Chapter 7

BRINGING SEMANTIC SECURITY TO SEMANTIC WEB SERVICES

Authors: Richard S. Patterson[1], John A. Miller[1], Jorge Cardoso[2], Mike Davis[3]
[1]University of Georgia, [2]University of Madeir, [3]U.S. Department of Veterans Affairs& Chair HL7 Security Committee

Abstract: Semantic Web services begin to emerge as the next evolution of the Service Oriented Architecture. It is become clear that authorization is going to be one of the biggest challenges. The typical obstacles which most areas of Semantic Web services have had to overcome are what parts of a Web services need semantic information, how best to use the semantics, and agreeing on standards. However, for authorization there are the fine grained security implications as well. For instance, how much authorization information is necessary to aid in Semantic Discovery of Web services? Is the authorization information opening any new security holes? We will examine a framework for expressing the proper authorization information in order to aid in the Semantic Discovery of Web services in which the requesting service most likely has the authority to invoke.

Key words: Web Services Discovery, Authorization, Semantic Matching of Web Services, Ontology-based matching of Authorization, Web Service Authorization Discovery, Semantic Web Services

1. INTRODUCTION

1.1 The Semantic Web

Currently, the World Wide Web is primarily composed of documents written in HTML (Hyper Text Markup Language), a language that is useful for visual presentation. HTML is a set of “markup” symbols contained in a Web page intended for display on a Web browser. Most of the information

on the Web is designed only for human consumption. Humans can read Web pages and understand them, but their inherent meaning is not shown in a way that allows their interpretation by computers.

The information on the Web can be defined in such a way that it can be used by computers not only for display purposes, but also for interoperability and integration between systems and applications. One way to enable machine-to-machine exchange and automated processing is to provide the information in such a way that computers can understand it. This is precisely the objective of the semantic Web – to make possible the processing of Web information by computers. "The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation." (Berners-Lee, Hendler et al. 2001). The next generation of the Web will combine existing Web technologies with knowledge representation formalisms (Grau 2004).

Currently the Web is under evolution and different approaches are being sought in order to come up with the solutions to add semantics to Web resources. To give meaning to resources and links, new standards and languages are being investigated and developed. The rules and descriptive information made available by these languages allow the type of resources on the Web and the relationships between resources to be characterized individually and precisely.

To give meaning to Web resource and links, the research community has developed semantic standards such as the Resource Description Framework (RDF) (RDF 2002) and the Web Ontology Language (OWL) (OWL 2004). RDF and OWL standards enable the Web to be a global infrastructure for sharing both documents and data, which make searching and reusing information easier and more reliable as well. RDF is a standard for creating descriptions of information, especially information available on the World Wide Web. What XML is for syntax, RDF is for semantics. The latter provides a clear set of rules for providing simple descriptive information. OWL is an extension of RDF and provides a language for defining structured Web-based ontologies which allows a richer integration and interoperability of data among communities and domains.

1.2 Semantic Web Services

Many believe that a new Web will emerge in the next few years, based on the large-scale research and development ongoing on the semantic Web and Web services. The intersection of these two, semantic Web services, may prove to be even more significant. Academia has mainly approached this area from the Semantic Web side, while industry is beginning to consider its importance from the Web services side (Cardoso,

Miller et al. 2005). Semantic Web services are the result of the evolution of the syntactic definition of Web services and the semantic Web as shown in Figure 1.

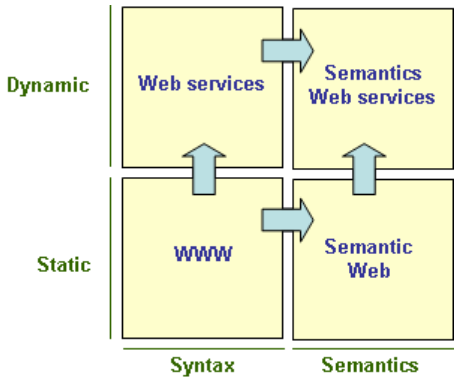

Figure 1 The nature of semantic Web services

Several approaches have been developed to bring semantics to Web services, including WSDL-S (Akkiraju, Farrell et al. 2006), OWL-S (Martin, Paolucci et al. 2004; OWL-S 2004), and WSMO (WSMO 2004; Feier, Roman et al. 2005). The work presented in this chapter uses the first approach, WSDL-S. This approach to creating semantic Web services consists in mapping concepts in a Web service description (WSDL specification) to ontological concepts and it is described into more detail in the next section.

1.3 Semantically Annotated Web services: WSDL-S

One solution to create semantic Web services is by mapping concepts in a Web service description to ontological concepts. Using this approach, users can explicitly define the semantics of a Web service for a given domain. With the help of ontologies, the semantics or the meaning of service data and functionality can be explained. As a result, integration can be accomplished in an automated way and with a higher degree of success.

WSDL-S (Patil, Oundhakar et al. 2004; Rajasekaran, Miller et al. 2004) establishes mapping between WSDL descriptions and ontological concepts. The idea of establishing mappings between service, task, or activity descriptions and ontological concepts was first presented in (Cardoso and Sheth 2003). Figure 2 illustrates METEOR-S WSDL-S Annotator tool (Patil, Oundhakar et al. 2004) and the mapping that have been established between WSDL descriptions and ontological concepts.

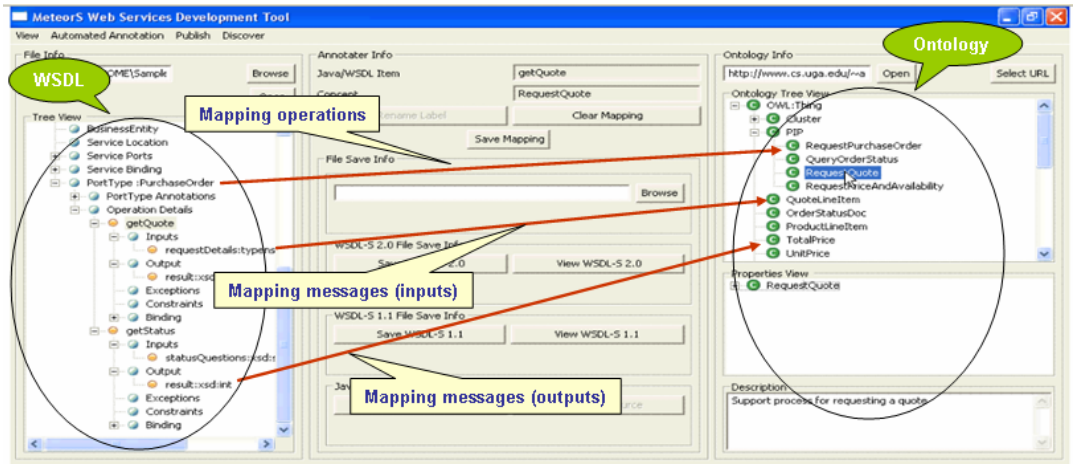

Figure 2 Annotating Web services with ontological concepts

Based on the analysis of WSDL descriptions, three types of elements can have their semantics increased by annotating them with ontological concepts: operations, messages, preconditions and effects. All the elements are explicitly declared in a WSDL description.

- **Operations**. Each WSDL description may have a number of operations with different functionalities. For example, a WSDL description can have operations for both booking and canceling flight tickets. In order to add semantics, the operations must be mapped to ontological concepts to describe their functionality.
- **Message**. Message parts, which are input and output parameters of operations, are defined in WSDL using the XML Schema. Ontologies – which are more expressive than the XML Schema – can be used to annotate WSDL message parts. Using ontologies not only brings user requirements and service advertisements to a common conceptual space, but also helps to use and apply reasoning mechanisms.
- **Preconditions and effects**. Each WSDL operation may have a number of preconditions and effects. The preconditions are usually logical conditions, which must be evaluated to true in order to execute a specific operation. Effects are changes in the world that occur after the execution of an operation. After annotating services' operations, inputs and outputs, preconditions and effects can also be annotated. The semantic annotation of preconditions and effects is important for Web services, since it is possible for a number of operations to have the same functionality, as well as the same inputs and outputs, but different effects.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name = "StudentManagement"
targetNamespace=
"http:.../StudentManagement.wsdl20"
        xmlns="http://www.w3.org/2004/03/wsdl"
        xmlns:tns="http.../StudentManagement.wsdl20"
        xmlns:sm=“http:.../StudentMng.owl#"
        xmlns:mep=http:.../TR/wsdl20-patterns>
<interface name = "StudentManagementUMA">
        <operation name = "RegisterStudent"
                pattern = "mep:in-out" >
        <action element =

"sm:RegisterStudent" />
        <input messageLabel = "student"
                        element = "sm:StudentInfo" />
        <output messageLabel = "ID"
                        element = "sm:StudentID" />
        </operation>
        <operation name = "StudentInformation"
         pattern = "mep:in-out" >

                <action element =

        "sm:StudentInformation" />
                <input messageLabel = "ID"
                                element = "sm:StudentID" />
                <output messageLabel = "student"
                                element = "sm:StudentInfo" />
        </operation>
        <operation name = "checkStatus"
         pattern="mep:in-out" >
...
        </operation>
</interface>
</definitions>
```

Figure 3 WSDL example

The WSDL-S specification indicates that the Web service supplies two operations: ‘RegisterStudent’ and ‘StudentInformation’. The first operation has an input named ‘student’, semantically described by the ontological concept “sm:StudentInfo”, and an output named ‘ID’, semantically described by the concept “sm:StudentID”. The operation ‘RegisterStudent’ is semantically annotated with the ontological concept “sm:RegisterStudent”. The second operation, ‘StudentInformation’, uses similar ontological concepts to annotate the input, output, and action. The ontological concepts are expressed in the ontology

http://dme.uma.pt/jcardoso/StudentMng.owl#, which is specified using OWL (OWL 2004).

To create, represent, and manipulate WSDL-S documents, WSDL4J (http://sourceforge.net/projects/wsdl4j/) can be used. WSDL4J provides JAVA API's for WSDL parsing and generation. WSDL4J supports extensibility elements providing an easy mechanism to add new extensions. This allows WSDL to represent a specific technology under various elements defined by WSDL.

2. WEB SERVICES SECURITY BACKGROUND

Web services can be used to expose inter-organizational components such as business critical data, business processes and internal workflows [Shivaram, 2003]. Organizations may expose some of these components in order to capitalize on the cost savings and reduced complexity that Web services can add to there SOA. Because the SOA is more dynamic, loosely defined, and ubiquitous, new security measures are needed to protect key business information. There are currently standards proposed or accepted regarding authentication, encryption, and identity management. These areas of Web service security use a combination of tried-and-true technologies such as keys, username token, and RSA encryption, along with newer technologies such as XML signature [XML-Signature, 2002] and SAML (Security Assertion Markup Language) [SAML 2.0, 2005].

In securing Web services, there are five fundamental areas to consider; Message Level Protection, Message Privacy, Parameter Checking, Authentication, and Authorization. When examining these areas it is important to stay within the context of Web services and not network security in general. This is because network security is at a different layer of the ISO model; Web service security is at the application layer. As we discuss these areas of security, observe the following. Some of solutions use the same or similar technologies to achieve vigilance. Not all of the technologies used were developed for Web services and may have been around for many years. Which of these areas could benefit from semantics? Four of the five areas have been addressed; however, authorization has not. Authorization aided by Semantics is not only important in Web services, but the Semantic Web as well.

As Web services continue to evolve into Semantic Web services for automated discovery and execution of business processes [Verma, 2005], two questions become more prevalent. From the Service Providers perspective, how much information should be shared with an entity to which there is no previous relationship. From the Requesters perspective, how

does the Requester know if they will have access to the information and resources they discover through automated discovery?

We will begin our discussion by reviewing the technologies currently in place to secure Web services. The following section take a look at an approach which uses semantics, along with current technologies, to aid Providers and Requesters in answering the questions posed above.

2.1 Message Privacy

Message privacy deals with the confidentiality of a message. Here unauthorized entities should not be able to access the information within the message. It is important to remember here that a part of this information is the XML Signature and Token, found in the message header, which can be seen in Figure 4. To ensure confidentiality an encryption scheme must be implemented.

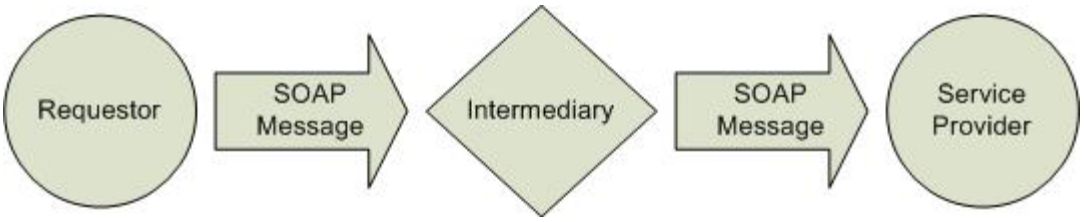

Figure 4 Soap Message in transit

Since Web services can be chained together to form complex services, traditional point-to-point encryption schemes, such as SSL, do not suffice. Point-to-point schemes work at the Network layer of the ISO model. Therefore, once the message has been received by an entity it is decrypted in its entirety. This entity may be an intermediary and not the provider of the service, Figure 4. Furthermore, a message may cross multiple trust domains due to routing caused by elaborate messaging communications [Web Services Architecture, 2004]. What is needed is an end-to-end encryption scheme.

The XML Encryption standard provides the necessary framework for accomplishing this task. XML Encryption allows for the encryption of any combination of the message body, header, attachments, and sub-structures [XML-Encryption, 2002].

When a message or part of a message is encrypted, the encryption information can be made available in the message header. This information is useful for complex services since each Web service in the chain will need to know how to decrypt the section of the message relevant to their service. This information should not be the actual key to decrypt the message.

An example will clarify. When a requester encrypts a message body and XML Signature information in the header, it may then specify in the header that it has used the providing service's public key. A public key allows for the encryption of data but only the private key may decrypt the data. Once the provider receives the message it sees that the message has been encrypted using its public key. The provider then decrypts the message using its private key.

XML Encryption allows multiple different keys to be used with in a message to encrypt different sections, elements, of the message. Each encrypted section is referenced in the message header and mapped to the key information if provided. This provides end-to-end encryption through intermediaries which may also be accessing the parts of the message.

Table 1 - XML Algorithms

Purpose	**Algorithm**	**Specified as**
Digest	SHA1	Required
Digest	SHA256	Recommended
Signature	DSAwithSHA1 (DSS)	Required
Signature	RSAwithSHA1	Recommended
Canonicalization	Canonical XML (omits comments)	Required
Canonicalization	XML with Comments	Recommended
Transform	XPath	Recommended
Transform	Enveloped Signature	Required

Table 1 above provides an overview of the algorithms specified in the XML-Signature and XML-Encryption standards. Required algorithms are the minimal to comply with the standard; while recommended is just that, recommended.

2.2 Message Level Protection

Message Level Protection has to do with message integrity. This means being able to detect when a SOAP message (message) has been modified from its original state and the ability to guarantee that the contents have not been modified [Web Services Architecture, 2004]. This is done by creating a message digest.

A message digest is an encoded message, a cryptographic checksum of an octet stream [WS-Security, 2002], which is created using an algorithm. The SHA-1 algorithm [NIST, 1993], in Table 1, is required in the XML Signature specification. The only parameter is the element to be signed. The provider of the Web service receives the message, the digest, and is told which algorithm has been used to create the digest. Using this information, the service provider is able to recreate the digest and compare it to the digest which it received from the requester.

When a message is passed from a Requester Web service to a provider Web service, the message body should be digitally signed using the XML Signature specification. There are several Token options for signing a message. These options fall under one of two categories; they can either be endorsed or unendorsed. An endorsed token is one which the claims of the Token can be validated by a trusted authority. An example of this kind of Token is a X.509 certificate. An unendorsed Token is one which the claims may not be validated by a trusted authority. However, there is such a thing as a proof-of–possession unendorsed Token. An example of this is a username-password Token.

When signing a message, the signature parameters consist of a security token and the message digest. It is worth noting that the second parameter can be an XPath node-set. The output is the message signature which will appear in the message header. Figure 2 [WS-Security, 2002] shows the key or token used to sign the message (1), the message digest (2), the signature value (3), and the unencrypted message body (4).

The provider service must have a way to verify the contents of the message. In order to do so the provider must have the message, the digest, determine the algorithm used to create the digest, and access the key or token. Security elements in the header of the message contain information on the algorithm and Token. The provider can use this information to compare the digest to the message. Any changes, even the addition of one white space to the original message can be detected. This clearly solves the problem of Message Level Protection.

2.3 Message Validity

Message Validity is ensuring that the contents of a message are appropriate to the service and that they are well formed. Checking the contents of a message can be subdivided into two categories, verifying data types and checking for malicious code. Verifying that the data types passed to an operation are those which the services are expecting is straight forward. Checking for malicious code within the message is not so straight forward.

```
<?xml version="1.0" encoding="utf-8"?>
<S:Envelope
        ....
   <S:Header>
      <m:path xmlns:m="http://schemas.xmlsoap.org/rp">
         .....
      </m:path>
1     <wsse:Security>
<wsse:BinarySecurityToken ValueType="wsse:X509v3"         EncodingType="wsse:Base64Binary"
Id="X509Token">
                  MIIEZzCCA9CgAwIBAgIQEmtJZc0rqrKh5i...
         </wsse:BinarySecurityToken>
         <ds:Signature>
            <ds:SignedInfo>
               <ds:CanonicalizationMethod Algorithm="http://....#"/>
               <ds:SignatureMethod Algorithm="http://...."/>
               <ds:Reference>
                  <ds:Transforms>
                     <ds:Transform Algorithm="http://...#RoutingTransform"/>
                     <ds:Transform Algorithm="http://.....#"/>
                  </ds:Transforms>
                  <ds:DigestMethod Algorithm="http://...#sha1"/>
2                 <ds:DigestValue>EULddytSol...</ds:DigestValue>
               </ds:Reference> </ds:SignedInfo>
3           <ds:SignatureValue>
              BL8jdfToEb11/vXcMZNNjPOV...
            </ds:SignatureValue>
            <ds:KeyInfo>
                <wsse:SecurityTokenReference>
                    <wsse:Reference URI="#X509Token"/>
                </wsse:SecurityTokenReference></ds:KeyInfo></ds:Signature>
</wsse:Security></S:Header>
   <S:Body>
4     <tru:StockSymbol xmlns:tru="http://..../payloads">
        QQQ
      </tru:StockSymbol>
   </S:Body>
</S:Envelope>
```

Figure 5 Soap Message with XML-Signature

Malicious code within a message can appear as part of the XML message or as parameters to be passed to operations. XML viruses and XML worms are commonly passed within the contents of any XML document or message [Lilly, 2002]. Because these are common in the Web environment there is software available to safely scan XML to determine if it contains either a virus or a worm.

Even after verifying that the parameters within a message are appropriate for the operation(s), their may be malicious code present. For example, it may be verified that a string is being passed to an operation which then queries a SQL database. SQL injection attacks are of the string data type. Therefore verifying that a string is being passed is not enough. Best practices for programming disallow and check for the presence of a ';' in any parameter which will be passed to a SQL database. The ';' in SQL allows for SQL commands to follow.

Ensuring that a message is well-formed is another step in Message Validity. Since the messages are in XML, it is possible that a message contains a circular-reference. A circular-reference may appear maliciously or through poor programming. Circular-references cause a system to encounter a run-out-of-memory error and shutdown [Lilly, 2002]. When done maliciously this is know as a denial-of-service attack. Proper parsing of a message will catch nested loops.

2.4 Authentication

Authentication can easily be described as verify to ones own level of certainty that an entity is who they claim to be. In its simplest form, authentication could be a username and password combination. However, this is only possible if there is already a relationship between the requester and provider.

Because of the distributed nature of Web services, a requester may be previously unknown to the provider. When an unknown requester authenticates it sends information about themselves to the provider. This information is known as credentials. It is up to the provider to verify this information. Now there are different degrees of verifying credentials and this can be directly affected by the type of credential that is sent. This is where the provider's own degree of scrutiny comes into play. In general, the more sensitive the information is which is being made available through a Web service, the higher the level of certainty must be. This certainty can be achieved through verification of the credentials. In the case of a previously unknown requester, the highest level of certainty can usually be achieved through a trusted authority. Trusted authorities issue certificates which can be used for authentication. A provider can evaluate the certificate and contact the trusted authority for verification.

However, their may be an intermediate service contacting the provider on behalf of the requester and once established the requester and provider will communicate. Assuming that the intermediary has authenticated the Requester and there is a trust relationship between the intermediary and the provider, the provider may take the 'word' of the intermediary and believe with a level of certainty that the requester is whom they claim to be. This can be done through the use of SAML or a certificate issued by the intermediary. Here the intermediary is providing the verification.

3. AUTHORIZATION

In organizations, highly sensitive data and information must be protected with access control systems. These control systems allow defining and controlling which users are authorized to access specific applications and data but prohibit the access of unauthorized users.

Nowadays, organizations are built on heterogeneous IT infrastructure. As a result, a variety of systems with proprietary access control mechanisms, such as Unix, Windows, MAC, and mainframes exist and are incompatible. In proprietary access control systems, information about resources and attributes is stored in repositories called Access Control Lists (ACL). This is a problem since different proprietary systems have different ACL implementations, making it difficult to exchange and share information between them.

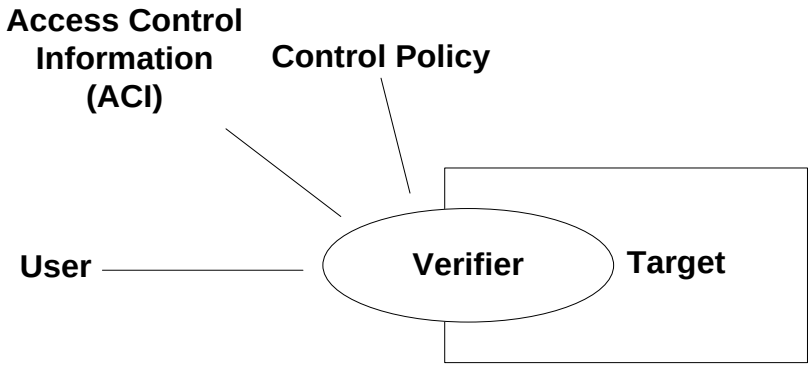

Figure 6 Access Control

Authorization is the granting of rights, which includes the granting of access based on access rights. This typically takes place after authentication. Authorization is often confused with authentication, however it is a separate issue altogether. An access control implementation compares access control information such as the rights of the Requester with the policies or

permissions needed to access the resource. If the rights of the Requester dominate the control policy, then access can be granted; otherwise access is denied. The two most common access control implementations are Access Control Lists (ACL), and RBAC. ISO 10181-3 specifies access control information used in making access control decisions.

3.1 ACL's

ACL's are often used in the Unix environment for file and directory security. Although ACL's offer much more granularity than pervious *nix access control mechanisms, they can be cumbersome to implement and manage. There are difficult to manage because of the lack of relationships between the access control entities, i.e., resources, permissions, groups, and users. There is an obvious relationship between users and groups, users belong to groups and groups contain users. However, each shared resource must have an ACL file specified for it and the associated permissions are held within the file.

```
# file: documents
# owner: somebody
# group: other
user::rwx
user:jackson:rwx  #effective:rwx
user:smith:rwx                  #effective:rwx
group:publ:rw-                  #effective:rw-
mask:rwx
other:---
```

Figure 7 ACL example

Users within groups can easily be managed, but for resources that change frequently like those in Web services it is difficult to modify the ACL's for all these resources. Therefore management of a ubiquitous and dynamic resource environment is cumbersome at best. Furthermore, performance is affected each time ACL is accessed and inspected. A simple example ACL is given below in Figure 4.

3.2 RBAC

In 2004 the National Institute of Standards and Technology (NIST) published a standard [NIST, 2004] for defining the features of Role Based Access Control. The standard was largely based on the various features found in commercial implementations of RBAC. There are two parts to an RBAC system. The first is the Reference Model which consists of objects, operations, permissions, roles, and users. The second is the System and

Administrative Functions which include system functionality, and administrative operations and reviews [NIST, 2004]. Our approach utilized the concepts of RBAC, so we will discuss it here. However, much has been written about RBAC over the past decade; in an effort not to be repetitive this is a summary-review.

RBAC contains Permissions sets. Generally speaking, Permissions express a privilege to access a resource. Permissions are a set of one or more objects and one or more operations. Objects refer to resources; for example a printer or a file. Operations are the invocation or execution of some function on a resource. An example of a Permission may be "create file in directory etc'. In this case, 'etc' is the Object and 'create file' is the operation.

Once Permissions are created, they may be assigned to Roles. A Role is a job function which is performed within an organization. A job function can be as concrete as a job title, 'Physician', or more general even abstract, 'internet user'. Once Roles are defined they may be assigned to Users who are actual people. Users may also include entire organizations, computers or networks [NIST, 2004].

The HL7 committee has developed a simple and effective way for creating Permissions (Object, Operation set) and Roles. They call it a scenario driven approach. The concept is to first create scenarios for the 'organization'. These scenarios include resources, actions taken on the resources, and who is performing these actions, in terms of job function. Our Detailed Scenario is an example of this approach.

Figure 5 shows the relationships between the elements of the RBAC Reference Model. Permissions are an Object, Operation set. There is an assignment between Permissions and Roles, and Roles and Users. These relationships/assignments are many-to-many. There is another element present in the figure which has a one-to-many assignment with Users. That element is the Session Role [NIST, 2004].

A Session is the activation of one or more Roles by a User. Simplistically and not entirely, a Session Role determines if a Users Role should be activated. This is determined by the constraints on the Roles assigned to the User and which Roles the User currently has active. The RBAC Reference Model provides fine granularity for authorization of resources. The RBAC Systems and Administrative Functions provide for distributed decision and enforcement points.

In our approach we conceptually map elements of RBAC to elements of Web services in order to an authorization function regarding the Web service and a prediction function regarding the Requester. In or approach, RBAC Operations are mapped to the action that an operation of a Web service performs. Keeping with our example, the Web service has an operation to review the medical history of a patient; this operation is mapped to the RBAC element 'read'. The RBAC element Object is mapped to the resource

which the service accesses and the parameters of the operation; i.e. the medical history of the patient and the patient respectively.

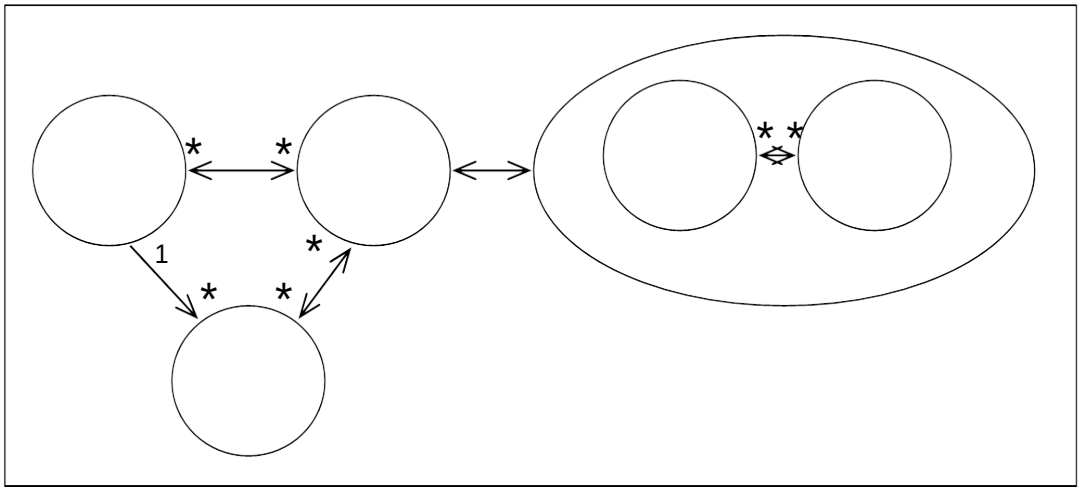

Figure 8 RBAC

We define an authorization function, and as mentioned in section 1.2, there is a variable of uncertainty between what a Web service provider is describing and what a Requester may assume the provider is describing. This variable is caused because our ontology is an upper level ontology; therefore each organization using the ontology must map Users, Roles, and possibly Groups to the ontology. This mapping is independent of any other organization; therefore there is uncertainty that a Requester and service provider have the same mapping. Our goal is to minimize the variable to achieve accurate results.

We define the authorization function as $f:<UA, S, O, P_{i,}\ldots,P_k) \rightarrow \{0,1\}$ where S is the service, O is an operation, P is a parameter and UA is defined as $UA \subseteq U \times R$. UA is the many-to-many user-to-role mapping relationship within the RBAC ontology. However, to predict authorization we use UA′ which is a many-to-many mapping between UA and (U′×R′), the many-to-many user-to-role mapping within an organization. Therefore, UA′ is defined as $UA' \subseteq UA \times (U' \times R')$, the many-to-many mapping between the ontology and the organization structure. So our prediction function is defined as $f:<UA', S, O, P_{i,}\ldots,P_k) \rightarrow \{0,1\}$

3.2.1 Structural Role Framework

Structural roles serve as access control decision information within the PMI (Privilege Management Infrastructure) by allowing authenticated users to establish a session or connect to a protected target.

To accomplish this function, the user asserts, in addition to authentication information, structural roles as a prerequisite authorization to "connect" to the task or workflow containing the requested session or target. An infrastructure access enforcement function grants or denies access to the session or target based on the structural role. Structural roles would be typically managed in identity certificates (per ASTM E2212) or directories. Structural roles are centrally managed, allowing any user to be granted access, suspended, or denied access (by means of the service-oriented Verifier) to any or all resources through this single point of control.

ASTM E1986 identifies healthcare persons for whom role based access control is warranted. These ASTM E1986 person types define basic healthcare role names as used within this standard.

3.2.2 Functional Role Framework

Functional role activation (session roles) cannot occur until the session is established, so structural role authorization/access is prerequisite to establishing a session or connection to the target. In the extended Control Model, what is desired is a decision on the user's authorizations to perform operations on the Target's protected objects. The result of the decision information is used as an input to the Verifier PEP (Policy Enforcement Point) for the purpose of access control.

Functional roles describe the permissions that a user has available once the session is established and his/her roles are activated. Functional roles are contained in applications, directories, attribute certificates, and XACML extensions. Functional roles specifically define, in terms of permissions, what authorizations are needed by an entity to access protected information technology or application resources. As a consequence, functional roles are much more healthcare specific than basic roles. Standard functional roles are applied across the enterprise and with business partners. Standard functional roles are aligned by mapping to underlying applications' enforcement mechanisms.

Functional roles should be expressed in a standards-compliant language for interoperability both inter and intra-enterprise. A standard language allows for leveraging policy and roles among applications, as well as consistent policy description and enforcement. The guideline standard language for this standard is OASIS XACML.

3.3 XACML

XACML is an OASIS standard for an XML representation of RBAC [XACML, 2005]. The Organization for the Advancement of Structured Information Standards (OASIS) standards group developed the eXtensible

Access Control Markup Language (XACML) as a language to express and evaluate access decisions.[i] The XACML technical specification includes a profile for RBAC using XACML that complies with the ANSI RBAC standard. Core RBAC, as defined above, is supported as shown in Table 2 below.

Table 2 RBAC Core Functionality Mapping

Core Element	XACML Profile
Users	XACML Subjects
Roles	XACML Subject Attributes
Objects	XACML Resources
Operations	XACML Actions
Permissions	XACML Role <PolicySet and Permission <PolicySet>

The XACML RBAC profile also supports hierarchical RBAC, allowing inheritance between roles. Dynamic Separation of Duty is supported by the profile, and structural Separation of Duty can be supported via the user-role assignment mechanism. Additional XACML policies are provided to support system and review functions described in the ANSI RBAC standard. Specifically, the Role PolicySet (RPS) associates holders of a given role attribute with a Permission PolicySet. The Permission PolicySet (PPS) describes the permissions associated with a specific role. The RPS and PPS replace the role assignment and role specification ACs in the X.509 based role model.

The XACML role based PMI features a rich policy language integrated throughout the design. The concept of structural versus functional roles is supported using a two tiered system comprised of a role attributes. That is, users can have roles assigned to them in the request context. An entity separate from the policy decision point can use an XACML Role Assignment Policy or PolicySet to enable attributes within the user session.

Figure 9 illustrates a typical XACML usage scenario. A subject (e.g., human user, application) wants to take some action on a specific network resource, such a file system or Web service. The subject submits its request. The request for authorization goes to the entity protecting the resource, the PEP (Policy Enforcement Point). The PEP uses XACML request language to create a request based on attributes of the subject, action, resources and sends it to the Policy Decision Point (PDP), which evaluates the request. The PDP invoke the Policy Information Point (PIP) service to retrieve applicable policies written in XACML that are applicable to the request. The PDP compares the request against policies and determines whether access should

be granted according to the XACML rules for evaluating policies. Policies contain information about the subject, the action, and other environmental properties. The result of the comparison can be either access granted or denied. The answer goes back to the PEP. If there is no match, the PEP denies user access; otherwise, it permits access by the user.

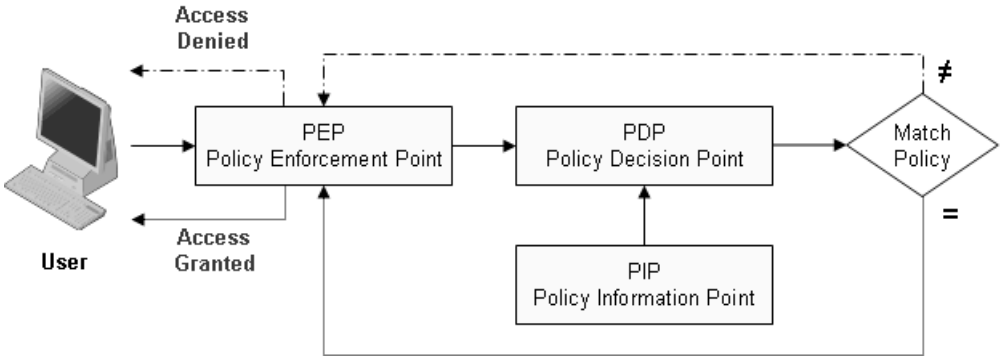

Figure 9 Typical XACML usage scenario

While there are many proprietary languages for controlling the access to resources, XACML has advantages. The use of a standard access control policy language can replace several proprietary languages making easier the interoperation of applications. Programmers and administrators can work more efficiently since they do not have to develop new policy languages and write code to support them and only need to understand one language. The use of a common language allows one policy to be used by many different applications, thus making policy management easier. Policies can also be distributed by referring to other policies stored in geographically disperse locations. For instance, a local-specific policy may refer to an organization-wide policy.

3.4 WS-Authorization

WS-Authorization is a proposed future specification regarding the description and management of authorization data and policies [IBM, Microsoft 2002]. In particular, WS-Authorization will specify a standard on how to describe authorization claims within a security token and how the end-point should interpret these claims. It is widely thought that this specification will be a follow-on specification to WS-Privacy and WS-Security, as seen in Figure 10. WS-Privacy and WS-Security are implemented in WS-Policy, and it is believed that WS-Authorization will be implemented in WS-Policy as well. It is also believed that this specification will be similar in structure to the XACML standard [Rosenberg, Remy, 2004].

IBM and Microsoft concur that the end-point policy files are the appropriate location for describing "execution capabilities" [IBM, Microsoft 2002] of an authenticated Requester. If the WS-Authorization specification ends up being similar to the XACML standard, this would offer many possibilities to an approach which uses RBAC and XACML concepts to annotate WS-Policy files with semantics to aid in the discovery of services which a Requester will be authorized to invoke or execute. By using ontological concepts, in this case RBAC and XACML concepts, to describe execution capabilities of an authenticated Requester, a potential Requester can automate the prediction of their authorization.

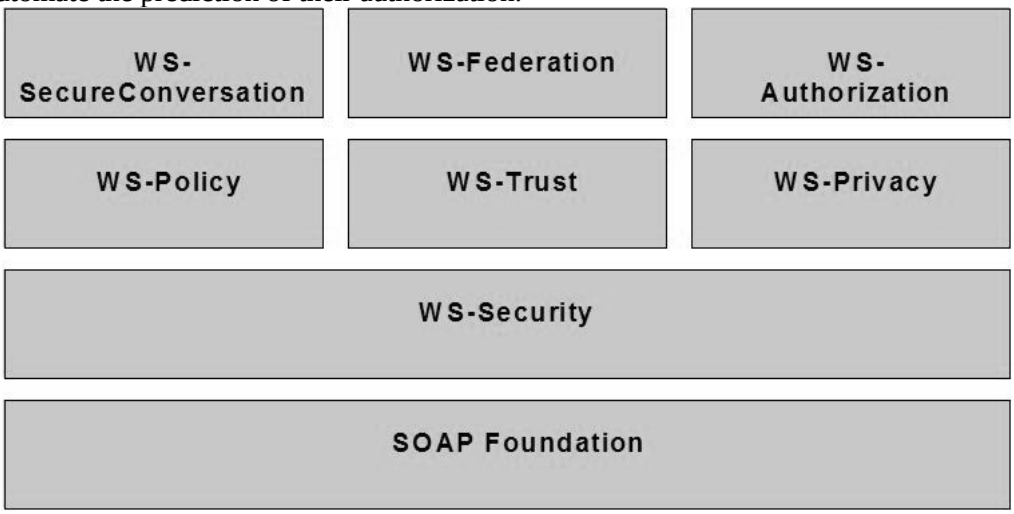

Figure 10 Web services stack

4. ADDING SEMANTICS FOR AUTHORIZATION

4.1 Why Use Semantics

The WS-Policy specification is a model and syntax for describing the policies of a Web services. It relies on its follow-on specifications, such as WS-Trust, WS-Agreement, WS-Security, and WS-Utility, which make within WS-Policy. The assertions are based on an XML based domain vocabulary. A Requester and service provider can make assertions in WS-Policy from any domain using the specifications which describe the vocabulary. When matching policies, if the policy matching mechanism is unaware of the domain context then it would be limited to using syntactical matching. Consider the following example where a Requester and a service provider have included authorization assertions from the Health Care domain.

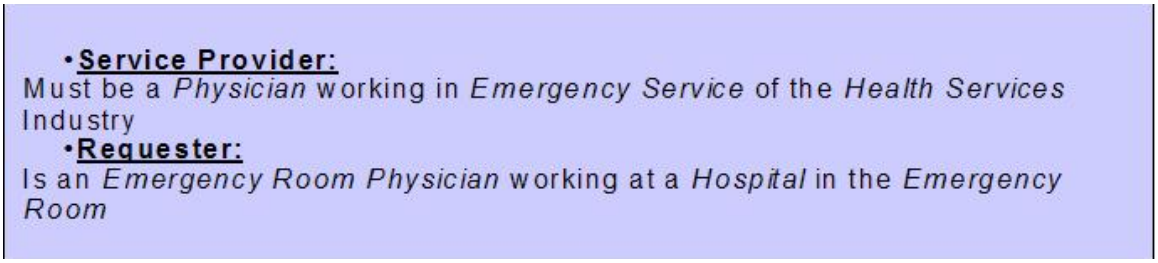

Figure 11 Roles in Web service example

These assertions are equivalent. The domain knowledge needed to determine that these assertions are equivalent are absent in a purely syntactic matching mechanism. Therefore, using a string matching algorithm would result in the denial of authorization for the Requester, which is a false negative result. These assertions can easily be determined to be equivalent by using domain information along with semantic reasoning. From the example, it can be determined that $\forall$*Physician WorkIn Emergency Room (Physician)* $\Rightarrow$ *provides Emergency Services*; that is to say that a *Physician working in an Emergency Room* is an entity that *provides Emergency Services*.

There are several key "ingredients" that are needed for a semantic solution to the distributed authorization problem; which is after all what we are talking about. The first ingredient is Domain Knowledge and as we discussed, the domain is security, more precisely authorization. There will most likely be a second domain, such as the medical domain which we will use in our examples. The second ingredient is a means to express constraints. Also as we discussed, WS-Policy seems to be the appropriate place for to express constraints for the Web services arena. The third ingredient is how to express the constraints in the Policy file. The last ingredient is a means to compare the constraints with information about the Requester. Let's now look at each of these ingredients in more detail.

4.2 Ontology

We will discuss a HL7 RBAC ontology which is represented in OWL-DL [OWL, 2004], *Web* Ontology Language - Description Logics. It begins with two separate Upper Level domain ontologies, a RBAC ontology and a HL7 ontology, as seen in Figure 8. Then a HL7 RBAC ontology is created by expanding the RBAC Upper Level ontology through the use of the HL7 RBAC Permissions Catalog [HL7 Security Technical Committee, 2005]. This catalog contains operations and objects which have been paired together to form permissions. For other domains in which an RBAC standard is not available, concepts from a domain may be imported into the RBAC ontology in order to create a domain specific Mid Level ontology.

A comprehensive list of medical departments [Hull and East, 2006], and a list of the 31 broad industry categories provides the information for additional Domain Knowledge.

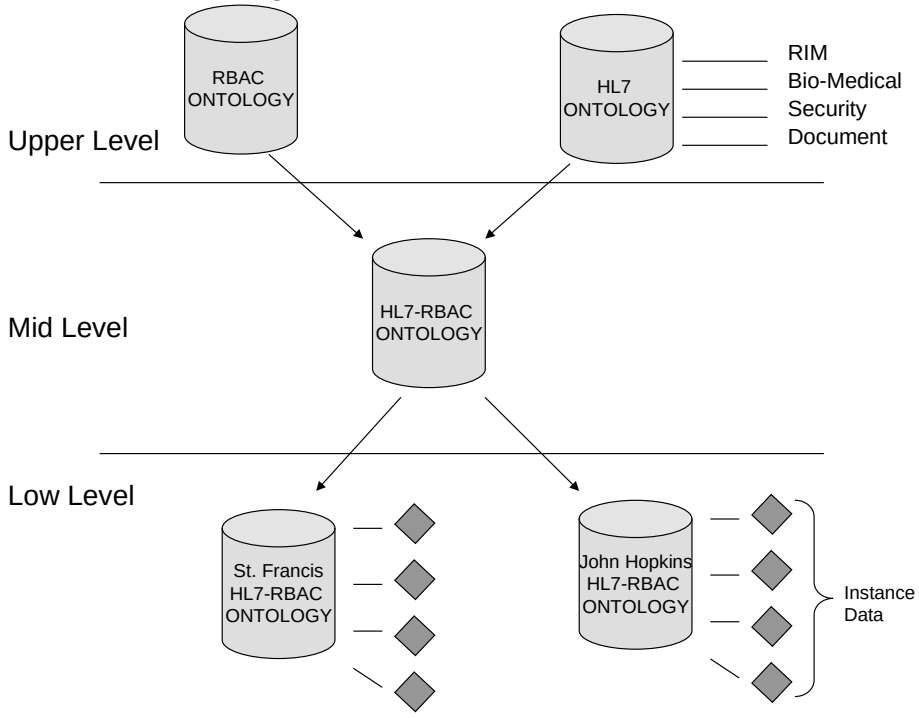

Figure 12 HL7 & RBAC Ontology Hierarchy

In the Figure there are two more specialized Lower Level ontologies, one for the requester and one for the service provider. These are created by extending the mid level ontology in an effort to more accurately model the real world. The requesters' ontology should be developed to reflect its organizational implementation. This can be done by adding users, assigning them to roles, using variations of the role names, and assigning appropriate permission to these roles. The service providers' ontology is extended in much the same way with the exception of not adding users, which is practical for security reasons. The fundamental difference between the ontologies is variation of role to permissions assignment, as well as role names. For the sake of a real world argument you will notice that we included some different role names between the ontologies, for the same job function. For example, 'Radiology Technician' and 'Radiology Tech', 'General Physician' and 'Family Practice Physician', and 'Pediatric Nurse' and 'Pediatric RN'. The above titles are all standard titles for positions in the Health Care domain. Since many organizations will implement systems

using variants of position titles. We will discuss the use of these two Ontologies in a few moments.

Figure 9 below shows a portion of our HL7 RBAC Mid Level Ontology. The 'RBAC Reference Model Elements' is the parent to the actual elements, namely Objects, Operations, Permissions, Role, Session, and Users. There are relationships between these elements, more accurately ontological concepts. A Semantic Authorization technique, such as this, exploits these relationships. Let us now describe some of the relationships.

Figure 13 Classes in the RBAC Ontology

As we stated earlier, permissions have a 'has object' relationship with Object and a 'has operation' with Operation. A Role has a 'assigned to' relationship with User, a 'department' relationship with Health Service (which is not visible in the figure), 'has permission' relationship with Permission, and so on. A User has a 'assigned to' relationship with Role, a 'employed by' relationship with Organization (which is not visible in the figure), a 'isA' relationship with Human, and so on.

As can be seen below, there are many instances of permissions. These are all from the HL7 Permissions specification. This is not an exhaustive list

of the possible permissions; rather the committees' goal was to provide a general starting point which provides examples so that health care organizations could correctly create permissions tailored to their organization.

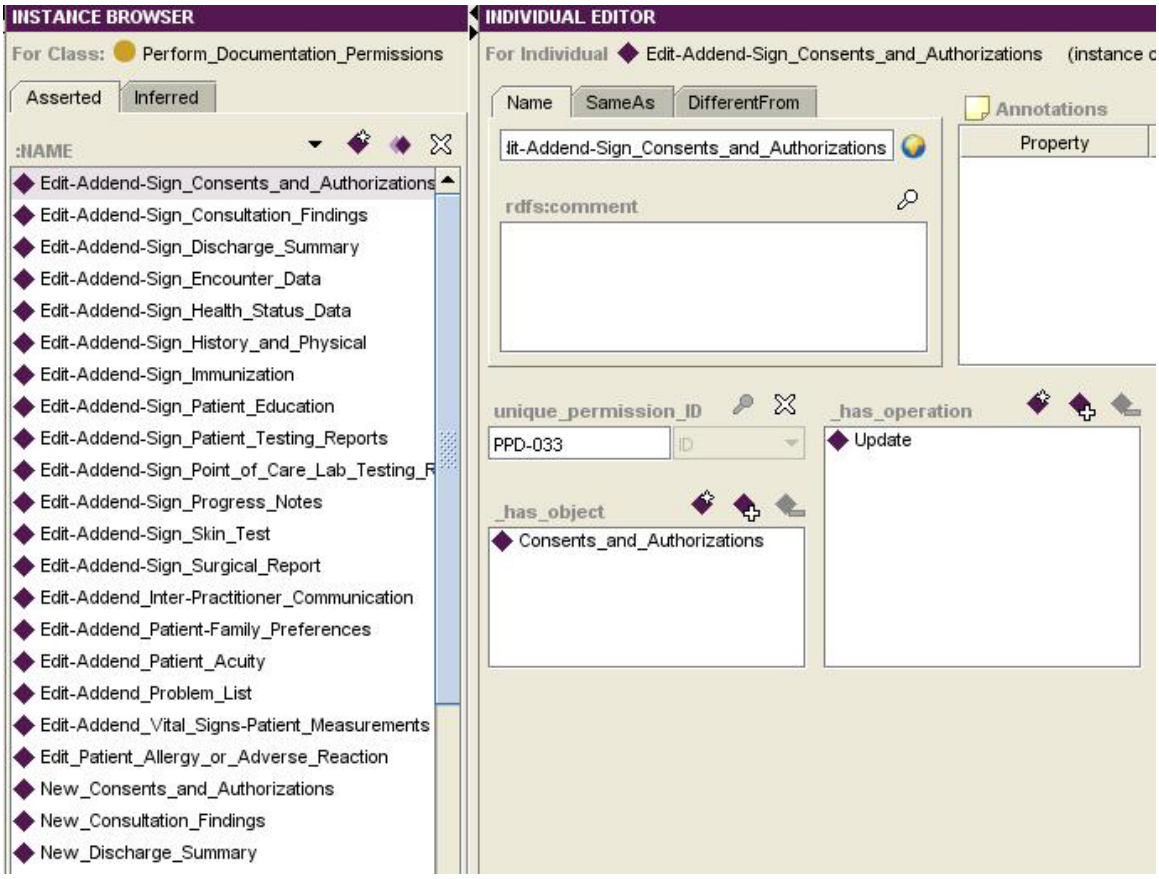

Figure 14 Instances example

4.3 Expressing Constraints – Extension Elements

This section covers the second and third ingredients for a Semantic Authorization approach. There currently is no a WS-Authorization specification. After reviewing current specifications that are built onto WS-Policy (WS-Agreement, WS-Transaction, WS-Security) we assume that a future WS-Authorization specification would follow suite and therefore lack the semantics needed for an automated process. Therefore in our examples we have extended the WS-Policy to include a WS-Authorization specification which incorporates semantics. By automating the predication

of authorization, a requester or consumer can save time and more efficiently allocate its resources.

The authorization annotations are extensibility elements similar to the extensibility elements provided in WSDL-S [Akkiraju et al., 2005], for example precondition and effect. The annotated WS-Policy file, called a SemPolicy in [Verma, 2006], will provide extensibility elements for semantic representation of authorization. The extensibility elements are derived from the RBAC standard [NIST, 2004] and XACML representation of RBAC.

RBAC was chosen because it is widely accepted, easily understood, and succinctly expresses authorization permissions. Here we will cover the extensibility elements, their descriptions, and give examples. Our first extension element is *permission*. *permission* is the operation an authenticated client is authorized to perform on a certain object, which is a resource.

The extension element *role* is a function within the context of an organization; some associated semantics regarding the authority and responsibility are conferred on the user assigned to the *role*. A role could be general, for example 'Employee', or more specific as in 'Radiologist'. As well, a *role* could be a group which confers authority and responsibility as in 'Hospital Executive'.

Since it is not feasible to name each user and the semantics of a subject can vary greatly we use an extension element *subjectCategory* to describe the type of a subject. Subjects can be users, which the National Institute of Standards and Technology define as a human, machine, network, or intelligent autonomous agent. In our context, this can also include an entire organization. Using *subjectCategory* as an extension element enables us to describe relationships. For example, the *subjectCategory* partner describes that the subject is in some kind of partnership agreement with the provider of the Web service.

Lastly, *modelReference* is used to handle the mapping of a schema element to an ontological concept. For example, this can be applicable when a Web service provider wants to demonstrate that authorization will be constrained to certain inputs for an operation. This might be done using an ontological concept like patient_identification_number.

An approach such as this provides the granularity needed for Web services. This is because a WS-Policy file may be attached to a message, a service binding, an operation, or a parameter such as an input. We assume that annotations are used to describe an explicit 'grant'; while we assume lack of the criteria or conditions is an implicit 'deny'.

The WSP-S is an annotated WSP. As seen in Figure 13, annotations can occur after the <All> tag in WSP. If there is one annotation for the entire WSP then it could be placed after the first <ExactlyOne> tag. The first annotation in Figure 13 describes authorization for a requestor whose

role is “Emergency Room Physician”. The second annotation is a *subjectCategory*, namely “Health Services”. From the namespace it is seen that the concepts are from the HL7 RBAC ontology.

This approach allows for multiple annotations within the policy file. This enables a provider to express multiple conditions regarding authorization. For instance, an ‘Emergency Room Physician’ who is also affiliated with an organization that is categorized as ‘Health Services’ may have authorization to a providers’ resource, while all other Physicians do not. This is accomplished by placing both annotations within the <ALL> tag. This can be seen in the example below.

```
<wsp:Policy
xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:base="http:/http://www.NationalEHR.com/policies"
xmlns:wsrm="http://schema.xmlsoap.org/ws/2004/03/rm"
xmlns:wsau="http://lsdis.cs.uga.edu/authorization/wsau"
xmlns:Ontology1="http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/ontologies/HL7_RBAC.owl">
 <wsp:ExactlyOne>
  <wsp:All>
   <wsau:role name="Emergency_Room_Physician" wsau:ModelReference="Ontology1# Emergency_Room_Physician "/>
   <wsau:subjectCategory name="Health_Services" wsau:ModelReference="Ontology1#Health_Services"/>
   <wsse:SecurityToken>
            <wsse:TokenType>
                     wsse:X509v3
            </wsse:TokenType>
   </wsse:SecurityToken>
   </wsp:All>
  <wsp:All>
   <wsau:permission name="read" wsau:ModelReference="Ontology1#read"/>
   <wsau:role name="Executive_Administration" wsau:ModelReference="Ontology1#Executive_Administration"/>
   <wsau:subjectCategory name="Health_Services" wsau:ModelReference="Ontology1#Health_Services"/>
  <wsse:SecurityToken>
            <wsse:TokenType>
                     wsse:X509v3
            </wsse:TokenType>
  </wsse:SecurityToken>
  </wsp:All>
</wsp:Policy>
```

Figure 15 Annotated WS-Policy file

There is also the situation in which a requester can have authorization to access a resource if it meets one condition or one set of conditions described by the provider. In this case the annotations are placed within the <ExactlyOne> tags. Figure 13 shows two sets of conditions with in the <ExactlyOne> tags. The authorization information from this figure can be read as authorization may be granted to someone that is an Emergency

Room Physician that is affiliated with an organization in Health Services or an Executive Administrator who has read privileges and is affiliated with an organization in Health Service.

Any domain specific ontology can be used for the annotations. However, a quality of RBAC is that it has a structural hierarchy with relationships which lends itself to the creation of an ontology schema. The concepts of RBAC include organizational and professional roles. This fits well with the extension elements derived from the XACML representation of RBAC.

The annotations begin with the namespace "wsau", as depicted in the previous figure, which is declared in XML declarations as follows: xmlns:wsau="http://lsdis.cs.uga.edu/authorization/wsau".

4.4 Constraint Comparison

We assume here that a Semantic Web services framework has been implemented. This may be a stretch of the imagination for some since there are only a handful of these around, and mostly in academia. Never the less, let's assume that we have discovered a set of services using one of these implementations.

Once Semantic Discovery has returned a set of candidate services, the requestor can perform constraint analysis to determine which of the candidate services it most likely has authorization to invoke. This predication uses information given about the client, WSP-S, and ontologies to make the 'best choice'. One approach is to have authorization information for the requester contained in client WSDL's attached policy file.

During the constraint analysis process, if an authorization annotation is found then that information it should be passed to a 'manager' or 'engine' which can perform Semantic Comparison Analysis. Information contained within the annotation, regarding the service provider, and information within the client policy, regarding the client, is used for ontology based inferencing to predict if the client has authorization to use the resource.

Ontology inference engines, also called reasoners, are software applications that derive new facts or associations from existing information. Inference and inference rules allow for deriving new data from data that is already known. Thus, new pieces of knowledge can be added based on previous ones. By creating a model of the information and relationships, we enable reasoners to draw logical conclusions based on the model. For example, with OWL it is possible to make inferences based on the associations represented in the models, which primarily means inferring transitive relationships. Jena has a built in rule-base reasoner that provides OWL inferencing support. The RBAC standard requires the ontology have

the ability to apply multiple restrictions and cardinalities to concepts in the hierarchy. The reasoner that is built into Jena is able to reason over these more complex ontologies.

If an authorization path is detected via dynamically generated queries then a relationship(s) exists between the concept(s) in the service policy file and the concept(s) in the client policy file. The RBAC standard provides a defined structure such a path exists only if the concepts are related in such a way that authorization should be granted. Therefore, we would predict that authorization will be granted. If a path is not detected then we move to a second phase.

In order to determine if two uniquely named concepts from different ontologies are equivalent, the relationships of those concepts to other concepts in their respective ontologies must be comapred. Even in a highly standardized domain such as Health Care, two concepts may have different names. For instance, an 'Emergency Room Physician" from one ontology may correspond to an "Emergency Physician' from another ontology, or "ER Doctor" or ER Physician".

One popular approach is to examine the relationships of the client and service concepts and those related concepts that are linked by these relationships. Because of the structure of the RBAC ontology, it is manually possible to quickly determine which relationships are most important. In most cases it will be necessary to place weights on the relationships in order to improve the accuracy of the results. This approach is based on [Dong et al., 2005] and [Aleman-Menza et al., 2006]. However, applying weights requires a human to review the relationships within the ontology. An automated approach to the problem of weighting the relationships is as follows.

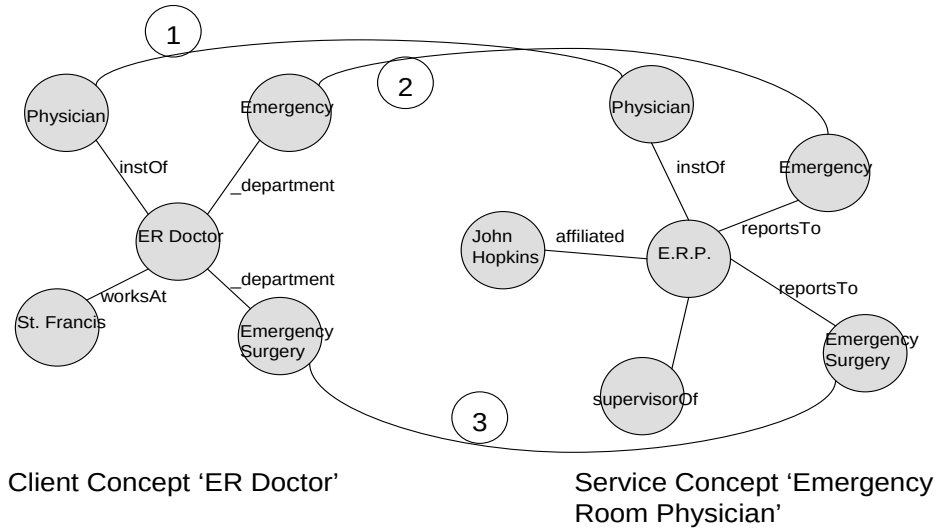

Figure 16 Concept Comparison

At each iteration of an algorithm, compare a concept related to the original service concept with one related to the original client concept. For example, in the first iteration compare the Physician concept related to ER Doctor from the client ontology to a concept related to Emergency Room Physician(ERP) in the service ontology. This continues and the concept Physician related to ERP is found, Figure 14 (1). As shown in the figure, after the algorithm terminates, we have found three concepts related to ER Doctor that are similar, if not equivalent to, the concepts related to ERP. This is a simplistic example because the names of the relating concepts are the same. However, if names of the related concepts are not the same this approach can be expanded upon.

Consider expanding the approach by comparing the names of the concepts as well as the relationship type through an advanced string comparison algorithm. We suggest the n-grams [Damashek, 1995] algorithm to compare the labels of the related concepts and the relationships, i.e. Physician and instanceOf respectively. By using a string comparison that returns a decimal between 0 and 1, one could create a weighted score and a threshold for predicting authorization. This approach, although using syntactic comparison, uses semantics from the schema to exploit relationships to other concepts, concepts names, and relationship names; as well the number of relationships.

Another approach to consider is to create SWRL (Semantic Web Rule Language) rules that are specific enough to capture relationship information regarding a resource but general enough to be applicable to an entire class. For example, we could have created a rule that a technician, like 'Radiology

Tech, belonging to the same department, 'Radiology.', as another technician, 'Radiology Technician, and are therefore equivalent concepts. However, this approach may result in an unacceptable amount of false positives and in a domain less structured than the HL7 domain, it would require many rules and a priori knowledge of the ontology.

5. OTHER APPROACHES TO MENTION

Authorization in Web services is currently a research area of great interest. WS-Authorization is one of a few Web service specifications remaining to be standardized [IBM, Microsoft 2002]. In the Semantic realm, including both Semantic Web and Semantic Web services, research interests in Semantic Authorization are rapidly growing.

Some previous research regarding authorization in Semantic Web services has been focused on implementing an access control enforcement structure [Yague et al., 2003]. [López et al., 2005] discusses an approach to access control in a distributed heterogeneous network in which XACML and SAML are used for access control. The novelty of their approach is to convert between these standards for the enforcement of policies.

[Agarwal et al., 2004] uses attributes from credentials like SAML or Digital Certificates to make access control decisions in their implementation. While this can be done, these credentials were designed for authentication. A similar approach to the one discussed in this chapter is [Kagal et al. 2004], in which the authors use ontologies to add authorization annotations to OWL-S. OWL-S is used to add semantics to Web services. This is done through a mapping of concepts in OWL-S to WSDL types. Their approach adds extensibility elements to the OWL-S constructs for the purpose of supplying authorization information.

An idea that has gained momentum recently is that of a hybrid approach. This approach incorporates real world concepts in ontology with rule based ontology and is described in [Kagal et al. 2004]. The strength of their approach is in describing access control policies with multiple ontologies. This provides for greater expressiveness since the semantics of rules may be incorporated.

When developing a system to perform a function that is currently in research, it is important not to reinvent the wheel. For example, extending the accepted standard WSP by adding semantic annotations. Instead of creating an entirely new standard it is better to build on an accepted standard. In addition, using the RBAC standard for an annotation scheme and for our ontologies. Other approaches have developed their own authorization ontologies, however RBAC is an accepted standard.

Policy matching in Semantic Web services is a complicated area of research. [Verma et al., 2005] details an implementation of Semantic Policy matching using the Semantic Web Rule Language (SWRL). [Wu et al., 2002] describes how to incorporate access control in a business process (workflow). It illustrates fundamental capabilities in a workflow and why authorization and access control need to be expressed semantically. This is particularly relevant to our research on the client side where we had to decide on the appropriate location for client information. Anyanwu et al., 2003] examines the issues of complex processes inherent in health care applications in a heterogeneous cross domain environment.

6. QUESTIONS FOR DISCUSSION

Beginner:

1. Why are traditional Web services authorization techniques not adequate for Semantic discovery? *(Traditional authorization techniques do not suggest what types of users may have access to the resource provided through the Web service.*
2. Why is the concept of Roles important in Semantic Authorization? *(They are the concepts built upon an existing standard which describe users. This information along with domain knowledge provides the basis for Semantic reasoning, or inferencing.)*

Intermediate:

1. How could independent enterprises exchange their authorization ontologies with those enterprises that have discovered their Web services? *(through accounts, place them freely on the internet for download)*
2. What security risks are involved with placing the ontologies on the internet? *(Exposing too much information regarding user accounts).*
3. Can you think of a solution that would alleviate the security risk mentioned in the previous question? (This answer may differ depending on your previous answer. However, for exposing too much information, a generic ontology for an entire domain would allow enterprises to share this information across any domain. The ontology implementation details in regards to a specific enterprise would not be shared.)

Advanced:

1. Discuss some of the relationships between RBAC concepts and a real world enterprise. How could these relationships be exploited in Securing Semantic Web services? (*Discussion question, meant to be thought provoking. No right or wrong answer.)*

2. Can you think of any ways that semantics could benefit the security technologies currently in use; i.e. authentication, encryption, etc.? What kind of ontology would you design to do this? (*Discussion question, meant to be thought provoking. No right or wrong answer.)*

7. SUGGESTED ADDITIONAL READING

- Rosenberg, J and Remy, D. *Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption*. Sams (May 12, 2004). 408 pp. This book explains the basics of securing Web services through traditional and current technologies.
- Alesso H. P. and Smith C. F. *Developing Semantic Web Services.* AK Peters; Bk&CD-Rom edition (October 2004). 445pp. This book provides further reading on creating semantic Web services and discusses there limitations.
- Ferraiolo D. F., Kuhn D. R., and Chandramouli R. *Role-Based Access Control*. Artech House Publishers (April 2003). 338pp. This book is an authoritative look at Role Base Access Control and discusses many of the complexities associated with a distributed implementation.

8. REFERENCES

Akkiraju R, Farell J, Miller J, Nagarajan M, Sheth A and Verma K, "Web Service Semantics - WSDL-S" Proceedings of the W3C Workshop on Frameworks for Semantics in Web Service (W3CW'05), Innsbruck, Austria (June 2005) pages 5.

Boanerges Aleman-Meza, Meenakshi Nagarajan1, Cartic Ramakrishnan1, Li Ding, Pranam Kolari, Amit P. Sheth1, I. Budak Arpinar, Anupam Joshi, Tim Finin, International World Wide Web Conference, Proceedings of the 15th international conference on World Wide Web, Edinburgh, Scotland, SESSION: Social networks, 2006, pp 407 - 416

Kemafor Anyanwu, Amit P. Sheth, Jorge Cardoso, John A. Miller and Krys J. Kochut, "Healthcare Enterprise Process Development and Integration," Journal of Research and Practice in Information Technology *(JRPIT),* Special Issue on Health Knowledge Management, Vol. 35, No. 2 (May 2003) pp. 83-98. Australian Computer Society, Inc.

Kemafor Anyanwu and Amit P. Sheth, "The ρ Operator: Discovering and Ranking Associations on the Semantic Web" ACM SPECIAL ISSUE:

Special section on semantic web and data management, Volume 31 , Issue 4 2002 pp 42 - 47

S. Agarwal, B. Sprick, S. Wortmann; "Credential Based Access Control for Semantic Web Services"; http://www.aifb.uni-karlsruhe.de/WBS/sag/papers/Agarwal_Sprick_Wortmann-CredentialBasedAccessControlForSemanticWebServices-AAAI_SS_SWS-04.pdf.

Census Bureau, 2000 Industry Categories for the Special EEO File, 2000.

Christensen E., Curbera F., Meredith G. and Weerawarana S., 2001, Web Services Description Language (WSDL) 1.1, W3C Note, http://www.w3.org/TR/wsdl.

M. Damashek. Gauging similarity with n-grams: language independent categorization of text, Science, 267(5199) pp 843--848, 1995

Dogac A., Cingil I., Laleci G., Kabak Y., Improving the Functionality of UDDI Registries through Web Service Semantics, 3rd VLDB Workshop on Technologies for Eservices (TES-02), Hong Kong, China, August 23-24, 2002

X.L. Dong, A. Halevy and J. Madhavan (2005) Reference reconciliation in complex information space, In Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, ACM Press: Baltimore, MD. Pp. 85-96

FaCT (2005) FaCT++, http://owl.man.ac.uk/factplusplus/.

Fensel D. and Bussler C., The Web Service Modeling Framework WSMF, http://informatik.uibk.ac.at/users/c70385/wese/wsmf.paper.pdf

S. Gavrila, D. Kuhn, R. Chandramouli; Proposed NIST Standard for Role-Based Access Control; http://csrc.nist.gov/rbac/rbacSTD-ACM.pdf

Gandon, F. L. and N. M. Sadeh, OWL inference engine using XSLT and JESS, http://www-2.cs.cmu.edu/~sadeh/MyCampusMirror/OWLEngine.html, 2003.

HL7 http://www.hl7.org

HL7 Security Technical Committee, Role Based Access Control (RBAC) Healthcare Scenarios Version 1.0, 2005.

HL7 Security Technical Committee, Role Based Access Control (RBAC) Healthcare Permissions Catalog Version 2.0, 2005

Hull and East Yorkshire Hospitals NHS Trust, 2006.

IBM Corporation and Microsoft Corporation, Security in a Web Services World: A Proposed Architecture and Roadmap Version 1.0, 2002.

Hewlett-Packard Development Company, LP., 2006. http://jena.sourceforge.net/

L. Kagal, M. Paolucci, N. Srinivasan, G. Denker, T. Finin, K. Sycara; Authorization and Privacy for Semantic Web Services; IEEE Intelligent Systems (Special Issue on Semantic Web Services), July 2004.

Leymann F, Roller D, Schmidt MT, Web services and business process management, IBM Systems Journal, 2002

G. López, Ó. Cánovas, A. Gómez-Skarmeta, S. Otenko, D. Chadwick; A Heterogeneous Network Access Service based on PERMIS and SAML; In Proceedings of 2nd EuroPKI Workshop, University of Kent, July 2005.

National Institute of Standards and Technology (NIST) FIPS Publication 180: Secure Hash Standard (SHS). May 1993.

National Institute of Standards and Technology (NIST) Role Based Access Control Standard (RBACS). April 2004.

Deborah L. McGuinness, Frank van Harmelen, W3C Recommendation 10 February 2004

Minswap, http://www.mindswap.org/2003/pellet/ , 2003

Cary Pennington, "Policy Based Optimal Composition of Web Services," Masters Thesis (M.S. in CS Degree) July 2006.

Jothy Rosenberg and David Remy, Securing Web Services Security with WS-Security, Sams, 2004.

SAML 2.0 profile of XACML v2.0 OASIS Standard, 1 February 2005, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf

Joel Farrell, IBM Semantic Annotations for WSDL, Editors Copy

Holger Lausen, DERI Innsbruck, August 08, 2006. http://www.w3.org/2002/ws/sawsdl/spec/SAWSDL.html

Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner. Submitted for publication to Journal of Web Semantics.

Sivashanmugam, K., Verma, K., Sheth, A., Miller, J., Adding Semantics to Web Services Standards, Proceedings of the 1st International Conference on Web Services (ICWS'03), Las Vegas, Nevada (June 2003).

A. Toninelli, J. Bradshaw, L. Kagal, R. Montanari; Rule-based and Ontology-based Policies: Toward a Hybrid Approach to Control Agents in Pervasive Environments; Proceedings of the Semantic Web and Policy Workshop, International Semantic Web Conference, 7 November, 2005.

UDDI Spec Technical Committee Specification, 2002. http://uddi.org/pubs/uddiv3.00- published-20020719.htm

Verma K, Sivashanmugam K, Sheth A, Abhijit Patil, Oundhakar S and Miller J, METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services, Journal of Information Technology and Management, Special Issue on Universal Global Integration, Vol. 6, No. 1 (2005) pp. 17-39. Kluwer Academic Publishers.

Kunal Verma, "Configuration and Adaptation of Semantic Web Processes" Doctoral Dissertation (Ph.D. in CS Degree) June 2006

K. Verma, R. Akkiraju, R Goodwin; Semantic Matching of Web Service Policies; Second International Workshop on Semantic and Dynamic Web Processes (SDWP 2005), Third International Conference on Web Services (ICWS'05), July, 2005.

Verma K, Aggarwal R, Miller J and Milnor W, "Constraint Driven Web Service Composition in METEOR-S" Proceedings of the 2004 IEEE International Conference on Services Computing (SCC'04), Shanghai, China (September 2004) pp. 23-32

Wielemaker, J., SWI-Prolog Semantic Web Library, http://www.swi-prolog.org/packages/semweb.html, 2005.

D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard, Web Services Architecture (WS Architecture), http://www.w3.org/TR/ws-arch/#security Feb. 2004

Siddharth Bajaj, Don Box; Web Services Policy Framework (WS-Policy), ftp://www6.software.ibm.com/software/developer/library/ws-policy.pdf

Web Services Security (WS-Security) Version 1.0 05, 2002 *et al* Bob Atkinson, Giovanni Della-Libera; Specification: Web Services Security, ftp://www6.software.ibm.com/software/developer/library/ws-secure.pdf; April 2002

S. Wu, A. Sheth, J. Miller, Z Luo; Authorization and Access Control of Application Data in Workflow Systems; Journal of Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies (JIIS), Vol. 18, No. 1 (January 2002) pp. 71-94. Kluwer Academic Publishers.

eXtensible Access Control Markup Language, (XACML) Version 2.0 OASIS Standard, 1 Feb 2005, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf

XML Signature Syntax and Processing (XML-Signature) W3C Recommendation 2002 http://www.w3.org/TR/xmldsig-core/

XML Encryption Syntax and Processing (XML-Encryption) W3C Recommendation 2002 http://www.w3.org/TR/xmlenc-core/

M. Yague, A. Mana, J. Lopez, J. Troya; Applying the Semantic Web Layers to Access Control; 14th International Workshop on Database and Expert Systems Applications (DEXA'03)