Applying KAoS Services to Ensure Policy Compliance for Semantic Web Services Workflow Composition and Enactment

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

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

Abstract

In this paper we describe our experience in applying KAoS services to ensure policy compliance for Semantic Web Services workflow composition and enactment. We are developing these capabilities within the context of two applications: Coalition Search and Rescue (CoSAR-TS) and Semantic Firewall (SFW). We describe how this work has uncovered requirements for increasing the expressivity of policy beyond what can be done with description logic (e.g., role-value-maps), and how we are extending our representation and reasoning mechanisms in a carefully controlled manner to that end. Since KAoS employs OWL for policy representation, it fits naturally with the use of OWL-S workflow descriptions generated by the AIAI I-X planning system in the CoSAR-TS application. The advanced reasoning mechanisms of KAoS are based on the JTP inference engine and enable the analysis of classes and instances of processes from a policy perspective. As the result of analysis, KAoS concludes whether a particular workflow step is allowed by policy and whether the performance of this step would incur additional policy-generated obligations. Issues in the representation of processes within OWL-S are described. Besides what is done during workflow composition, aspects of policy compliance can be checked at runtime when a workflow is enacted. We illustrate these capabilities through two application examples. Finally, we outline plans for future work.

1. 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 many new and significant 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. Semantic Web Languages such as OWL extend RDF to allow users to specify ontologies composed of taxonomies of classes and inference rules.

Semantic Web Services can be effectively used not only by people but also by software agents [10]. 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 on agent interaction are respected. Policy-based controls of Semantic Web Services can also be used to govern interaction with traditional (non-agent) clients.

2. 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/*) [18]. 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 for the next decade [7; 12].

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.

In addition to our own work on KAoS (see below), some initial efforts in the use of Semantic Web representations for basic security applications (authentication, access control, data integrity, and encryption) of policy have begun to bear fruit. For example, Denker *et al.* have integrated a set of ontologies (credentials, security mechanisms) and security extensions for OWL-S Service profiles with the CMU Semantic Matchmaker [12] to enable security brokering between agents and services. Future work will allow security services to be composed with other services. Kagal *et al.* [8] are developing Rei, a Semantic Web language-based policy language that is being used as part of the described above OWL-S Service profiles extension and other applications.

In another promising direction, Li, Grosof, and Feigenbaum [9] have developed a logic-based approach to distributed authorization in large-scale, open, distributed systems.

3. KAoS Policy and Domain Management Services

KAoS is one of the first efforts to represent policy using a Semantic Web language—in this case OWL[3]. 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; 16; 17]. 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 (e.g., CORBA, Web Services, Grid Computing (Globus GT3)).

3.1 Ontological Representation of KAoS Policies

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

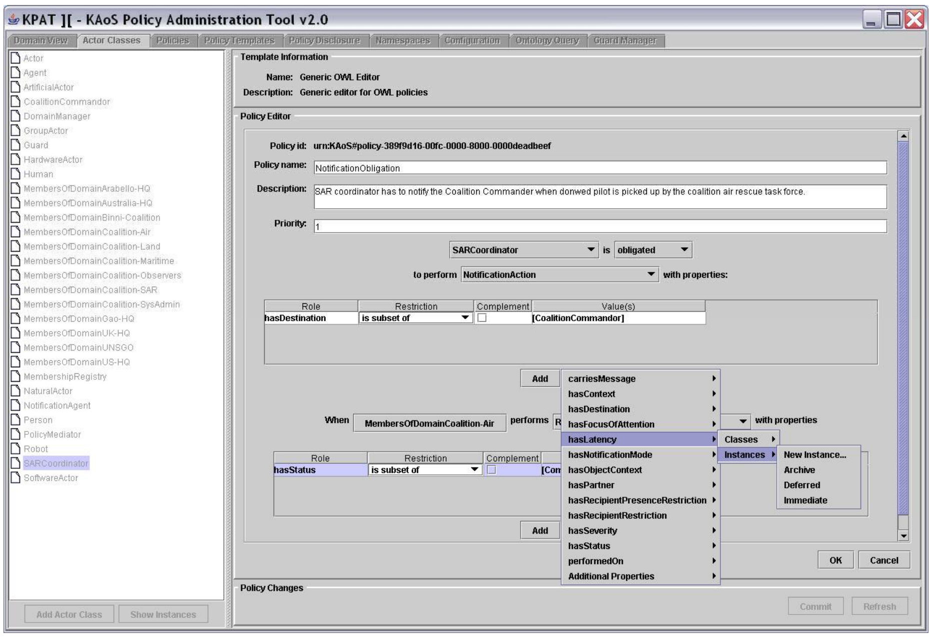
[1] e.g., *http://www-106.ibm.com/developerworks/webservices/library/ws-secure/*

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

[3] A comparison among two semantically-rich representations of policy (KAoS, Rei) and amore traditional policy language (Ponder[5]) can be found in [15].

describe a generic actors' environment and policies within this context (*http://ontology.ihmc.us/*). Then KAoS loads additional ontology, 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.



Template Information

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

Policy Editor

Policy id: urn:KAoS#policy-389f9d16-00fc-0000-8000-0000deadbeef
Policy name: NotificationObligation
Description: SAR coordinator has to notify the Coalition Commander when donwed pilot is picked up by the coalition air rescue task force.
Priority: 1

SARCoordinator is obligated

to perform NotificationAction with properties:

Role	Restriction	Complement	Value(s)
hasDestination	is subset of	☐	[CoalitionCommandor]

When MembersOfDomainCoalition-Air performs R with properties

Role	Restriction	Complement	
hasStatus	is subset of	☐	[Com

Policy Changes

Fig. 1. Graphical interface of the OWL policy editor and administration tool: KPAT.

KAoS policy's OWL definition of (Fig. 1 shows the tool to define such policies) is an instance of one of four basic policy classes, that is: *PositiveAuthorization*, *NegativeAuthorization*, *PositiveObligation* or *NegativeObligation*. The property values determine management information for a particular policy (for example, its priority). The type of policy instance determines the kind of constraint KAoS should apply to the action, while a policy's action class is used to determine a policy's applicability in a given situation. The action class uses OWL restrictions to narrow scopes-of-action properties to a particular policy's needs. Every action contains a definition of the range of actors performing it. This range can be defined using any available OWL construct. For example, the range can be an enumeration of actor instances, a class of actors defining its type, or any description of the actor context (for instance, the class of actors executed on some host and possessing a given resource). The same is true for the action class's other properties but additionally XML Schema expressions can be used to restrict ranges of datatype properties. Consequently, policy can contain arbitrarily complex definitions of a situation. So, KAoS policies represent policies without conditional rules, relying instead on the context restrictions associated with the action class to determine policy applicability in a given situation

An action class helps classify action instances that actors intend to take or are currently undertaking. Components (such as KAoS Guards) that are interested in checking policy impact on these actions construct RDF descriptions of action instances. KAoS classifies these instances, relying on the inference capabilities of Stanford University's Java Theorem Prover (JTP, *www.ksl.stanford.edu/software/JTP*). It then obtains a list of any policies whose action classes are relevant to the current situation. In the next step, KAoS determines the relative precedence of the obtained policies and sorts them accordingly in order to find the dominating authorization policy. If the dominating authorization is positive, KAoS then collects, in order of precedence, obligations from any triggered obligation policies. KAoS returns the result to the interested parties—in most cases, these parties are the enforcement mechanisms that are jointly responsible for blocking forbidden actions and assuring the performance of obligations.

Representing policies in OWL facilitates reasoning about the controlled environment, policy relations and disclosure, policy conflict detection, and harmonization. It also facilitates reasoning about domain structure and concepts exploiting the description logic subsumption and instance classification algorithms. KAoS can identify and, if desired, harmonize conflicting policies through algorithms that we have implemented in JTP.

3.2 Important KAoS Features

We highlight a few important features of KAoS below:

- *Homogeneous policy representation.* Because all aspects of KAoS policy representation are encoded purely in OWL, any third-party tool or environment supporting OWL can perform 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 ontology is easily loaded on top of the core concepts. Moreover, the policy enforcement elements have been straightforwardly adapted to a wide range of computing environments, both traditional distributed computing platforms (e.g., Web Services, Grid Computing, CORBA) and various software and robotic agent platforms (e.g., Nomads, Brahms, SFX, CoABS Grid, Cougaar).
- *Scalability and performance.* We optimized the policy disclosure methods such that response to a query from an enforcer is provided on average in less than 1 ms. This performance is due in part to our reliance on efficient and logically decideable description logic subsumption and classification methods. Furthermore, queries can be executed concurrently by multiple enforcers, letting KAoS export multiprocessor machines. In rigorous evaluations in the

DARPA UltraLog program, we've found that performance is acceptable even in large societies of more than a thousand agents, running on a dozen or more platforms, with hundreds of policies. Here, dynamic policy updates can be committed, deconflicted, and distributed in a matter of a few seconds. Further enhancements to underlying reasoners and advances in computer hardware will continue to improve this performance.

3.3 Beyond Description Logic for Policy Representation

Until recently, KAoS used only OWL-DL (initially DAML) to describe policy-governed entities and their actions. The semantic richness OWL enables in comparison to traditional policy languages allowed us much greater expressivity in specifying policies. However, we found ourselves limited in situations where we needed to define policies where one element of an action's context depended on the value of another part of the context. A simple example is an action of loop communication, where you must constrain the source and the destination of communication so that they're one and the same. A more complex example would be when we want to constrain the action to return the results of a calculation to only the parties that provided the data used to perform it (or to the specific entities the data's providers authorized). Such an action description might be needed to specify a policy controlling the distribution of calculation results. All such action descriptions go beyond what OWL-DL can express.

The required missing aspect of representational semantics has, however, been well studied under the name of role-value maps [10]. These maps should express equality or containment of values that has been reached through two chains of instance properties. The emerging standard for OWL rules, the Semantic Web Rule Language (SWRL, www.daml.org/2003/11/swrl), allows the use of role-value-map semantics. However, the required syntax is complex, and we've begun to think that an OWL-based representation expressing this same semantics might be valuable for a broad range of uses. For instance, the OWL-S developers found the need to express similar dataflow semantics and developed their own formulation (process:sameValues) that allowed the representation of such chains, albeit with the limitation that they could contain only single-chain elements [11].

We have equipped KAoS with mechanisms that will allow adding role-value-map semantics to defined policy action using the KAoS Policy Administration Tool. For the interim, we're basing our syntax for this semantics on the current version of the SWRL OWL ontology. However, the code that generates this syntax is encapsulated in a specialized Java class allowing later modification if the SWRL ontology changes or if an OWL-based syntax eventually emerges. Our classification algorithm can also use this information to classify action instances. This algorithm verifies if an instance satisfies the OWL-DL part of the action class and, if so, checks the appropriate role-value-map constraints. For example, if KAoS needs to determine whether an intercepted communication is a loop communication, it would determine whether the current communication source is also one of the values of the property describing the communication's destination.

To perform more complex policy analyses relying on role-value-map semantics, we've begun joint exploration with Stanford on extending JTP to allow subsumption reasoning on role-value-map semantics.

4. Example Application Contexts

In the remainder of the paper, we will discuss how KAoS is being extended to address two complementary requirements in a Semantic Web Services context:

- Verification for policy compliance for Semantic Web Services workflow composition (section 5),
- Enforcement of policies during the workflow enactment (section 6).

In this section, we briefly introduce the application contexts that motivate these investigations.

4.1 Coalition Search and Rescue Scenario

Within the CoSAR-TS[1] (Coalition Search and Rescue Task Support) project we are testing the integration of KAoS and AIAI's I-X technology with Semantic Web Services. Search and rescue operations, especially coalition based, by nature require the kind of rapid dynamic composition of available policy-constrained heterogeneous resources that make it a good use case to describe them using Semantic Web technologies. Additionally, military operations usually are conducted according to the well defined procedure, which however have to be concretized and grounded to the given situation. This presents a good planning under policy imposed constrained. Other participants in this application include BBN Technologies, SPAWAR, AFRL, and Carnegie Mellon University.

The fictitious scenario, which is an extension of the well-know collation agent experiment CoAX[2], 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. Next is an investigation of the facilities available to rescue the airman. There are different possibilities: a US ship-borne helicopter; a Gaoan helicopter from a land base in Binni; a patrol boat from off the Arabello coastline, etc. 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.

Selection of these resources is constrained by different policies originated from different partners of the coalition. If for instance a hospital in Arabello is best placed to provide the facilities, due to the fact that it has the necessary treatment facilities, choices of rescues resources are then restricted. There is a coalition policy that no Gaoan helicopters may be used by coalition members to transport injured airmen.

In addition to IHMC's KAoS, the CoSAR-TS application relies on a variety of I-X technologies from AIAI. I-X Process Panels (*http://i-x.info*; [13; 14]) provide task support by reasoning about and exchanging with other agents and services any combination of Issues, Activities, Constraints and Annotations (elements of 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 a 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.

[1] *http://www.aiai.ed.ac.uk/project/cosar-ts/*

[2] *http://www.aiai.ed.ac.uk/project/coax/*

Figure 2 shows an I-X Process Panel (I-P^2) and associated I-X Tools. The I-Space tool maintains agent relationships. The relationships can be obtained from agent services such as KAoS. I-X Process Panels can also link to semantic web information and web services, and can be integrated via "I-Q" adaptors [11] to appear in a natural way during planning and in plan execution support.

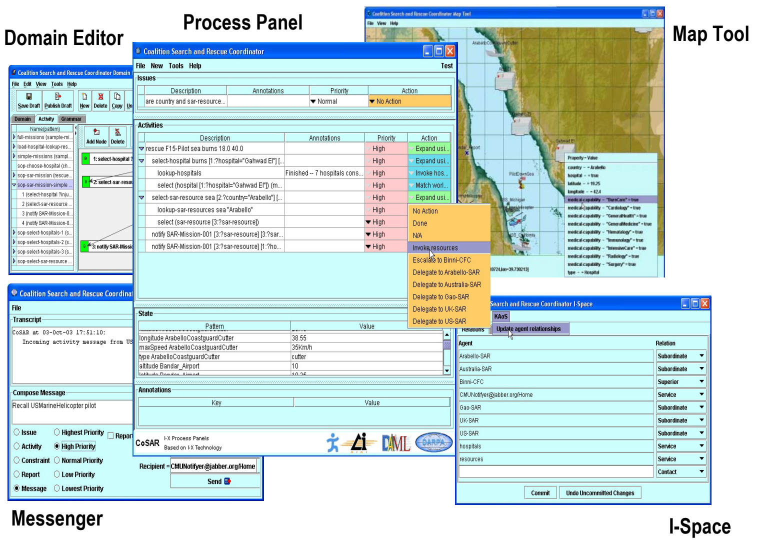


Fig. 2. I-**X Process Panel for a Coalition Search and Rescue Task**

I-X work has concentrated on dynamically determined workflows at execution time, using knowledge of services, other agent availability, and so on. However, it also offers 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.

4.2 Semantic Firewall

Another application area allowing us to validate our approach is the Semantic Firewall (SFW) project, developed in collaboration with University of Southampton, IT Innovation, and SRI International [1][1]. In addition to performing standard policy management functions, the system will take as an input a desired client workflow of Grid Services invocations and verify whether the client is authorized to execute such a workflow in the domain controlled by a given instance of the SFW environment. Additionally the policy system may generate obligations in the form of grid service invocations. These obligations have to be executed during the original workflow; for instance in order to preserving provenance[2] of the calculation results. In effect, the

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

[2] *http://www.pasoa.org/index.html*

initial workflow can be modified and amended with the policies. The resulting policies embedded within the contract governing the transaction will be then enforced by the system as the workflow is enacted.

5. Verification for Policy Compliance in Semantic Web Services Workflow Composition

As a research topic, automatic composition of feasible workflows from a dynamic set of available Semantic Web Services is drawing increasing attention [19]. We argue for applying existing technology and mapping of already developed planners input and output formats to the emerging Semantic Web Services Process Model standard (www.daml.org/services/owl-s/1.0). To this end, we are extending our implementations of I-X and KAoS.

5.1 I-K-C Tool

In the context of CoSAR-TS, we've integrated KAoS and I-X to let I-X obtain information about the role relationships among human and software actors (peers, subordinates, and superiors, for example) represented in domains and stored in KAoS as ontological concepts. I-X can also use the KAoS policy disclosure interface to learn about policy impact on its planned actions. This is the first step toward mutual integration of the planning and policy verification components.

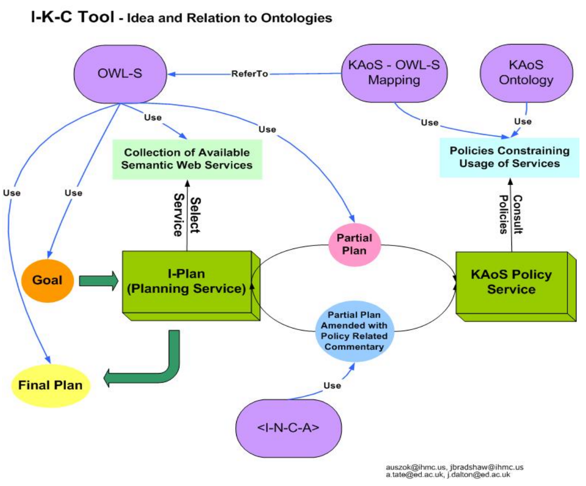

I-K-C Tool - Idea and Relation to Ontologies
OWL-S
ReferTo
KAoS - OWL-S Mapping
KAoS Ontology
Use
Use
Use
Use
Collection of Available Semantic Web Services
Policies Constraining Usage of Services
Use
Use
Select Service
Consult Policies
Partial Plan
Goal
I-Plan (Planning Service)
KAoS Policy Service
Partial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

Fig. 3. Cooperation between I-X and KAoS in the process of semantic workflow composition

The new I-K-C tool goes beyond the initial integration of I-X and KAoS to enable Semantic Web Services workflow composition consistent with policies that govern composition and enactment (see Figure 3). This approach lets I-X import services described in OWL-S into the planner, augmenting any predefined processes already in the process library. KAoS verifies constructed partial plans for policy compliance. We can export the final plan, represented in OWL-S ontology form, and use it in various enactment systems or to guide the dynamic reactive execution of those plans in I-P[2].

5.2 Mapping the OWL-S Representation of Process to the KAoS Concept of Action

The OWL-S concept of *Process* maps semantically to the KAoS concept of Action[1]. Unfortunately, OWL-S made a dramatic change in representing workflow processes in the transitioning from the earlier ontology called DAML-S. In DAML-S, processes were represented as classes whose instances were process executions and whose input and output parameters were defined as properties of those classes. Parameter restrictions were represented as range constraints on those parameter properties. In contrast, OWL-S represents processes as instances, and parameters are defined as instances of the class Parameter or its subclasses Input and Output, with their corresponding parameter restrictions defined by the value of the *process:parameterType* property for each parameter. This significant change does not allow for a straightforward mapping between OWL-S and KAoS concepts using *owl:equivalentClass* and *owl:equivalentProperty* as it had been previously possible in the case of DAML-S. OWL-S will define process executions as instances of a *ProcessInstance* class that refers to its process type. This approach is similar to that taken in the Process Specification Language (PSL) [20].

In order to use KAoS reasoning capabilities it is now necessary to create an OWL class based on the OWL-S process definition instance. This is done by changing the *process:parameterType* mentioned above to represent the appropriate restrictions. We are using OWL-S API[2] to load OWL-S process workflows, to find all processes within a workflow, and then to get detailed definitions in order to build, using Jena[1], the corresponding OWL class which is a subclass of the KAoS Action class.

The change in the representation of the process from DAML-S to OWL-S has other consequences:

- You can't build process hierarchies at different abstraction levels using *rdfs:subClassOf*, while you can in the KAoS ontology of actions.
- You can't represent an actual instance of a process—a very concrete realization of the process. Again, in KAoS we use the instance of an action to describe the currently enacted event and then to find whether policies exist that apply to this situation. The envisioned process control ontology, announced as part of OWL-S's future release, will clearly need methods to represent actual events and their relation to processes.
- The process instance doesn't represent the actual event anymore, so the fact that the process in OWL-S is a subclass of *time-entry:IntervalEvent* carried over from DAML-S is a self-contradiction. (OWL-S's developers have promised to resolve this issue in the near future.)

In short, the change of representation of processes between DAML-S and OWL-S was motivated by difficulties related to usage of classes of processes in collections and other issues. However, addressing this problem has created the challenges in the representation of policies in KAoS mentioned above. We hope that the promised improvements in future versions of OWL-S will help to address these issues.

[1] *http://ontology.ihmc.us/Action.owl*
[2] *http://www.mindswap.org/2004/owl-s/api/*
[1] *http://jena.sourceforge.net/*

5.3. KAoS Capabilities for Analyzing Action Classes

After KAoS extracts a particular action from the workflow and converts it to a corresponding action class, we examine the action to determine its compliance with the relevant policies in force. The process of workflow policy compliance checking differs from that of checking authorization and obligations of an action instance in policy enforcement that we described earlier. In workflow policy compliance checking, we're not dealing with an action instance but an action class. So, we must use subsumption reasoning instead of classification reasoning - KAoS must find relations between the current action class and action classes associated with policies. Fortunately, we use this kind of reasoning to perform policy analyses such as policy deconfliction.8 These analyses also involve discovering relations (subsumption or disjointness, for example) between action classes associated with policies.

Such analyses will often lead to deterministic conclusions - for example, that a given process will be authorized or forbidden or that it will definitely generate an obligation. Results will always be deterministic if the given action class representing the investigated process is a subclass of either a single policy action class or a union of some policy action classes, respectively representing either authorization or obligation policies.

Sometimes, however, the analyses can be nondeterministic—that is, we might be able to conclude only that a given process instance could possibly be authorized or that it might generate obligations. This kind of result will occur if the given action class, representing the process in question, is neither fully subsumed nor fully disjoint, with a single policy action class or their unions respectively representing either authorization or obligation policies. In this case, KAoS can build a representation of the action class (either the class that corresponds to the portion of the action class in the authorization request or the one that generates a given obligation) by computing the difference between the current action class and the relevant policy action class. The algorithm is identical to the one we have previously described [3] for policy harmonization. However, we're still working out how to generically translate that new class to an OWL-S process instance representation.

We've developed a first cut of additional KAoS ontology components, enabling workflow annotation with the results of the policy analyses we described. The appropriate markup was added to the original OWL-S workflow using the OWL-S API and sent back from KAoS to the I-X planner.

5.4. Example: Planning Rescue Operation under Coalition Policy Constraints

The CoSAR-TS scenario described in section 4 is being used to test the capabilities just described. Each time a new search and rescue situation is undertaken; the SAR coordinator gathers available information about the accident and constructs an appropriate goal for the planner. The goal could, for instance, contain information about the kind of injuries sustained and the approximate location of the victim. The planner begins with the selection of an initial plan template that is best for the given situation. It then builds OWL-S profiles for each of the necessary services and queries the Coalition Matchmaker to learn about OWL-S descriptions of registered search and rescue resources. This results in the first approximation of the plan expressed as the OWL-S Process Model. For instance, if the downed pilot has serious burn injuries, the planner will ask the Matchmaker about which services are offered by the burn injuries treatment unit in each medical care center. Subsequently it will ask for available rescue resources, which can pick-up pilot from the sea and deliver it to the chosen hospital (i.e., Arabello). The best result is selected and the OWL-S Process Model is submitted for verification. During workflow analysis, KAoS determines that there is an obligation policy requiring notification of the coalition commander when the downed pilot is successfully recovered. The appropriate process invoking the Notification Service available in the environment as the Web service is inserted into the model and returned to the planner.

6. Enforcement of Policies during Workflow Enactment

Not every aspect of policy compliance can be checked at planning time. Moreover, sometimes the possibility of buggy or malicious code requires runtime checking of compliance. Thus we have designed KAoS so that the policy service can independently enforce policies during workflow execution. The policies governing both authorization and obligation of clients and servers are stored in KAoS and checked by authorized parties. Whereas other 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 can automatically enforce any sort of policy through integration of Semantic Web Services with KAoS enforcers. These enforcers intercept requests to a service and consult KAoS about relevant authorizations and obligatiosn. KAoS is able to reason about the entire action performed by the services, not just about security credentials attached to the request. Additionally, KAoS is used to generate obligations created during use of the services, not just up front during initial service invocation.

6.1 Matchmaker Policy Enforcement – CoSAR-TS scenario

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 always 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.

6.2 Generic Semantic Web Service Enforcer

We have defined enforcers that intercept SOAP messages from the Matchmaker and filter results consistent with coalition policies. In our CoSAR-TS demonstration, these policies prevent the use of Gaoan resources.

Our implementation of a SOAP-enabled enforcer is capable of understanding arbitrary Semantic Web Service invocations so it can apply appropriate authorization policies to them. Additionally, it is equipped with a mechanism to perform obligation policies, which will be in the form of other Web Service invocations. For instance, an obligation policy may require the recording of certain kinds of service transactions through a logging service.

7. Conclusions

KAoS provides necessary capabilities to verify and enforce user-defined policy in the automatic process of planning and executing workflows of semantically described processes in the area of processes building such workflows. Future work will investigate how to take a context surrounding the process (i.e., processes and control constructs) in a given workflow into account.
Currently, KAoS is able to analyze OWL-S encoded workflows, however it can be extended to understand other form of descriptions (e.g., WSMO (Web Service Modeling Ontology)[1]) that share similar concepts of basic process and workflow composition abstractions.

Acknowledgements

This material is based on research sponsored by the Defense Advanced Research Projects Agency (DARPA) CoABS, DAML, and Ultra*Log programs the US Air Force Research Laboratory under agreement numbers F30602-00-2-0577 and F30602-03-2-0014, by NASA IS and X-Enterprise grants, by ONR, and the Army Research Labs. 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. Thanks to the other members of the KAoS project team: Maggie Breedy, Larry Bunch, Matthew Johnson, Hyuckchul Jung, Shri Kulkarni, James Lott, William Taysom, and Gianluca Tonti. We are also grateful for the contributions of Mark Burstein, Pat Hayes, Luc Moreau, Grit Denker, Darren Marvin, Mike Surridge, Ron Ashri, Terry Payne, Katia Sycara, Massimo Paolucci, Naveen Srinivasan, Niranjan Suri, Paul Feltovich, Richard Fikes, Jessica Jenkins, Bill Millar, Deborah McGuinness, Rich Feiertag, Timothy Redmond, Rebecca Montanari, Sue Rho, Ken Ford, Mark Greaves, Jack Hansen, James Allen, Lars Olson, and Robert Hoffman.

[1] *http://www.wsmo.org/*

References

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

[2] Bradshaw, J. M., Beautement, M. Breedy, L. Bunch, S. Drakunov, P. Feltovich, P., Raj, A., Johnson, M., Kulkarni, S., Suri, N. & A. Uszok (2004). 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. 361-400). 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, 20 October 2000.

[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.), *The Semantic Web—ISWC 2003. Proceedings of the Second International Semantic Web Conference, Sanibel Island, Florida, USA, October 2003, LNCS 2870.* (pp. 335-350). Berlin: Springer.

[6] Feltovich, P., Bradshaw, J. M., Jeffers, R., Suri, N., & Uszok, A. (2004). Social order and adaptability in animal and human cultures as an analogue for agent communities: Toward a policy-based approach. In *Engineering Societies in the Agents World IV.* (pp. 21-48). Berlin, Germany: Springer-Verlag.

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

[8] 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).: Springer.

[9] 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.

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

[11] 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,

[12] Seamons, K. E., Winslett, M., Yu, T., Smith, B., Child, E., Jacobson, J., Mills, H., & Yu, L. (2002). Requirements for policy languages for trust negotiation. *Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks (POLICY 2002)*. Monterey, CA,

[13] 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.

[14] Tate, A., Dalton, J., Siebra, C., Aitken, S., Bradshaw, J.M. and Uszok, A. (2004) Intelligent Agents for Coalition Search and Rescue Task Support, AAAI-2004 Intelligent Systems Demonstrator, in Proceedings of the Nineteenth National Conference of the American Association of Artificial Intelligence, (AAAI-2004), San Jose, California, USA, July 2004.,

[15] 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 (Ed.), *The Semantic Web—ISWC 2003. Proceedings of the Second International Semantic Web Conference, Sanibel Island, USA, 2003, LNCS 2870.* (pp. 419-437). Berlin: Springer.

[16] Uszok, A., Bradshaw, J. M., Hayes, P., Jeffers, R., Johnson, M., Kulkarni, S., Breedy, M. R., Lott, J., & Bunch, L. (2003). DAML reality check: A case study of KAoS domain and policy services. *Submitted to the International Semantic Web Conference (ISWC 03).* Sanibel Island, Florida,

[17] 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,

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

[19] 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.), *The Semantic Web—ISWC 2003. Proceedings of the Second International Semantic Web Conference, Sanibel Island, Florida, USA, October 2003, LNCS 2870.* (pp. 195-210). Berlin: Springer.

[20] Schlenoff, C., Gruninger M., Tissot, F., Valois, J., Lubell, J., Lee, J. (2000). The Process Specification Language (PSL): Overview and Version 1.0 Specification," NISTIR 6459, *National Institute of Standards and Technology*, Gaithersburg, MD.