Journal of Computer Science 2 (4): 303-313, 2006
ISSN 1549-3636

A UDDI Search Engine for SVG Federated Medical Imaging Web Services

Sabah Mohammed, Jinan Fiaidhi and Marshal Hahn
Department of Computer Science, Lakehead University, 955 Oliver Road
Thunder Bay, Ontario P7B 5E1, Canada

Abstract: With more and more medical web services appearing on the web, web service's discovery mechanism becomes essential. UDDI is an online registry standard to facilitate the discovery of business partners and services. However, most medical imaging applications exist within their own protected domain and were never designed to participate and operate with other applications across the web. However, private UDDI registries in federated organizations should be able to share the service descriptions as well as to access them if they are authorized. The new initiatives on Federated Web Services Identity Management can resolve a range of both technical and political barriers to enable wide-scale participation and interoperation of separate domains into a singular, robust user experience. However, there is no widely acceptable standard for federated web services and most of the available venders frameworks concentrate only on the security issue of the federation leaving the issue of searching and discovering web services largely primitive. Federated web services security and web services searching are uniquely intertwined, mutually reliant on each other and are poised to finally solve a long-running problem in both IT and systems security. Traditional keyword search is insufficient for web services search as the very small text fragments in web services are unsuitable for keyword search and the underlying structure and semantics of the web service are not exploited. Engineering solutions that address the security and accessibility concerns of web services, however, is a challenging task. This article introduces an extension to the traditional UDDI that enables sophisticated types of searching based on a lightweight web services federated security infrastructure.

Keywords: Web service federation, web service security, svg image security, medical imaging, medical informatics

INTRODUCTION

Like many complex distributed systems, healthcare information systems involve a variety of services and participants. When giving and receiving medical care, for example, participants such as doctors, radiologists, technicians, administrative staff and patients frequently interact with information services such as medical records databases, radiology image stores and billing systems. In addition, users and services also communicate with external entities such as insurance companies, pharmacies and health clinics. While regular communication is essential between healthcare providers, these exchanges are largely inefficient. Currently, exchanges generally occur in paper form or electronically using mostly custom, incompatible legacy systems (e.g. PACS, RIS, DICOM)[1]. Because these disparate users and services lack a common communications framework, it is difficult for healthcare participants to obtain comprehensive medical information about patients when providing care. A patient may have multiple medical records stored at various locations (e.g., at a hospital, doctor's clinic, pharmacy) and data such as lab results, drug prescriptions and disease histories are often not consolidated. Thus, it is likely that healthcare participants could provide higher quality medical care if they had access to such information and resources, especially during emergency situations. There is a range of other challenging properties of medical work, which makes it fundamentally different from typical distributed office network: extreme mobility, ad hoc collaboration, interruptions, high degree of communication, etc. – attributes that are in strong contrast to normal office work[2]. Moreover, healthcare services are turning to stronger authentication methods. Biometric methods (e.g., fingerprints, iris scans, signature and voice recognition) and non-biometric digital techniques (e.g., e-tokens, RFID, key fobs) are rapidly replacing passwords for authentication purposes. Thus, the security requirements in the healthcare systems require very dynamic and flexible policy enforcement.

As a remedy the recent advent of XML and web services can be seen as an effective solution to the security issues. In particular web services present a standardized, loosely-coupled framework that can incorporate the complex, cross-boundary interactions of a healthcare system into a fully-connected, distributed computer system. Utilizing a standardized computer

Corresponding Author: Sabah Mohammed, Department of Computer Science, Lakehead University, 955 Oliver Road, Thunder Bay, Ontario P7B 5E1, Canada

language like XML, allows a wide and diverse group of individuals or organizations to "talk" to each other, which greatly facilitates information gathering and on-line transactions. On the other hand, Web services are applications that can share information and services with other applications over the Internet using a common interface and messaging system. Such an integrated environment for exchanging information may revolutionize communication and information-sharing practices not only for healthcare systems, but also for a variety of other industries. This technology provides an easy way for entities to share data and services with other entities using a common framework and a standardized messaging protocol. Thus, there is a growing number of international associations like MedBiquitous Consortium (http://www.medbiq.org/) and the HL7 new initiative on Medical Informatics (or what is called HL7 V3 Initiative(http://www.hl7.org/)) dedicating their efforts to accommodate this new trend of technology for constructing a new type of medical healthcare systems. Such initiatives provided an environment for a growing number of web services and XML-based data and applications available within hospitals and on the Web which raises new and challenging research problems. Particularly on how to *locate desired web services and how to securely access these web services*.

Unfortunately the traditional keyword search is insufficient in context of web services: the specific types of queries users require are not captured, the very small text fragments in web services are unsuitable for keyword search and the underlying structure and semantics of the web services are not exploited. Moreover, in creating mechanisms to collect and to retrieve medical information, however, one must recognize that protecting patient privacy is a fundamental system requirement. Engineering solutions that address the security and accessibility concerns of web services especially for medical imaging services, however, is a challenging task. Many corporations and standards organizations currently undertaking this task have developed specifications and tools to address these concerns, but this effort is largely a work in progress.

MEDICAL IMAGING BASED ON SVG WEB SERVICES

Using XML to represent patient data records is a new trend in medical systems[3,4]. However, including binary images within the format of XML limits the ubiquity of XML as well as prevents searchability for these images or their contents. Medical images are at the heart of the patient's diagnosis, therapy treatment, surgical planning, disease screening and long-term follow-up for outcome assessment. Medical imaging is becoming increasingly important in patient records. In the past three decades, we have witnessed tremendous changes in medical imaging; new techniques include diagnostic ultrasound, X-ray computed tomography (CT), magnetic resonance imaging (MRI), magnetic resonance spectroscopy (MRS), functional magnetic resonance imaging (fMRI), digital subtraction angiography (DSA), positron emission tomography (PET), magnetic source imaging (MSI) and so on. These digital imaging modalities, which currently constitute about 30 percent of medical imaging examinations and records in the United States[5]. These advancements in medical imaging requires better means to acquire patient images from patient records[6]. In this direction Scalable Vector Graphics (SVG) imaging promises to revolutionize the Web through the introduction of standard based on vector graphics for imaging, animation and multimedia interactivity. SVG standard allows to represent complex graphical scenes by a collection of vectorial-based primitives, offering several advantages compared to classical raster images[6,7].

The broad support behind SVG comes from its many advantages. SVG has sophisticated graphic features, which is naturally important for a graphic format, but it also benefits from being an XML grammar. SVG has all the advantages XML brings such as internationalization (Unicode support), wide tool support, easy manipulation through standard APIs (e.g., the Document Object Model, DOM API, Batik API) and easy transformation (e.g., through XML style sheet Language Transformations, XSLT). In the graphical arena and especially compared to raster graphics formats (such as GIF, JPEG or PNG images), SVG has the advantage of being:

* Lightweight. For many types of graphics, an SVG graphic will be more compact than its raster equivalent
* Zoomable. SVG content can be viewed at different resolutions, e.g., enlarged or shrunk without losing quality.
* Searchable. Because SVG content is XML, it becomes possible to search the content of an SVG image for text elements, comments or any kind of meta-data.
* Structured and Accessible. Graphic objects can be grouped and organized hierarchically.

It is natural for Web Services to accommodate SVG for graphical and image based services. In addition to being open and XML, SVG has a rich structure and preserves semantic because of its descriptive element and metadata. This richness provides an opportunity to Web Services to generate, modify or search rich graphical content[8]. Traditionally SVG has been used as a flexible imaging viewer only, which limited its' potential for advanced imaging applications. Security issues have been the main challenge in SVG applications. A key challenge, therefore, is to enable the interoperability between SVG Web Services to take place seamlessly and securely[9].

BUILDING FEDERATED WEB SERVICES

To meet the challenge of current industry trends such as growth in increased mobility and the need for persistent connectivity, healthcare organizations are extending internal systems to external users providing connectivity to customers, partners, suppliers and mobile healthcare users. However, providing efficient and seamless connectivity requires building "trust-based" relationships that enable organizations to securely share a user's identity information. Trust relationships allow identity and policy information to flow between healthcare organizations independent of platform, application, or security model. Trust relationships need to be formed quickly and efficiently to maximize productivity and eliminate the manual processes that often take place today. Web Service Federation describes the technology and business arrangements necessary for this interconnection[10]. The term federation derives from the Latin word for trust. In the world of distributed network services, the term refers to the need for trust agreements among decentralized security and policy domains. Federation lets access-management functions span diverse organizations, business units, sites, platforms, products and applications. Federation requires that an organization trust each trading partner to authenticate its own users' identities. In a federated environment, a user can log on to his home domain and access resources transparently in external domains, such as those managed by customers or suppliers, subject to various policies defined by home and external administrators. Federated systems need to interoperate across organizational boundaries and connect processes utilizing different technologies, identity storage, security approaches and programming models. Within a federated system, identities and their associated credentials are still stored, owned and managed separately. Each individual member of the federation continues to manage its own identities, but is capable of securely sharing and accepting identities and credentials from other members' sources. Within a federated system, an organization needs a standardized and secure way of expressing not only the services it makes available to trusted partners and customers, but also the policies by which it runs its business such as which other organizations and users it trusts, what types of credentials and requests it accepts and its privacy policies.

In this direction, web service federation requires standards, specifications and frameworks that will describe the model for establishing both direct and brokered trust relationships (including third parties and intermediaries). The industry is yet to agreed about one standard. Although there are some serious attempt to have a united framework, but we cannot yet see the light in the tunnel. Currently there are many standards and techniques to develop and managed federated web services. The most notable of all is the *ebXML* which provides an open, industry-wide standard for building support for collaborative Web services, including reliable messaging. This standard provides a suite of managing specifications including:

* Reliable messaging: *(ebMS)* -- Provides guaranteed, once-and-once-only delivery, layered on SOAP messaging.
* Business process specifications *(BPSS)* -- Defines business activities, collaborations and transactions and describes their relationships. Also provides a machine-readable specification instance.
* Partner profile and agreements *(CPP/A)* -- Holds configuration information for partners' runtime systems and stores quality-of-service (QOS) information.
* Registries and repositories *(Reg/Rep)* -- Provides a powerful classification and storage mechanism for artifacts, including *BPSS* and *CPP/A*.

More recent standards like the *WS-Federation* standard has been developed recently by a group of major vendors (BEA, IBM, Microsoft, RSA Security and VeriSign http://xml.coverpages.org/WS-Federation.pdf). This standard provides a language *(WS-Federation)* that defines mechanisms "used to enable identity, account, attribute, authentication and authorization federation across different trust realms. Other group of vendors (Oracle, BEA, IBM, Microsoft) are also aggressively working on another standard for orchestrating Web-services-based end-to-end business processes. They introduced the *BPEL* (Business Process Execution Language). BPEL is the XML standard that can create trusted and federated environment for web services. Moreover, there are many other standards that can be used to enforce federated web services environment:

* The Universal Business Language (UBL) (http://docs.oasis-open.org/ubl/cd-UBL-1.0/)
* The Extensible Access Control Markup Language (XACML) (http://java.sun.com/developer/technicalArticles/Security/xacml/xacml.html)
* The Business Transaction Protocol (BTP) (http://www.developer.com/java/data/article.php/10932_3066301_2)
* The Global XML Web Services Architecture (GXA) (http://www.ftponline.com/wss/2003%5F01/magazine/features/mmercuri/)
* The Liberty Alliance Project (http://www.projectliberty.org/)
* XML Key Management Specifications (http://www.w3.org/TR/xkms/)

Among the many evolving mentioned security standards for Web services there are only two major basic initiatives:

1. Security assertion markup language (SAML): SMAL protocol relies on Single-Sign-On (SSO) services to deliver authentication through *federated web services. ClearTrust 5 is an example of such protocol*[11]. Actually, SMAL defines set of XML formats for representing identity and attribute information, as well as protocols for requests and responses for access control information. The key principle behind SAML is an assertion, a statement made by a trusted party about another. Assertions can be encoded in browser requests or included in Web services transactions, enabling logins for both person-to-machine and machine-to-machine communications.

2. WS-security (WSS): This is a security standard that focuses on message integrity, confidentiality and authentication. WSS does not address SSO but covers message encryption in detail. WS-Security will standardize how security information is added to SOAP messages. One important class of such security information is one which WS-Security calls *Security Tokens* -- a security token is "a collection of claims" about the sender (the sender typically proving their right to this claim through a digital signature). WS-Security begins with the assumption that, if one of the parties uses a particular type of security token(e.g. X.509 certificates, Kerberos tickets, SAML Assertions, XACML policies, etc.) within the WS-Security header, then the other party will be able to interpret and process this token. The basic architecture of WSS is shown in the Fig. 1[12]. A SOAP client sends a SOAP message to a business application SOAP service (for the sake of this example a purchasing application), which, after appropriate business processing, sends the SOAP response back. Supporting this fundamental exchange are the components of the security infrastructure, a SOAP gateway and a security token service (STS), which work together to ensure that the SOAP service receives only messages with "appropriate" security tokens.

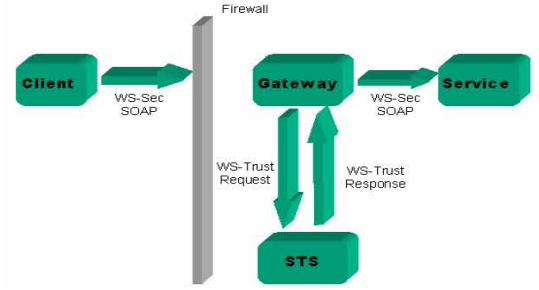


Fig. 1: Basic model of web service security

Obviously the SAML approach to solving the WS security has been based on SSO, the centralization of access control information into one server or servers that requires special plugins (e.g., "Web agents" for Web servers) to retrieve the information. Every application needs to be "SSO enabled" by programming to the proprietary API, different for each competing vendor. The coding task usually falls to the IT organization. Overall, this technology has not been as successful as originally hoped, with many SSO implementations either behind or experiencing scalability challenges. Indeed, identity information and access control policies are some of the most valuable and frequently used data in any IT organization. Instead of coding to a proprietary agent (which in turn uses a proprietary protocol to communicate with a particular brand of identity management server), applications can make Web services (SOAP) requests to authenticate users or authorize transactions. In conclusion, we may see WSS is more mature than SAML. Even though both SAML and WSS overlap, they are not mutually exclusive and may be merged into a single standard in the future. However, to be successful with either SAML or WSS, you must build a layered system that supports current and future security implementations.

Thus, one interesting of our ongoing research is to extend SOAP to gain access to the actual network stream before it is decentralized into objects and vice versa. For instance, through these extensions encryption algorithms can be developed on top of the Web Service call. However, most of the SOAP extensions are transport Protocols dependent, especially for multimedia applications, such as RTP (http://www.ietf.org/rfc/rfc1889.txt.), Multicast and UDP. To deal with this issue, brokering-based architectures can be used. In this direction we proposed an AXIS Based Web Service Management Network (AWSMN), as management architecture for federated service management[13]. Since we believe it is safe to assume Internet services will be implemented using web service technology (SOAP, XML, WSDL), AWSMN is based on such technology. The critical concept in AWSMN is that of Axis Security Handlers (ASH). SAHs are intermediary components that can explicitly defined to manage agreements between services. The SAH concept allows us to frame and solve many problems rather elegantly and effectively. AWSMN, then, is a network of cooperating intermediaries, each such intermediary implemented as a proxy sitting between a service and the outside world and a set of protocols to manage service relationships expressed through SAHs (Fig. 2).

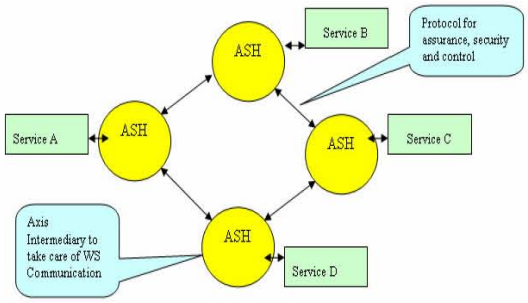


Fig. 2: The AWSMN System

The AWSMN is concerned with the following security-related tasks:

* XML Encryption\Decryption (used by various ASH handlers).
* XML Signature\Verification (also used by various ASH handlers).
* Generation and storage of RSA public\private key pairs. The public key is always wrapped in X509 Certificate.
* Authenticates user requests for Web Service usage.
* Authorizes users for Web Service usage.
* Handles group management tasks such group creation, member addition, WS addition and related tasks such as organization and registration\deletion.
* Assist in handling WS search.
* Processes messages.

Tasks 1-3 and 8 are performed locally using the Secure class. However, tasks 4-6 are performed remotely by the other classes of the AWSMN Security Subsystem (Fig. 3). Task 7 is a compound task that includes tasks 1-6 and work in collaboration with the UDDI Subsystem.

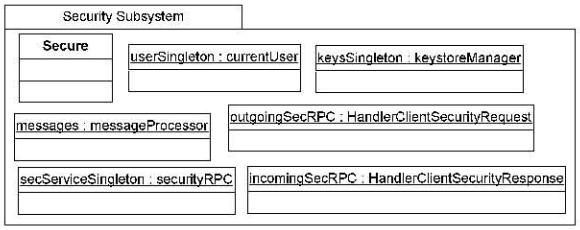


Fig. 3: The AWSMN security subsystem

Details of the AWSMN classes implementation can be found at[13,4].

UDDI SEARCH ENGINE

Basically a UDDI client works by searching UDDI servers for all services that matches the service required name through parsing the WSDL specifications. Each web service has an associated WSDL file describing its functionality and interface. A web service is typically published by registering its WSDL file in UDDI registries. Each web service consists of a set of operations. Through the WSDL, we have access to the following information on web services: Name and text description, Operation descriptions and Input/Output descriptions. To the outside world, UDDI provides two sets of interfaces: one for service registration and one for service discovery. For discovery, UDDI can be considered as White Pages registry that provides business contact information. However, UDDI search engines in this context does not help in categorizing web services based-on accepted business classifications nor it comply to the federated security initiatives. For this purpose we are extending the UDDI client to accommodate the category-based searching facilities as well as to comply to the AWSMN federated web service model. The advantage of this approach is that several candidate services can be considered and searched and by using a unified classification pattern, a single interface provided to all of the candidates. In this context, UDDI can be extended to represent Yellow Pages registry that can categorize business and their services according to standard taxonomies. To extend UDDI so to be aware of a business classification scheme, we used mammography cancer staging categories as described by the American Joint Committee on Cancer (http://www.imaginis.com/breasthealth/staging.asp).
The categories are kept in an external XML file and can be changed to any other application. However, the categories use the tumor, nodes, metastasis (TNM) classification system. The stage of a breast cancer describes its size and the extent to which it has spread. The staging system ranges from stage 0 to stage IV according to tumor size, lymph nodes involved and distant metastasis. T indicates tumor size. The letter T is followed by a number from 0 to 4, which describes the size of the tumor and whether it has spread to the skin or chest wall under the breast. Higher T numbers indicate a larger tumor and/or more extensive spread to tissues surrounding the breast.

a. TX: The tumor cannot be assessed.
b. T0: No evidence of a tumor is present.
c. Tis: The cancer may be LCIS, DCIS, or Paget disease.
d. T1: The tumor is 2 cm or smaller in diameter.
e. T2: The tumor is 2-5 cm in diameter.
f. T3: The tumor is more than 5 cm in diameter.
g. T4: The tumor is any size and it has attached itself to the chest wall and spread to the pectoral (chest) lymph nodes.

The letter N indicates palpable nodes. The letter N is followed by a number from 0 to 3, which indicates whether the cancer has spread to lymph nodes near the breast and if so, whether the affected nodes are fixed to other structures under the arm.

a. NX: Lymph nodes cannot be assessed (eg, lymph nodes were previously removed).
b. N0: Cancer has not spread to lymph nodes.
c. N1: Cancer has spread to the movable ipsilateral axillary lymph nodes (underarm lymph nodes on the same side as the breast cancer).
d. N2: Cancer has spread to ipsilateral lymph nodes (on the same side of the body as the breast cancer), fixed to one another or to other structures under the arm.

e. N3: Cancer has spread to the ipsilateral mammary lymph nodes or the ipsilateral supraclavicular lymph nodes (on the same side of the body as the breast cancer).

The letter M indicates metastasis. The letter M is followed by a 0 or 1, which indicates whether the cancer has metastasized (spread) to distant organs (eg, lungs or bones) or to lymph nodes that are not next to the breast, such as those above the collarbone.

a. MX: Metastasis cannot be assessed.
b. M0: No distant metastasis to other organs is present.
c. M1: Distant metastasis to other organs has occurred.

Hence the main purpose of the extended UDDI client is to search for SVG based mammograms based on the beast cancer staging categories and via Security Server[16]. The object which provides the main functionality of this UDDI client (Search and Retrieve Subsystem) is a singleton of type uddiConnectivity. Fig. 4 illustrates the main classes involved in the UDDI client. The uddiMessage class inherits from class java.lang.Exception. Methods in the uddiConnectivity singleton object sometimes will create and throw objects of type uddiMessage to pass messages to the GUI subsystem such as "No Services Found". The methods of class uddiConnectivity throw other exceptions as well that are mostly JAXR-related. Information (name, which SVG service offers it, …) regarding each SVG found by an SVG search is stored in a svgImageResult object and these objects are passed to the GUI subsystem.If the user indicates the desire to view an SVG listed in the search results list, this subsystem will retrieve the URL of the SVG Web Service offering that SVG and store it in an accessPoints object. Next, the accessPoints object will be passed to the SVG Image Retrieval Subsystem.

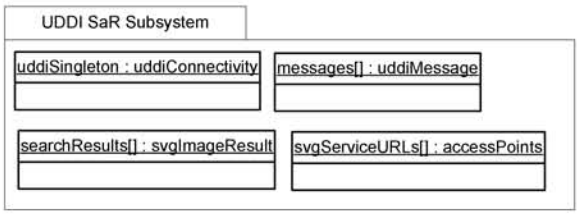


Fig. 4: The UDDI search and retrieval subsystem

This image retrieval subsystem is responsible for the secure retrieval of SVG Images and is shown in Fig. 5. The axisRPC class is designed specifically to encapsulate remote procedure calls to an RPC Style SVG Web Service. To create an object of this class, the only parameter required is an accessPoints object (which contains the URL needed to contact a particular SVG Service). Using this object, the constructor will create the necessary Axis call objects. The Axis-related code of this class is similar to that of the securityRPC class of the Security Subsystem.

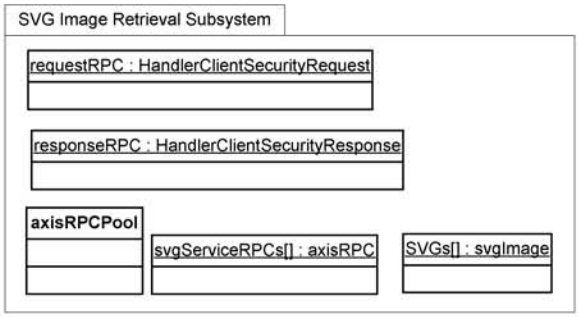


Fig. 5: The SVG image retrieval subsystem

The axisRPCPool class stores a number of axisRPC objects in a static vector object. Each axisRPC object in the pool is configured to connect to exactly one SVG Web Service and none of these objects connect to the same SVG Web Service. Each time the Client user attempts to retrieve an SVG from an SVG Web Service that he has not accessed before, a new axisRPC object configured to connect to that SVG Web Service will be added to the pool. When an axisRPC object is created, the public key certificate of the SVG Web Service it will connect to must be retrieved. An advantage gained by using the axisRPCPool class regards the fact that the certificate of each SVG web service must only be downloaded once. Note that the pool must be rebuilt each time the Client is restarted (the public key certificates are not saved locally). The axisRPC getSVG(…) method returns an object of type svgImage. Each svgImage object contains an SVG image (stored in an org.w3c.dom.Document) as well as the name and uuid of that SVG image. Objects of type svgImage are passed to the GUI Subsystem.

The HandlerClientSecurityRequest and HandlerClientSecurityResponse instances behave similarly to the ASHs discussed in[15] with only one difference where the public key of the SVG Web Service being contacted is used for the encryption operation in the Request Handler and the signature verification operation in the Response Handler.

The UDDI client or the SVG Search Engine client can then query the UDDI registry to discover the required Web services and can make use of them. Searches can be performed based on a number of criteria types such as the service name, Breast Cancer classification and the service groups. The system supports hierarchal classification schemes and classification-based searches will return all SVGs with a more specific classification then the one the user chose to search for (Fig. 6).

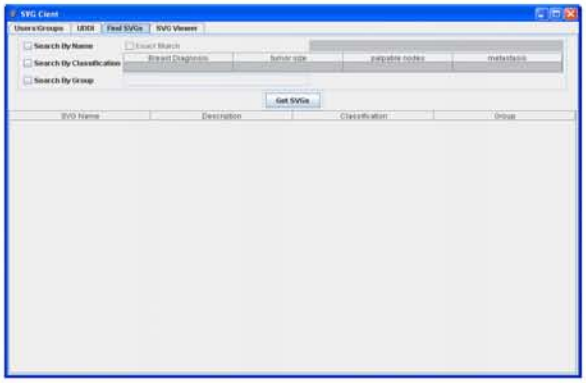
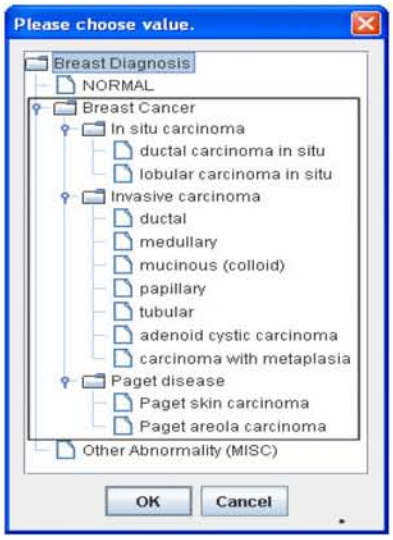
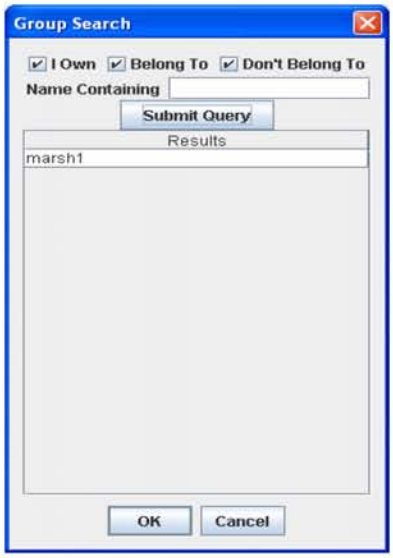


Fig. 6: UDDI SVG search client



a. Searching for breast cancer web service



b. Searching via the group type

Fig. 7: Searching by category and by group

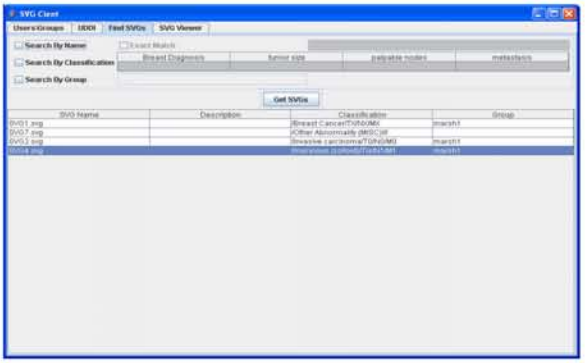


Fig. 8: The SVG web services search results

After the results of a search are displayed, the user can attempt to view any of the SVGs listed. After the user chooses to view an SVG, the Client will retrieve the location of the SVG Web Service from the UDDI Server. Next, the Client will attempt retrieval of the SVG image. Whether or not this retrieval is successful depends on whether or not the user is a member of the group the SVG belongs to. The SVG Web Service will contact the Security Server to both authenticate (based on the user's digital signature) and authorize users who request SVGs.

To prevent someone from setting up a fake Security Server in an attempt to gain access to SVGs that have been registered, all messages sent from the Security Server are digitally signed. Moreover, every message sent by each user is digitally signed by that user. Although digital signatures prove that a message is from a particular user and that this message has not been changed, they do not prevent others from seeing what that message is. Therefore, most messages sent within the system are encrypted.

Searching by SVG web service name: A search in which name criteria is specified will find all SVGs whose names meet certain criteria. Some of the SQL-92 syntax such % (matches any 0 or more characters) and _ (matches any one character) is supported (this support is actually built into the JAXR\UDDI API). For example, if you enter *%SVG%* in the "Search By Name" field, all SVGs which have the string "SVG" inside their names will be found. However, if you enter just SVG, all SVGs which begin with the string "SVG" will be found. If the "Exact Match" search criterion is specified, only SVGs whose names match the contents of the "Search By Name" field exactly will be found. For example, if I were to enter "Blue", only SVGs whose names are "Blue" would be found. Suppose you want to find all SVGs whose names include the strings "SVG" and "Service". This could be done by entering the following string into the "Search By Name" field: %SVG%Service%. Moreover, suppose that you want to find all SVGs whose names include the string "SVG" or "Service". This can be done use the | character.

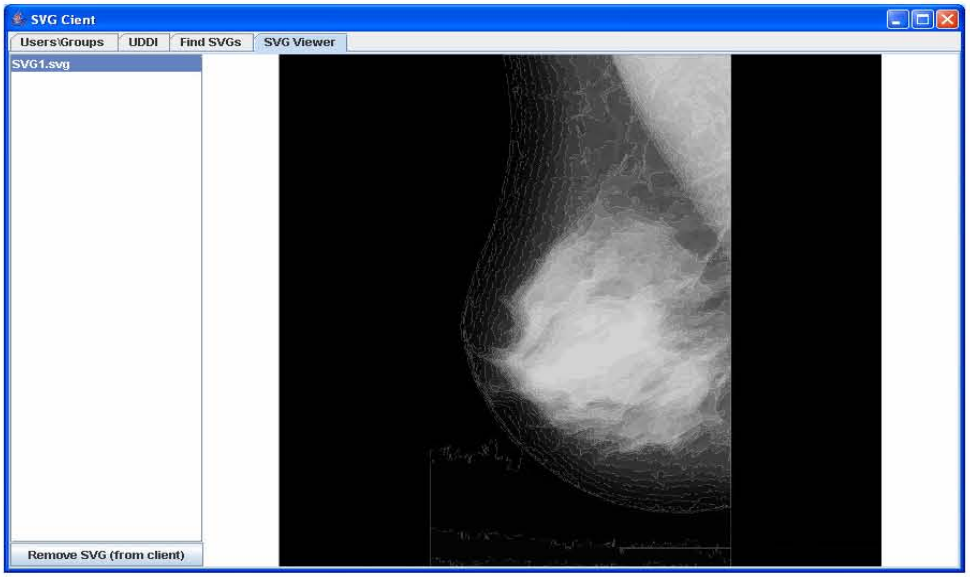


Fig. 9: The SVG viewer

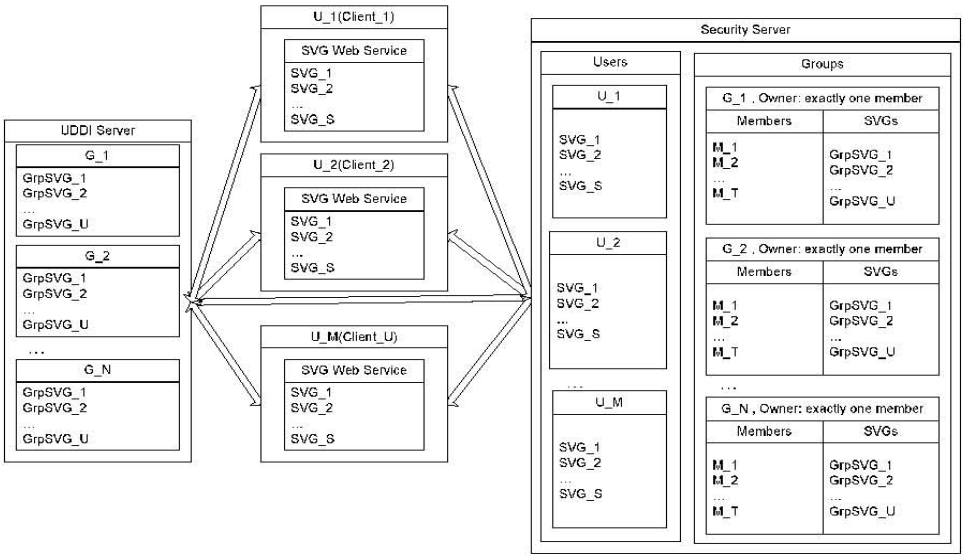


Fig. 10: The UDDI SVG client system components

The following string would be entered into the "Search By Name" field: %SVG%|%Service%. If the | character is entered in the "Search By Name" field more than once, the search will always fail to produce results. I'm not sure why this is happening and I will investigate this matter eventually.

Searching by breast cancer classification: To search for an SVG based upon "Breast Diagnosis", "tumor size", "palpable nodes", or "metastasis", double click the appropriate "Classification" table entry. This will cause the Client to display a window similar to that shown in Fig. 7a. After selecting the appropriate classification, click "OK" to confirm your selection. You can search for SVGs based upon any combination of these classifications.

Note that classification-based searches will always return all SVGs with a more specific classification then the one the user chose to search for as well as with the classification chosen. For example, if you chose to search based upon the "Breast Cancer" classification shown in Fig. 7a, all SVGs with any of the classifications in the black box shown in Fig. 8 would be returned.

Once you have specified your search criteria, click the "Get SVGs" button. This will cause a list of SVGs to be retrieved from the UDDI server. If any SVG in the list is clicked, that SVG will display in the "SVG Viewer" tab, as shown in Fig. 9.

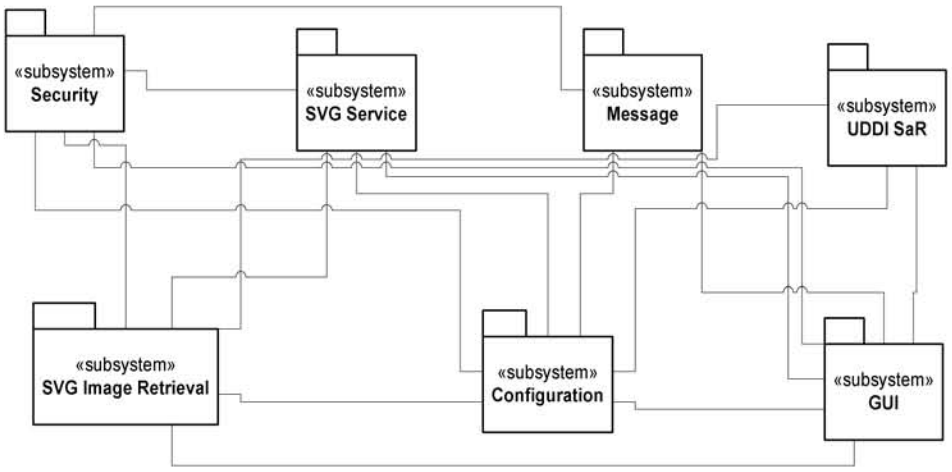


Fig. 11: The UDDI SVG client subsystems

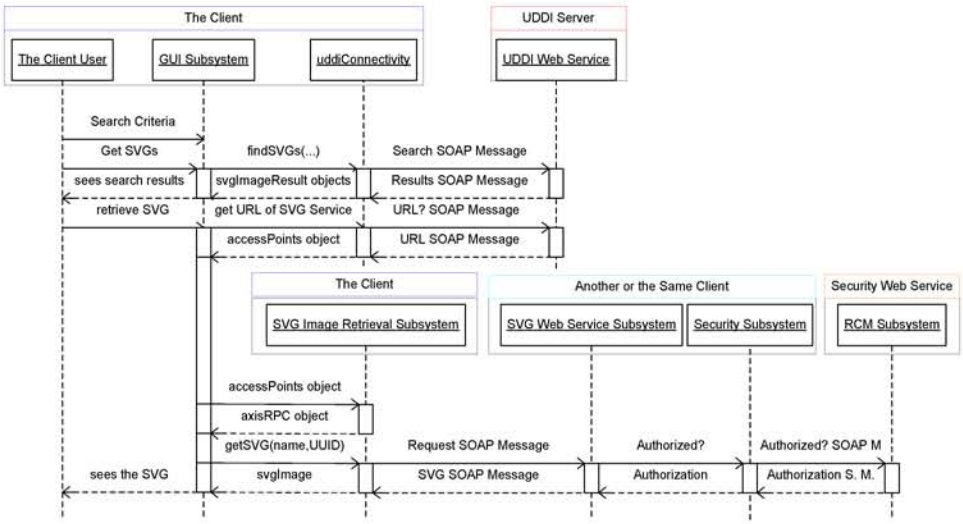


Fig. 12: A sequence diagram illustrating how the various subsystems are called after the user press get SVG button

Searching by group: To search by group, double click the text field next to the "Search By Group" check box shown in Fig. 7b. You can retrieve a list of groups based on any combination of the following criteria: whether or not you "own", "belong to", or "don't belong to" a group and whether or not the name of the group contains a certain string. If the "Name Containing" text field is left blank, it will be ignored. Once you have selected your search criteria, click the "Submit Query" button to retrieve a list of groups. If the group that you are looking for appears in the results list, select that group and then click "OK". The name of this group should now appear in the text field next to the "Search By Group" check box.

CONCLUSION

The AWSMN provides a secure means of sharing medical SVG files between Clients which utilize Apache Axis. The architecture of the system is shown in Fig. 10. We can see that the system has three main components: a Security Server (Security Web Service), a UDDI Server (UDDI Web Service) and any number of Clients.

At each client there is exactly one user and one SVG Web Service associated with each Client. However, the SVG Web Service allows the user to share any number of SVGs with other users who belong to particular groups. Each Client has a special folder in which the user can place all SVGs that should be accessible by the SVG Web Service and therefore, other users who belong to the appropriate groups. Each user can create and therefore, own any number of groups. The owner of a group decides who is allowed to join that group and is also a member. Each group can have a subset of all users registered as members and will always have at least one member (the owner). Each user can and should register the SVGs offered by his

SVG Web Service with the Security Server. During this registration process, the user will specify which groups his SVGs can be accessed by (Currently, an SVG can be accessed by exactly one group) and therefore, the set of SVGs belonging to a group is a subset of the union of all SVGs owned by its members. Of course, before a user can register SVGs to a particular group, he must be a member of that group. The Security Server will register each group and the SVGs that belong to it with the UDDI Server.

When a user wishes to search for SVGs, they will contact the UDDI Server directly. Searches can be performed based on a number of criteria types such as name, Breast Cancer classification and group. The system supports hierarchal classification schemes and classification-based searches will return all SVGs with a more specific classification then the one the user chose to search for.

The UDDI SVG Client is composed of seven subsystems, as shown as Fig. 11. In this Fig. 11, a line between two subsystems indicates that some kind of interaction takes place between them. The type interaction taking place is not indicated. Possible interactions include a subsystem accessing an instance variable located in an object in another subsystem, a subsystem calling one of the methods of a singleton object contained in another subsystem, etc. As indicated in the diagram, many interactions take place between the seven subsystems. In fact, there are many interactions taking place between the objects and classes of each subsystem as well.

Moreover, Fig. 12 shows the interactions that occur when a user searches for SVGs. This diagram shows what happens when a user attempts to view an SVG. After the user enters the search criteria, he will click the "Get SVGs" button shown in Fig. 7. The GUI Subsystem will pass the search criteria to uddiConnectivity findSVGs(…) method which will use JAXR to create and send a SOAP message that specifies search criteria for a tModel classification-based search. This SOAP message will be sent to the UDDI Web Service. The UDDI Web Service will reply to uddiConnectivity with a SOAP message containing the results of the search. UddiConnectivity will store the results of the search in a number of svgImageResult objects and will send them to the GUI Subsystem. Finally, the GUI Subsystem will display the search results for the user. However, it is important to mention that the GUI subsystem has the classification stored as JTable. With each client the GUI load the following four classification schemes: Breast Diagnosis, tumor size, palpable nodes and metastasis. Indeed, the GUI could easily be changed to check which classifications exist in the myClassificationSchemes.xml file. It could then simply create a tree object to store each classification it finds and associate each tree object with a column of the JTables. Therefore, assuming that the program would come preinstalled with appropriate classification schemes, only minimal changes are needed.

When the user indicates his desire to view an SVG, the GUI Subsystem will ask uddiConnectivity to retrieve the URL of the SVG Web Service offering the SVG from the UDDI Web Service. When it receives the UDDI Web Service's response, uddiConnectivity will return an accessPoints object, which contains the URL, to the GUI Subsystem. The GUI Subsystem will retrieve a reference to the appropriate axisRPC object from the axisRPC pool found in the SVG Image Retrieval (SVGIR) Subsystem. Using this object, it will tell the SVGIR Subsystem to retrieve the SVG. The SVGIR Subsystem will send an encrypted and signed SVG Request SOAP message to the appropriate SVG Web Service, which may be located within the same Client or another user's Client.

When the SVG Web Service receives the user's request, it will tell the Security Subsystem to ask the Security Web Service if the user is authorized to view the SVG. The Security Subsystem will react by sending the Security Web Service a signed and encrypted Authorized? SOAP Message. The Security Web Service will reply with a signed and encrypted Authorization SOAP Message, which will indicate to SVG Web Service whether or not the user is authorized to view the SVG. If the user is authorized, the SVG will be sent to the SVGIR Subsystem of the requesting user's Client within a signed and encrypted SOAP Message. The SVGIR Subsystem will send an svgImage object containing the SVG to the GUI Subsystem, which will display the SVG for the user.

This research also aims to extend AWSMN to work for peer-t-peer environments based on JXTA[17] through adding a bridge between AXIS and JXTA[18].

REFERENCES

1. Le Bozec, C., E. Zapletal, M.C. Jaulent, D. Heudes and P. Degoulet, 2000. Towards content {based image retrieval in HIS} integrated PACS. In Proc. of the Ann. Symp. of the 18 Am. Soc. for Med. Informatics (AMIA), pp: 477-481, Los Angeles, CA, USA.
2. Alfred, C.W., J. Samuel Dwyer III and Andrew M. Snyder, *et al.,* 2003. Federated, secure trust networks for distributed healthcare IT services. IEEE IECON, Roanoke, Virginia, USA.
3. Jung, B., E.P. Andersen and J. Grimson, 2000. Using XML for seamless integration of distributed electronic patient records. In Proc. of XML Scandinavia Conf., Gothenburg, Sweden.
4. Health Level Seven XML Patient Record Architecture [http://xml.coverpages.org/hl7PRA.html]
5. Stephen, I., T.C. Wong and A. Donny, 1999. Tjandra, A digital library for biomedical imaging on the internet. IEEE Commun. Mag.

6. Quint, A., 2003. Scalable vector graphics. IEEE Multimedia, 3: 99-101.
7. Scalable Vector Graphics (SVG) – XML Graphics for the Web – http://www.w3c.org/Graphics/SVG (2003).
8. Vincent, H., 2003. Using SVG to create compelling user interfaces for web services. XML Europe 2003 Conf., London, England, May 5-8. http://www.idealliance.org/papers/dx_xmle03/papers/02-04-05/02-04-05.pdf
9. Hondo, M., N. Nagaratnam and A. Nadalin, 2002. Securing web services. IBM Systems J., 41: 2.
10. Yao, J., K.J. Lin and R. Mathieu, 2003. Web services computing: Advancing software interoperability. IEEE Computer J., pp: 35-37.
11. George, W., 2004. Secure SSO for web services. Information Security J., Jan. 2004.
12. Paul, M., 2003. WS-trust: Interoperable security for web service. XML.COM Online J., Jun. 2003, http://www.xml.com/pub/a/ws/2003/06/24/ws-trust.html
13. Sabah, M., J. Fiaidhi, H. Ghenniwa and M. Hahn, 2006. Developing a secure web service architecture for SVG image delivery. J. Computer Sci., 2: 171-179.
14. Hahn, M. and S. Mohammed, 2005. The axis SVG exchange system. NSERC/Coop Placement Technical Report. Department of Computer Science, Lakehead University.
15. Steve, G., D. Davis and S. Simeonov, 2005. Building Web Services with Java. Sec. Edn. Sams Publishing.
16. Frank Sommers, "Publish and find UDDI tModels with JAXR and WSDL", Java World online Journal, Dec., 2002, http://www.javaworld.com/javaworld/jw-12-2002/jw-1213-webservices.html
17. Mohammed, S. and J. Fiaidhi, 2005. Developing secure transcoding intermediary for SVG medical images within peer-to-peer ubiquitous environment. 3rd Ann. Commun. Networks and Services Research Conf. (CNSR'05), pp: 151-156.
18. Burton, K., 2002. JXTA soap bridge project. http://relativity.yi.org/jxta-bridge