Toward a Flexible Ontology-Based Policy Approach for Network Operations Using the KAoS Framework

Andrzej Uszok, Jeffrey M. Bradshaw, James Lott, Matthew Johnson, Maggie Breedy, Michael Vignati
Florida Institute for Human and Machine Cognition (IHMC)
40 South Alcaniz St., Pensacola, FL 32502
{auszok, jbradshaw, jlott, mjohnson, mbreedy, mvignati}@ihmc.us

Keith Whittaker, Kim Jakubowski, Jeffrey Bowcock
RDECOM CERDEC STCD
Ft. Monmouth, NJ 07703
{keith.d.whittaker.civ, kimberly.a.jakubowski2.civ, jeffrey.c.bowcock.civ@mail.mil}

Abstract - *This paper addresses the challenges of flexible and uniform policy management in complex military network operations. The ontology-based approach of KAoS policy services provides flexibility in level of abstraction, in mapping to third-party policy approaches, and in managing policies across multiple application domains and across different operating environments. KAoS allows for specification of policy in contrained English with an ontology-based vocabulary. Following an overview of the major components of the KAoS Policy Services framework, we describe ontology-related components, including the UCore and KAoS ontologies. We then describe our work on network and cognitive radio management. We present our approach to mapping SNMP MIB information to ontologies. Finally, we discuss results of an ongoing performance study of the policy system.*

Keywords: *policy, ontology, OWL, KAoS, SNMP, network operation management*

I. INTRODUCTION

The growing dependence of the military on network-centric warfare increases the need for an innovative approach to the problems of integrated management for network operations. Policy services have become the most widely-used approach to such management problems. Many different kinds of policy approaches have been proposed (e.g., [1][4][9][10][13]). An ideal fit for military requirements would be a flexible and uniform policy approach that supports multiple application domains and platforms in complex, large-scale, and dynamic network operations. The approach would need to support enterprise-wide control of low-level network, security, access control, and radio configurations based on high-level mission objectives. We believe that ontology-based policy management schemes hold great promise in providing the flexibility needed to meet these and other demanding military requirements.

The KAoS Policy Services framework [12] described in this article was the first to offer an ontology-based [2] approach and is currently the most successful and mature of such efforts. In a recent policy language overview presented to the US Government Digital Policy Management Standards Subgroup, KAoS was highlighted as the "recommended policy ontology starting point" [15].

The flexibility of the KAoS ontology-based policy approach is apparent in the following ways:

1. *Flexibility in level of abstraction.* The rich semantics of ontology-based policies coupled with powerful descriptive-logic-based reasoning mechanisms enable KAoS to generate low-level operational policies from high-level mission objectives (expressing "commander's intent").
2. *Flexibility in mapping to third-party policy approaches.* Special-purpose policy languages are inherently limited, making it difficult or impossible to meet requirements for system-wide control. A major advantage of using ontology-based policy representations is that any policy element (e.g., system components, actions, and context) can be described by appropriate concepts and relationships at the desired level of abstraction. Because the semantics of such representations typically are a superset of the semantics of specialized "niche" policy languages, it is possible to convert ontology-based representations into the more specific languages.
3. *Flexibility in managing policies across multiple application domains*. For a policy-based system to be effective in multiple application domains, it must develop beyond a specialized focus to handle richer policy semantics—ideally, an easily extensible semantics like OWL 2[1]. For example, administration of access control requires the specification of who can or cannot use specific resources, while network management applications control and schedule bandwidth on a controlled network. In contrast to approaches that focus on a particular application niche, the ontology-based approach of KAoS allows it to be easily extended to manage policies across multiple application domains through adding new concepts and relationships to the ontology.
4. *Flexibility in managing policies across different operating environments.* Available enforcement strategies and mechanisms vary across platforms and

[1]Ontology Web Language - http://www.w3.org/TR/owl-features

application domains, and the policy framework must be flexible enough to adapt to each of them. The policy system must also deal with the particular idiosyncrasies of the environment in which the application will be deployed. For example, some operating environments afford virtually guaranteed connectivity with ample network bandwidth, while other environments suffer from intermittent connectivity and highly-constrained bandwidth — or no network connection at all. The architecture of KAoS, combined with its ontology-based approach, allows it to be dynamically adapted to a wide variety of platforms and operating environments.

In the next section, we describe KAoS in more detail.

II. KAoS Policy Services

IHMC's KAoS policy services framework is a mature system that relies on ontologies in the specification, analysis, and enforcement of policy constraints across a wide variety of distributed computing platforms. KAoS enables the specification, management, conflict resolution, and enforcement of policies. The use of ontologies to represent policies enables reasoning about the controlled environment, policy relations and disclosure, policy conflict resolution, and domain structures and resources. KAoS reasoning methods exploit description-logic-based subsumption and instance classification algorithms and, if necessary, controlled extensions to description logic (e.g., role-value maps).

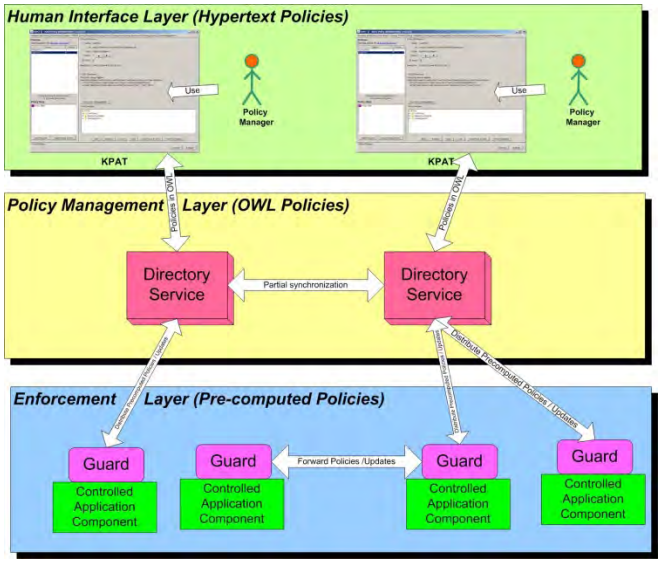

Figure 1: KAoS Policy Services Conceptual Architecture.

Two important requirements for the KAoS architecture are modularity and extensibility. These requirements are supported through a framework with well-defined interfaces that can be extended, if necessary, with the components required to support application-specific policies. The basic elements of the KAoS architecture are shown in Figure 1. The three layers of functionality correspond to three different policy representations:

- *Human Interface Layer*: This layer uses a hypertext-like graphical interface (KAoS Policy Administration Tool — KPAT) for policy specification in the form of constrained English sentences. The vocabulary is automatically provided from the relevant ontologies, consisting of highly-reusable core concepts augmented by application-specific ones. Besides KPAT's use in policy specification and analysis, it is employed for administrative tasks such as browsing and loading ontologies, and domain and Guard management. The generic KPAT interface can be easily customized or replaced.
- *Policy Management Layer*: Within this layer, OWL is used to encode and manage policy-related information. The KAoS Distributed Directory Service (DDS) residing in this layer encapsulates a set of ontology reasoning mechanisms over the policies, used for policy deconfliction and various kinds of analysis.
- *Policy Monitoring and Enforcement Layer*: KAoS automatically "compiles" OWL policies to a very efficient format that can be used for monitoring and enforcement. This representation provides the grounding for abstract ontology terms, connecting them to the instances in the runtime environment and to other policy-related information. The KAoS Guard residing in this layer is integrated with the controlled application and provides an API for policy checking.

Maintaining consistency among the three layers is handled automatically by KAoS, a task made more challenging because each layer implements its functionality in a distributed rather than a centralized manner.

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

KPAT's generic Policy Editor presents an administrator with a starting point for policy construction — essentially, a very generic policy statement shown as hypertext. Clicking on a specific link in this statement that represents a variable provides users with menu choices allowing them to make the generic policy more specific.

Policies defined using this menu-driven process follow a predetermined syntax in constrained natural language for either authorization or obligation policies. Authorization policies permit or forbid some action while obligation policies either require some action to be performed or waive such a requirement. Figure 2 shows an example of an authorization policy being defined in KPAT.

During use, KPAT accesses the ontologies that have been loaded into the DDS and provides the user with the list of choices narrowed to the current context of the policy construction. New ontology classes and instances needed for specific kinds of policies can also be created within KPAT. Since the ontologies directly determine what

choices are provided to users when they build policies,, the correctness of the semantics of the policy is dependent only on the correctness of the ontology. KAoS tools to create ontologies directly from the environment (currently SNMP and WSDL; soon, Java) are designed to further ensure the correctness of the ontology. The translation from the form of the constrained English policy to its OWL representation in KPAT is deterministic.

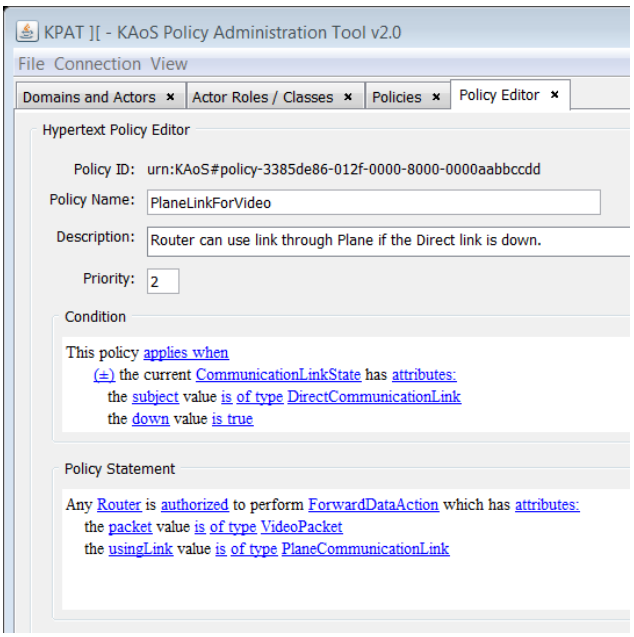

Figure 2: Authorization Policy in the KPAT Generic Policy Editor.

To further simplify policy construction, KPAT provides two additional policy creation interfaces:

- The *Policy Wizard* **(**Error! Reference source not found.**) takes a user step-by-step through the policy creation process. Information selected for presentation is conditioned on whatever has been selected previously, making the experience as simple and foolproof as possible.
- The *Policy Template Editor* allows custom policy editors for a given kind of policies to be created by point-and-click methods. For instance, if an application will require the definition of several policies governing publish/subscribe actions, a custom policy editor can be quickly created by limiting choices to just what is needed, thus eliminating the requirement for repetitive selections when a given type of policy has to be created multiple times.

As another example of KPAT extensibility, when filling in values of type "area," users are presented with a custom area editor. The editor allows them to define a polygonal region on top of a domain-specific background map by using the mouse to define edge points.

The Guard is where KAoS meets the controlled system. Its primary role is as a policy decision point. Guards register to receive policies about particular entities or classes of entities for a given set of action classes. Because guards can save their policies and reload them directly from a snapshot, they can be bootstrapped in a standalone mode without a need to connect to the DDS. This functionality allows policies to govern the actions of standalone sensors or similar components.

Guards not only receive information about policies, but also about the state of the system and the entities being managed. Guards do not by themselves provide monitoring functions, but they do provide interfaces to plug in outside monitors or databases providing access to external state or event-related information.

The KAoS Guard Policy Checking Interface provides a set of methods that allow checking for:

- *Authorization.* If an action is not authorized, an exception is thrown with information about the policy that prevented it. In some secure applications, however, it would not be desirable to release information about the cause of the policy exception.
- *Obligations.* A list of obligations for a given actor is returned, sorted in rank order of importance. In addition, if there are obligations for other actors that are triggered by an external event, then KAoS will try to locate them and forward the obligations to them.
- *Configuration options.* If a partial description of the action is sent to KAoS, a range of allowed values for properties of a given action is returned. For instance, if an application were to query the guard about a planned radio transmission, information about the maximum power and range of frequencies allowed to be used in the given geographical area would be returned to it. Disclosure policies would be used to filter out unauthorized information in the results.

In order to support the semantics of complex application-specific policies, guards accommodate a variety of extensions. These can be activated on demand, as specified in each guard's configuration information. Specific extensions are:

- *History Monitor:* tracks the history of specific actions and allows verifying whether a given history is present (e.g., three successive login failures).
- *State Manager:* manages set of environment-specific sensors that provide information about dynamic aspects of the environment or situation (e.g., threat level, resource availability).

III. Ontology

An ontology [2] is a formal representation of knowledge as a set of concepts within a domain, and the relationships between those concepts. Ontologies structure these concepts at various levels of abstraction, where the upper ontology consists of relatively generic terms and lower-level ontologies extend the basic concepts with more specific definitions relevant to the particular domain of application. For analytical purposes, relationships among lower-level concepts can be maintained through the upper-level parent concepts. KAoS ontologies are used as a source of vocabulary for policies and to reason about the relationships

among policies. They are used to find the mapping between abstract concepts used in policy definition and the actual concrete entities controlled by the given policy.

A. Upper Ontology – KAoS and UCore

The KAoS Core[2] ontology is defined in OWL 2 and contains about a hundred class definitions providing basic concepts of entity, actor, action, group, situation, history, state, and policy. These basic concepts are extended with essential subclasses and properties. The development of KAoS Core ontology began in 2001. Recently the US Military has developed its own upper layer ontology called UCore Semantic Layer[3] ontology which defines 144 classes and 16 relations. Its terms are very generic and pertain mainly to the physical aspect of environment such as vehicles, cargo, economic events, and so forth. We linked our ontologies with UCore Semantic Layer to help provide interoperability with military systems that rely on it.

B. Network Ontology

The role of the Computer Network ontology is to serve as a mediating virtualization layer between the network manager (whether a human or automated system) and the complexities of heterogeneous network infrastructure. The network ontology represents both active nodes (models as subclasses of Actor) and passive links (models as subclasses of Entity). The links are associated with a class that represents the changing state of connections. Network nodes are associated with physical locations, organizations, and other context. Nodes are related with actions they can perform:

- network modification (add, move, reconfigure, remove),
- changes in connections
- information generation, translation, and dissemination.

The ontology models a variety of basic categories of network devices and servers, their interfaces and ports, links types (including radio links) and communication protocols. The Network ontology concepts have been connected to UCoreSL at the level of its classes *slr:Entity* and *slr:Event*. Additionally, specific UCoreSL terms are used to define the state of the physical environment in which the network operates. UCoreSL properties are used as superproperty of specific network properties such as *slr:located_at* or *slr:part_of*.

C. Radio Ontology

The Radio Network OWL concept is a specialization of the Computer Network concept, and thus the Radio ontology relies on the Network ontology. In the ontology, a Radio is a subclass of KAoS Actor class, under the assumption that it will be performing actions such as transmission. It is defined as consisting of a transmitter, receiver and channels and tied into generic equipment class. The appropriate radio frequency parameters are defined to each class. The Radio class has properties depicting its physical location, condition, mission, ownership, and so forth. It also has properties relating it to radio links and relevant states. The Radio class is associated with actions related to, among others, selection and modification of channels and power. This ontology also has definitions of action classes related to network modification (add, move, reconfigure, remove radio and links), beacon, and broadcasting of other messages. Additional ontologies define classes of units for radio frequencies and parameters, and enumerations of commonly-used band designations. A separate ontology defines terms contained in the US Spectrum Allocation chart[4] and those related to waveform classification. Since radios are typically tactical equipment, we have also defined a Tactical System Ontology based on a MITRE study, and have used it to provide classification of radios from the perspective of the tactical environment.

D. Ontology Mapping Tool

The Computer Network ontology has existed for quite a while, and already contains a rich set of knowledge about specific network devices. These include SNMP MIB files as well as other formats. In addition, these technologies allow for operational network management. In order to link the high-level network ontology described in the previous section with actual network devices we created a tool (Figure 3) that enables semi-automatic mapping from specific representation of network configurations to OWL. The tool facilitates the migration of knowledge about network configuration and functionality from a MIB database into ontologies. The mapping from the ontologies to the MIB files and other representations—used when policies employing a given vocabulary are enforced—is also supported in a fashion that is transparent to the end user. The tool possesses an easy-to-learn interface showing results of the automatically-generated mapping to OWL, but allowing for user modification. An important option is the ability to link a newly-mapped concept to an existing concept in the ontology. We are working on assisting the user with this process by providing suggestions as a list of possible candidate concepts for linkage using heuristic search methods and sources such as WordNet[5].

SNMP MIB files are parsed using the Mibble[6] parser, allowing the tool to retrieve imported modules, textual conventions, object names, and sequences. The tool generates an ontology name based on the MIB name for each element found, and then relates them to appropriate ontology concepts from the Network ontology. The same mapping methods are employed for other representations such as Web Service WSDL.[7] Our goal is to make this tool generic, allowing mapping from different representations of

[2] http://ontology.ihmc.us/ontology.html

[3] https://wiki.kc.us.army.mil/wiki/UCore-SL_Implementation_Guidance

[4] http://www.ntia.doc.gov/osmhome/allochrt.pdf

[5] http://wordnet.princeton.edu/

[6] http://www.mibble.org

[7] http://www.w3.org/TR/wsdl

real-world knowledge to ontologies and integrating them into a uniform semantic layer. This tool would typically be used by a person responsible for deploying KAoS in a given environment. The creator of polices will not be exposed to its inherent complexity.

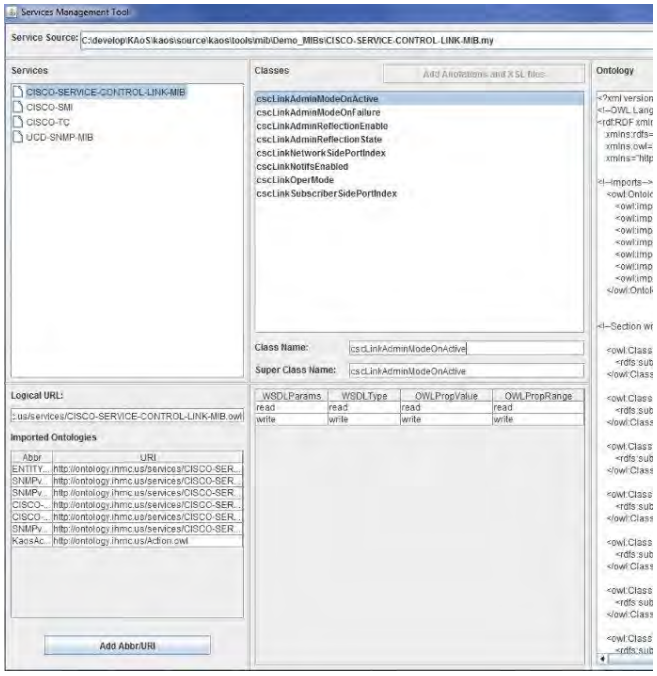

Figure 3: Fragment of the SNMP to Ontology mapping tool GUI.

IV. Network Management

For network management applications, we define policies that automate network configuration and reconfiguration based on changes in network state or in the operational situation. Using KAoS, it is possible to combine policies from different sources of regulation with default configuration and tactical intent for network operation into the coherent set of policies. The policy generic terms are then mapped (Figure 4) to the actual controlled network elements through ontology reasoning and the exploitation of the previous MIB (and other representations) to OWL mapping.

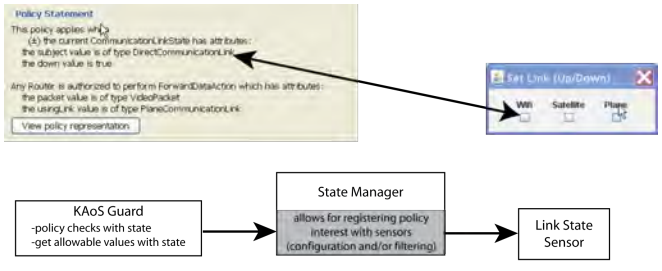

Figure 4: Example of the network policy controlling selection of the used link by the network gateway.

Ontological concepts representing specific MIB elements are annotated with OID numbers. This allows the KAoS Guard extension *SNMPSensor* and *SNMPEnforcer* to map ontology concepts from the policies to SNMP requests - and vice-versa (Figure 5).

The role of the sensor is to monitor the state of one or more SNMP-enabled nodes, based on policy interest. It maps ontology-based state definitions from policies to SNMP requests. Either it periodically sends SNMP GET requests to obtain the current value of parameters of interest to policies or it sets up TRAP, if available. It converts SNMP replies to their ontological equivalent and notifies the KAoS Guard's StateManager of changes to the network node's state.

The enforcer configures network nodes in accordance with policy decisions, whether as part of initial configuration or in response to state or situation changes. It sends an SNMP SET request containing information about the new configuration. Implementation of these components may also rely on the SNMP4J library.

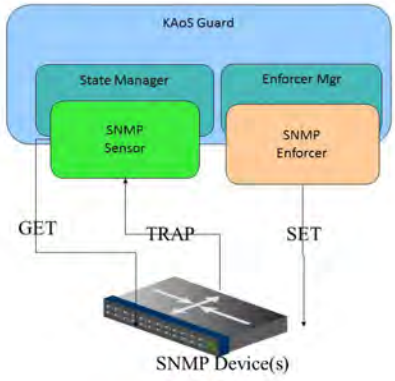

Figure 5: KAoS SNMP integration.

As a next step, we have begun to integrate KAoS with the JINX network management system. This effort will facilitate management of networks composed of joint assets during a joint mission. We are extending our ontology mapping tool to take advantage of our current SNMP and MIB integration. The mapping tool is also being extended to cover additional network information standards (e.g., SCOM[8]). We plan to use KAoS to generate specific network configurations based on high-level requirements (intent) for network operations. These high-level requirements are expressed as policies. The configuration is provided to JINX through its MUSIC (Multi-System Integrated Configurator) interface. The KAoS Guard State Monitor is being integrated with JINX in order to obtain feedback about network state changes and, as policy dictates, to update the original network configuration accordingly.

V. Cognitive Radio Policy Management

Radio configuration and operation can also be controlled by policy services. When a radio is bootstrapped, its software provides KAoS with information such as its location, mission, or usage. Based on this information, the policy services calculate the configuration of viable operational parameters for this radio. The feedback about any changes

[8] http://technet.microsoft.com/en-us/systemcenter/om/bb497976

to the radio context is provided to the KAoS Guard State monitor by the radio software. Based on these changes, the configuration will be recalculated and reconfigured as needed.

Radio software also consults policies about any obligations that may be triggered when it receives radio network management signals or when changes in the radio state or network are detected. KAoS generates the required response of the radio software, determining how the generic policy form should be shaped to the local radio context.

We are creating templates for common policy types in this domain—for instance, geospatial, time-based, identity-based, frequency-based, technical parameter enforcement, directive control, group behavior, monitoring behaviors, and network specification.

VI. Mapping of High –Level to Low-Level Policies

KAoS policy refinement mechanisms map from high-level policies (e.g., expressions of Commander's intent, mission-level objectives) to low-level policies that dictate operational aspects of network configuration and operation. As our research progresses, we will upgrade our initial static mapping approaches with more advanced synthetic methods supported by a planner.

Intent-level objectives for network operations may be expressed relative to a given mission type or a specific mission instance. Specific needs for connections between units based on current conditions can also be expressed. By providing priorities for network resources based on the situation of units (e.g., engagement status, whether they have reached target location), resilient response to meet specific high-level objectives can be accommodated (e.g., needs chat connections to all units, needs video connection to unit Bravo).

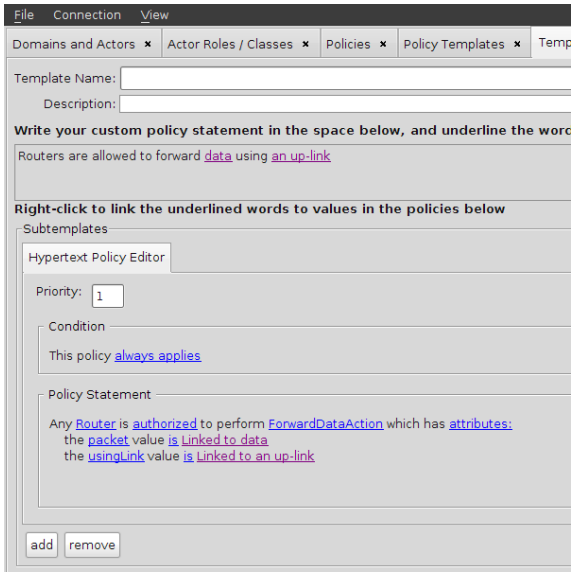

Figure 6: Creating a hypertext template (links are purple)

KPAT allows administrators to create policy templates composed of free-form English sentences tailored to the vocabulary and types of policies that are common in a particular application context (Figure 6). Very simple templates may be configured to generate a set of arbitrarily complex policies as output. In this way, even novices can create sensible, well-crafted policies without requiring specialized training. Templates can be used equally-well to define policies expressing high-level intent and low-level tactics. To create a hypertext template for a new class of policies, an administrator begins by writing a plain-English statement of the policies which the template will output, just as if he were using a normal text editor or word processor. For example, "Routers are allowed to forward data using an up-link". The underlined words become variables that the end-user fills in.

Next, one or more policies are added to the template, which will be the output. Static values within these policies are filled in, while values to be chosen by the end-user of the template are left empty. For example, in the policy below, the values for the attributes *packet* and *usingLink* are left empty:

Any Router is authorized to perform ForwardDataAction which has attributes:
the packet value is of type [Select...]
the usingLink value is of type [Select...]

To complete the template, words or phrases from the plain-English statement must be linked to the empty values in the policies. To create a link, an administrator highlights and underlines some text from the statement for which the end-user must choose a value. Then he simply drags and drops the underlined text to an empty value in one of the policies, thereby completing the link. A single underlined phrase in the statement may be linked to multiple values in the resultant policies (assuming the range of values has a non-empty intersection).

Later, when the template is used to specify a new policy, end-users are presented with a plain-English policy statement, including the underlined phrases as hypertext (Figure 7). The user simply clicks on the underlined phrases to select the values. These values will be incorporated into the final OWL representation of the policy that is generated by the template.

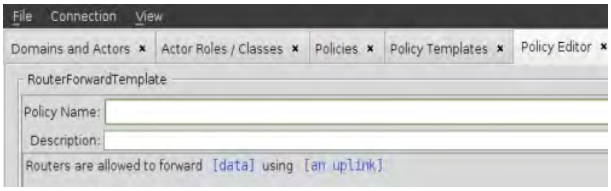

Figure 7: Filling in a completed hypertext template

VII. Scalability and Performance

The policy system should be efficient in order to be deployable in realistic network management applications. Performance studies provide insight into system overhead as a baseline against which future improvements can be compared. Among others, we measured two critical phases in policy lifecycle. The first phase is when policies are

initially committed by the user to the KAoS Directory Service and distributed to Guards (Figure 8). The second is when the policies are used in the Guard to calculate decisions (Figure 9).

When policies are committed to the KAoS DDS using KPAT, they must be translated to an ontological representation, added to the ontology, and then distributed to relevant Guards. Figure 8 shows the average time needed for each policy: about 120ms when more than 50 policies are added are the same time. When a smaller number of policies is committed, the time is about 200ms for each policy. In the case of larger number of policies the overhead of network communication is spread across a larger number of policies.

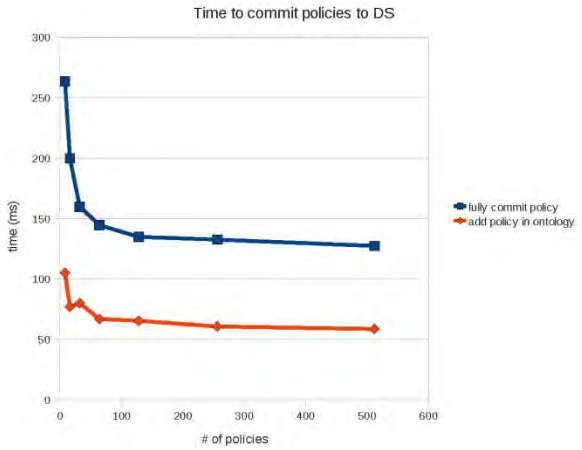

Figure 8: Diagram presenting the time needed to commit increasing number of policies

In contrast to the policy commitment, the time to obtain authorization policy decisions is fast (40ms) even when there are a large number of active policies. Such efficiency is achieved by the effective use of indexes and hash tables.

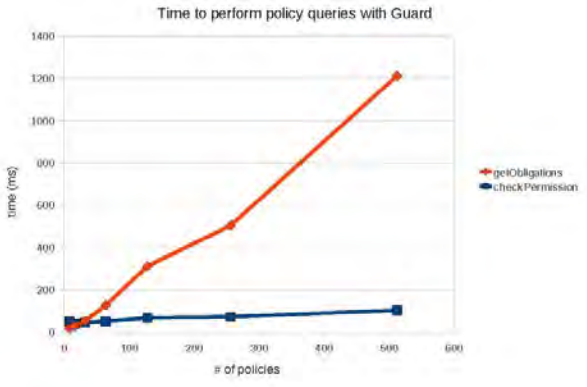

Figure 9: Diagram presenting the time needed to obtain decision base depending on the number of policies

Obligation policy decision time is linearly proportional to the number of obligation policies. The reason for the difference between authorizations and obligations is because in the case of authorizations the system has to find only the single deciding policy but in the case of obligations it has to analyze all obligation policies matching the relevant action. These performance studies will provide the basis for future improvements in KAoS.

VIII. Conlusion

We believe that the KAoS ontology-based policy management approach—with its rich semantics, its affordances for layered abstractions, and its ability to accommodate dynamic system evolution—holds great promise for the challenges of military network operations.

Acknowledgments

This research effort is supported by the US Army CERDEC Grant W15P7T-06-D-E402. Sub: S11-114109.

References

[1] Agrawal, D., Calo, S, Lee, K, Lobo, J and Verma D. (2009). Policy Technology for Self-Managing Systems. IBM Press, ISBN-978-0-13-221307-3.

[2] Allemang, D. and Hendler, J. (2008). Semantic Web for the Working Ontologist. Morgan Kaufmann, ISBN-978-0-12-373556.

[3] CERDEC Dynamic Spectrum Access Policy Assessment report.

[4] Chadha, R. and Kant, L. (2008). Policy-Driven Mobile Ad Hoc Network Management. Willey Interscience, ISBN-978-0-470-05537-3.

[5] Dandashi, F., et al (2008). Tactical Edge Characterization Framework -Volume 1: Common Vocabulary for Tactical Environments. MITRE Corporation, http://www.mitre.org/work/tech_papers/tech_papers_08/08_0037/.

[6] Elmasry, G., Jain, M., Jakubowski, K. and Whittaker, K. (2010). Conflict Resolution for Shared Resources between Network Mansagers. Proceedings of MILCOM 2010: 1866-1871.

[7] Guerrero, A., Villagrá, V., de Vergara, J. and Berrocal, J. (2005). Ontology-Based Integration of Management Behaviour and Information Definitions Using SWRL and OWL. Proceedings of DSOM 2005: 12-23.

[8] Kodeswaran, P., Li, W., Joshi, A., Finin, T. and Perich, F. (2010). Enforcing Secure and Robust Routing with Declarative Policies. Proceedings of MILCOM 2010: 1653-1658.

[9] http://www.ponder2.net/ (accessed 25 May 2011).

[10] Strassner, J. (2004). Policy-Based Network Managmement. Morgan Kaufmann, ISBN-1-55-860-859-1.

[11] Uszok, A., Bradshaw, J., Lott, J. Breedy, M., Bunch, L., Feltovich, P., Johnson, M. and Jung, H., (2008). New Developments in Ontology-Based Policy Management: Increasing the Practicality and Comprehensiveness of KAoS. In Proceedings of the IEEE Workshop on Policy 2008, IEEE Press.

[12] Uszok, A., Bradshaw, J., Jeffers, R., Johnson, M., Tate A., Dalton, J., Aitken, S. (2004). KAoS Policy Management for Semantic Web Services. IEEE Intelligent Systems, July/August, 19(4), pp. 32-41.

[13] Verma, D. (2001). Policy-Based Networking. New Riders, ISBN-1-57870-226-7.

[14] Walsh, L. (2008). SNMP MIB Handbook. Wyndham Press, ISBN 978-0-9814922-0-9.

[15] Westerinen, A., Digital Policy Management: Policy Language Overview. Presentation at the DPM Meeting, Jan 19, 2011 / Updated Mar 27, 2011.