KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction, and Enforcement

A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, J. Lott
Institute for Human and Machine Cognition (IHMC), Univ. West Florida, 40 S. Alcaniz, Pensacola, FL 32501
{auszok, jbradshaw, rjeffers, nsuri, phayes, mbreedy, lbunch, mjohnson, skulkarni, jlott}@ai.uwf.edu

Abstract

In this paper, we describe our initial implementation of the KAoS policy and domain services. While primarily oriented to the dynamic and complex requirements of software agent applications, the services are also being adapted to general-purpose grid computing and web services environments as well. The KAoS services rely on a DAML description-logic-based ontology of the computational environment, application context, and the policies themselves that enables runtime extensibility and adaptability of the system, as well as the ability to analyze policies relating to entities described at different levels of abstraction.

Keywords: *policy, agent, ontology, DAML, domains, KAoS, description logic, policy conflict resolution.*

1. Introduction

The increased intelligence and autonomy afforded by software agents is both a boon and a danger. This greater autonomy, if unchecked, has the potential of effecting severe damage in the case of buggy or malicious agents. Techniques and tools must be developed to assure that agents will always operate within the bounds of established behavioral constraints and will be continually responsive to human control.

Under DARPA and NASA sponsorship, we have been developing the KAoS [2, 3] policy and domain services to increase the assurance with which agents can be deployed in a wide variety of operational settings. In conjunction with Nomads [7] strong mobility and safe execution features, KAoS services and tools allow for the specification, management, conflict resolution, and enforcement of policies within the specific contexts established by complex organizational structures. While initially oriented to the dynamic and complex requirements of software agent applications, the services are also being adapted to general-purpose grid computing (*http://www.gridforum.org*) and web services (*http://www.w3.org/2002/ws/*) environments as well.

There are some important differences between the objectives of our approach and that of other more typical policy systems. Our approach seeks to enable policy uniformity in domains that might be simultaneously distributed across multiple platforms and execution environments, as long as semantically equivalent monitoring and enforcement mechanisms are available. Second, insofar as possible the framework needs to support dynamic runtime policy changes, and not merely static configurations determined in advance. Third, the framework needs to be extensible to a variety of execution platforms.

2. KAoS policy ontologies

The representation chosen to describe the policies and their context largely determines the flexibility, extensibility, and amenability to analysis of a given implementation. KAoS services rely on a DAML (*http://www.daml.org*) description-logic-based ontology of the computational environment, application context, and the policies themselves [1]. It makes possibly to represent subjects, actions, and situation at multiple levels of abstraction and to dynamically calculate relations between policies and environment entities and other policies based on ontology relations. The use of an ontology also facilitates a dynamic adaptation of the policy framework by specifying the ontology of a given environment and linking it with generic framework ontology. The DAML description logic features enable, using a inferencing engine, reasoning about policy disclosure, conflict detection, and harmonization, as well as about domain structure and concepts exploiting the subsumption and instance classification algorithms.

The current version of KAoS Policy Ontologies (*http://ontology.coginst.uwf.edu/*) defines basic ontologies for actions, actors, groups, places, various entities related to actions (e.g., computing resources), and policies. There are currently 79 classes and 41

properties defined in the basic ontologies. It is expected that for a given application, the ontologies will be further extended with additional classes, individuals, and rules.

The actor ontology distinguishes between people and various classes of software agents that can be the subject of policy. Groups of actors or other entities may be distinguished according to whether the set of members is defined extensionally (i.e., through explicit enumeration in some kind of registry) or intentionally (i.e., by virtue of some common property such as a joint goal that all actors possess or a given place where various entities may be currently located).

The actions ontology defines the subclass relation between action classes as well the *partOf* relation (for composite actions) and the *implementedBy* relation between abstract action classes and operations in the given software environment; defining *grounding* of the ontology concepts.

A policy is a statement enabling or constraining execution of some type of action by one or more actors in relation to various aspects of some situation. In DAML, a policy is represented as an instance of the appropriate policy type (i.e., positive or negative authorization, positive or negative obligation) with associated values for properties: priority, update time stamp and a site of enforcement. The most important property value is, however, the name of a controlled action class. Usually, a new action class is built automatically when a policy is defined. Through various property restrictions, a given policy can be variously scoped, for example, either to individual agents, to agents of a given class or to agents belonging to a particular group, etc. Additionally, action context can be precisely described by restricting values of its properties. The KAoS Policy Administration Tool (KPAT) is a graphical interface hides the complexity of the DAML policy representation from users.

3. Policy analyses

The KAoS Policy Ontologies are intended for a variety of purposes. One obvious application is during inference relating to different forms of online or offline analysis of policies. They are used for a variety of purposes, including policy disclosure management, reasoning about future actions based on knowledge of policies in force, and in assisting users of policy specification tools to understand the implications of defining new policies given the current context and the set of policies already in force.

In the current version of KAoS, changes or additions to policies or a change in status of an actor (e.g., a human administrator being given new permissions; software agent joining a new domain or moving to a new host) requires logical inference to determine first of all which policies are in conflict and second, in an optional step, resolving these conflicts [4]. We have implemented a general-purpose algorithm for policy conflict detection and, if chosen, harmonization.

The three types of conflict that can currently be handled are: positive vs. negative authorization (i.e., being simultaneously permitted and forbidden from performing some action), positive vs. negative obligation (i.e., being both required and not required to perform some action), and positive obligation vs. negative authorization (i.e., being required to perform a forbidden action). The policy conflict detection and harmonization algorithms within KAoS allow policy conflicts to be found and resolved even when the actors, actions, or targets of the policies are specified at very different levels of abstraction. These algorithms rely on JTP (*http://www.ksl.stanford.edu/software/JTP/*) ontology inferencing engine that we have integrated with KAoS.

Policy precedence conditions are needed to properly execute the automatic conflict resolution algorithm. When policy conflicts occur, these conditions are used to determine which of the two policies being compared is most important. The conflict can then be resolved automatically in favor of the most important policy. Alternatively, the conflicts can be brought to the attention of a human administrator who can make the decision manually.

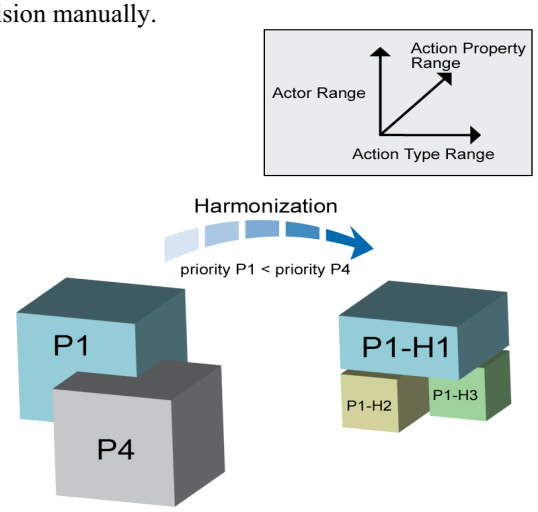

Figure 1. Representation of policy harmonization

The derivation of the newly generated set of harmonized policies (Figure 1), if chosen, can be understood by imagining an intersection of two N-dimensional Cartesian products. If *P1* and *P2* are two Cartesian products defined as:

$$P1 = D11 \times D12 \times \ldots \times D1n$$
$$P4 = D21 \times D22 \times \ldots \times D2n$$

then

$$P1 \backslash P2 = subP1 + subP2 + \ldots + subPn$$

where

$$subPk = (D11 \cap D21) \times \ldots \times (D1(k-1) \cap D2(k-1)) \times (D1k \setminus D2k) \times D1(k+1) \times \ldots \times D1n$$

4. KAoS Policy and Domain Services

KAoS is a collection of componentized services compatible with several popular agent/distributed system frameworks, including Nomads [7], the DARPA CoABS Grid [5], the DARPA ALP/Ultra*Log Cougaar framework (*http://www.cougaar.net*), Brahms [6] and CORBA (*http://www.omg.org*). The adaptability of KAoS is due in large part to its pluggable infrastructure based on Sun's Java Agent Services (JAS) (*http://www.java-agent.org*). For a full description of KAoS, the reader is referred to [2].

Groups of people and computational entities are logically structured into domains and subdomains to facilitate policy administration. Domains may represent any sort of group imaginable, from potentially complex organizational structures to administrative units to dynamic task-oriented teams with continually changing membership. Membership in a given domain can extend across host boundaries and, conversely, multiple domains can exist concurrently on the same host. Domains may be nested indefinitely and, depending on whether policy allows, membership in more than one domain at a time is possible.

KAoS Policy Framework (Figure 2) generic functionality includes:

- Policy ontology management,
- Creating/editing of policies using KPAT,
- Storing, deconflicting and querying policies using the Directory Service,
- Distribution of policies to Guards, which control agents' actions using Enforcers,
- Policy disclosure mechanisms.

The framework can be extended to support a specific environment by:

- Defining new ontologies describing; resources and types of actions which can be performed on them,
- Creating Plug-ins for:
  - Policy Template editors,
  - Enforcers controlling specific actions or with generic enforcement capability,
  - Defining Semantic Matchers to determine if a given instance is in the scope of the given class to support specific actions.

These plug-ins are linked with the framework by association with appropriate ontology concepts so that, for instance, if a user decides to create a policy for a specific action class, the policy editor can be found using this action class as a reference. The same method is used to find an enforcer for a policy controlling some action class.

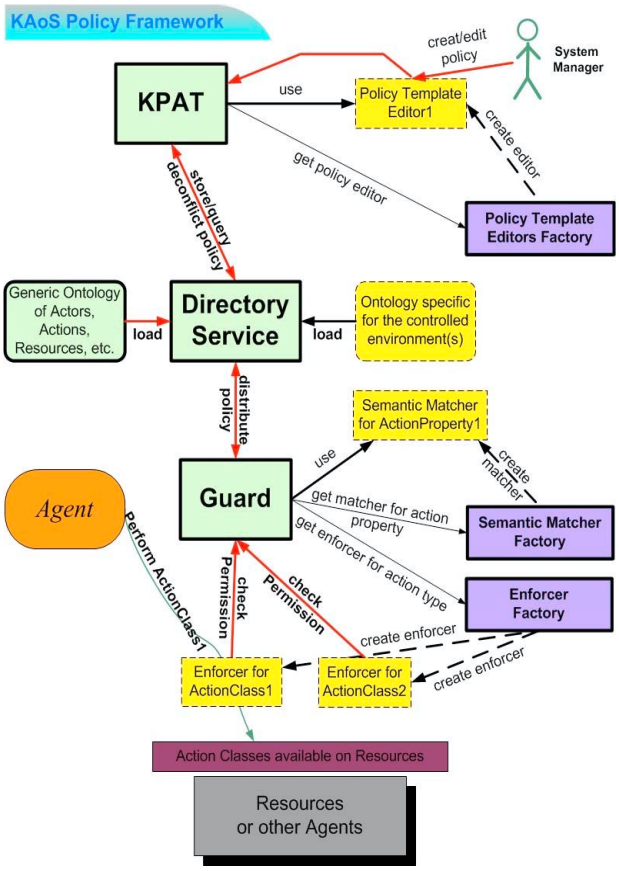

Figure 2. KAoS Policy Framework Architecture

KAoS Policy Administration Tool (KPAT) is used to browse and load ontologies, to define, deconflict, and commit new policies, and to modify or delete them. The generic graphical DAML Policy Editor, integrated with KPAT, is driven by the ontologies loaded into the Directory Service, during the bootstrapping process, and always provides the user with the list of choices narrowed to the current context by querying the loaded ontologies. When a user commits a change to ontology (e.g., a new or edited policy, changes to domain structure) the Jena (*http://www.hpl.hp.com/semweb/*) toolkit is used to dynamically build a DAML representation based on the values selected by the user.

Following conflict detection, policies are distributed to guards based on information about types of agents controlled by them. Guards activate appropriate enforcers based on received policy types.

Enforcers are the mechanism by which Guards ensure compliance with authorization/obligation policies. The *grounding* of enforcers to the particular agent environment cannot always be made fully generic.

Depending on the environment; they can be made fully general and understand abstract ontology action classes via their property *implementedBy*, which maps them to concrete environment operations, and by using reflection and security mechanism of the environment. Other environments require pre-building enforcers based on the ontology description of the controlled action class, potentially using preprocessor. Finally, some cases required fully custom built enforcers. What can be made generic however is the interface to the policy disclosure system answering for instance the question, "is a given action authorized or not?„

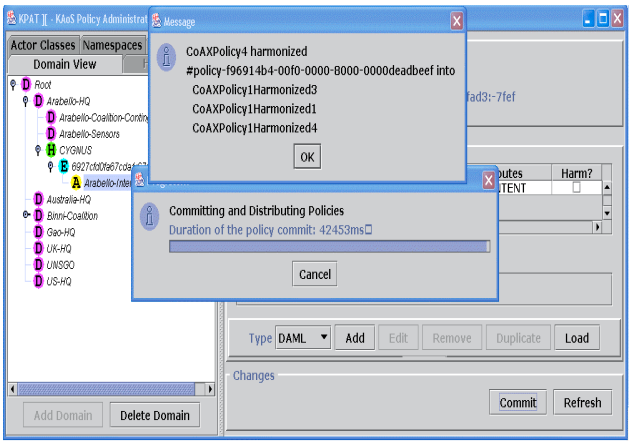

Figure 3. KPAT enables policy management

In KAoS, Guards disclose policies to authorized parties (e.g. enforcers), which provide the description of the interested action instance. The system can:

- Check permission to perform the action,
- Find the values of a partially described action for which the final action would be allowed,
- Discover if some action is obliged for the given condition.

5. Example applications

Many of the capabilities were developed as part of the DARPA CoABS-sponsored Coalition Operations Experiment (*http://www.aiai.ed.ac.uk/project/coax/*, CoAX). CoAX models military coalition operations and implement agent-based systems to mirror coalition structures, policies, and doctrines. KAoS provides mechanisms for overall management of coalition organizational structures represented as domains and policies.

Within the DARPA Ultra*Log program (*http://www.ultralog.net*) we are developing agent policy and domain services to assure the robustness and survivability of logistics functionality in the face of information warfare attacks or severely constrained or compromised computing and network resources.

Another application is within the NASA Cross-Enterprise and Intelligent Systems Programs, where we are investigating the use of policy-based models to drive human-robotic teamwork and adjustable autonomy for highly interactive autonomous systems such as the Personal Satellite Assistant (PSA).

6. Future work

Some areas of future work include: full support for obligation policies and condition monitoring in the framework, grounding of the enforcement mechanism to new software environments, improved support for reasoning about composite actions, augmentation of policies with additional information such as penalties for breaking policies and the rationale for them (expressed as a risk) and equipping agents with mechanism to reason about these information. We will also continue to develop versions of KAoS suitable for deployment in Web Services and Grid Computing environments.

References

[1] Baader, F., Calvanese, D., McGuinness, D., Nardi, D. & Patel-Schneider, P. (Ed.) (2003). The Description Logic Handbook. Cambridge University Press,

[2] Bradshaw, J. M., Beautement, P., Breedy, M. R., Bunch, L., Drakunov, S. V., Feltovich, P., Hoffman, R. R., Jeffers, R., Johnson, M., Kulkarni, S., Raj, A. K., Suri, N., & Uszok, A. (2003). Making agents acceptable to people. In N. Zhong & J. Liu (Ed.), Handbook of Intelligent Information Technology. IOS Press.

[3] Bradshaw, J. M., Dutfield, S., Benoit, P., & Woolley, J. D. (1997). KAoS: Toward an industrial-strength generic agent architecture. In J. M. Bradshaw (Ed.), Software Agents. (pp. 375-418). Cambridge, MA: AAAI Press/The MIT 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, London,

[5] Kahn, M., & Cicalese, C. (2001). CoABS Grid Scalability Experiments. O. F. Rana (Ed.), Second International Workshop on Infrastructure for Scalable Multi-Agent Systems at the Fifth International Conference on Autonomous Agents. ACM Press,

[6] Sierhuis, M. (2002). Brahms - Modeling and Simulating Work Practice. Univ. of Amsterdam Press,

[7] Suri, N., Bradshaw, J. M., Breedy, M. R., Groth, P. T., Hill, G. A., Jeffers, R., Mitrovich, T. R., Pouliot, B. R., & Smith, D. S. (2000). NOMADS: Toward an environment for strong and safe agent mobility. Proceedings of Autonomous Agents 2000. Barcelona, Spain, New York: ACM Press.