Policy and Contract Management for Semantic Web Services

Andrzej Uszok, Jeffrey M. Bradshaw, Renia Jeffers, Matthew Johnson
Institute for Human and Machine Cognition (IHMC), 40 S. Alcaniz, Pensacola, FL 32501, USA
{auszok, jbradshaw, rjeffers, mjohnson}@ihmc.us

Austin Tate, Jeff Dalton, Stuart Aitken
Artificial Intelligence Applications Institute, University of Edinburgh, Edinburgh EH8 9LE, UK
{a.tate, j.dalton, s.aitken}@ed.ac.uk

Abstract

This paper summarizes our efforts to develop capabilities for policy and contract management for Semantic Web Services applications. KAoS services and tools allow for the specification, management, analyzes, disclosure and enforcement of policies represented in OWL. We discuss three current Semantic Web Services applications as examples of the kinds of roles that a policy management framework can play: as an authorization service in grid computing environments, as a distributed policy specification and enforcement capability for a semantic matchmaker, and as a verification tool for services composition and contract management.

Introduction

Despite rapid advances in Web Services, the demanding requirements of the user community continue to outstrip currently available technology solutions. To help close this gap, advocates of Semantic Web Services have begun to define and implement new capabilities (*http://www.swsi.org/*). These new capabilities are intended to more fully harness the power of Web Services through explicit representations of the semantics underlying Web resources, and the development of intelligent Web infrastructure capable of fully exploiting them to provide services that can be effectively used not only by people but also by software agents [12]. Semantic Web Languages such as OWL extend RDF to allow users to specify ontologies composed of taxonomies of classes and inference rules. Extending the initial use of the Web by people, agents will increasingly use the combination of semantic markup languages and Semantic Web Services to understand and autonomously manipulate Web content in significant ways. Agents will discover, communicate, and cooperate with other agents and services—and, as described in this paper, will rely on policy-based management and control mechanisms to ensure that human-imposed constraints are respected.

Policies and Semantic Web Services

Policies, which constrain the behavior of system components, are becoming an increasingly popular approach to dynamic adjustability of applications in academia and industry (*http://www.policy-workshop.org/*). Elsewhere we have pointed out the many benefits of policy-based approaches, including reusability, efficiency, extensibility, context-sensitivity, verifiability, support for both simple and sophisticated components, protection from poorly-designed, buggy, or malicious components, and reasoning about their behavior [2]. Policies have important analogues in animal societies and human cultures [6].

Policy-based network and distributed system management has been the subject of extensive research over the last decade (*http://www-dse.doc.ic.ac.uk/Research/policies/*) [19]. Policies are often applied to automate network administration tasks, such as configuration, security, recovery, or quality of service (QoS). In the network management field, policies are expressed as sets of rules governing choices in the behavior of the network. There are also ongoing standardization efforts toward common policy information models and frameworks. The Internet Engineering Task Force, for instance, has been investigating policies as a means for managing IP-multiservice networks by focusing on the specification of protocols and object-oriented models for representing policies (*http://www.ietf.org/html.charters/policy-charter.html*).

The scope of policy management is increasingly going beyond these traditional applications in significant ways. New challenges for policy management include:

- Sources and methods protection, digital rights management, information filtering and transformation, and capability-based access;
- Active networks, agile computing, pervasive and mobile systems;
- Organizational modeling, coalition formation, formalizing cross-organizational agreements;
- Trust models, trust management, information pedigrees;
- Effective human-machine interaction: interruption and notification management, presence management, adjustable autonomy, teamwork facilitation, safety; and
- Support for humans trying to retrieve, understand, and analyze all policies relevant to some situation.

Multiple approaches for policy specification have been proposed that range from formal policy languages that can be processed and interpreted easily and directly by a computer, to rule-based policy notation using an if-then-else format, to the representation of policies as entries in a table consisting of multiple attributes.

In the Web Services world, standards for SOAP-based message security[1] and XML-based languages for access control (e.g., XACML[2]) have begun to appear. However the immaturity of the current tools along with the limited scope and semantics of the new languages make them less-than-ideal candidates for the sorts of sophisticated Web-based applications its visionaries have imagined in the next ten years [7].

The use of XML as a standard for policy expression has both advantages and disadvantages. The major advantage of using XML is its straightforward extensibility (a feature shared with languages such as RDF and OWL, which are built using XML as a foundation). The problem with mere XML is that its semantics are mostly *implicit*. Meaning is conveyed based on a shared understanding derived from human consensus. The disadvantage of implicit semantics is that they are rife with ambiguity, promote fragmentation into incompatible representation variations, and require extra manual work that could be eliminated by a richer representation. However Semantic Web-based policy representations, such as those described in this paper, could be mapped to lower level representations if required by an implementation by applying contextual information.

Some initial efforts in the use of Semantic Web representations for basic security applications (authentication, access control, data integrity, encryption) of policy have begun to bear fruit. For example, Denker *et al.* [5] have integrated a set of ontologies (credentials, security mechanisms) and security extensions for Web Service profiles with the CMU Semantic Matchmaker [13] to enable security brokering between agents and services. Future work will allow security services to be composed with other services. Kagal *et al.* [10] are developing Rei, a Semantic Web language-based policy language that is being used as part of this and other applications.

In another promising direction, Li, Grosof, and Feigenbaum [11] have developed a logic-based approach to distributed authorization in large-scale, open, distributed systems.

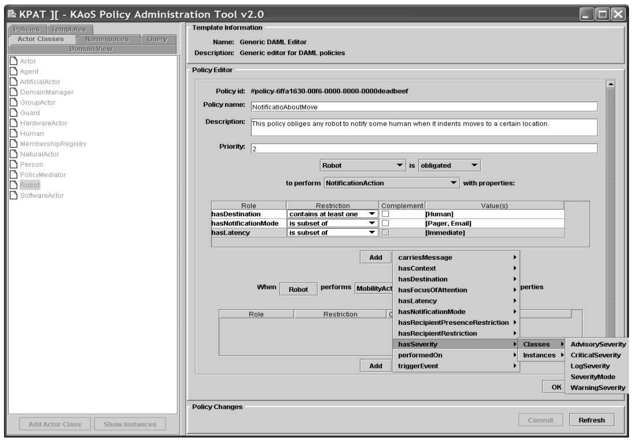
KAoS Policy and Domain Management Services

KAoS services and tools allow for the specification, management, conflict resolution, and enforcement of policies within the specific contexts established by complex organizational structures represented as *domains* [2; 3; 18]. While initially oriented to the dynamic and complex requirements of software agent applications, KAoS services have been extended to work equally well with both agent and traditional clients on a variety of general distributed computing platforms.

KAoS uses ontology concepts (encoded in OWL) to build policies. During its bootstrap, KAoS first loads a KAoS Policy Ontology (KPO) defining concepts used to describe a generic actors' environment and policies within this context (*http://ontology.ihmc.us/*) and then, on top of it, an additional ontology is loaded, extending concepts from the generic ontology, with notions specific to the particular controlled environment.

The KAoS Policy Service distinguishes between *authorizations* (i.e., constraints that permit or forbid some action) and *obligations* (i.e., constraints that require some action to be performed when a state- or event-based trigger occurs, or else serve to waive such a requirement) [4]. Other policy constructs (e.g., delegation, role-based authorization) are built out of the basic primitives of domains plus these four policy types.

The concept of an action is central to the definition of policies in KAoS (Fig. 1). Action is defined as an ontological class used to classify instances of actions that are intended or currently underway. If the particular action instance is of the type of the action class associated with the given policy then this policy is applicable to the current situation.



Name: Generic DAML Editor
Description: Generic editor for DAML policies

Policy id: #policy-6ffa1630-00f6-0000-8000-0000deadbeef
Policy name: NotificatioAboutMove
Description: This policy obliges any robot to notify some human when it indents moves to a certain location.
Priority: 2

Robot is obligated
to perform NotificationAction with properties:

Role	Restriction	Complement	Value(s)
hasDestination	contains at least one	☐	[Human]
hasNotificationMode	is subset of	☐	[Pager, Email]
hasLatency	is subset of	☐	[Immediate]

Figure 1. Graphical interface of the OWL policy editor

The use of OWL enables reasoning about the controlled environment, policy relations and disclosure, policy conflict detection, and harmonization, as well as about domain structure and concepts exploiting the description logic subsumption and instance classification algorithms. Conflicting policies can be identified and, if desired, harmonized through KAoS use of algorithms relying on the inference capabilities of Stanford's Java Theorem Prover (JTP; *http://www.ksl.stanford.edu/software/JTP/*).

A comparison of KAoS, Rei, and Ponder approaches to policy can be found in [17]. We highlight a few important features below.

Homogeneous representation. Because all aspects of KAoS representation are encoded purely in OWL, any third-party tool or environment supporting OWL can perform

[1] e.g., *http://www-106.ibm.com/developerworks/webservices/library/ws-secure/*

[2] *http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security*

specialized analyses of the full knowledge base completely independently of KAoS itself, thus easing integration with an increasingly sophisticated range of new OWL tools and language enhancements in the future.

Maturity. Over the past few years, KAoS has been used in a wide range of applications and operating environments.

Comprehensiveness. Unlike many approaches that deal with only simple forms of access control or authorization, KAoS supports both authorization and obligation policies. In addition, a complete infrastructure for policy management has been implemented including a full range of capabilities from sophisticated user interfaces for policy specification and analysis to a generic policy disclosure mechanism. Facilities for policy enforcement automation (i.e., automatic generation of code for enforcers) are under further development.

Pluggability. Platform-specific and application-specific ontologies are easily loaded on top of the core policy classes. Moreover, the policy enforcement elements have been straightforwardly adapted to a wide range of computing environments.

In the remainder of the paper, we discuss three current applications of KAoS to Semantic Web Services, as examples of the kinds of roles that a policy management framework can play in providing:

- Policy management for the Grid Computing environments (*http://www.globus.org/ogsa/*);
- Distributed policy specification and enforcement to a Semantic Matchmaker;
- Verification for Semantic Web Services composition and contract management.

Policy Management for Grid Computing

Our first foray into Web Services has been the development of an initial OGSA-compliant[1] version of KAoS services, allowing fine-grained policy-based management of registered Grid Computing services on the Globus platform [9]. Our goal has been to extend and generalize this capability so as to work with Web Services outside of Grid Computing environments.

Globus provides effective resource management, authentication and local resource control for the grid-computing environment, but has a need for domain and policy services. KAoS seemed to be a perfect complement to the Globus system, providing a wide range of policy management capabilities that rely on platform-specific enforcement mechanisms. By providing an interface between the Globus Grid and KAoS, we enable the use of KAoS mechanisms to manage GSI (Grid Security Infrastructure) enabled Grid services. GSI was the only component of the GT3 (Globus Toolkit) we used in the integration. The interface itself is a Grid service, which we called a KAoS Grid service. It provides Grid clients and services the ability to register with KAoS services, and to check weather a given action is authorized or not based on current policies. The basic architecture is shown in Fig 2.

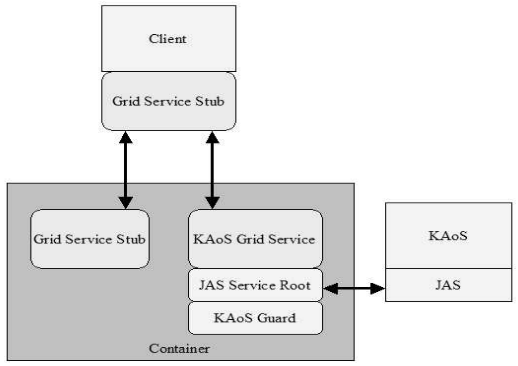


Figure 2. KAoS Grid Service Architecture

Creating a KAoS Grid Service

In order to create a KAoS Grid service, we used tools provided with GT3 to create a normal Grid service and then added to it the required KAoS framework components to make it KAoS aware. This framework links Grid services to the KAoS-implemented JAS[2] services: Naming, Message Transport, and Directory. It also associates a Guard with the KAoS Grid Service, described above.

Registration

To use domain services, we needed to establish a method for clients and resources to register within a KAoS domain. The clients or resources use their credential to request to be registered into one or more domains. The credential is a standard X.509 certificate that Globus uses for authentication. The credential is verified using the GT GSI. If the certificate is valid the registration request is sent to KAoS for registration into the desired domains. If the resource uses an application specific ontology to describe its capabilities, it will have to be loaded into the KAoS ontology using a utility provided by KAoS. Inside the KAoS Grid service, the registration is handled through the associated Guard. This allows KAoS to distribute all applicable policies to the appropriate Guard.

Expressing Policies

The basic components of any authorization policy are the actor(s), action and target(s). A sample policy would read as follows:

It is permitted for actor(s) X to perform action(s) Y on target(s) Z.

Actors requesting to execute an action are mapped to various actor classes and instances in the KAoS Policy Ontology (KPO). In this case, actors consist of various software clients and the groups they belong to. Registration adds each client to the existing KAoS knowledge base stored within JTP, offline or at runtime, enabling policies to be written about the client or its domain.

[1] OGSA - Open Grid Services Architecture, a Web Services-compatible standard for defining Grid Computing Services

[2] JAS – Java Agent Services (*http://sourceforge.net/projects/jas/*)

The actions can be represented at different levels of generality. A policy defined on a more general action might permit or forbid overall access to a service, which is useful for simple services or services that do not provide varying levels of access. For example, a policy defining overall permissions for a chat service might make use of generic communication concepts in the existing KPO as in the following:

It is forbidden for Client X to perform a communication action if the action has a destination of Chat Service Y.

This policy would be used to prevent Client X from using Chat Service Y. KAoS already understands the concepts of "forbidden", "communication action" and "has destination." KAoS will also understand "Client X" and "Chat Service Y" once each entity registers.

More complex services may require new concepts in the ontologies that map to specific actions on a Grid Service. For example, Reliable File Transfer Service has a variety of methods that may not map to an existing ontology. To provide fine-grained control of this service, the KAoS ontology can be extended for the specific domain space and loaded into KAoS using KPAT (KAoS Policy Administration Tool), a graphical user interface for interacting with KAoS (see Fig. 1 for example of KPAT window). We are currently working on a tool to automatically generate OWL ontology for a given WSDL[1] specification of the OGSI-Compliant Grid Service.

Targets can be clients, services, or domain specific entities, such as different computing resources. The first two cases are added to the KAoS ontology upon their registration within KAoS Directory Service, but the last case, requires extensions to the ontology, either before loading into KAoS or using graphical interface in KPAT.

Policies may be written to restrict a client's use of a resource, or to restrict the set of access rights delegated to the KAoS Grid service.

Checking Authorization

Since the KAoS Grid Service has full control of access to a given resource based on the rights permitted by participating resources, it serves as the policy enforcer, using Globus local enforcement mechanisms. The KAoS Grid service coordinates with the KAoS Guard to determine authorization for a requested action. Once registered, clients will have access to the Grid service based on the policies in KAoS. As policies are added into KAoS through KPAT, they are automatically converted to OWL for use in reasoning, and to a simple and efficient representation in the Guard associated with the KAoS Grid service for enforcement purposes. When a client requests a service, the KAoS Grid service will check if the requested action is authorized based on current policies by querying the Guard. If the Guard allows the requested action, KAoS Grid service initializes a GIS restricted proxy certificate by putting the permissions needed to execute the action in its own end GIS entity certificate. This certificate is the one provided by the resource at registration and maps to the local control mechanism. The KAoS Grid Service also sets the proxy lifetime and signs it. The restricted proxy certificate is returned to the client. The client then uses this proxy certificate to access the given grid service.

When a service receives a request it checks the submitted certificate against the local GIS control mechanism. Services can also check permissions by querying the KAoS Grid service directly. The service checks to ensure that action requested is covered by the intersection of the rights given to the KAoS service and the rights embedded in the certificate by the KAoS service. This allows the local resource owner to write policies restricting the rights it allows KAoS to delegate.

A current limitation of our implementation is that there is no mechanism for proxy certificate revocation. Globus relies on short lifetimes to limit proxy credentials. An updated policy in KAoS would not take effect until the current proxy credential expired forcing the user to return to KAoS for an update.

Policy Management for Semantic Matchmaking

Within the CoSAR-TS (Coalition Search and Rescue Task Support; Principal Investigator: Austin Tate) project (*http://www.aiai.ed.ac.uk/project/cosar-ts/*) we are testing the integration of KAoS and AIAI's I-X technology with Semantic Web Services. Military search and rescue operations by nature require the kind of rapid dynamic composition of available policy-constrained resources for a task that make it a good use case for Semantic Web technologies. Other participants in the application include BBN Technologies, SPAWAR, AFRL, and Carnegie Mellon University.

I-X Technologies

I-X Process Panels (*http://i-x.info*; [15; 16]) can provide task support by reasoning about and exchanging with other agents and services any combination of Issues, Activities, Constraints and Annotations (in the <I-N-C-A> ontology). I-X can therefore provide collaborative task support and exchange of structured messages related to plans, activity and the results of such activity. These types of information can be exchanged with other tools via OWL, RDF or other languages. The system includes an AI planner that can compose a suitable plan for the given tasks when provided with a library of standard operating procedures or processes, and knowledge of other agents or services that it may use.

Fig. 3 shows an I-X Process Panel (I-P^2) and associated I-X Tools. I-X can make use of multiple communications methods ranging from simple XML instant messaging (e.g. Jabber) to sophisticated policy constrained agent communications environments (e.g. CoABS Grid, KAoS). The I-Space tool maintains agent relationships. The relationships can be obtained from agent services such as KAoS if that is used to describe agents, domains and

[1] WSDL - Web Services Description Language (*http://www.w3.org/TR/wsdl*)

policies. Communications methods and new contacts can be added or changed dynamically while an I-X system is running. I-X Process Panels can also link to semantic web information and web services, and can be integrated via "I-Q" adaptors [14] to appear in a natural way during planning and in plan execution support.

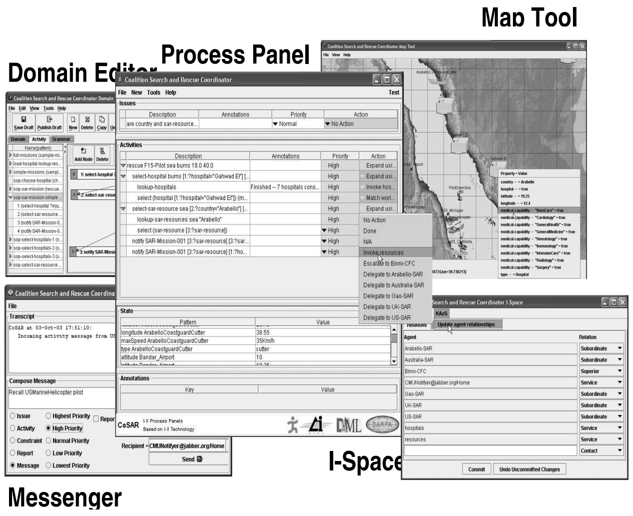


Figure 3. I-X Process Panel for a Coalition Search and Rescue Task

Constraints sent to I-X immediately change the model state that is visualized in all views and used throughout the system. These changes can trigger preconditions on activities and affect the action options presented in the selection menus. So, for example, web services availability information, agent presence or status, and agent or people GPS positions can be sent to I-X as world state constraint messages and appear immediately. This allows for high levels of dynamic workflow support.

I-X work to date has concentrated on dynamically determined workflows at execution time – using knowledge of services and other agent availability, etc. However, there is also a process editor for creating process models (I-DE) to populate the domain model and an AI planner (I-Plan) which allows for hierarchical plan creation, precondition achievement, consistent binding of multiple variables, temporal constraint checking, and so forth.

CoSAR-TS Scenario

The scenario begins with an event that reports a downed airman between the coastlines of four fictional nations bordering the Red Sea: Agadez, Binni and Gao (to the West), and Arabello (to the East). In this initial scenario it is assumed that excellent location knowledge is available, and that there are no local threats to counter or avoid in the rescue. The airman reports his own injuries via his suit sensors in this initial scenario.

Next is an investigation of the facilities available to rescue the airman. There will be three possibilities: a US ship-borne helicopter; a Gaoan helicopter from a land base in Binni; or a patrol boat from off the Arabello coastline.

Finally, there is a process to establish available medical facilities for the specialized injury reported using the information provided about the countries in the region. A hospital in Arabello is best placed to provide the facilities, due to the fact that it has the necessary treatment facilities. However, there is a coalition policy that no Gaoan helicopters may be used by coalition members to transport injured airmen.

CoSAR-TS Scenario Knowledge and Ontologies

Several OWL ontologies define the SAR domain and the services that are available. Knowledge of medical facilities is obtained from a medical OWL ontology stored in the BBN SONAT database. This has been extended to include data on the fictional countries in the Binni scenario: Agadez, Arabello, Binni and Gao. The medical ontology includes several instances of *Hospital* and other medical facilities, and associated information provides the latitude and longitude of hospital locations. Services that may be invoked are defined in OWL-S. For example, we defined the *GaoMarineHelicopter* service profile, which has the associated atomic process *PickUpDownedPilot*. Gao provides this profile, and has an input defined by the (constrained) parameter description *PickUpLocation* which refers to the property *pickUpLocation_In* restricted to *Location*. The *HospitalLocation* and *CountryOfHospital* are further inputs to the service profile, and these have similar definitions. These resources provide the domain knowledge and service capabilities that are required to plan the SAR mission.

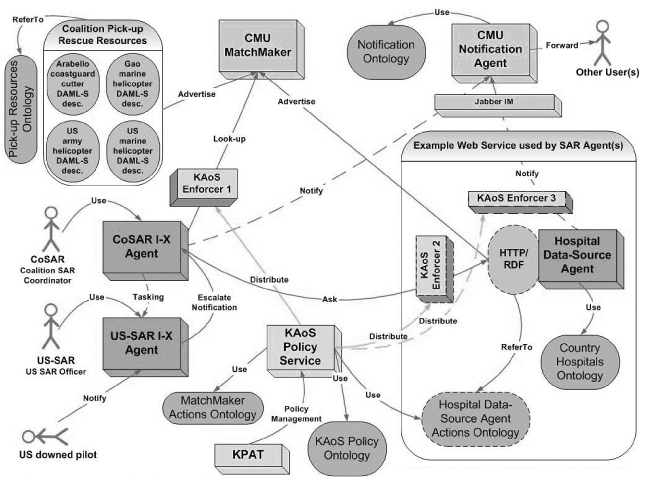


Figure 4. CoSAR-TS architecture.

CoSAR-TS Components

Four coalition agents are used, representing the roles and functions of the Coalition SAR coordinator, US SAR officer, hospital information provider, and SAR resource provider. The Coalition SAR coordinator has an I-X process panel, which can be used to follow a standard operating procedure. This is represented by a rescue process in I-X, which, at the top-level, contains the four activities: *select hospital*; *select SAR resource*; *notify SAR resource*; *notify hospital*; which are executed sequentially. The *select hospital* activity is broken down further into three steps, one of which is *lookup hospital*—an action that can be carried out by querying the SONAT database of OWL-encoded facts (as described above). The decomposition of the *select SAR resource* activity includes *lookup SAR resource*, an activity that can be carried out by querying the CMU Semantic Matchmaker [13] for a service with a matching profile (i.e. one of the rescue services encoded in OWL-S such as *GaoMarineHelicopter*). Notifications are done using Sadeh's Notification Agent[1], which relies upon profiles defined using concepts from the notification ontology to forward messages.

CoSAR-TS Policy Services Description

While annotation of the Semantic Matchmaker service profiles allows registered service providers to describe required security profiles [5], it does not allow owners of infrastructure resources (e.g., computers, networks), client organizations (coalition organizations, national interest groups), or individuals to specify or enforce policy from their unique perspectives. For example, the policy that coalition members cannot use Gaoan transports is not something that can be anticipated and specified within the Matchmaker service profile. Neither would Matchmaker service profile annotations be an adequate implementation for a US policy obligating encryption, prioritizing the allocation of network bandwidth, or requiring the logging of certain sorts of messages.

Moreover, the semantics of these policies cannot currently be expressed in terms of the current OWL-S specification of conditional constraints. Even if they were expressible, organizations and individuals may prefer to keep policy stores, reasoners, and enforcement capabilities within their private enclaves. This may be motivated by both the desire to maintain secure control over sensitive components as well as to keep other coalition members from becoming aware of private policies. For example, coalition members may not want Gao to be aware that the offer of their helicopters to rescue the downed airman will be automatically filtered out by policy.

Within the current CoSAR-TS implementation, KAoS is used to define, represent, analyze, query, and deconflict policies about access to Semantic Matchmaker information. Additionally, we have defined enforcers that intercept SOAP messages from the Matchmaker and filter results consistent with coalition policies, specifically those that prevent the use of Gaoan resources in the demo situation.

We are extending the SOAP-enabled enforcer to understand arbitrary Web Semantic Service invocations so it can apply appropriate authorization policies to them. Additionally, we plan to equip it with a mechanism to perform obligation policies, which will be in the form of other Web Service invocations. For instance it can be imagined that some policy may require consultation or registration of performed transactions in some logging service available as a Web Service audit entity.

[1] *http://www-2.cs.cmu.edu/~sadeh/*

Verification for Semantic Web Services Composition

Automatic composition of feasible workflows from available Semantic Web Services is an ongoing research topic. An obvious solution argued in this paper, as also proposed in [20], is the application of existing solutions: the input and output formats can be straightforwardly mapped to the emerging standard of the Semantic Web Services Process Model (*http://www.owl.org/services/owl-s/1.0/*). To this end, we are extending our implementations of I-X and KAoS.

I-K-C Tool

In the context of CoSAR-TS, we have already integrated KAoS and I-X to allow I-X to obtain information about the relations about actors (human and software) such as peers, subordinates and superiors and others represented in domains and stored in the KAoS Directory Services. I-X is also already able to use the KAoS policy disclosure interface to learn about the impact of policies on its planned actions. These represent first steps toward mutual understanding of workflow specifications.

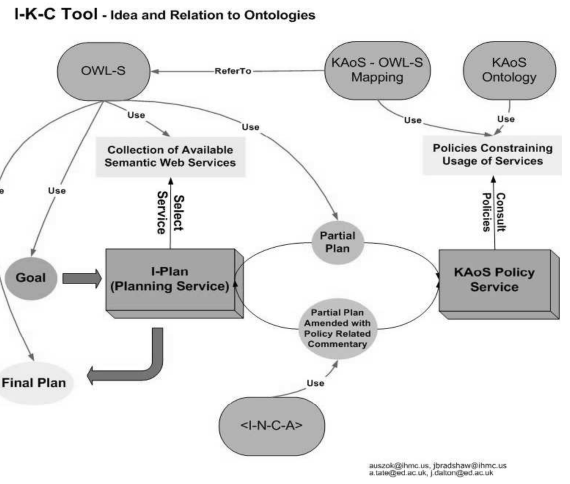

I-K-C Tool - Idea and Relation to Ontologies
OWL-S
ReferTo
KAoS - OWL-S Mapping
KAoS Ontology
Use
Collection of Available Semantic Web Services
Policies Constraining Usage of Services
Select Service
Consult Policies
Partial Plan
Goal
I-Plan (Planning Service)
KAoS Policy Service
Partial Plan Amended with Policy Related Commentary
Final Plan
<I-N-C-A>
auszok@ihmc.us, jbradshaw@ihmc.us
a.tate@ed.ac.uk, j.dalton@ed.ac.uk

Figure 5. Cooperation between I-X and KAoS in the Process of Semantic Workflow Composition

I-K-C is a name of the tool, integrating I-X and KAoS, enabling composition of Semantic Web Services into workflow taking into account policies constraining usage of these services and their composition. At first, this tool is going to build a workflow from the client perspective taking into account only its policies, constraining usage of different services and their composition, when building the workflow. In order to realize the architecture in Fig. 5[2], the

[2] *http://ontology.ihmc.us/SemanticServices/I-K-C/I-K-C-Ontologies.htm* provides links to the actual ontology files in OWL

I-X Process editor and I-Plan planning elements are being extended to allow the creation of composed workflows in advance of execution. This will allow for the import of services described in OWL-S to be used within the planner, augmenting any predefined processes in the process library. KAoS verifies constructed partial plans for policy compliance. In order to achieve that a mapping from the KAoS ontology of action[1] to the OWL-S process ontology was developed. Additional ontologies enabling modification of partial plans with policy-related markup to describe the results of policy checks for composed workflows is also required. We are using <I-N-C-A> so that the results can be expressed either as specific constraints that must be added into the 'plan,' as specific issues to address, or possibly activities to be added. Additional information that cannot be conveyed as one of these three types can be conveyed as <I-N-C-A> annotations. The final plan in OWL-S ontology can be exported for use in other enactment systems or can be utilized to guide the dynamic reactive execution of those plans in I-P[2] or other tools.

An open research issue, common to others now exploring the use of HTN planning approaches to web services composition [20], is how far to go with plan time composition and how much to leave unselected to dynamic enactment support which can account for available services, discovery of services, and so forth.

During plan execution, the KAoS Policy Service can independently ensure that interactions between services comply with policies constraining their usage. The policies controlling both authorization and obligation of clients and servers are stored in KAoS and checked by interested parties[2]. It is possible to automatically enforce these policies, both authorizations and obligations, by integration of Semantic Web Services with the KAoS enforcer, as a generalization of what is being done in the CoSAR-TS application. The existing approaches to securing Semantic Web Services are limited to either marking Service advertisement with requirements for authentication and communication and enforcing compliance with these requirements [5] or by attaching conditions to inputs, outputs and effects of services. KAoS is able to reason about the entire action performed by the services and soon will be able to understand workflows defined by I-X technologies. Additionally we plan to use KAoS to generate obligations created during use of the services, which can be passed as constraints back to I-X.

Semantic Firewall

A necessary requirement for the support of complex, dynamic groups of service providers in a business context is the notion of a contract. While KAoS policies represent constraints on behavior involuntarily imposed on software entities, contracts represent voluntary agreements that mutually bind the participants to various authorizations, obligations, and modes of interaction. As an example of the application of contracts to Semantic Web Services, Grosof and Poon [8] have developed SweetDeal, a rule-based approach to automating "law in the small." SweetDeal represents business contracts to allow software agents to create, evaluate, negotiate, and execute contracts among themselves for the performance of Semantic Web Services.

Within KAoS, we plan to extend the existing representation of policy sets to include rules and other constructs necessary to enable the creation and execution of contracts. As part of contract creation, KAoS already has capabilities for detecting policy conflicts and suggesting harmonization [3]. These are being extended and combined with new facilities for negotiation, and extensions to existing capabilities for enforcement.

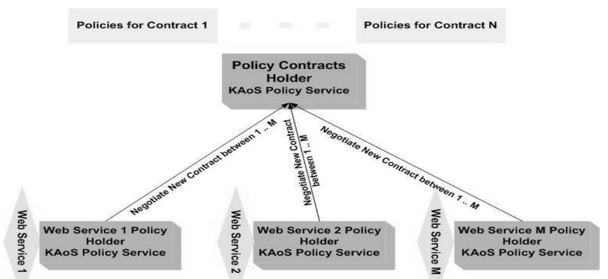

Policies for Contract 1
Policies for Contract N
Policy Contracts Holder KAoS Policy Service
Negotiate New Contract between 1 .. M
Negotiate New Contract between 1 .. M
Negotiate New Contract between 1 .. M
Web Service 1
Web Service 1 Policy Holder KAoS Policy Service
Web Service 2
Web Service 2 Policy Holder KAoS Policy Service
Web Service M
Web Service M Policy Holder KAoS Policy Service

Figure 6. KAoS Policy Service as Negotiator, Holder and Enforcer of Contracts Policies between Web Services

Contracts can be stored either within instances of KAoS (or perhaps some other interoperable policy service) associated with each Web Service or else, when stakeholders prefer, as independent KAoS instances representing neutral third parties (Fig. 6). We have begun to explore these new issues in policy and contract management and execution in the context of a collaboration with University of Southampton, IT Innovation, and SRI International to develop a *Semantic Firewall* [1][3]. The tool will take as an input a desired client workflow and starts negotiation with the policy services associated with semantic service provider to be used in the proposed workflow. In effect, the initial workflow can be modified and amended with the agreed policy contract. This contract will be then enforced by the system.

Tools Integration

An Integration of these two complementary tools, I-K-C and Semantic Firewall, is envisioned for the next phase. The resulting combination can use I-K-C to produced initial workflow and then a policy contract can be negotiated and enforced by the Semantic Firewall. Further development may combine these tools even closer by interleaving workflow planning and policy negotiation phases.

[1] *http://ontology.ihmc.us/Action.owl*

[2] Of course these parties would have to be authorized to have access to the policies in question.

[3] See *http://ontology.ihmc.us/SemanticServices/S-F/Example/index.html* for an example scenario with policies encoded using KAoS Policy syntax.

Acknowledgements

This material is based on research sponsored by the Defense Advanced Research Projects Agency (DARPA) and US Air Force Research Laboratory under agreement numbers F30602-00-2-0577 and F30602-03-2-0014. The U.S. Government, IHMC, and the University of Edinburgh are authorized to reproduce and distribute reprints and on-line copies for their purposes notwithstanding any copyright annotation hereon.

References

[1] Ashri, R., Payne, T., & Surridge, M. (2004). Towards a Semantic Web Security Infrastructure. AAAI Spring Symposium on Semantic Web Services. Stanford Univ.

[2] Bradshaw, J. M., Beautement, M. Breedy, L. Bunch, S. Drakunov, P. Feltovich, P., Raj, A., Johnson, M., Kulkarni, S., Suri, N. & A. Uszok (2003). Making agents acceptable to people. In N. Zhong & J. Liu (Ed.), Intelligent Technologies for Information Analysis: Advances in Agents, Data Mining, and Statistical Learning. (pp. in press). Berlin: Springer Verlag.

[3] 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). Melbourne, Australia, New York, NY: ACM Press.

[4] Damianou, N., Dulay, N., Lupu, E. C., & Sloman, M. S. (2000). Ponder: A Language for Specifying Security and Management Policies for Distributed Systems, Version 2.3. Imperial College of Science, Technology and Medicine, Department of Computing.

[5] Denker, G., Kagal, L., Finin, T., Paolucci, M., & Sycara, K. (2003). Security for DAML Web Services: Annotation and Matchmaking. In D. Fensel, K. Sycara, & J. Mylopoulos (Ed.), Proceedings of the Second International Semantic Web Conference, Sanibel Island, Florida, 2003, LNCS 2870. Berlin: Springer Verlag.

[6] Feltovich, P., Bradshaw, J. M., Jeffers, R., & Uszok, A. (2003). Social order and adaptability in animal, human, and agent communities. Proceedings of the Fourth International Workshop on Engineering Societies in the Agents World, (pp. 73-85). Imperial College, London,

[7] Fensel, D., Hendler, J., Lieberman, H., & Wahlster, W. (Ed.). (2003). Spinning the Semantic Web. Cambridge, MA: The MIT Press.

[8] Grosof, B. N., & Poon, T. C. (2003). SweetDeal: Representing agent contracts with exceptions using XML rules, ontologies, and process descriptions. Submitted for publication.

[9] Johnson, M., Chang, P., Jeffers, R., Bradshaw, J. M., Soo, V.-W., Breedy, M. R., Bunch, L., Kulkarni, S., Lott, J., Suri, N., & Uszok, A. (2003). KAoS semantic policy and domain services: An application of DAML to Web services-based grid architectures. Proceedings of the AAMAS 03 Workshop on Web Services and Agent-Based Engineering. Melbourne, Australia.

[10] Kagal, L., Finin, T., & Joshi, A. (2003). A policy-based approach to security for the Semantic Web. In D. Fensel, K. Sycara, & J. Mylopoulos (Ed.), The Semantic Web—ISWC 2003. Proceedings of the Second International Semantic Web Conference, Sanibel Island, Florida, USA, October 2003, LNCS 2870. (pp. 402-418). Berlin: Springer.

[11] Li, N., Grosof, B. N., & Feigenbaum, J. (2003). Delegation logic: A logic-based approach to distributed authorization. ACM Transactions on Information Systems Security (TISSEC), 1-42.

[12] McIlraith, S. A., Son, T. C., & Zeng, H. (2001). Semantic Web Services. IEEE Intelligent Systems, 46-53.

[13] Paolucci, M., Kawamura, T., Payne, T. R., & Sycara, K. (2002). Semantic matching of Web Services capabilities. Proceedings of the First International Semantic Web Conference. Sardegna, Italy.

[14] Potter, S., Tate, A., & Dalton, J. (2003). I-X Task support on the Semantic Web. Poster and Demonstration Proceedings for the Second International Semantic Web Conference (ISWC 2003). Sanibel Island, FL.

[15] Tate, A. (2003). Coalition task support using I-X and <I-N-C-A>. In Proceedings of the Third International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 2003), 16-18 June, Prague, Czech Republic, LNAI 2691. (pp. 7-16). Berlin: Springer Verlag.

[16] Tate, A., Dalton, J., & Potter, S. (2004). Intelligible Messaging: Activity-oriented instant messaging. Unpublished. *http://i-x.info/documents/*.

[17] 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. Proceedings of the Second International Semantic Web Conference, Sanibel Island, Florida, 2003, LNCS 2870. Berlin: Springer Verlag.

[18] Uszok, A., Bradshaw, J. M., Jeffers, R., Suri, N., Hayes, P., Breedy, M. R., Bunch, L., Johnson, M., Kulkarni, S., & Lott, J. (2003). KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. Proceedings of Policy 2003. Como, Italy.

[19] Wright, S., Chadha, R., & Lapiotis, G. (2002). Special Issue on Policy-Based Networking. IEEE Network, 16(2), 8-56.

[20] Wu, D., Parsia, B., Sirin, E., Hendler, J., & Nau, D. (2003). Automating DAML-S Web Services composition using SHOP2. In D. Fensel, K. Sycara, & J. Mylopoulos (Ed.), Proceedings of the Second International Semantic Web Conference, Sanibel Island, Florida, 2003, LNCS 2870. Berlin: Springer.