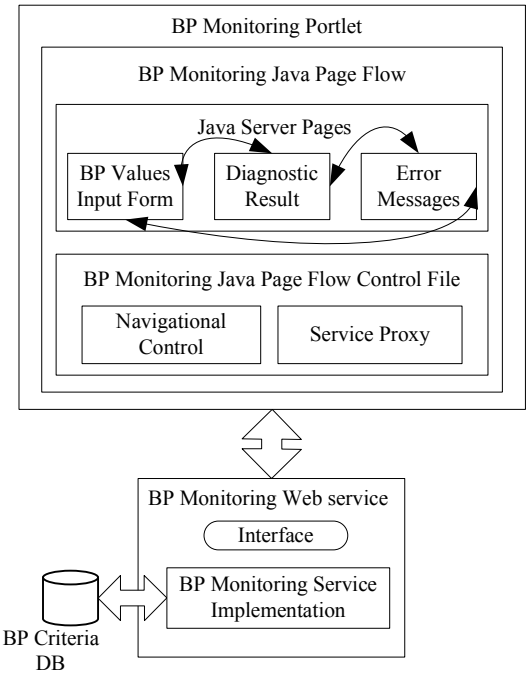

Figure 11: BP Monitoring Service Implementation

component is written in C++.

The third component is a Java application that sends request to and receives response from the Prolog server and exposes the validation interface as a Web service which can be invoked by access control proxies resided in applications. This component can also be implemented using other languages supported by SICStus, such as .NET, C or C++.

Figure 14 is the interactions among the components involved in Rule BAC procedure, as described below.

1. The access control proxy sends the authorization request to the access control engine.

2. Upon receiving the request, the interface component forwards it to the validator (Prolog server).

3. According to the "action" tag in the request, corresponding rule set will be activated. Each rule within the rule set contains one or more predicates. The evaluator will invoke each predicate implementation via predicates API.

4-5. The predicates API may collect necessary data from databases or environment parameters to in-

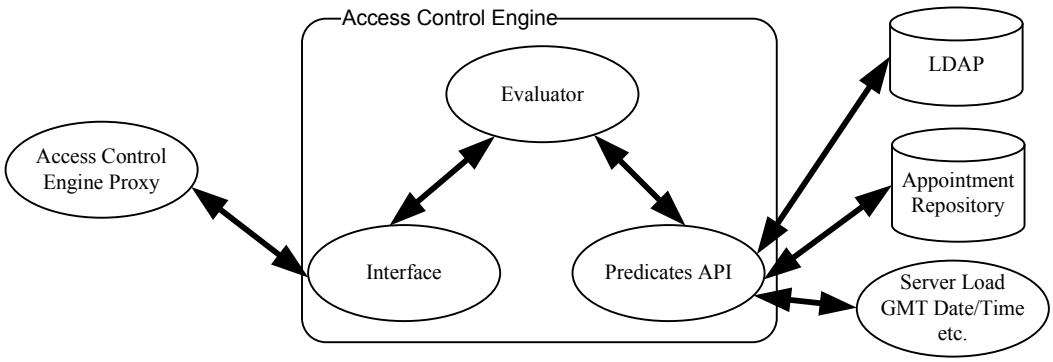

Figure 12: Components of the Access Control Engine

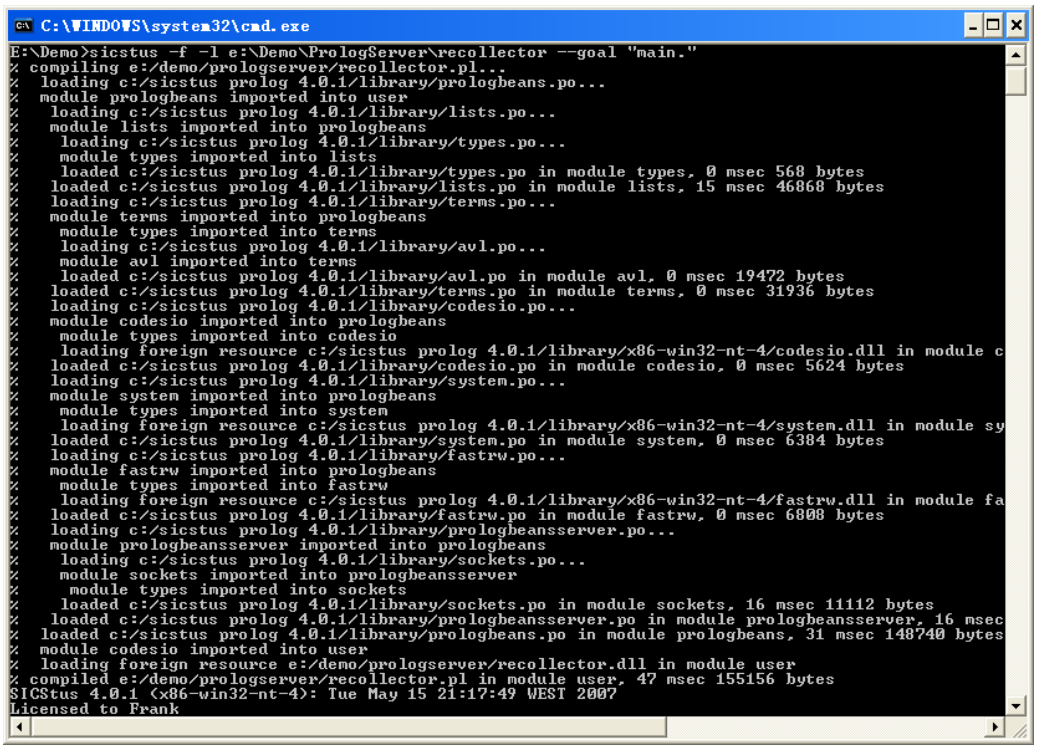

Figure 13: Snapshot of the Prolog Server

stantiate variables within each predicate.

6. The predicates API returns to the evaluator with "True" or "False" after executing the implementation of the predicate.

This procedure may occur multiple times until predicates API returns all predicates executing result (as shown in 7-10).

11. After collecting all the results from the predicates API, the evaluator will perform the inference procedure to obtain authorization decision and return the decision to the interface component.

12. The interface component returns this decision back to the access control proxy, and the proxy will enforce access control according to the decision.

Hereafter, we illustrate in details how the access control engine is configured, compiled and linked under Windows XP SP2.

1. Software and System requirements: JDK 1.5, Microsoft Visual Studio 2005 Team Suite with SP1, SICStus Prolog 4.0.1, MySQL 5.0.

2. MySQL Server in its original configuration. In the current implementation, the MySQL database is used to store all necessary information, such as Appointment Information, User Profile, etc.

3. Under the Windows environment, when compiling the database (MySQL 5.0) operation interface "*.cpp" file, we must include the following "*.h" files and must obey the order:

 # include "myglobal.h"

 # include "mysql.h"

Moreover, we must also include "*_glue.h" to "*.cpp" file where "*" is identical to the "*.cpp" prefix. This glue file will be generated by the Prolog compiler when compiling the prolog program.

4. Visual Studio C++ 6.0 is not supported by SICStus 4.0.1 anymore since its compiler does not support some options that SICStus needed. Thus, as recommended, we choose the Visual Studio C++ 2005 (VSC2005) as the default C++ program compiler. In order for the SICStus prolog compiler and linker to

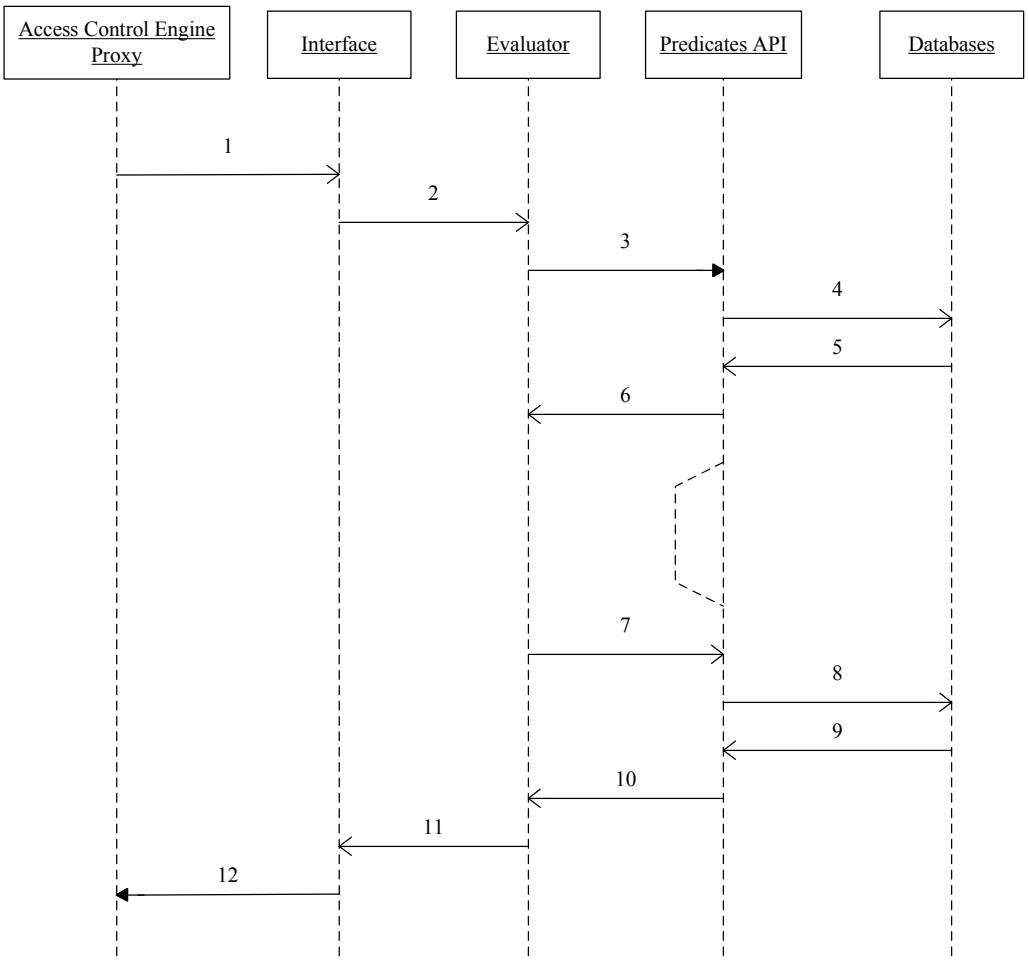

Figure 14: Sequence Diagram for Access Control Engine

locate the "cl.exe" of the VSC2005, the Windows system environment variables must be set properly. Here is an example setting:

PATH = C:\Program Files\Microsoft Visual Studio 8\VC\bin

5. Configuration of SICStus 4.0.1. The default configuration file of SICStus is "spconfig-4.0.1". Since the default setting for the VSC2005 linker sets the flag "SAFESEH" on, some libraries used by MySQL is incompatible with the "SAFESEH" linker flag and this flag is completely optional. Thus, we turn "SAFESEH" off by modifying the configuration file by simply removing all occurrences of the "SAFESEH" flag. The following are two occurrences of such flag:

WIN32_SPLD_CC_SPECIAL=/link /NXCOMPAT /SAFESEH

SPLFR_SHLDFLAGS=-nologo -dll /INCREMENTAL:NO /SAFESEH /NXCOMPAT ...

Moreover, we should also set the path of the MySQL library as (where the path of the library may be different according to the installation location):

SPLD_EXE_LIBS=D:\MySQL\lib\opt\libmysql.lib kernel32.lib user32.lib...

6. Setting the environment variables of JDK 1.5.

7. Compiling the project. In order to run the program in an environment that does not have VSC2005, we must set the Manifest file for the project. Copy all "∗.dll" and manifest files from the "...Microsoft Visual Studio 8\VC\redist\x86 \Microsoft.VC80.CRT" folder to the project folder where contains the "∗.cpp" and "∗.pl" files. Then rename the original "Microsoft.VC80.CRT.manifest" to "∗.dll.manifest" where "∗" should be identical to the name of the "cpp" file in step 3. Then execute the following command to compile the "cpp" and "pl" file:

splfr –verbose –keep recollector.pl recollector.cpp

Then a "∗.dll" file will be generated. The foreign resource will be imported by the Prolog program automatically when the Prolog program is running.

8. Running the access control engine. Executing the following command to start the Prolog server:

%sicstus -f -l recollector –goal "main."

After that, a Java application will communicate with the Prolog server and the interface of this Java application will be exposed as Web service, and can be invoked by any access control engine proxy deployed in any applications.

Notice that in this access control engine, many application-specific predicates are defined. These predicates are implemented using C++ language (Predicates API). The return type of these predicates is boolean. Thus, if a predicate is approved (or not), true (or false) will be returned to the Prolog program. In Prolog, a rule is constructed by a rule head and a body and the body contains one or more predicates and each predicate has a prefix operator (positive or negative). If the logical operation of all predicates with their prefix operators are approved then the head, i.e. the goal will be approved. Otherwise, the goal cannot be achieved. Each business policy can be represented as a rule set, which contains multiple rules and meta-policies. The Java applicaiton hides the rule validation details and only exposes one interface with few parameters. When the access control proxy wants to validate user's request, it only needs to provide the "object", the "subject" and the "action" values.

4.4 Use Cases

In this section, two use cases will be given to demonstrate how access control works together with e-Health services. The first is a Web service-based BP monitoring scenario, and the second is an applet-based ECG monitoring scenario.

4.4.1 Web Service-based BP Monitoring Scenario

After the patient successfully logs in to the system, the portal engine will redirect the user to the patient portal interface. By clicking the navigation bar on the portal interface, the interface of the BP monitoring service will be rendered to the patient. This interface contains two components, i.e. BP monitoring request history panel and the service panel. When the patient clicks the desired request hyperlink in the request history panel, the service panel will be activated to allow the patient to input the systolic and diastolic

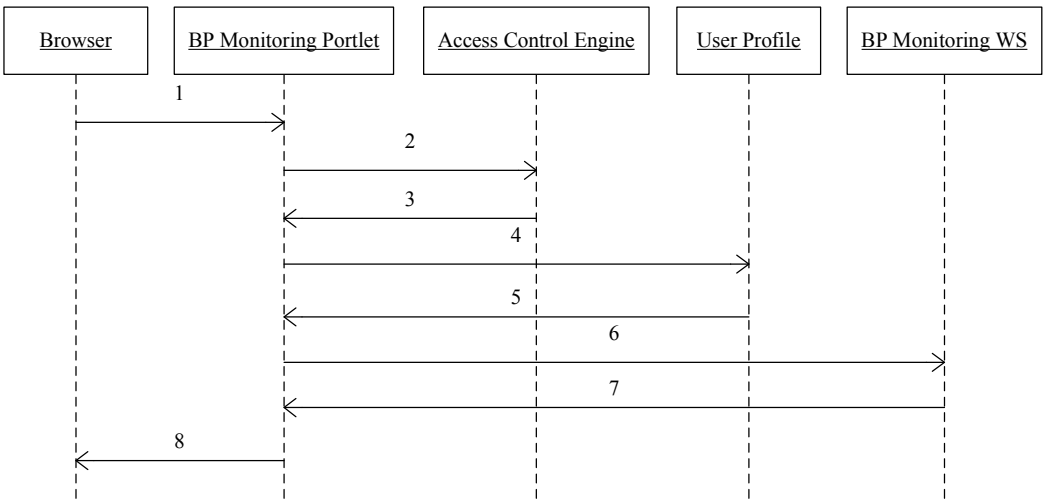

Figure 15: Sequence Diagram for BP Monitoring Service

values (In reality, these data should also be gathered directly from the BP measurement device installed on patients' body). Figure 15 is the sequence diagram for BP monitoring service. The procedure of how the messages interact among components within the portal and backend services is shown as follows.

1. After the patient fills in the systolic and diastolic values and clicks the "Submit BP Values" button, this request will be forwarded to the BP Monitoring Portlet.

2. Upon receiving this request, the portlet will get the necessary parameters from the request, assemble a new authorization request, and send it to the access control engine.

3. According to the rule set defined for the "Submit BP Values" action, the access control engine will evaluate this request, decide whether to allow the user to access, and then return the decision back to the BP monitoring portlet.

4. If the authorization is negative, the portlet will show an error message to the patient. Otherwise, it will contact the user profile database and fetch the gender and age information for future usage.

5. The user profile database returns the gender and age information of the patient to the BP monitoring portlet.

6. The BP Monitoring portlet assembles the BP values with gender and age information as parameters, and invokes the service via SOAP.

7. The BP monitoring service will process this request and return the result to the portlet.

8. Return the result to the patient.

4.4.2 Applet-based ECG Monitoring Scenario

Like the BP monitoring service, the ECG monitoring service user interface also contains two panels, i.e. a request history panel and a service panel. When the patient clicks the desired request hyperlink, the applet will be downloaded into the service panel, and all the parameters used by this applet will be initialized. After that, the user can use the service via the applet. Figure 16 shows the sequence diagram on behalf of the patient.

1. The ECG applet authenticates itself to the Kerberos server in term of the patient.

2. The Kerberos server returns the client-to-server ticket to the ECG applet. In step 1 and 2, we ignore multiple messages exchanged between the ECG applet and the Kerberos server (for more details for how to get the client-to-server ticket, refer to chapter 2).

3. ECG applet uses the client-to-server ticket to authenticate itself and sends a request to establish the connection for ECG signal transmission.

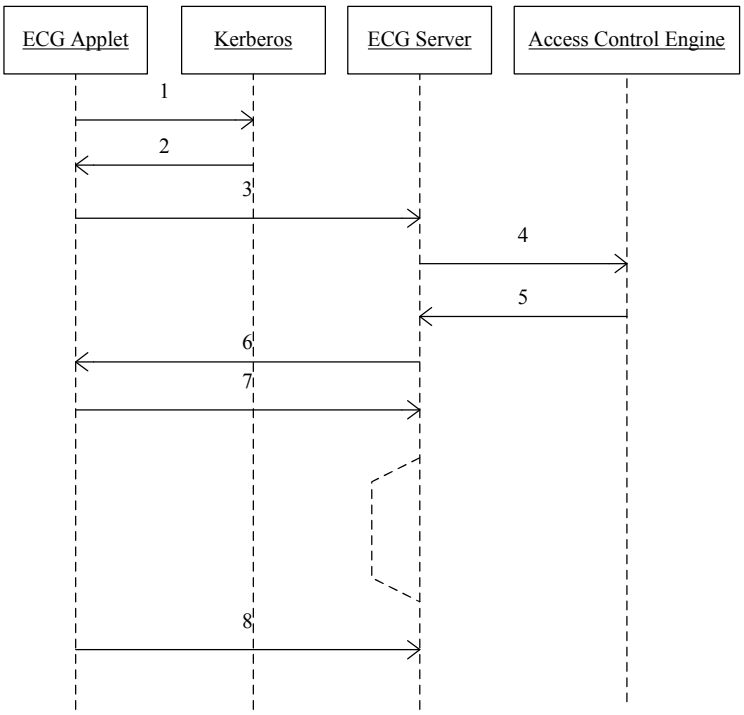

Figure 16: Sequence Diagram for Real Time ECG Monitoring Service on Behalf of Patients

4. If the authentication is successful, the ECG monitoring server will send an authorization request to the access control engine to confirm that this user is allowed to use this service.

5. The access control engine returns the authorization decision to the ECG monitoring server.

6. If the authorization is positive, the ECG monitoring server will accept the connection to the applet. At the same time, the ECG monitoring server will check whether the desired medical staff has connected to the server. If yes, the server will inform the patient-side applet to send data.

7. Upon receiving the instruction, the patient-side applet will transmit ECG signal to the ECG monitoring server.

Figure 17 shows the sequence diagram in term of the medical staff.

1. The medical staff-side ECG applet authenticates itself to the Kerberos server on behalf of the medical staff.

2. The Kerberos server returns the client-to-server ticket to the ECG applet.

3. ECG applet uses the client-to-server ticket to authenticate itself and sends a request to establish the connection for ECG signal transmission.

4. If the authentication is successful, the ECG monitoring server will send an authorization request to the access control engine to confirm that the medical staff is allowed to use this service.

5. The access control engine returns the authorization decision to the ECG monitoring server.

6. If the authorization is positive, the ECG monitoring server will accept the connection to the applet. At the same time, the ECG monitoring server will check whether the desired patient has connected to the server. If yes, the server will inform the medical staff-side applet to prepare receiving data.

7. Upon receiving the order, the medical staff-side applet will prepare receiving ECG signal from the ECG monitoring server in sequence and display the data to the medical staff.

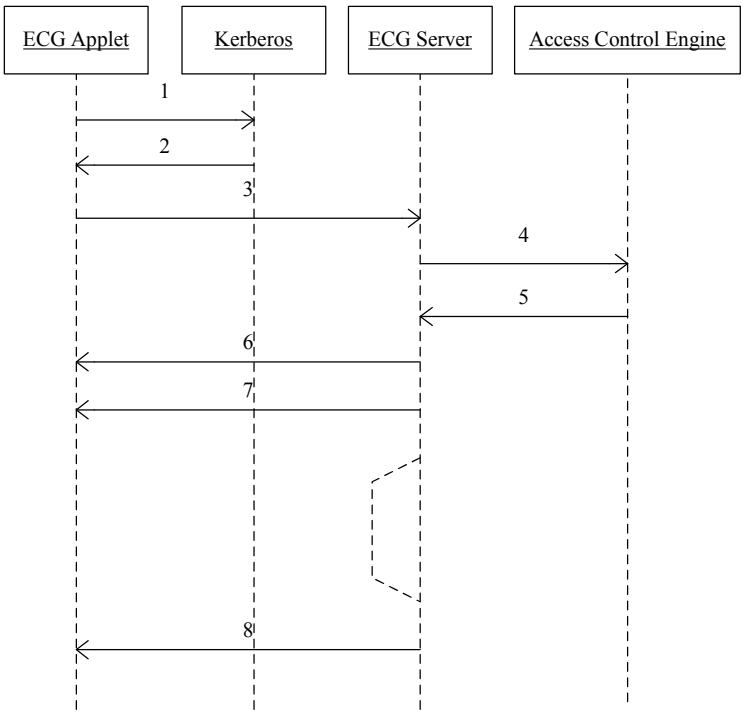

Figure 17: Sequence Diagram for Real Time ECG Monitoring Service on Behalf of Medical Staffs

5 Conclusion and Future Work

5.1 Conclusion

Traditional designs of software systems failed to meet the requirements of our system. We thus based our design upon a service-oriented architecture that can satisfy the stated functional requirements. Our e-Health portals can integrate different medical services and applications. In our prototype system, real-time ECG monitoring service, BP monitoring service, EPR system and Teleconsultation service have been implemented and integrated into our e-Health portal.

We also pointed out limitations found in the access control module of many off-the-shelf software components. Our solution was based on a two-tier access control architecture that integrated existing RBAC modules with a rule-based access control extension. This design inherited the advantages of both models and was cost-efficient. We also indicated and proposed solutions for the inconsistency issue within the two-tier model, the predicate attribute naming inconsistent issue, and the restricted attribute issue in applying our model to a federation environment.

5.2 Future Work

The proposed architecture and access control mechanism have been formulated in the thesis. Future researches are listed below:

1. Extend rule-based access control engine to enforce authorization of workflow-based services.

2. Quality of Services (QoS) is an important factor to be considered. QoS includes the scalability and performance of Web service-based services, the performance of the access control engine, the delay, jitter, response time etc., of real-time services etc. In the future work, we will investigate those QoS issues.

3. The WSRP standard does not address any security standard currently. Thus, for the authentication of consumer, the authentication of end user to the producer, message integrity and confidentiality should be

enforced in a standard way.

References

[1] G. Eysenbach, "What Is e-Health?," *Journal of Medical Internet Research*, vol. 3, no. 2, p. e20, 2001.

[2] G. Bisson, "e-Health Portal and SNOMED for a More Personalized Integrated EHR," *Proc. UM2005 Workshop on Personalization for e-Health*, 2005.

[3] S. Lu, Y. Hong, Q. Liu, L. Wang, and R. Dssouli, "Access control for e-health system portal," *Proc. 4th International Conference on Innovations in Information Technology (Innovations 2007), IEEE*, 2007.

[4] Y. Hong, S. Lu, Q. Liu, L. Wang, and R. Dssouli, "A hierarchical approach to the specification of privacy preferences," *Proc. 4th International Conference on Innovations in Information Technology (Innovations 2007), IEEE*, 2007.

[5] Q. Liu, S. Lu, Y. Hong, L. Wang, and R. Dssouli, "Securing telehealth applications in a web-based e-health portal," *Proc. 3rd International Conference on Availability, Reliability and Security (ARES 2008), IEEE*, 2008.

[6] Y. Hong, S. Lu, Q. Liu, L. Wang, and R. Dssouli, "Preserving privacy in e-health systems using hippocratic databases," *in Proceedings of the 32nd Annual International Computer Software and Applications Conference*, 2008.

[7] "http://www.oasis-open.org/,"

[8] C. Koch, "A new blueprint for the enterprise," *CIO Magazine*, 2005.

[9] A. Gokhale, B. Kumar, and A. Sahuguet, "Reinventing the wheel? corba vs. web services," *The 11th International World Wide Web Conference*, 2002.

[10] "http://www.w3c.org/,"

[11] "http://edocs.bea.com/wlp/docs81/index.html,"

[12] "http://jcp.org/en/jsr/detail?id=168,"

[13] "http://portals.apache.org/pluto/,"

[14] S. Jajodia, P. Samarati, M. Sapino, and V. Subrahmanian, "Flexible Support for Multiple Access Control Policie," *ACM Transactions on Database Systems (TODS)*, vol. 26, no. 2, pp. 214–260, 2001.

[15] D. W. N. Zannone, S. Jajodia, "Creating objects in the flexible authorization,"

[16] "http://web.mit.edu/kerberos/,"

[17] G. Kaur and N. Gupta, "E-Health: A New Perspective on Global Health," *Journal of Evolution and Technology*, vol. 15(1), pp. 23–35, 2006.

[18] Y. Chu and A. Ganz, "A mobile teletrauma system using 3G networks," *IEEE Transactions on Information Technology in Biomedicine*, vol. 8, no. 4, 2004.

[19] J. Fayn, C. Ghedira, D. Telisson, H. Atoui, J. Placide, L. Simon-Chautemps, P. Chevalier, and P. Rubel, "Towards new integrated information and communication infrastructures in e-health: Examples from cardiology," *Computers in Cardiology*, 2003.

[20] C. Riedel, T. Choudhri, D. Wilson, N. Khanafer, A. Alaoui, W. Tohme, and S. Mun, "Telemedicine in neurosurgery: peri-operative management," *Proceedings of Medical Technology Symposium, IEEE*, pp. 80–82, 1998.

[21] B. D. M. and B. L. M., "An interactive telemedicine system for remote speech-language pathology treatment," *Proceedings of the 26th Annual International Conference of the IEEE EMBS*, 2004.

[22] S. Shin, C. Ryu, J. Kang, S. Nam, Y. Song, T. Lim, J. Lee, D. Park, S. Kim, and Y. Kim, "Realization of an e-Health System to Perceive Emergency Situations," *Proceedings of the 26th Annual International Conference of the IEEE EMBS*, 2004.

[23] "World Wide Emergency Telemedicine Service (WETS)," *Telemat. Application Programme Project HC-4025*, 1998.

[24] B. F., B. K., and P. Maryni, "Adopting telemedicine services in the airline framework," *IEEE Transactions on Information Technology in Biomedicine*, vol. 5, no. 2, p. 171ÍC174, 2001.

[25] W. Chimiak, R. Rainer, J. Chimiak, and R. Martinez, "An Architecture for Naval Telemedicine," *IEEE Transactions On Information Technology In Biomedicine*, vol. 1, no. 1, 1997.

[26] E. Kyriacou, S. Pavlopoulos, D. Koutsouris, A. S. Andreou, and C. Pattichis, "Multipurpose health care telemedicine system, Engineering in Medicine and Biology Society," *Proceedings of the 23rd Annual International Conference of the IEEE*, vol. 4, pp. 3544–3547, 2001.

[27] J. E, "Wireless Technology and System Integration in Body Area Networks for m-Health Applications," *Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference Shanghai, China*, 2005.

[28] S. Voskarides., C. Pattichis., R. Istepanian., E. Kyriacou., M. Pattichis., and C. Schizas., "Mobile health systems: A brief overview," 2001.

[29] C. Pattichis, E. Kyriacou, S. Voskarides, M. Pattichis, R. Istepanian, and C. Schizas, "Wireless Telemedicine Systems: An Overview," *IEEE Antenna's and Propagation Magazine*, vol. 44, no. 2, 2002.

[30] R. Istepanian, E. Jovanov, and Y. Zhang, "Guest Editorial Introduction to the Special Section on M-Health: Beyond Seamless Mobility and Global Wireless Health-Care Connectivity," *IEEE Transactions On Information Technology In Biomedicine*, vol. 8, no. 4, 2004.

[31] S. J. R. and S. S., "Healthcare Information System Architecture (HISA) and its Middleware Models," 1999.

[32] "SAMTA Project, http://samta.offis.de/,"

[33] "The Telemedicine System Interoperability Architecture, http://telemedicine.sandia.gov/,"

[34] Y. Xiang, Q. Gu, and Z. Li, "Computer-Based Medical Systems," *Proc. 16th IEEE Symposium*, p. 108 ÍC 113, 2003.

[35] O. W. M. and B. A.T, "E-Health Support Services Based on Service-Oriented Architecture," *IEEE Computer Society*, 2006.

[36] E. M. and B. S., "A Case Study in Access Control Requirements for a Health Information System," *Australasian Information Security Workshop 2004 (AISW 2004), Dunedin, New Zealand. Conferences in Research and Practice in Information Technology*, vol. 32, 2004.

[37] L. Rostad and O. Edsberg, "A Study of Access Control Requirements for Healthcare Systems Based on Audit Trails from Access Logs," *Computer Security Applications Conference, ACSAC '06, IEEE*, 2006.

[38] L. B. W., "Protection," *Proceedings of the 5th Princeton Conference on Information Sciences and Systems*, p. 437, 1971.

[39] R. S. Sandhu, E. Coyne, H. Feinstein, and C. Youman, "Role-Based Access Control Models," *IEEE Computer*, vol. 29(2), pp. 38–47, 1996.

[40] E. Friedman-Hill, "Jess in action. rule-based systems in java.," *Manning Publications, Greenwich (USA)*, 2003.

[41] "http://www.physionet.org/physiotools/ecgsyn/java/ecgsyn-java.html,"

[42] "http://www.nlm.nih.gov/medlineplus/ency/article/003398.htm,"

[43] "http://www.ama-assn.org/ama1/pub/upload/mm/38/a-06csaph.pdf,"

[44] P. Roussel, "Prolog, manuel de rĺęfĺęrence et d'utilisation," *Groupe Intelligence Artificielle, Facultĺę des Sciences de Luminy, Universitĺę Aix-Marseille II*, 1975.

[45] "http://www.swi-prolog.org/,"

[46] "http://www.ncc.up.pt/ vsc/yap/,"

[47] "www.sics.se/sicstus/,"