AFRL-RI-RS-TR-2011-182

DELEGATION MANAGEMENT

RAYTHEON

JULY 2011

FINALTECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■**UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

AFRL-RI-RS-TR-2011-182 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/

JAMES MILLIGAN
Work Unit Manager

/s/

JULIE BRICHACEK, Chief
Information Systems Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*
JUL 2011

2. REPORT TYPE
Final Technical Report

3. DATES COVERED *(From - To)*
FEB 2010 – FEB 2011

4. TITLE AND SUBTITLE
DELEGATION MANAGEMENT

5a. CONTRACT NUMBER
FA8750-10-C-0034

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62788F

6. AUTHOR(S)
Jim Jacobs

5d. PROJECT NUMBER
S2DM

5e. TASK NUMBER
05

5f. WORK UNIT NUMBER
06

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Raytheon Company
1010 Production Road
Fort Wayne IN 46808

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/RISD
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2011-182

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. PA# **88ABW-2011-2934**
Date Cleared: **25 May 2011**

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The DoD defines delegation of authority as the action by which a commander assigns part of his authority commensurate with the assigned task to a subordinate commander. While ultimate responsibility cannot be relinquished, delegation of authority carries with it the imposition of a measure of responsibility, the extent of which must be clearly stated. Similarly, access to enterprise services and information can be controlled through delegation of credentials by an authority as established by formal semantics and explicit policies. In this report we describe a prototype system for policy-based access control of web services. Policies, which are written in the Web Ontology Language (OWL), govern both web service access and delegation of authority, and are enforced by IHMC's KAoS policy services framework and management system. Each delegation of authority policy permits or denies access to a web service based on the credentials of the principal requesting access. A powerful feature of our approach is that it can be applied to existing web services with little or no modification of service implementation. It also allows the schema used for web service design to evolve independently of the policy and domain ontologies.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON / 19b. TELEPHONE NUMBER (*Include area code*)
U	U	U	UU	22	JAMES MILLIGAN / N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

In this paper we describe an architecture and demonstration system for policy-based access control of Web services. Our architectural framework is derived from ISO Standard 10181–3 (ITU-T, 1995), which defines an architectural model for controlling access to networked resources. Web services and access policies are drawn from activities and procedures associated with an Air Operations Center (AOC) and a small set of operational scenarios. These scenarios incorporate realistic patterns of service invocations while exercising essential capabilities of access control and delegation of authority within a federated environment. Policies govern both Web service access and delegation of authority. Policies, which are written in the Web Ontology Language (OWL) [OWL 2004], are defined and enforced by the KAoS policy services framework. Each policy permits or denies access to a Web service based on credentials. Some credentials accompany the request, while others are looked up based on the requestor's identity.

Central to our governance approach is a Delegation Management Web service. This web service exposes operations for assigning and revoking roles. Such roles infer subsets of credentials associated with a specific delegation of authority. Underlying these policies and their supporting web services, we have constructed a formal model of delegation-of-authority as practiced in an AOC. This model, which is also written in OWL, was integrated with the core KAoS policy ontologies to create semantically rich policies that enable fined-grained control of both Web service access and delegation of authority.

Within the DoD, delegation of authority is the act by which a commander transfers part of his authority to a subordinate commander in order to complete an assigned task or carry out additional duties. Delegation of authority is often limited to specific tasks or for specific time periods and is commonly governed by policies that specify what may be delegated, to whom it may be delegated, and under what circumstances delegation may occur. Furthermore, policies may also dictate whether or not a person may perform tasks for which he has been given the authority to delegate. For example, suppose a flight operations manager has been delegated the authority to assign pilots to flights. A delegation policy should prevent the manager from assigning himself to a flight unless he is also a pilot.

Any recipient who is asked to perform a service should be able to verify that the requestor has the authority to make such a request. If the requestor has not been properly authorized, the request should be denied. Authorization is commonly based on presenting the recipient with a set of credentials. Using this information the recipient can decide if the request should be accepted or denied. Within the context of delegation, the requestor may be a delegate, and the recipient would also enforce the delegation policy of its organization when considering service requests.

Increasingly, delegation of authority takes place within a computing context. Managers may need to delegate some privileges to subordinates to enable them to carry out computer-based tasks. In an enterprise system, Web services themselves may need the ability to delegate the ability to invoke operations to other services. Service providers need to be able to verify that each service requestor is properly authorized. If the service requestor has received dynamically-delegated authority, service providers need to be able to verify that this was done in accordance with their delegation policy. In addition, whenever delegation of authority is attempted, there must be a mechanism to ensure that such delegation is permitted.

In designing our access control mechanism, we addressed the requirements specified by Chadwick [Periorellis 2008] for a general purpose delegation of authority service (DoAS). We summarize these below. Since we are already assuming that the DoAS is operating within a service-oriented architecture, we have omitted the last one.

- The DoAS should be able to support delegation from person to person, person to task, task to task, and service to service.
- Every principal should authenticate with its own independent identity, enabling delegation to be performed from one named entity to another.
- To support a scalable authorization infrastructure, access controls should support attribute- or role-based, where each principal is assigned an attribute set, and each set of attributes may be used to grant selected access rights to a given resource or set of resources, e.g. Web service operations.
- Principals should to be able to delegate any of their attributes to other principals. Such delegation enables the delegee to perform additional tasks that are authorized through its association with the delegated attributes.
- The DoAS should embody a delegation policy along with an enforcement mechanism that will control both the delegation process itself and the authorization process for the requested Web service.
- The DoAS should support fine-grained delegation, i.e. the ability to delegate authority to access a particular operation of a Web service or perform a particular operation on a data resource.
- Users should be able to authenticate and prove their identity without having to possess a public key certificate.
- The DoAS should support immediate revocation of delegated attributes, cutting short the originally intended duration of effectivity. Furthermore, acts of delegation themselves should take effect instantaneously.

In the Section 3.0, we present our architecture and discuss show how it satisfies these requirements.

2.0 SUMMARY

We have built a demonstration system, based on scenarios from an air operations center, which utilizes KAoS to govern delegation of authority in the context of web service access control. We discussed the architecture of our demonstration system, described the mechanisms for authorization of delegation actions and web service requests, and showed how KAoS integrates with existing standards for web service modeling, implementation and security. A powerful feature of our approach is that it can be applied to existing web services with little or no modification of service implementation. It also allows the schema used for web service design to evolve independently of the policy and domain ontologies. Future work will focus on developing tools for automatically generating the necessary transformation files, more fully supporting composite and orchestrated web services, and extending the delegation-of-authority micro-theory to incorporate more concepts and relationships from the AOC domain.

3.0 METHODS, ASSUMPTIONS AND PROCEDURES

3.1 Architectural Framework.

Our approach integrates technologies for semantic modeling, Web service access control, and policy management within an enterprise environment. Software components are written in Java EE. Access control and delegation management services are implemented as Web services that conform to OASIS and W3C standards including SOAP, WSDL and XML. For authentication, these services leverage existing Web service security infrastructure that includes a variety of WS-* standards and specifications. Semantic models and policies use the OWL and Resource Description Framework (RDF) standards. ISO Standard 10181–3 (ITU-T, 1995) defines an architectural model for controlling access to networked resources (see Figure 1). In the ISO model, access control is implemented by two components, the Policy Enforcement Point (PEP) and the Policy Decision Point (PDP). The PEP intercepts incoming requests and asks the PDP if the requestor has the authority to perform the requested action on the protected resource. The PDP maintains a set of policies that define necessary credentials for each type of access for each protected resource. Based on the applicable policy and supplied credentials, the PDP determines if the requester is granted access to the resource. It returns its response to the PEP, which then either grants or denies the original request. In this model, the credentials may be provided with the access request, or the PDP can retrieve them from a credential repository using the requester's identity.

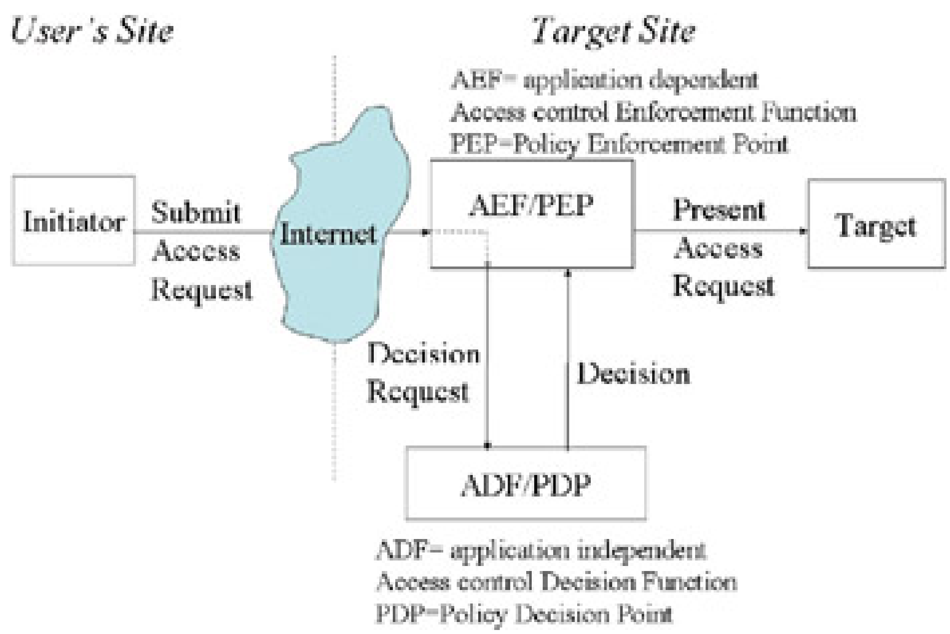

Figure 1. ISO Standard 10181–3 Architectural Model for Network Resource Access Control.

Our architecture is consistent with the ISO Standard 10181–3 model. Figure 2 details components relevant to both Web service access control and delegation management.

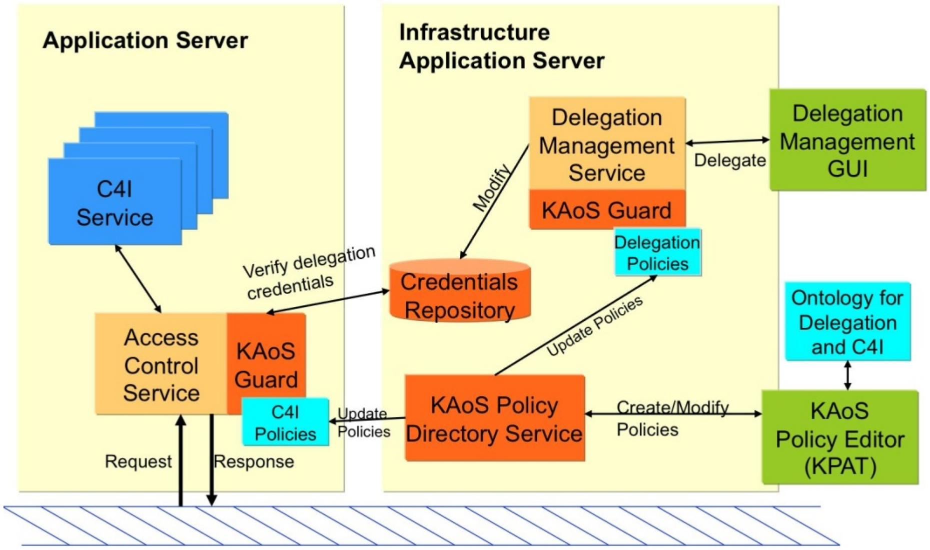

Figure 2. Architecture for Policy-Based Access Control and Delegation Management.

3.1.1 Runtime Management of Delegation and Access Control Policies.

Functions of the PEP and PDP are distributed among the Access Control Service (ACS), KAoS Guard and KAoS Directory Service (KDS). The ACS intercepts each Web service request. It extracts salient information from the request including the requestor's identity, Web service operation, and any pertinent contextual information. (Our architecture does not include an authentication component, but assumes authentication information – at a minimum, the requestor's identify – is transmitted with each service request.) The requester's identity is used to query the Credentials Repository. The ACS then invokes the KAoS Guard with the supplied credentials to perform an authorization check. The Guard contains a set of policies that control access to the hosted Web services. These policies are maintained by the KDS. The KDS ensures that the Guard is configured with the latest policy set as policies may be updated at any time. The Guard applies the relevant policy against the supplied credentials. The request is either authorized or denied. Authorized requests are forwarded to the appropriate Web service. Within our demonstration system, these web services are used to perform notional AOC Command and Control (C2) capabilities.

3.1.2 KAoS Policy Framework.

KAoS is the foundation of our solution for policy-based access control. The KAoS framework is a policy management system that has sufficient generality and expressive power to span the breadth of requirements for enterprise applications [Uszok 2004, 2008]. A singular advantage of KAoS' OWL-based policies is that they can either be used directly or, because of their rich semantics, as abstract models that can be converted to special-purpose policy language representations as necessary. KAoS has been integrated with a variety of agent, robotic, Web services, Grid computing (e.g., Globus), and traditional distributed computing platforms, and across a variety of industrial, military, and space applications. Particularly relevant to the SOA domain, KAoS has been successfully integrated with service-oriented technologies such as JBoss and Spring, allowing for policy-based control of the interaction among web services.

KAoS also provides basic services for distributed computing, including message transport and directory services. Because the services are accessed through a well-defined Common Services

Interface (CSI), application developers can selectively use subsets of its capabilities (e.g., registration, transport, publish-subscribe, domain management, remote request forwarding, queries) as appropriate.

The basic elements of the KAoS architecture are shown in Figure 3. Its three layers of functionality correspond to three different policy representations. The Human Interface Layer provides administrative tools to construct, edit and distribute KAoS policies. The Policy Management Layer encodes OWL policies and manages policy-related information for further analysis. The Distributed Directory Service (DDS) encapsulates a set of OWL reasoning mechanisms based on two open source components: Jena [McBride 2001] and Pellet [Sirin]. The Policy Monitoring and Enforcement Layer establishes and maintains KAoS enforcement components known as Guards. Guards embody "compiled" OWL policies, a representation that affords extremely efficient run-time monitoring and enforcement at "table look up" speeds. Because, apart from policy updates, Guards operate independently from the rest of KAoS, they can be used as small-footprint standalone policy enforcement platforms in disconnected operations. This representation also provides the grounding for abstract ontology terms, connecting them to instances in the runtime environment and to other policy-related information.

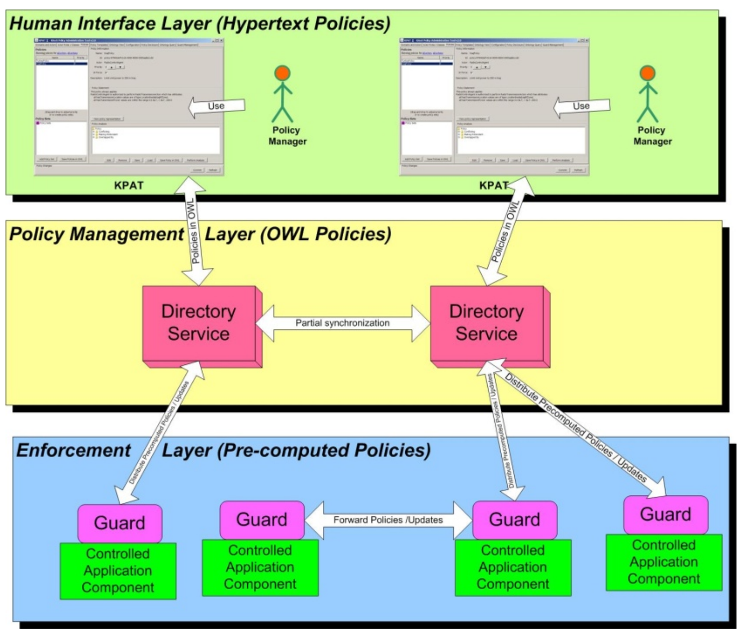

Figure 3: KAoS Policy Service Conceptual Architecture

Within each of the layers, the end user may plug in specialized extension components if needed. Such components are typically developed as Java classes and described using ontology concepts in the configuration file. They can then be used by KAoS in policy specification, reasoning and enforcement.

Policy negotiation provides the mechanism for policy reconciliation and deconfliction between different nodes/users/applications/groups. Conflicts and ambiguities may emerge for a number of reasons such actual differences in the administrative requirements of each domain, or the possibility that different regions of a segmented network may independently learn conflicting policies, which have to be reconciled (and negotiated) at a later time when connectivity is re-established.

3.1.3 Specification of Access Control and Delegation Management Policies.

The KAoS Policy Administration Tool (KPAT) graphical user interface allows end users to manually specify, analyze, and modify authorization and obligation policies at runtime. KPAT hides the complexity of the OWL representation from users. The reasoning and representation capabilities of OWL are used to full advantage to make the process as simple as possible. Whenever users are required to provide an input, they are presented with a complete set of context-driven values from which to select.

KPAT's generic Policy Editor presents an administrator with a starting point for policy construction – essentially, a very generic policy statement shown as hypertext. Clicking on a specific link that represents a variable provides the user with choices allowing him to make a more specific policy statement. During use, KPAT accesses the loaded ontologies and provides the user with the list of choices, narrowed to the current context of the policy construction. New classes and instances can also be created from KPAT. To further simplify policy construction, KPAT provides two additional policy creation interfaces: A Policy Wizard to guide users step-by-step, and a Policy Template Editor that allows custom policy editors for a given kind of policy to be created by point-and-click methods. For the purposes of defining access control and delegation management policies for this project, we have developed a specialized template editor containing just the functionality required for the use case scenarios, allowing delegation policies to be easily defined and analyzed by users without requiring specialized training.

3.1.4 Delegation Management Service.

The Delegation Management Service (DMS) governs the process of delegation of Web service access privileges. The delegator may be a person interacting with the DMS via a user interface or a software agent of some kind (e.g., Web service). Likewise, the role of the delegee can be assumed by either entity. This functionality fulfills DoAS Requirement 1, as it enables delegation of authority from person to person, person to software agent, software agent to person or software agent to software agent.

The DMS will intercept the delegator's request and pass it to the Guard to determine if this Principal is allowed to access the DMS. If the request is granted then the request is forwarded to the DMS. The DMS then determines whether the delegator has sufficient credentials to delegate the specified attributes to the delegee. KAoS policies determine what delegation of authority actions can be taken by specific requestors acting in particular roles or who have been assigned particular responsibilities. The DMS Guard will apply an appropriate delegation policy. This addresses DoAS Requirement 5.

The primary functionality of the DMS is to augment the credentials of the specified delegee on behalf of the delegator, and to publish the updated credentials into the repository. Afterwards, the delegee will be able to use the augmented credentials to gain access to the accompanying delegated services and may be empowered to further delegate these additional attributes if allowed by the delegation policy. Common representations for credentials include the X.509 attribute certificate and signed SAML attribute assertions. Periorellis has argued that the SAML format might be more flexible [Periorellis 2008b]. To address Requirement 7, the credentials are digitally signed by the DMS (or related software that actually creates the new credentials) so that future authorization activities can verify them.

Delegation of authority is seldom permanent. The revocation of authority is a challenging problem. The primary objective of revocation is to remove a credential from a delegee so that it can no longer be used to gain access to associated resources. The effects of revocation should be instantaneous. If this is not feasible, a secondary objective is to inform resource providers that an existing credential has been revoked. The preferred mechanism for the latter objective is to require providers to periodically check with the credential issuer.

Our revocation mechanism follows that proposed by Chadwick [Periorellis 2008a]. His approach overcomes limitations by existing strategies, including short lived credentials [Tuecke et al., 2004][Alfieri et al., 2005][OASIS, 2005]), credential revocation lists [ITU-T 2005], and the Online Certificate Status Protocol (OCSP) [Myers, Ankney, Malpani, Galperin, and Adams, 1999]. In Chadwick's approach, a credential is issued just once and stored in the issuer's repository with its own unique Uniform Resource Locator (URL). The credential is then valid for as long as delegation is required and can be used many times by many different service providers without having to be reissued. Revocation is simply and instantly achieved by simply deleting the credential from the repository. Providers are required to periodically check the presence of the credential using the URL. This period can vary per application or per request as determined by the provider. Our demonstration system checks the credentials on a per request basis and assume they remain valid for the duration of the request. The preferred manner for credential checking could itself be determined by policy. This revocation mechanism satisfies DoAS Requirement 8.

3.1.5 Domain and Policy Ontologies.

Our basic approach to knowledge capture is to use a description logic representation for domain knowledge expressed as OWL ontologies. An ontology is a formal description of concepts, relationships, constraints, and axioms that exist for a specified domain [Gruber 2003]. Unlike basic XML, which embodies semantics implicitly and by convention, an ontology defines a common vocabulary along with the semantics, and is in a machine-interpretable form to enable people and machines to reason about them. It explicitly states assumptions by clearly defining relationships between entities. An ontology has the advantage of separating the domain knowledge from the implementation, such that operational experts are able to define the ontology, with minimal training [Noy and McGuinness 2001]. A variety of graphical tools are now available to make the process even easier.

Rather than construct a single ontology for all of the knowledge in the application, we chose to work from the key scenarios to arrive at a list of important terms and concepts that would form the specific elements of policies. This is supported by an established foundational ontology (Raytheon's Hematite™) and a new 'micro-theory' describing the semantics of delegation. The micro-theory approach to partitioning was pioneered in the Cyc project [Cyc][CycL] and is used to define a particular area of knowledge in a contradiction-free manner. We went a bit further to sharpen and narrow a micro-theory to a particular set of inter-related concepts forming a reusable core within a domain of analysis.

With the foundational ontology and the delegation micro-theory, we were able to construct a domain ontology that provides all of the semantics needed to support inferencing and policy-based reasoning. Figure 4 offers a relation-focused concept map of delegation. Note that it incorporates concepts and relationships from the AOC, policy, and web service domains. The policies themselves are likewise represented in an ontology within KAoS and edited with KPAT.

3.1.6 Authentication.

When a requester desires access to a Web service, the requester must first be authenticated. In our demonstration system, user authentication (DoAS Requirement 2) is performed via a standard login mechanism consisting of a username and password. The architecture itself is agnostic of the authentication mechanism utilized. Most likely, for operation within a federated environment, an authenticated name will be mapped into an authorization name (possibly with accompanying attributes) and stored in that user's credentials. PicketLink Federation [PicketLink] was used for this purpose. PicketLink is a JBoss Community Project, and the Federation subproject provides support for Federated Identity and Single Sign On. PicketLink's Security Token Server (STS) was utilized to generate a simple OASIS SAML v2.0 token containing the requestor's identity. This identity serves as the look-up key for Credentials when applying the authorization policies.

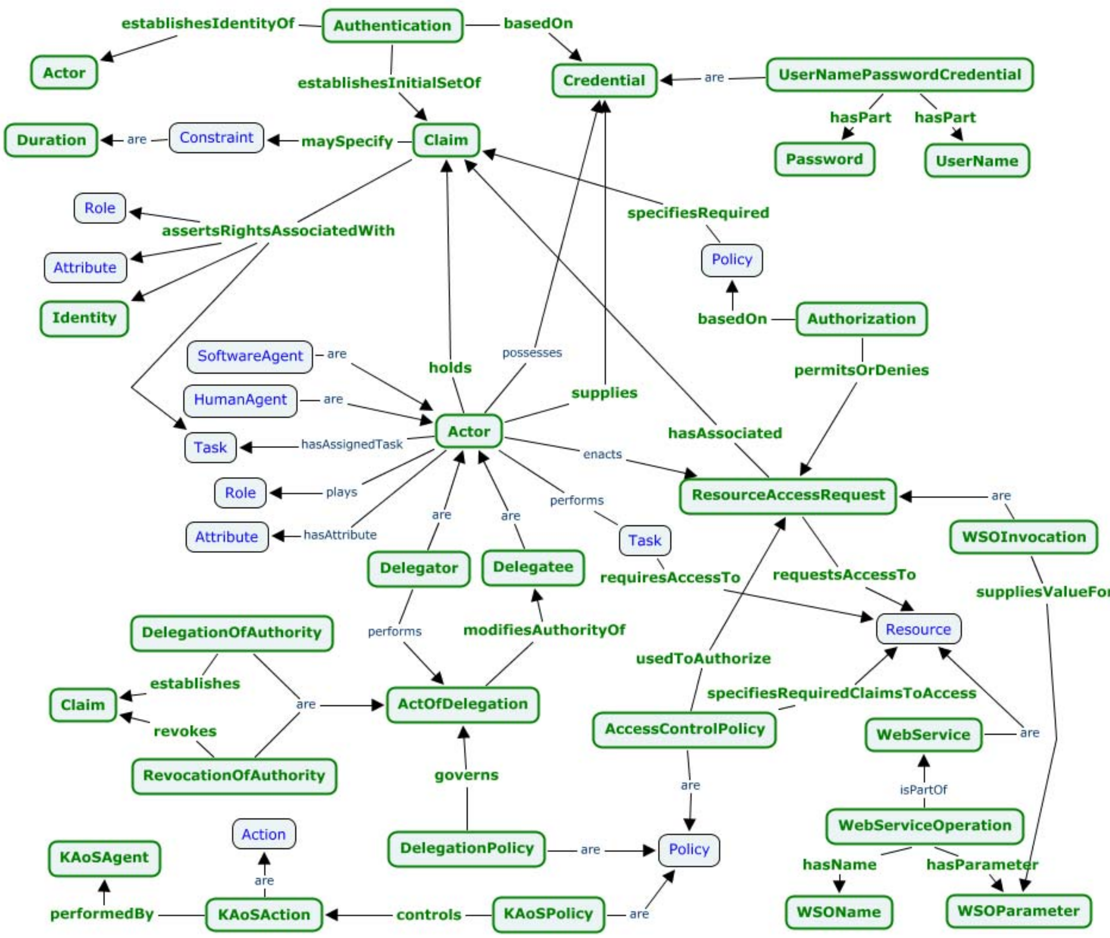

Figure 4. A Micro-theory of Delegation: Relational View

3.2 Operational Scenario and Demonstration System.

Our operational scenario centers on notional activities within an Air Operations Center (AOC) that support target weaponeering. Figure 5 details some of the actions that might be performed by AOC personnel assigned the Targeteer role, while Figure 6 does the same for the Senior Offensive Duty Officer (SODO) role. In this scenario, the Senior Intelligence Duty Officer (SIDO) identifies a new, high-value targeting opportunity (a bridge). This begins a chain of activities that are carried out by personnel acting in the various roles. These activities include posting the target, determining and selecting weapons options, assessing collateral damage, formulating an Air Tasking Order (ATO) change, and posting that change. The SIDO and SODO also are responsible for delegating the roles of Targeteer and Interdiction Officer (INTDO) to personnel whose initial roles do not give them authority to carry out all the required activities.

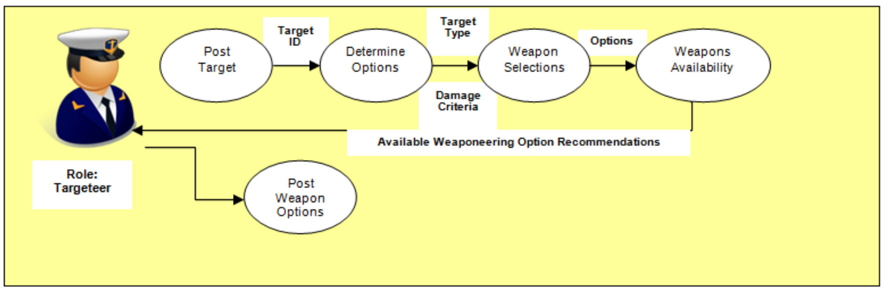

Figure 5. Targeteer Activities.

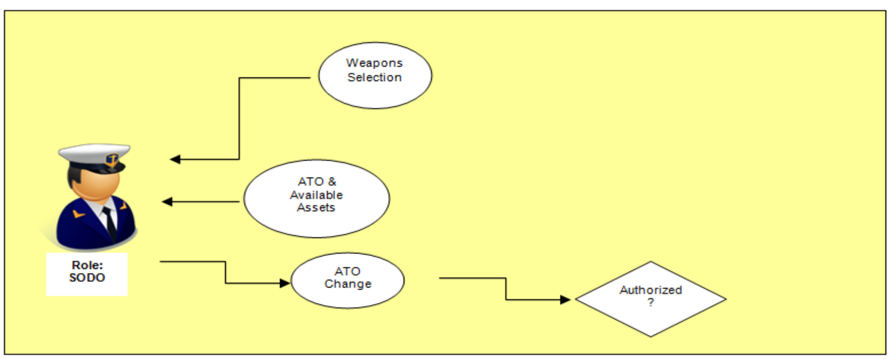

Figure 6. Senior Offensive Duty Officer (SODO) Activities.

To exercise our delegation of authority and web service access control mechanisms, we implemented a demonstration system. The system consists of four Java web services to directly support AOC actions, one Java web service to handle delegation and revocation of authority, and seven KAoS policies. Each web service is configured with the access control service, which is implemented as a Java API for XML Web Services (JAX-WS) handler. A simple web application initiates service requests through a browser interface. The browser interface simulates the application consoles of the various AOC personnel. A screenshot of the Targeteer's weaponeering console is shown in Figure 7.

Figure 7. Targeteer's Console.

The operational scenario described here afforded us a rich set of use cases to exercise our approach. We successfully demonstrated capabilities to control access via policies for both an entire service and individual service operations, to assign and revoke delegations-of-authority, and to handle both user and software agent web service requests.

3.2.1 Technical Details.

To illustrate our technical approach in more detail, we present salient details of the access control and delegation-of-authority mechanisms for the 'Target' web service. The Target Web service is a primitive service, i.e., one which does not invoke operations of another web service. It implements Create, Retrieve, Update, Delete (CRUD) operations on a target object. We suppose that such a service already exists; our objective is to limit access to the create, update and delete operations to personnel serving in the Targeteer role.

To enable access control, the Target service must be associated with the Access Control Service (ACS). The ACS is implemented as a JAX-WS Handler. A simple way to link the web service to the ACS is to use the "@HandlerChain" annotation and specify the ACS as the only handler. The WSDL document is augmented to identify those operations which will be enforced by KAoS policies. The WSDL element corresponding to the create target operation is shown in Figure 8. A "liftingSchemaMapping" attribute of the Security Annotations for WSDL (SAWSDL) schema [SAWSDL] has been added. The purpose of this attribute is to identify an Extensible Stylesheet Language (XSL) file that maps the web service vocabulary to that used by KAoS. This is a powerful mechanism that allows the KAoS policy and domain ontologies to be develop and evolve independently from the web service schema. The associated XSL mapping file is provided in Figure 9. In this case, only a simple translation is needed to map the web service operation requested, CreateTarget, into the KAoS domain concept, CreateTargetAction. In general, the web service operation and its parameters, and possibly parameter values, may require transformation.

```
<xsd:element name="CreateTarget"
      sawsdl:liftingSchemaMapping="CreateTarget2Ont.xsl">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="dm:Target" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure 8. A Portion of the WSDL Definition for the 'Create Target' Operation.

```
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:ns2="http://ont.ray.com/TargetService/"
      xmlns:java="http://xml.apache.org/xalan/java"
    exclude-result-prefixes="java">
<xsl:template match="ns2:CreateTarget">
<rdf:Description rdf:about="REPLACE-WITH-KAOS-URI">
<rdf:type
rdf:resource="http://ontology.ihmc.us/TargetAction.owl#CreateTargetAction"/>
</rdf:Description>
</xsl:template>
</xsl:stylesheet>
```

Figure 9. The XSL Stylesheet that Specifies a Mapping Between the 'Create Target' Web Service Request and KAoS Ontology.

When the Target web service is initialized, the associated instance of the ACS is instantiated. This ACS reads the WSDL and XSL files, then creates a XSL Transformations (XSLT) transformer for the CreateTarget request. It also initializes a KAoS Guard that will be responsible for applying the authorization policies. Whenever a CreateTarget request occurs, the ACS intercepts it. The requestor's identity is extracted, and the XSLT transformer is applied. The resulting data are used to construct a call to the KAoS Guard to determine if the request is authorized. The KAoS Guard applies the relevant policy. In simple terms, this policy states: "Any Targeteer is authorized to perform CreateTargetAction which has any attributes." If the requestor has been assigned the Targeteer role, then the request is allowed and the handler forwards it to the Target web service. If not, an exception is raised and no further request processing occurs.

The DMS is designed in the same manner; however, its operations require more sophisticated interaction with KAoS. First, we note that the delegation operation itself is controlled by policy. The associated XSL file for the delegation-of-authority operation is shown in Figure 10. We note that there are transformation rules for both the operation and parameter names.

```
<xsl:stylesheet version="1.0"
      :
<xsl:template match="ns2:DelegateRole">
<rdf:Description rdf:about="REPLACE-WITH-KAOS-URI">
<rdf:type
rdf:resource="http://ontology.ihmc.us/DelegationAction.owl#DelegationAction"/
>
    <action:hasDelegatedRole rdf:resource="{delegatedRole}"/>
    <action:hasDelegee rdf:resource="{delegateeId}"/>
    <action:hasDelegationContext rdf:resource="{delegationContext}"/>
</rdf:Description>
</xsl:template>
</xsl:stylesheet>
```

Figure 10. XSL File for Mapping a Delegation Web Service Request.

There are several policies that apply to delegation operations. One such policy states: "Any SeniorIntelligenceDutyOfficer is permitted to delegate the Targeteer role to any DutyOfficer." If a delegation operation is permitted, the credentials of the associated delegatee must be modified. This is accomplished through calls to the KAoS API that modify instance data. For example, the invocation, delegateRole("Targeteer", "baker", null), would result in a "hasDelegatedRole" property with the value "Targeteer" to be add to the "baker" instance of an "DutyOfficer". Each role delegation is identified by a unique Uniform Resource Identifier (URI). Revocation operations reference this URI. Since revocation effects changes to the global Credentials repository, revocations are immediate.

4.0 RESULTS AND DICUSSION

The final research development tasks have been completed. The team completed tests and patches on the integrated delegation management system, addressing all outstanding issues with the framework and reasoner. On 28 January 2011, we conducted the final demonstration session at AFRL Rome Research Site and delivered the software to an AFRL-designated machine.

The operational scenario set described here afforded us a rich set of use cases to exercise our approach. We successfully demonstrated capabilities to control access via policies for both an entire service and individual service operations, to assign and revoke delegations-of-authority, and to handle both user and software agent web service requests.

The policy-based concept has long been discussed for many access control systems. In the context of delegation management, the relevant policies are far more dynamic and complex than typical user permission structures. This requires a stronger, more flexible approach, and the ontology-supported policy reasoning technique delivers the needed power. The policy services technique implemented in KAoS provides the reasoning needed, the domain ontology and micro-theory approach provided the knowledge representation structure required and the architecture we developed brought these capabilities together in an integrated solution.

5.0 CONCLUSIONS

We have built a demonstration system, based on scenarios from an air operations center, which utilizes KAoS to govern delegation of authority in the context of web service access control. We discussed the architecture of our demonstration system, described the mechanisms for authorization of delegation actions and web service requests, and showed how KAoS integrates with existing standards for web service modeling, implementation and security. A powerful feature of our approach is that it can be applied to existing web services with little or no modification of service implementation. It also allows the schema used for web service design to evolve independently of the policy and domain ontologies. Future work will focus on developing tools for automatically generating the necessary transformation files, more fully supporting composite and orchestrated web services, and extending the delegation-of-authority micro-theory to incorporate more concepts and relationships from operational military domains.

6.0 REFERENCES

Arp, Robert and Smith, Barry. "Function, Role, and Disposition in Basic Formal Ontology." *Available from Nature Proceedings* <http://hdl.handle.net/10101/npre.2008.1941.1> (2008) Accessed December 27, 2010.

Bradshaw, J. M., Uszok, A., Jeffers, R., Suri, N., Hayes, P., Burstein, M. H., Acquisti, A., Benyo, B., Breedy, M. R., Carvalho, M., Diller, D., Johnson, M., Kulkarni, S., Lott, J., Sierhuis, M., & Van Hoof, R. (2003). "Representation and reasoning for DAML-based policy and domain services in KAoS and Nomads." *Proceedings of the Autonomous Agents and Multi-Agent Systems Conference (AAMAS 2003).* 14-18 July, Melbourne, Australia. New York, NY: ACM Press, pp. 835-842.

Bradshaw, J. M., Jung, H., Kulkarni, S., Johnson, M., Feltovich, P., Allen, J., Bunch, L., Chambers, N., Galescu, L., Jeffers, R., Suri, N., Taysom, W., & Uszok, A. (2005). "Toward trustworthy adjustable autonomy in KAoS". In R. Falcone, S. Barber, J. Sabater, and M. Singh (Eds.), *Trusting Agents for Trustworthy Electronic Societies.* LNAI. Berlin: Springer.

Bradshaw, J. M., Feltovich, P. J., Johnson, M., Bunch, L., Breedy, M., Jung, H., Lott, J. & Uszok, A. (2008). "Coordination in human-agent-robot teamwork." *Proceedings of the 2008 International Symposium on Collaborative Technologies and Systems (CTS 2008), Special Session on Collaborative Robots and Human Robot Interaction*, Irvine, CA, 19-23 May.

Bunch, L., Bradshaw, J. M. & Young, C. O. (2008). "Policy-governed information exchange in a US Army operational scenario." *Demonstration track. 2008 IEEE Conference on Policy*, Palisades, NY, 2-4 June.

Cyc, http://en.wikipedia.org/wiki/Cyc#Knowledge_base Accessed December 27, 2010.

CycL, http://en.wikipedia.org/wiki/CycL#Microtheories Accessed December 27, 2010.

Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., Schneider, L. "Sweetening Ontologies with DOLCE." In A. Gómez-Pérez, V.R. Benjamins (eds.) *Knowledge Engineering and Knowledge Management.* "Ontologies and the Semantic Web," *13th International Conference, EKAW 2002*, Siguenza, Spain, October 1-4, 2002, Springer Verlag, pp. 166-181.

Gruber, T. 2003. "What is an Ontology?" at http://www-ksl.stanford.edu/kst/what-is-an-ontology.html. Accessed December 27, 2010.

McBride, Brian. "Jena: Implementing the RDF Model and Syntax Specification." *Semantic Web Workshop, WWW2001.* 2001.

Myers, M., Ankney, R., Malpani, A., Galperin, S., & Adams, C. (1999). X.509 "Internet public key infrastructure: Online certificate status protocol —OCSP", RFC 2560.

Noy, N. and McGuinness, D. March 2001. ``Ontology Development 101: A Guide to Creating Your First Ontology". Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880 at http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html. Accessed December 27, 2010.

OWL Web Ontology Language. http://www.w3.org/TR/owl-features/. Accessed December 27, 2010.

Periorellis (ed), Panos. "Chapter V - Dynamic Delegation of Authority in Web Services". *Securing Web Services: Practical Usage of Standards and Specifications.* IGI Global. 2008.

Periorellis (ed), Panos. "Chapter VIII - Using SAML and XACML for Web Service Security and Privacy". *Securing Web Services: Practical Usage of Standards and Specifications.* IGI Global. 2008.

Periorellis (ed), Panos. "Chapter VII - Description of Policies Enriched by Semantics for Security Management". *Securing Web Services: Practical Usage of Standards and Specifications.* IGI Global. 2008.

PicketLink http://www.jboss.org/picketlink. Accessed December 27, 2010.

Martin, D., Paolucci, M., and Wagner, M. “Toward Semantic Annotations of Web Services: OWL-S from the SAWSDL Perspective.” *Proceedings of the 4th European Semantic Web Conference (ESWC 2007).* June 2007.

Sirin, E., Parsia B., Cuenca-Grau, B., Kalyanpur, A., and Katz, Y. “Pellet: A Practical OWL-DL Reasoner.” http://www.mindswap.org/papers/PelletJWS.pdf.

Sowa, John F. **Knowledge Representation: Logical, Philosophical, and Computational Foundations,** Brooks Cole Publishing Co., Pacific Grove, CA, 2000.

Tonti, G., Bradshaw, J. M., Jeffers, R., Montanari, R., Suri, N., & Uszok, A. (2003). “Semantic Web languages for policy representation and reasoning: A comparison of KAoS, Rei, and Ponder.” In D. Fensel, K. Sycara & J. Mylopoulos (Eds.), **The Semantic Web—ISWC 2003**. *Proceedings of the Second International Semantic Web Conference,* Sanibel Island, Florida, USA, October 2003, LNCS 2870. Berlin, Germany: Springer, pp. 419-437.

Uszok, A., Bradshaw, J. M., Johnson, M., Jeffers, R., Tate, A., Dalton, J., & Aitken, S. (2004). “KAoS policy management for semantic Web services.” *IEEE Intelligent Systems,* July/August, 19(4), pp. 32-41.

Uszok, A., Bradshaw, J. M., Breedy, M., Bunch, L., Feltovich, P., Johnson, M. & Jung, H. (2008). “New developments in ontology-based policy management: Increasing the practicality and comprehensiveness of KAoS.” *Proceedings of the 2008 IEEE Conference on Policy*, Palisades, NY, 2-4 June.

APPENDIX A: Program Schedule and Milestones

Table A1. Program Schedule and Milestones

Activity	Status	Start	Finish
Design & Development (Spirals 1-3)	Spirals 1 & 2 development complete. Spiral 3 design underway.	Mon 5/3/10	Wed 1/19/11
Spiral 1: Reqmts Mgmt & Analysis	Spiral 1 complete.	Mon 5/3/10	Wed 7/28/10
Spiral 1: Ontology • Delegation Micro Theory • Domain • Domain Extensions for Policy Definitions	Basic domain ontology complete. Delegation Micro Theory complete. Policy ontology complete.	Mon 5/3/10	Wed 7/28/10
Spiral 1: Policy • Access Control Policy Set • Delegation Policy Set	Spirals 1&2 policy development complete. Spiral 3 policy development underway.	Mon 5/3/10	Wed 7/28/10
Spiral 1: Software • Access Control Service • Delegation Mgmt Service • C4I Services • Demo Clients	Basic service stubs, demo clients and access control complete. DM policy service integration complete.	Mon 5/3/10	Wed 7/28/10
Spiral 1 Working Session	Complete.	Wed 7/28/10	Fri 7/30/10
Spiral 2: Reqmts Mgmt & Analysis	Spiral 2 analysis complete.	Mon 2Aug10	Fri 13Aug10
Spiral 2: Ontology • Delegation Micro Theory • Domain • Domain Extensions for Policy	Micro-theory complete, ontology for domain knowledge complete for spiral 2.	Mon 2Aug10	Fri 13Aug10
Spiral 2: Policy • Access Control Policy Set • Delegation Policy Set	Spiral 2 policies complete.	Mon 2Aug10	Fri 20Aug10
Spiral 2: Software • Access Control Service • Delegation Mgmt Service • C4I Services • Demo Clients	Service development complete, clients stable, KAoS fully integrated with services.	Mon 2Aug10	Fri 20Aug10
Spiral 3: Reqmts Mgmt & Analysis	Spiral 3 complete.	Mon 30Aug10	Fri 3Dec10
Spiral 3: Ontology, Policy & Software	Development complete, integration of enhanced KAoS features complete.	Mon 4Oct10	Fri 7Jan11
Final Demonstration	Successful demonstration on 28Jan .	28Jan11	28Jan11

APPENDIX B: Technical Notes: Questions and Answers

- ? "Is this (the Delegation micro-theory) a Delegation Management Meta-Policy"?

  Not really. An ontology provides the basic building blocks for building knowledge models. A micro-theory is a particular collection of specifically shaped blocks intended to help build models for a particular sub-domain. Policies are specific types of models that we build to model knowledge within the domain. The micro-theory defines the pieces; policies are things we build with those pieces.

- ? "What does this buy us over what we had before in terms of service invocation security?"

  Dramatically more flexibility in specifying and applying service invocation restrictions. Linking policies to roles also eases access management by more clearly targeting the correct access privileges necessary for users to perform their current – and only their current – roles. For additional benefits see the summary paper appended below.

- ? "How hard is it to go around the guard?"

  To do so requires specific knowledge of the configuration and internals of the service host – in our system the JBoss holder. The guard is not a panacea for all security issues but it does add a substantial layer of protection that enforces not just access restrictions, but detailed specific policies appropriate at any point. Users using standard clients do not have access to protected services. The "Rogue Client" and "Misconfigured Service" scenarios demonstrated in our final system show how two attack vectors are correctly handled via the KAoS guard approach.

- ? "Is it possible to delegate your authorities to delegate from one principal to another?"

  Yes. This is a matter that can be specified with KAoS policies. It is possible to either allow or prohibit the transfer of roles. The delegation capability is defined as an operation attached to a role, and that role can itself be delegated.

- ? "Is it possible to assign delegation privileges to people and services, or simply access control to services based on assigned roles?"

  Yes. See previous response.

- ? "Can you assign roles to services as well as people?"

  We've not explored this aspect, but the design of our approach would allow for this. Since a person is represented with credentials and communications via a client, it would be straightforward to construct an analogous set of credentials for agent-level services

- ? "Are credentials/authorities being passed from the invoker of a service to the service being invoked and so on and so forth?"

  The authentication approach is using a token-based system. We are passing a token representing the authentication along with each service invocation. The guard is using those tokens to evaluate the governing policies. A major goal was to avoid having to alter the protected services – to avoid having to rewrite them. Without the guards, the services would have to be altered to handle the tokens directly.

LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS

ACS	Access Control Service
ADF	Access-control Enforcement Function
AFRL	Air Force Research Laboratory
AOC	Air Operations Center
API	Application Programming Interface
ATO	Air Tasking Order
C4I	Command, Control, Communications, Computers & Intelligence
CRUD	Create, Retrieve, Update, and Delete
CSI	Common Services Interface
DDS	Distributed Directory Service
DMS	Delegation Management Service
DoAS	Delegation of Authority Service
DoD	Department of Defense
ISO	International Standards Organization
JAX-WS	Java Extensions for Web Services
KDS	KAoS Directory Service
KPAT	KAoS Policy Administration Tool
OASIS	Organization for the Advancement of Structured Information Standards
OWL	Web Ontology Language
PDP	Policy Decision Point
PEP	Policy Enforcement Point
RDF	Resource Description Framework
SAML	Secure Authentication Mark-up Language
SAWSDL	Security Annotations for WSDL
SIDO	Senior Intelligence Duty Officer
SOA	Services Oriented Architecture
SODO	Senior Offensive Duty Officer
STS	Security Token Server
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
XML	Extensible Mark-up Language
XSL	XML Schema Language
XSLT	XSL Transformations