KAoS Policy Management for Semantic Web Services

Andrzej Uszok, Jeffrey M. Bradshaw, Matthew Johnson, and Renia Jeffers, *Institute for Human and Machine Cognition*

Austin Tate, Jeff Dalton, and Stuart Aitken, *University of Edinburgh*

KAoS policy and domain services help with policy specification, analysis, disclosure, and enforcement for Semantic Web Services. The authors describe these capabilities in the context of three applications: Grid Policy Management, Coalition Search and Rescue, and the Semantic Firewall.

Despite rapid advances in Web Services, the user community's demanding requirements continue to outstrip available technology solutions. To help close this gap, Semantic Web Services advocates are defining and implementing many new and significant capabilities (www.swsi.org). These new capabilities should more fully harness Web Services' power through explicit representations of Web resources' underlying semantics and the development of an intelligent Web infrastructure that can fully exploit them. Semantic Web languages, such as OWL, extend RDF to let users specify ontologies comprising taxonomies of classes and inference rules.

Both people and software agents can effectively use Semantic Web Services.[1] 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 we'll describe—will rely on policy-based management and control mechanisms to ensure respect for human-imposed constraints on agent interaction. Policy-based controls of Semantic Web Services can also help govern interaction with traditional (nonagent) clients.

In the mid 1990s, we began to define the initial version of KAoS, a set of platform-independent services that let people define policies ensuring adequate predictability and controllability of both agents and traditional distributed systems. With various research partners, we're also developing and evaluating a generic model of human-agent teamwork that includes policies to assure natural and effective interaction in mixed teams of people and agents—both software and robotic.[2–4] We're exploiting the power of Semantic Web representations to address some of the challenges currently limiting Semantic Web Services' widespread deployment.

KAoS policy and domain management services

KAoS is one of the first efforts for representing policy using a Semantic Web language—in this case, OWL (see the sidebar on page 36 for more about policies and the Semantic Web). Gianluca Tonti and his colleagues have compared two semantically rich policy representations, KAoS and Rei,[5] and a more traditional policy language, Ponder.[6] KAoS services and tools allow for the specification, management, conflict resolution, and enforcement of policies in the specific contexts established by complex organizational structures represented as domains.[2,7–9]

Ontological representation of KAoS policies

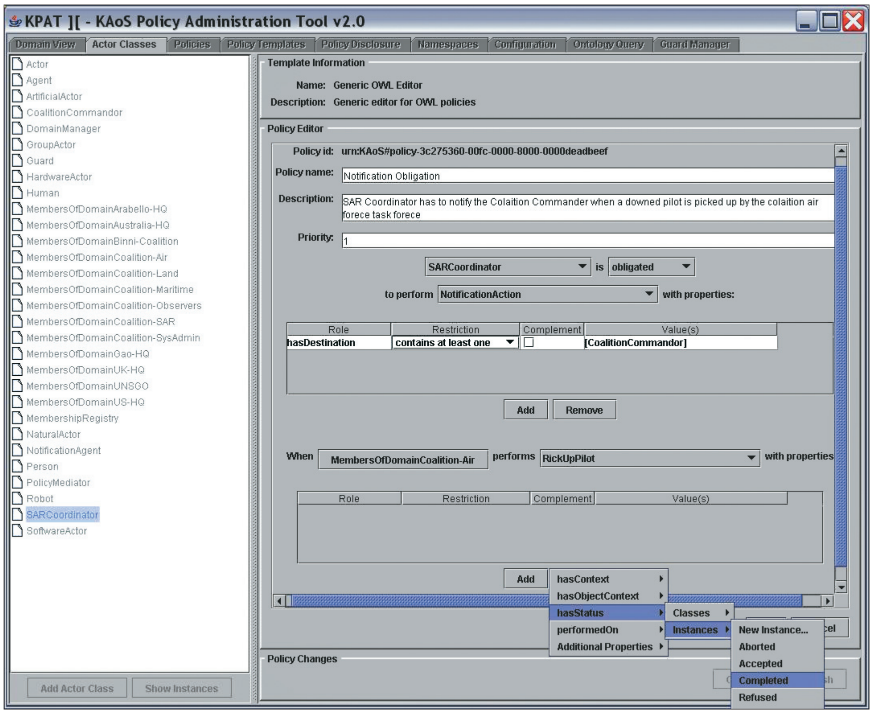
KAoS uses ontology concepts (encoded in OWL) to build policies. During its bootstrap, KAoS first loads the core KAoS Policy Ontology defining concepts used to describe a generic actor's environment and policies in this context (http://ontology.ihmc.us). Then, KAoS loads additional ontologies on top of this, extending concepts from the core ontology, with notions specific to the particular controlled environment and application domain.

The KAoS Policy Service distinguishes between positive and negative *authorizations* (constraints that permit or forbid some action) and positive and negative *obligations* (constraints that require some action when a state- or event-based trigger occurs or that serve to waive such a requirement).[2,6] Other policy constructs (for example, delegation or role-based authorization) are built from the basic domain primitives plus the four policy types.

KAoS policy's OWL definition (see Figure 1) is an instance of one of these four basic policy classes: 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. 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 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 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've implemented in JTP.



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

Policy id: urn:KAoS#policy-3c275360-00fc-0000-8000-0000deadbeef
Policy name: Notification Obligation
Description: SAR Coordinator has to notify the Colaition Commander when a downed pilot is picked up by the colaition air forece task forece
Priority: 1

SARCoordinator is obligated
to perform NotificationAction with properties:

Role	Restriction	Complement	Value(s)
hasDestination	contains at least one	☐	[CoalitionCommandor]

When MembersOfDomainCoalition-Air performs RickUpPilot with properties

Role	Restriction	Complement	Value(s)

Figure 1. The graphical interface of the OWL policy editor and administration tool, the KAoS Policy Administration Tool.

KAoS features

KAoS has several important features.

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 independently of KAoS. As the Semantic Web becomes more widely deployed, this will ease integration with an increasingly sophisticated range of new OWL tools and language enhancements.

Maturity. Over the past few years, we've used KAoS services in conjunction with a wide range of applications and operating platforms.

Comprehensiveness. Unlike many approaches that deal with only simple forms of access control or authorization, KAoS supports both authorization and obligation policies. Additionally, we've implemented a complete infrastructure for policy management, including a full range of capabilities from sophisticated user interfaces for policy specification and analysis to a generic policy disclosure mechanism. We're further developing facilities for policy enforcement automation (that is, automatic code generation for enforcers).

Pluggability. A platform-specific and application-specific ontology is easily loaded on top of the core concepts. Moreover, we've straightforwardly adapted the policy enforcement elements to a wide range of computing environments, both traditional distributed-computing platforms (for example, Web Services, grid computing, and CORBA) and various software and robotic agent platforms (for example, Nomads, Brahms, SFX, CoABS Grid, and 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

Policies and Semantic Web Services

Policies are a means to dynamically constrain and regulate a system's behavior without changing code or requiring the cooperation of the components being governed. They are becoming an increasingly popular approach to dynamic adjustability of applications in academia and industry (www.policy-workshop.org). Policy-based approaches have many benefits, 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.[1] Policies have important analogs in animal societies and human cultures.[2]

Policy-based network and distributed-system management has been researched extensively over the last decade (www-dse.doc.ic.ac.uk/research/policies).[3] Policies are often applied to automate network administration tasks, such as configuration, security, recovery, or quality of service. In network management, policies govern choices in a network's behavior. Additionally, standardization efforts are heading toward common policy information models and frameworks. The Internet Engineering Task Force, for instance, has been investigating policies as a way to manage IP-multiservice networks by focusing on the specification of protocols and object-oriented models for representing policies (www.ietf.org/html.charters/policy-charter.html).

Increasingly, the scope of policy management is 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, disruption-tolerant networks, and pervasive and mobile systems
- Organizational modeling, coalition formation, and formalizing cross-organizational agreements
- Trust models, trust management, provenance, and information pedigrees
- Effective human–machine interaction: interruption and notification management, presence management, adjustable autonomy, mixed-initiative interaction, teamwork facilitation, and safety
- Support for humans trying to retrieve, understand, and analyze all policies relevant to some situation

Researchers have proposed multiple approaches for policy specification. They range from formal policy languages that a computer can easily and directly process and interpret, to rule-based policy notation using an if-then-else format, to the representation of policies as entries in a table with multiple attributes.

In Web Services, standards for SOAP-based message security (for example, www-106.ibm.com/developerworks/webservices/library/ws-secure) and XML-based languages for access control (for example, the eXtensible Access Control Markup Language, www.oasis-open.org/committees/tc_home.php?wg_abbrev=security) have begun to appear. However, the current tools' immaturity along with the new languages' limited scope and semantics make them less-than-ideal candidates for the sophisticated applications that Semantic Web visionaries have imagined for the next decade.[4,5]

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 (for example, current work on general "untell" mechanisms) and advances in computer hardware will continue to improve this performance.

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 in which 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're equipping 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 (www.daml.org/2003/11/swrl/swrl.owl). 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

Using XML as a standard for policy expression has advantages and disadvantages. The major advantage is its straightforward extensibility (a feature shared with languages such as RDF and OWL, which use XML as a foundation). The problem with mere XML is that its semantics are mostly implicit—meaning is conveyed on the basis of a shared understanding derived from human consensus. Implicit semantics are ambiguous, they promote fragmentation into incompatible representation variations, and they require extra manual work that a richer representation could eliminate. However, if an implementation requires the use of an XML approach, you could map Semantic Web-based policy representations, such as those we describe, to these lower-level representations by applying contextual information.

In addition to KAoS, some initial efforts in using Semantic Web representations for basic security applications (authentication, access control, data integrity, and encryption) of policy are bearing fruit. For example, Grit Denker and her colleagues have integrated a set of ontologies (credentials and security mechanisms) and security extensions for OWL-S Service profiles with Carnegie Mellon University's Semantic Matchmaker[5] to enable security brokering between agents and services. Future work will let security services be composed with other services. Lalana Kagal and her colleagues are developing Rei, a Semantic Web language-based policy language they're using as part of the OWL-S service profiles extension and other applications.[6] In another promising direction, Ninguhi Li, Benjamin Grosof, and Joan Feigenbaum have developed a logic-based approach to distributed authorization in large-scale, open, distributed systems.[7]

References

1. J.M. Bradshaw et al., "Making Agents Acceptable to People," *Intelligent Technologies for Information Analysis: Advances in Agents, Data Mining, and Statistical Learning*, N. Zhong and J. Liu, eds., Springer-Verlag, 2004, pp. 355–400.
2. P. Feltovich et al., "Social Order and Adaptability in Animal, Human, and Agent Communities," *Proc. 4th Int'l Workshop Eng. Societies in the Agents World,* LNAI 3071, Springer-Verlag, 2003, pp. 73–85.
3. S. Wright, R. Chadha, and G. Lapiotis, eds., *IEEE Network*, special issue on policy-based networking, vol. 16, no. 2, 2002, pp. 8–56.
4. D. Fensel et al., *Spinning the Semantic Web,* MIT Press, 2003.
5. K. Sycara et al., "Automated Discovery, Interaction and Composition of Semantic Web Services," *J. Web Semantics*, vol. 1, no. 1, 2003, pp. 27–46.
6. L. Kagal, T. Finin, and A. Joshi, "A Policy-Based Approach to Security for the Semantic Web," *Proc. 2nd Int'l Semantic Web Conf.* (ISWC 2003), LNCS 2870, Springer-Verlag, 2003, pp. 402–418.
7. N. Li, B.N. Grosof, and J. Feigenbaum, "Delegation Logic: A Logic-Based Approach to Distributed Authorization," *ACM Trans. Information Systems Security* (TISSEC), vol. 6 , no. 1, 2003, pp. 128–171.

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.

Example application contexts

We're extending KAoS to address requirements of Semantic Web Services. We describe three examples of the kinds of roles that a policy management framework can play in providing

- Policy management for grid-computing environments
- Verification of policy compliance for Semantic Web Services workflow composition
- Policy enforcement during workflow enactment

Policy management for grid computing

Our first foray into Web Services was developing an initial OGSA-compliant (Open Grid Services Architecture) version of KAoS services, allowing fine-grained policy-based management of registered grid-computing services on the Globus platform.[12] OGSA is a Web Services-compatible standard for defining grid-computing services (www.globus.org/ogsa). We've aimed to extend and generalize this capability 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 needs 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 Grid Security Infrastructure-enabled grid services. GSI was the only component of the Globus Toolkit (GT3) we used in the integration. The interface itself is a grid service, which we called a KAoS grid service. It gives grid clients and services the ability to register with KAoS services and to check whether a given action is authorized on the basis of current policies. Figure 2 shows the basic architecture.

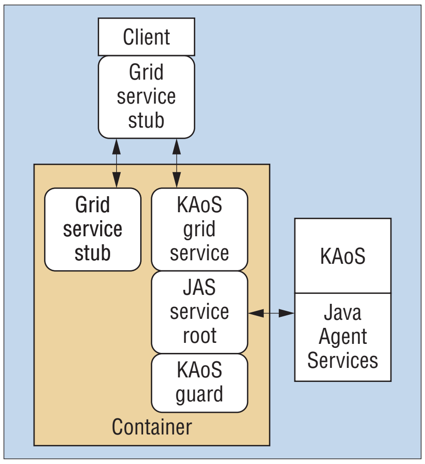


Figure 2. The KAoS grid service architecture.

Creating a KAoS grid service. To create a KAoS grid service, we used tools provided with GT3 to create a normal grid service, then added to it the required KAoS framework components to make it KAoS aware. This framework links grid services to the KAoS-implemented JAS (Java Agent Services, http://sourceforge.net/projects/jas): naming, message transport, and directory. It

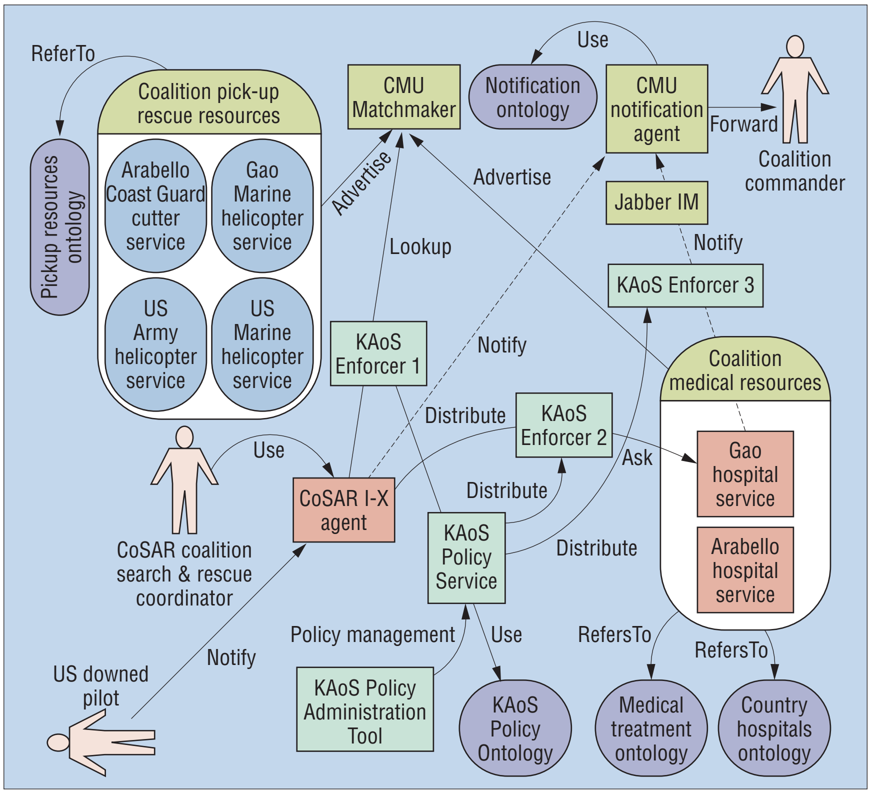

ReferTo
Pickup resources ontology
Coalition pick-up rescue resources
Arabello Coast Guard cutter service
Gao Marine helicopter service
US Army helicopter service
US Marine helicopter service
Advertise
CMU Matchmaker
Lookup
Notification ontology
Use
CMU notification agent
Forward
Coalition commander
Advertise
Jabber IM
Notify
KAoS Enforcer 3
KAoS Enforcer 1
Notify
Coalition medical resources
Gao hospital service
Arabello hospital service
Distribute
KAoS Enforcer 2
Ask
Use
CoSAR coalition search & rescue coordinator
CoSAR I-X agent
Distribute
KAoS Policy Service
Distribute
Policy management
Use
RefersTo
RefersTo
US downed pilot
Notify
KAoS Policy Administration Tool
KAoS Policy Ontology
Medical treatment ontology
Country hospitals ontology

Figure 3. A CoSAR-TS demo diagram.

also associates the KAoS Guard (policy decision point) with the KAoS grid service that Figure 2 shows.

Registration. To use domain services, we needed to establish a method for clients and resources to register in a KAoS domain. The clients or resources use their credential to request registration in one or more domains. The credential is a standard X.509 certificate that Globus uses for authentication and is verified using the GT Grid Security Infrastructure. If the certificate is valid, the registration request goes to KAoS for registration into the desired domains. If the resource uses an application-specific ontology to describe its capabilities, it must be loaded into the KAoS ontology using a utility KAoS provides. Inside the KAoS grid service, the registration is handled through the associated guard. This lets KAoS distribute all applicable policies to the appropriate guard.

Expressing policies. The basic components of any authorization policy are the actors, action, and context. A sample policy would read

> It is permitted for actor(s) X to perform action(s) Y in context 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 in JTP, offline or at runtime, letting policies be written about the client or its domain.

The system can represent actions 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 don't provide varying access levels. For example, a policy defining overall permissions for a chat service might use 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.

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

More complex services might require new concepts in the ontologies that map to specific actions on a grid service. For example, Reliable File Transfer Service has various methods that might not map to an existing ontology. To provide fine-grained control of this service, we can extend the KAoS ontology for the specific domain space and load it into KAoS using KPAT (KAoS Policy Administration Tool), a graphical user interface for interacting with KAoS (see Figure 1 for an example of a KPAT window). We're working on a tool to automatically generate an OWL ontology for a given WSDL (Web Services Description Language, www.w3.org/TR/wsdl) specification of the OGSI-compliant grid service.

The context can refer to objects that are the targets of action, such as clients or services, or domain-specific entities, such as different computing resources. The context might also include other information about the situation in which the action is performed. Clients and services are added to the KAoS ontology when they register in the KAoS Directory Service, while using domain-specific entities requires extensions to the ontology, either before loading into KAoS or using the graphical interface in KPAT.

Policies might 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. Recently, we've added the capability of defining simple obligation policies.

Checking authorization. Because 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 on the basis of KAoS policies. As policies are added to KAoS through KPAT, they're 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 on the basis of current policies by querying the guard. If the guard allows the requested action, the 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 the resource provided at registration and maps to the local control mechanism. The KAoS grid service also sets the proxy lifetime and signs it. It then returns the restricted proxy certificate 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 the KAoS service embedded in the certificate. This lets the local resource owner write policies restricting the rights it lets KAoS delegate.

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

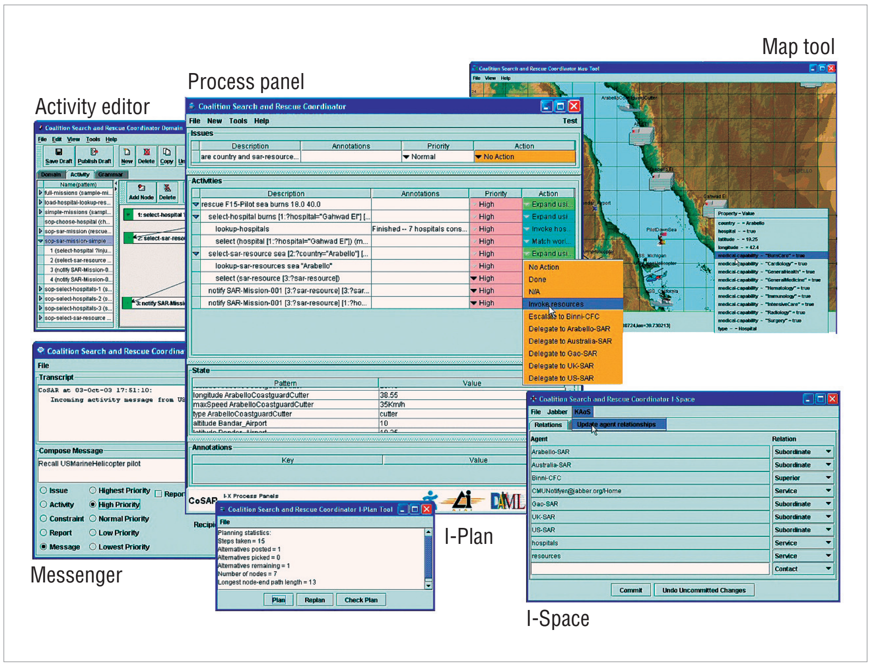


Figure 4. I-X process panel and task support tools.

Coalition search and rescue

In the Coalition Search and Rescue Task Support project (CoSAR-TS, www.aiai.ed.ac.uk/project/cosar-ts), we're testing the integration of KAoS and the Artificial Intelligence Applications Institute (AIAI)'s I-X technology with Semantic Web Services. Other participants in the CoSAR-TS project include BBN Technologies, the Space and Naval Warfare Systems Command, the Air Force Research Laboratory, and Carnegie Mellon University.

Search and rescue operations, especially coalition-based ones, require rapid dynamic composition of available policy-constrained heterogeneous resources. A good use case is to describe them using Semantic Web technologies. Additionally, military operations are usually conducted according to some well-defined procedure that must be made concrete and grounded to the given situation. Such a scenario is good for illustrating planning under policy-imposed constraints.

The fictitious scenario (see Figure 3), which extends the well-known Coalition Agent Experiment (CoAX, www.aiai.ed.ac.uk/project/coax), 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, we assume that excellent location knowledge is available and that no local threats exist 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 for rescuing the airman. Different possibilities exist: a US shipborne helicopter, a Gaoan helicopter from a land base in Binni, a patrol boat off the Arabello coastline, and so on. Finally, there's a process to establish available medical facilities for the specialized injury reported, using the information provided about the region's countries.

Different policies originate from different coalition partners, which constrains selection of these resources. If, for instance, a hospital in Arabello has the best treatment facilities for dealing with the airman's injuries, choices of rescue resources are then restricted. Additionally, the coalition has a policy prohibiting members from using Gaoan helicopters to transport injured airmen.

In addition to relying on KAoS, CoSAR-TS relies on various I-X technologies from the 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 such activity's results. This information can be exchanged with other tools using OWL, RDF, or other languages. The system includes a planner that can compose a suitable plan for the given tasks when it receives a library of standard operating procedures or processes and knowledge of other agents or services that it can use.

Figure 4 shows an I-X process panel (I-P^2) and associated I-X tools. The I-Space tool maintains agent relationships. I-X can obtain the relationships 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 naturally during planning and in plan execution support.

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.

The Semantic Firewall

We developed the Semantic Firewall (SFW) project in collaboration with the University of Southampton, IT Innovation, and

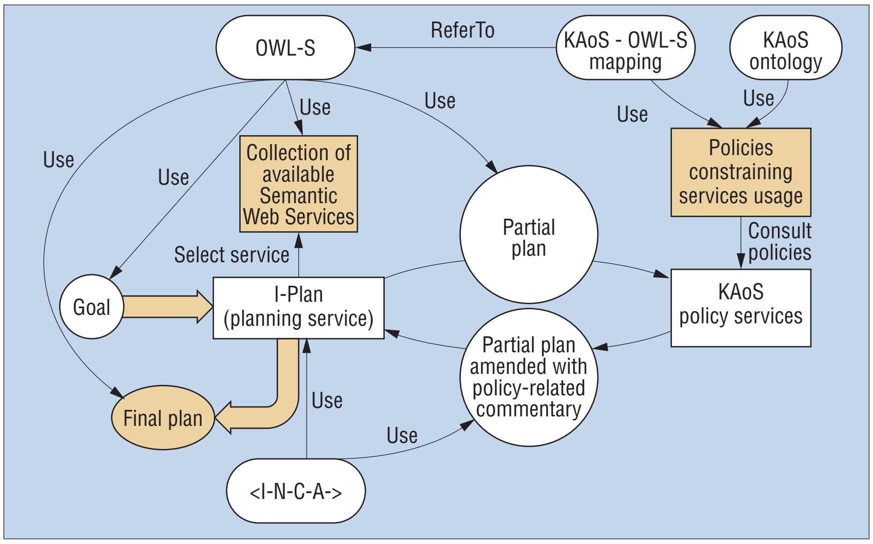

OWL-S
ReferTo
KAoS - OWL-S mapping
KAoS ontology
Use
Use
Use
Use
Use
Use
Collection of available Semantic Web Services
Policies constraining services usage
Partial plan
Consult policies
Select service
Goal
I-Plan (planning service)
KAoS policy services
Partial plan amended with policy-related commentary
Final plan
Use
Use
<I-N-C-A->

Figure 5. Cooperation between I-X and KAoS in the process of semantic workflow composition.

SRI International.[10] (See http://ontology.ihmc.us/SemanticServices/S-F/Example/index.html for an example scenario with policies encoded using the KAoS Policy syntax.) In addition to performing standard policy management functions, KAoS will take as an input a desired client workflow of grid services invocations. It will then 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 might generate obligations in the form of grid service invocations. These obligations must execute during the original workflow—for example, to preserve provenance (www.pasoa.org/index.html) of the calculation results. In effect, we can modify and amend the initial workflow with the policies. The system will then enforce resulting policies embedded in the contract governing the transaction as the workflow is enacted.

Policy compliance verification 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. We argue for applying existing technology and mapping already developed planners' input and output formats to the emerging Semantic Web Services Process Model standard (www.daml.org/services/owl-s/1.0).[15] To this end, we are extending our implementations of I-X and KAoS.

The 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.

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 5). 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.

Mapping process to action

The OWL-S concept of process maps semantically to the KAoS concept of action (http://ontology.ihmc.us/Action.owl). Unfortunately, OWL-S dramatically changed how it represents workflow processes compared with DAML-S. DAML-S represented processes as classes whose instances were process executions and whose input and output parameters were defined as properties of those classes. It represented parameter restrictions as range constraints on those parameter properties. In contrast, OWL-S represents processes as instances and defines parameters as instances of the class Parameter or its subclasses Input and Output. Processes' parameter restrictions are defined by the value of the process:parameterType property for each parameter. This significant change doesn't allow for a straightforward mapping between OWL-S and KAoS concepts using owl:equivalentClass and owl:equivalentProperty, which had been previously possible in the case of DAML-S. In the near future, the OWL-S definition of process executions should change again and will be defined as an instance of a ProcessInstance class that refers to the process type. This approach is similar to that taken in the Process Specification Language.[16]

To use KAoS reasoning capabilities, we must create an OWL class based on the OWL-S process definition instance. We do this by changing the process:parameterType to represent the appropriate restrictions. We're using the OWL-S API (www.mindswap.org/2004/owl-s/api) to load OWL-S process workflows, find all processes in a workflow, and then get detailed definitions to build the corresponding OWL class—a subclass of the KAoS Action class—using Jena (http://jena.sourceforge.net).

The change 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, difficulties related to using process classes in collections and other issues motivated changing the representation of processes between DAML-S and OWL-S. However, addressing this problem has created the challenges in the representation of policies in KAoS we just mentioned. We hope that the promised improvements in future versions of OWL-S will help resolve these issues.

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, which 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.[6] 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 previously described[7] 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.

KAoS can't check every aspect of policy compliance at planning time. So, we've designed it so that the policy service can independently enforce policies during workflow execution.

Example: Planning a rescue operation under coalition policy constraints

Project participants are using the CoSAR-TS scenario to test the capabilities we just described. With each new search-and-rescue situation, 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 the victim sustained and his or her approximate location. The planner begins by selecting the best initial plan template for the given situation. It then builds OWL-S profiles for each necessary service and queries the Coalition Matchmaker to learn about OWL-S descriptions of registered SAR 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 which services are offered by the burn injuries treatment unit in each medical care center. Subsequently, it will ask for available rescue resources that can pick up a pilot from the sea and deliver him or her to the chosen hospital (that is, Arabello). The system selects the best result and submits the OWL-S process model for verification. During workflow analysis, KAoS determines that an obligation policy exists requiring that the coalition commander receive notification when the downed pilot is successfully recovered. It inserts the appropriate process, invoking the notification service—available in the environment as the Web Service—into the model, and returns it to the planner.

Policy enforcement during workflow enactment

KAoS can't check every aspect of policy compliance at planning time. So, we've designed it 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. Other approaches to securing Semantic Web Services are limited. They can mark service advertisements with requirements for authentication and communication and enforce compliance with these requirements,[17] or they attach conditions to inputs, outputs, and effects of services. KAoS, on the other hand, can automatically enforce any sort of policy by integrating Semantic Web Services with KAoS enforcers, components that intercept requests to a service and consult KAoS about their authorization and obligation. KAoS can reason about the entire action the services perform, not just about security credentials attached to the request. Additionally, KAoS helps generate obligations created when the services are used. We describe three applications of KAoS during workflow enactment: Matchmaker policy enforcement, a generic Semantic Web Service enforcer, and agreements and contracts in the Semantic Firewall application.

Matchmaker policy enforcement—CoSAR-TS scenario

Although annotating the Semantic Matchmaker service profiles lets registered service providers describe required security profiles,[15] it doesn't let owners of infrastructure resources (for example, computers or networks), client organizations (coalition organizations or national interest groups), or individuals specify or enforce policy from their unique perspectives. For example, the policy that coalition members can't use Gaoan transports can't always be anticipated and specified in the Matchmaker service profile. Neither would Matchmaker service profile annotations be an adequate implementation for a US policy obligating encryption, prioritizing network bandwidth allocation, or requiring the logging of certain sorts of messages.

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

A 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.

We're enhancing the SOAP-enabled enforcers to understand arbitrary Semantic Web Service invocations so that they can apply appropriate authorization policies to them. The enforcer is equipped with a mechanism to perform obligation policies, which are in the form of other Web Service invocations. For instance, you can imagine that some policy might require consultation or registration of performed transactions in some logging service available as a Web Service audit entity.

Agreements and contracts—a Semantic Firewall application

A necessary requirement for supporting complex, dynamic groups of service providers in a business context is the notion of a contract. Although 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, Benjamin Grosof and Terrence Poon have developed SweetDeal, a rule-based approach to automating "law in the small."[18] SweetDeal represents business contracts to let software agents 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 the representation and reasoning constructs necessary to allow for creating and executing agreements and contracts. As part of contract creation, KAoS can already detect policy conflicts and suggest harmonization. We're extending these and combining them with new facilities for negotiation and extensions of existing capabilities for enforcement.

Contracts can be stored in instances of KAoS (or perhaps some other interoperable policy service) associated with each Web Service or, when stakeholders prefer, as independent KAoS instances representing neutral third parties.

KAoS provides capabilities for verifying and enforcing user-defined policy when automatically planning and executing semantically described process workflows. To advance this, we plan to investigate how to take a context surrounding the process (that is, processes and control constructs) in a given workflow into account during policy analyses.

Currently, KAoS can analyze OWL-S encoded workflows; however, we can imagine how to extend it to understand other forms of descriptions (for example, the Web Service Modeling Ontology, www.wsmo.org) that share similar concepts of basic process and workflow composition abstractions. ■

Acknowledgments

Our research was sponsored by the DARPA CoABS, DAML, and UltraLog programs at the US Air Force Research Laboratory under agreements F30602-00-2-0577 and F30602-03-2-0014. The US government, the Institute for Human and Machine Cognition, and the University of Edinburgh are authorized to reproduce and distribute reprints and online copies for their purposes notwithstanding any copyright annotation hereon. Thanks to the other members of the KAoS project team: Maggie Breedy, Larry Bunch, 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, Niranjan Suri, Paul Feltovich, Richard Fikes, Jessica Jenkins, Bill Millaar, Deborah McGuinness, Rich Feiertag, Timothy Redmond, Rebecca Montanari, Sue Rho, Ken Ford, Mark Greaves, Jack Hansen, James Allen, Ron Ashri, Terry Payne, Mike Surridge, Darren Marvin, Grit Denker, Kate Keahey, Katia Sycara, Massimo Paolucci, Naveen Srinivasan, and Robert Hoffman.

References

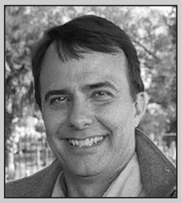
1. S.A. McIlraith, T.C. Son, and H. Zeng, "Semantic Web Services," *IEEE Intelligent Systems*, vol. 16, no. 2, 2001, pp. 46–53.
2. J.M. Bradshaw et al., "Making Agents Acceptable to People," *Intelligent Technologies for Information Analysis: Advances in Agents, Data Mining, and Statistical Learning*, N. Zhong and J. Liu, eds., Springer-Verlag, 2004, pp. 355–400.
3. J.M. Bradshaw et al., "Dimensions of Adjustable Autonomy and Mixed-Initiative Interaction," to be published in *Computational Anatomy*, Springer-Verlag, 2004.
4. G. Klein et al., "Common Ground and Coordination in Joint Activity," *Organizational Simulation*, John Wiley & Sons, 2004.
5. G. Tonti et al., "Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder," *The Semantic Web–ISWC 2003: 2nd Int'l Semantic Web Conf.*, LNCS 2870, Springer-Verlag, 2003, pp. 419–437.
6. N. Damianou et al., *Ponder: A Language for Specifying Security and Management Policies for Distributed Systems,* tech. report Doc2000/1, Dept. of Computing, Imperial College of Science, Technology and Medicine, 20 Oct. 2000.
7. J.M. Bradshaw et al., "Representation and Reasoning for DAML-Based Policy and Domain Services in KAoS and Nomads," *Proc. 2nd Int'l Joint Conf. Autonomous Agents and Multi-Agent Systems* (AAMAS 2003), ACM Press, 2003, pp. 835–842.
8. A. Uszok, J.M. Bradshaw, and R. Jeffers, "KAoS: A Policy and Domain Services Framework for Grid Computing and Grid Computing and Semantic Web Services," *Trust Management: 2nd Int'l Conf. Proc.* (iTrust 2004), LNCS 2995, Springer-Verlag, 2004, pp. 16–26.
9. A. Uszok et al., "KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction, and Enforcement," *Proc. IEEE 4th Int'l Workshop*

Policies for Distributed Systems and Networks, IEEE CS Press, 2003, pp. 93–96.

10. R. Ashri, T.R. Payne, and M. Surridge, “Towards a Semantic Web Security Infrastructure,” *Proc. AAAI Spring Symp. Semantic Web Services*, AAAI Press, 2004, pp. 84–91.
11. S. Potter, A. Tate, and J. Dalton, “I-X Task Support on the Semantic Web,” *Poster Proc. 2nd Int’l Semantic Web Conf.*, 2003; http://i-x.info/documents/2003/2003-iswc-poster-potter-ix.pdf.
12. M. Johnson et al., “KAoS Semantic Policy and Domain Services: An Application of DAML to Web Services-Based Grid Architectures,” *Proc. AAMAS 03 Workshop Web Services and Agent-Based Eng.*, 2003; www.ihmc.us/research/projects/KAoS/OGSAIntegration.pdf.
13. A. Tate, “Coalition Task Support Using I-X and <I-N-C-A>,” *Proc. 3rd Int’l Central and Eastern European Conf. Multi-Agent Systems* (CEEMAS 2003), LNAI 2691, Springer-Verlag, 2003, pp. 7–16.
14. A. Tate, J. Dalton, and S. Potter, “Intelligible Messaging: Activity-Oriented Instant Messaging,” submitted to *Proc. 14th Int’l Conf. Automated Planning and Scheduling* (ICAPS 2004), AAAI Press, 2004; http://i-x.info/documents/2004/2004-x-tate-ime.pdf.
15. D. Wu et al., “Automating DAML-S Web Services Composition Using SHOP2,” *The Semantic Web—ISWC 2003: 2nd Int’l Semantic Web Conf.*, LNCS 2870, Springer-Verlag, pp. 195–210.
16. C. Schlenoff et al., *The Process Specification Language (PSL): Overview and Version 1.0 Specification*, NISTIR 6459, Nat’l Inst. Standards and Technology, 2000.
17. G. Denker et al., “Security for DAML Web Services: Annotation and Matchmaking,” *The Semantic Web—ISWC 2003: 2nd Int’l Semantic Web Conf.*, LNCS 2870, Springer-Verlag, 2003, pp. 335–350.
18. B.N. Grosof and T.C. Poon, “SweetDeal: Representing Agent Contracts with Exceptions using XML Rules, Ontologies, and Process Descriptions,” *Proc. 12th Int’l Conf. World Wide Web*, ACM Press, 2003, pp. 340–349.

The Authors

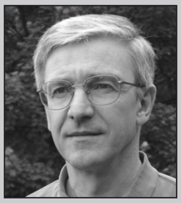
Andrzej Uszok is a research scientist at the Institute for Human and Machine Cognition. His research interests include ontology, policy specification, agent systems, and transparent interoperability. He received his PhD in computer science from the AGH University, Krakow. He is a member of the AAAI. Contact him at the Inst. for Human and Machine Cognition, 40 S. Alcaniz, Pensacola, FL 32502; auszok@ihmc.us.



Jeffrey M. Bradshaw is a senior research scientist at the Institute for Human and Machine Cognition. His research interests include knowledge acquisition, software and robotic agent technology, human–agent teamwork, adjustable autonomy, and mixed-initiative interaction. He received his PhD in cognitive science from the University of Washington. He is a member of the AAAI, the ACM, and the IEEE Computer Society. Contact him at the Inst. for Human and Machine Cognition, 40 S. Alcaniz, Pensacola, FL 32502; jbradshaw@ihmc.us.

Renia Jeffers is a research associate at the Institute for Human and Machine Cognition and a lead designer and developer for the KAoS policy and domain services. Her research interests include software agent technologies; policy specification, management, and enforcement; and human–robotic teamwork. She received her MS in software engineering from Seattle University. Contact her at the Inst. for Human and Machine Cognition, 40 S. Alcaniz, Pensacola, FL 32502; rjeffers@ihmc.us.

Matthew Johnson is a research associate at the Institute for Human and Machine Cognition. His research interests include human–machine interface, teamwork, policy, and mixed-initiative interactions. He received his MS in computer science from Texas A&M University at Corpus Christi. Contact him at the Inst. for Human and Machine Cognition, 40 S. Alcaniz, Pensacola, FL 32502; mjohnson@ihmc.us.



Austin Tate is the technical director of the Artificial Intelligence Applications Institute and holds the Personal Chair of Knowledge-Based Systems at the University of Edinburgh. His research interests include AI planning (O-Plan and I-X/I-Plan), collaboration and process, and Web Services standards activities. He received his PhD in machine intelligence at the University of Edinburgh. He is a fellow of the Royal Society of Edinburgh (Scotland’s National Academy) and the AAAI. Contact him at the Artificial Intelligence Applications Inst., Univ. of Edinburgh, Appleton Tower, Crichton St., Edinburgh EH8 9LE, UK; a.tate@ed.ac.uk.

Jeff Dalton is a research scientist at the University of Edinburgh’s Artificial Intelligence Applications Institute. His research interests include planning, simulation, Web-based software, and programming language design and implementation. He received his BA in mathematics from Dartmouth College. Contact him at the Artificial Intelligence Applications Inst., Univ. of Edinburgh, Appleton Tower, Crichton St., Edinburgh EH8 9LE, UK; jeff@inf.ed.ac.uk.

Stuart Aitken is a member of the University of Edinburgh’s Artificial Intelligence Applications Institute. His research interests include ontology, bioinformatics, intelligent tools for knowledge acquisition, and machine learning. He received his PhD in computer science from the University of Glasgow. Contact him at the Artificial Intelligence Applications Inst., Univ. of Edinburgh, Appleton Tower, Crichton St., Edinburgh EH8 9LE, UK; stuart@aiai.ed.ac.uk.