

Paper Review

**New Developments in Ontology-Based Policy
Management:
Increasing the Practicality and Comprehensiveness
of KAoS**

Andrzej Uszok, Jeffrey M. Bradshaw, James Lott,
Maggie Breedy, Larry Bunch,
Paul Feltovich, Matthew Johnson, and Hyuckchul
Jung

Ankesh

March - 24 - 2009

Paper about?

- KAoS - <http://ihmc.us:16080/research/projects/KAoS>
- Institute for Human and Machine Cognition
- “KAoS policy management framework **pioneered the use of semantically-rich ontological representation** and reasoning to specify, analyze, deconflict and enforce policies.”
- [3] “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. We’re exploiting the power of Semantic Web representations to address some of the challenges currently limiting Semantic Web Services’ widespread deployment.”

2008 IEEE Workshop on Policies for Distributed Systems and Networks submission

Submission Categories:

- **Regular Papers**
- Short Papers
- System demonstrations

Prior published works

1. **KAoS Policy Management for Semantic Web Services.** IEEE INTELLIGENT SYSTEMS (2004).
2. **Semantic web languages for policy representation and reasoning: A comparison of kaos, rei, and ponder.** (ISWC 2003)
3. **Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement** (Policy 2003)

Quotes

- “ontologies provide a **natural means for supporting alternate sets of policy vocabulary** for different applications.”
- “The flexibility and power of a policy management framework is to a large degree determined by the expressivity of its policy representation, provided it is also computationally efficient. In our experience, OWL has not only proven to be remarkably **expressive** and **efficient** but is also straightforwardly **extensible** and **adaptable**, *even at runtime when sophisticated real-time management and analysis of new or existing policies may be required.*”

Outline

- Policy representation
 - Syntax + Semantics
 - Grounding
 - Role value maps
 - Policy precedence
 - History determines policy applicability
- 3 layer architecture
 - Policy management layer
 - Policy distribution
 - Policy enforcement- the guard architecture
- KAoS applications
- Review

Ontologies, syntax, semantics

POLICY REPRESENTATION

Ontologies

- > 100 classes and properties this basic ontology
- Concepts to define the policies themselves
- root concepts for policy-governed actions, actors, places, states, history, situations, environmental properties related to actions (e.g., computing or network resources), groups (e.g., domains, roles, teams), and so forth.
- Core ontology → application-specific vocab
(ApplicationAction, ApplicationActor, ApplicationEntity)

Grounding of Ontologies

- Static elements defined as individuals- e.g. robots, range of radio spectrums or set of producers of weather reports.
- Dynamic elements registered via Guard interface
 - Information can be provided to guard at policy enforcement time (new areas of operation, new radios, new teams or removal of any)

Policy Syntax + Semantics

- supports two main types of policy: **authorization** and **obligation**
- All other kinds of policies (*e.g.*, **delegation**, **teamwork coordination**) are built from these two primitive types.
- *positive / negative* authorization policies specify which actions an actor or set of actors is allowed/ not allowed in a given context.
- *Positive / negative* obligation policies specify actions that an actor or set of actors is required to perform/ *is waived*.

Basic form of KAoS policy:

[Actor] is [constrained] to perform [controlled action] which has [any attributes]

- *[Actor]* - a variable that refers to the subject of the policy-controlled action. (single or a class)
- *[constrained]* - a variable that refers to the basic type of the policy. (+ve/ -ve authorization, +ve/ -ve obligation)
- *[controlled action]* - a variable that refers to the action class controlled by the policy. (e.g. Movement)
- *[any attributes]* - optional variable referring to attributes of the controlled action. (e.g. configuration parameters). Attributes can have simple value restriction or related to other attributes.

[all | some] [attribute] values are [within the set of enumerated instances | of a given type]

Constraints on attributes

- some constraints on attributes can not be expressed in OWL.

Role-value maps- E.g.

- user department membership must equal the ownership property of the printer used in the print action.
- receivers of a radio transmission must all be members of the set of security clearance holders.
- At least one of the credentials of the user initiating communication must be in the set of credentials of the receiver of the message.
- None of the attribute values is equal to the values of another action attribute.

Relative attributes

- relate the obligation policy trigger to obliged action.
- E.g. If [some message] is of class 'secret' or greater, [some message] must be logged to the audit queue.

In addition to DL reasoner:

- Role-value-map reasoner
- spatial reasoning component to reason about location and orientation of objects, defined as regions described in polygons.

History + state information

- Applicability of policy dependent on history. e.g.
 - forbid system access to anyone who has a history of two or more failed logon attempts in the last five days.
- Similarly, state information. e.g.
 - policy applies when the weather has the following attributes: temperature > 75 and sunlight is bright.

Policy precedence

Why?

- To resolve conflicts
- Order by importance, and check in that order

How?

- Numeric policy assignments – executes quickly
- Logical precedence mechanism
 - **policies of domain administrator** take precedence over user policies.
 - More **recent policies** take precedence over older policies
 - **superdomain policies** take precedence over subdomain policies
 - policies about **writing to a specific directory** take precedence over policies about writing to the Volume
 - **Negative authorizations** take precedence over Positive authorizations
 - **Priority level of the policy** - users can define and name any number of arbitrarily-ordered priority categories (e.g. high-medium-low).
 - Default authorization mode

3-LAYER POLICY MANAGEMENT ARCHITECTURE

Architecture

- hypertext like graphical interface for policy specification in natural english
- Vocabulary automatically provided from relevant ontologies
- Jena framework used to build OWL policy

- encode and manage policies(repres.)
- DDS encapsulate OWL reasoning mechanisms (JTP, Pellet- any)

- compiled OWL policies, along with
- grounding for abstract ontology terms, connecting them to instances in RE and to other policy-related information

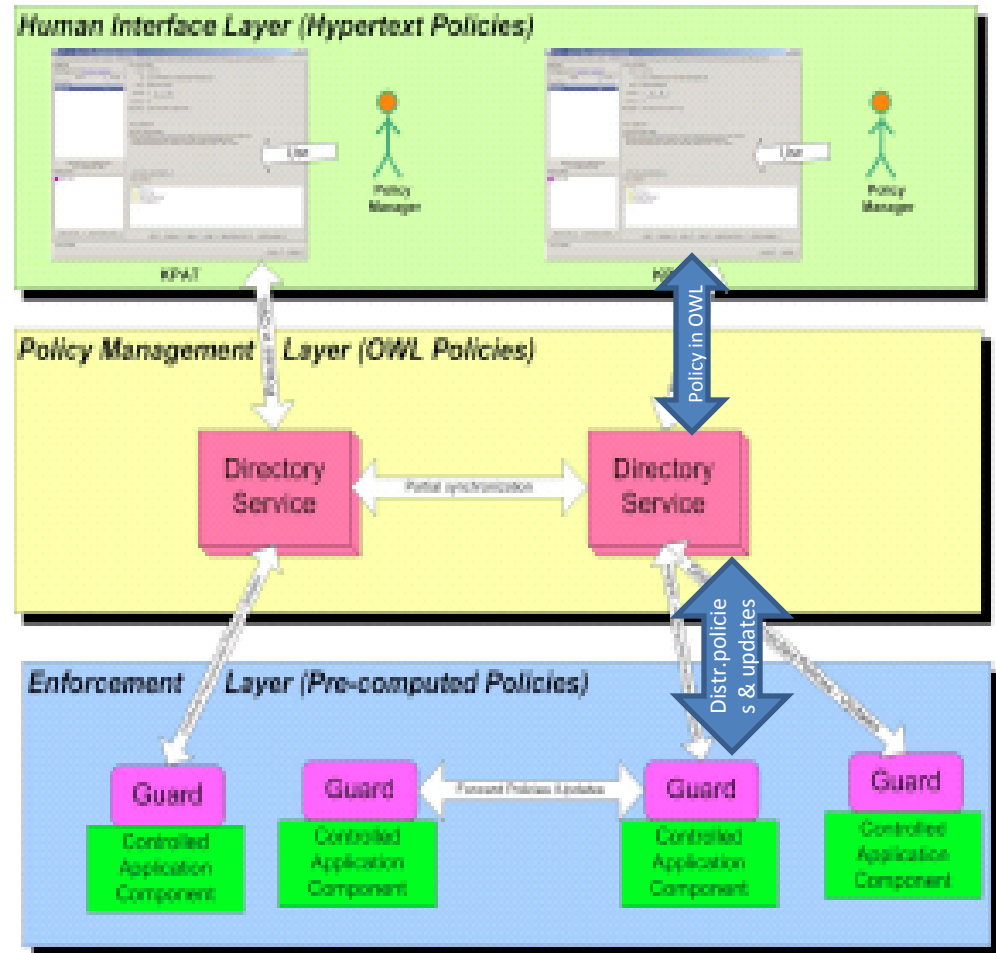


Figure 1: KAoS Policy Services Architecture

KAoS Policy Management Layer

Reasoning

- at the time of creation of policy- supplying lists of vocabulary terms.
- at the time of policy analysis- find relations between action classes controlled by policies through subsumption reasoning. Further check for role-value map semantics.
- at the time of policy distribution to guards- classify existing instances (e.g. list of actors) so that other relevant information can also be sent.

Policy distribution

- When policies added/ modified or when a guard (re)connects DDS sends the appropriate updates to particular guard.
- DDS maintains interests of each guard and history of what has been sent.
- The OWL policy representations are “compiled” into a hash-table-like structure where entries for each policy describe policy action attributes, a range of acceptable values, a list of super-attributes and sub-attributes defined in the ontology, and the encoded definition of the range class.
 - Enables the guards to make complex enforcement decisions very efficiently.

Policy monitoring and Enforcement Layer

- KAoS – **policy decision maker** for the applications
- Guard **registers to receive policies** about particular entities, entity classes or action classes

The Policy Checking Interface method for a given action instance to be checked for:

- **Authorization of an action** (can provide information about the policy that prevented it)
- **Obligations: A list of obligations for a given actor** is returned. If there are obligations for other actors KAoS locates them & forwards the obligations to them. Then there is *obligation monitor*.
- **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.

The data structure exchanged with Apps – Action Instance Description. **Java constants** for ontological concept **URIs**.

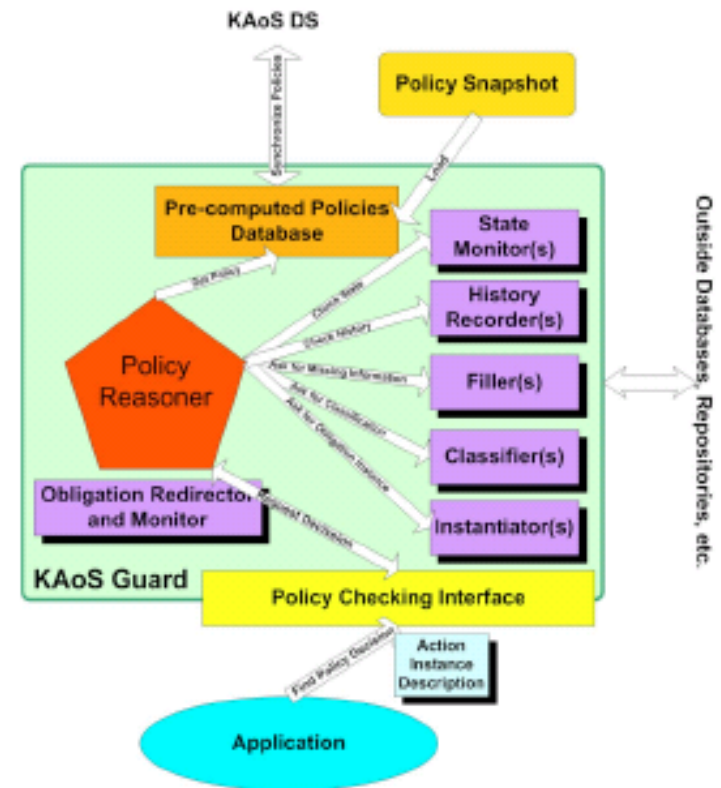


Figure 4: Architecture of the KAoS Guard

Current KAoS applications

- **Information Management System** – security and QoS policies in large-scale highly-distributed publish-subscribe system.
- **Cross-Domain Information Exchange (CDIX)** - a system that represents and reasons about domain-specific policies to help recognize what otherwise sensitive documents a soldier is allowed to receive given the current mission context.
- **Human-Agent-Robot Teamwork** - an application involving a variety of application domains and enforcement at different levels of control, from low level network resource control to high level organizational constraints, spatial reasoning, and coordination management.
- **Radio Spectrum and Transmission Control applications** - for settings governing power and other configuration parameters of the transmission-given geographical areas are approximated from reasoning based on spatial knowledge and the current type and usage of a radio.

REVIEW

(New-) features of KAoS

- support for representing and reasoning about history, state, and spatial properties as they relate to policy
- support for logical policy precedence
- three layer architecture
- the hypertext policy editor and policy wizard
- network efficient policy distribution methods
- a revamped Guard architecture
- (additional) mechanisms to support obligation policies.

My review

	Score	
RELEVANCE	9	Fairly detailed description of policy framework architecture
SIGNIFICANCE	7	Usefulness of OWL representation. Reference for general policy languages.
NOVELTY	7	Logical precedence mechanism
QUALITY OF EVALUATION	-/ 7	Good application case studies
TECHNICAL SOUNDNESS	8	
CLARITY	7	+ : Well explained with examples - : But.. business code?
PRESENTATION	-	
OVERALL SCORE	6	-: Summary of earlier published papers. provides an architectural design (System Demonstrations?)
CONFIDENCE SCORE	5	

Some suggestions for improving the paper @ clarity

- (Section 6.2, page 150) “Abstract actions not explicitly included in the **business code** are changed into the enforceable action with a definition of restrictions that distinguish the abstract class from a base class encoded as an attribute. For instance, if a given policy talks about the publication of weather reports and the business code just has a method of publication with a parameter specifying the information, then the range of values for the corresponding attribute of the published action used in the enforceable representation of policy is set to the weather report type.”
- (Section 7, page 151) “In order to cope with the OWL open-world assumption, KAoS **uses intersection to define classes of policy controlled actions** for deterministic policy decisions.”

Thank you!

Questions?