A SEMANTICALLY RICH POLICY BASED APPROACH TO ROBOT CONTROL

Matthew Johnson, Jeffery Bradshaw, Paul Feltovich, Renia Jeffers, Hyuckchul Jung, Andrzej Uszok
Institute for Human and Machine Cognition, 40 South Alcaniz, Pensacola, Florida, U.S.A.

Keywords: Policy, semantic, authorization, obligation, ontology.

Abstract: In this paper we describe our approach to enhancing control of robotic systems by providing domain and policy services via KAoS. Recently developed languages such as OWL provide a powerful descriptive logic foundation that can be used to express semantically rich relationships between entities and actions, and thus create complex context sensitive policies. KAoS provides a tool to create policies using OWL and an infrastructure to enforce these policies on robots. We contend that a policy-based approach can provide significant advantages in controlling robotic systems and is a much more natural way for operators to interact with and manage multiple robots.

1 INTRODUCTION

Robot control is achieved through a variety of techniques. Some aspects require tight real-time control loops, such as balancing and avoiding dynamic obstacles. Many aspects however do not have these real-time restrictions and simply provide constraints on the system, for example speed limits. These types of constraints can also adjust the parameters of the real-time control systems, like adjusting the standoff range for obstacle avoidance. The majority of robotics work to date embeds non-real-time constraints into the robotic system. Some are built into reactive behaviors and others are hard coded into deliberative layers. This makes it difficult for anyone but the robot's developer to have an understanding of them. Even with good architectural design that allows for adjustment of these constraints, it can still be difficult for an operator to manage constraints. The problem is exacerbated in heterogeneous multi-robot environments. Another limitation is that there is typically no capability to add context to a constraint adjustment. For example, the operator may be able to adjust the speed limit, but cannot say that the speed limit can be higher for emergency situations. This lack of semantic description also limits the use of software reasoning that could be used to aid the operator.

In this paper we describe our approach to enhancing control of robotic systems by providing semantically rich domain and policy services via KAoS. KAoS policies are mechanisms to make non-real-time constraints accessible to operators and a means to adjust these constraints at run time. Since the term "policy" is a very general term with a variety of meanings, in Section 2 we start by defining what we mean by policy. We will then describe how we have designed and implemented policies in Section 3 and provide a brief description of our applications in Section 4. These applications are used as examples as we discuss the advantages of policy based robot control in Section 5.

2 WHAT IS A POLICY?

The term "policy" is a word that takes different meanings according to context. In the robotics domain, the term policy is often used to mean a complete mapping from states to actions (Mataric, 2001). While there has been much work on expressive mathematical frameworks such as Markov Decision Process models, recently developed rich descriptive languages such as the Web Ontology Language (OWL: http://www.w3.org/2004/OWL) have not been exploited to express high level notions such as authorization, obligation, and preference. As a

Johnson M., Bradshaw J., Feltovich P., Jeffers R., Jung H. and Uszok A. (2006).
A SEMANTICALLY RICH POLICY BASED APPROACH TO ROBOT CONTROL.
In *Proceedings of the Third International Conference on Informatics in Control, Automation and Robotics*, pages 318-325
DOI: 10.5220/0001219003180325
Copyright © SciTePress

derivative of XML, OWL provides standard, open, and extensible data representations as well as a powerful descriptive logic foundation that can be used to produce semantically rich descriptions of relationships between entities and actions. This provides the ability to reason about relationships between actions and create semantically rich policies.

Our approach is significantly different from mapping states to actions. We define a policy as:

"An enforceable, well-specified constraint on the performance of a machine-executable action by an actor in a given situation" (Bradshaw, 2004)

The basic components of a policy are an actor, an action, and a constraint. Authorities may dynamically impose or remove involuntary policy constraints on the actions of actors. Alternatively, actors may voluntarily enter into agreements that mutually bind them to some set of policies for the duration of the agreement. There are two types of policies; *authorization* and *obligation*. The set of permitted actions is determined by *authorization policies* that specify which actions an actor or set of actors is allowed (*positive authorizations* policies) or not allowed (*negative authorizations* policies) to perform in a given context. *Obligation policies* specify actions that an actor or set of actors is required to perform (*positive obligations*) or for which such a requirement is waived (*negative obligations*).

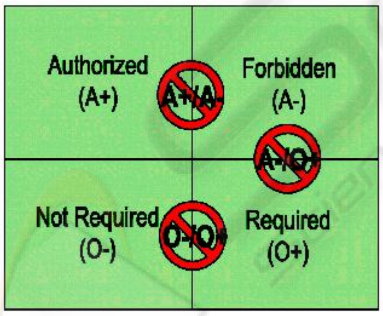

Figure 1: Policy modalities.

As an example, consider the simple obligations:

"Robot X is obligated to beep before it moves."

This could easily be handled by adding the pertinent code to the robot. A more flexible way would be to have a flag that an operator could set to enable or disable the beeping. There are still some limitations to this design. For example, it only applies to Robot X, and it only applies to beeping. Semantic policies provide a much higher level of expressiveness and flexibility. The operator could specify:

"Actors who are robots are obligated to warn before moving".

This is a much more general statement and would immediately apply to new robots added to the system without the operator explicitly setting flags on each one. It also allows each robot to warn in its own way (beep, flash lights etc.) based on its particular capabilities. The next section addresses how we implement and enforce general statements like the one used in our simple example.

3 HOW WE IMPLEMENT POLICIES

KAoS policies are written using concepts defined in *the KAoS Policy Ontology (KPO)* and are represented in OWL. KPO is a generic ontology, which contains the basic concepts such as actors, actions, and various generic properties needed to write policies. An diagram of the ontology describing a policy is shown in Figure 2.

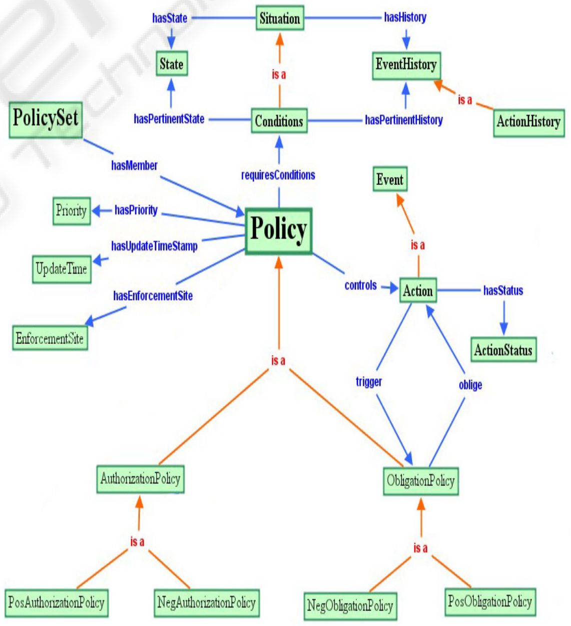

Figure 2: Policy Ontology Diagram.

We have also extended this ontology with concepts specific to the mobile robot domain. The KAoS robot ontology contains descriptions of the various robot classes and their capabilities such as sonar, camera, and mobile bases. There is no requirement that every action that a robot may take be represented in the ontology; only those that are potentially subject to policy constraints.

OWL provides a rich descriptive language, but at a cost of complexity. In order to free users from this complexity we have created a graphical interface that allows users to specify policies without any knowledge of OWL. As the complexity and number of policies increase, it is likely that conflicts between policies will exist. These conflicts can be difficult or impossible to find if the constraints are coded into the system. However, KAoS uses the Stanford Java Theorem Prover (JTP: http://www.ksl.stanford.edu/software/JTP) to provide conflict detection, as well as automatic conflict resolution in some cases. Even if the conflict cannot be automatically resolved, the system can still notify the administrator and direct the policy creators to reconsider the problematic policies in order to find a solution.

In order to enforce policies on a system, it must be possible to screen the actions as they are attempted, translate them into the representation of the policy language, check the action against policies and then enforce the result. Robotic Systems, being mobile in nature, usually rely on distributed communication. This is particularly true in multi-robot control. We leverage this in our implementation to provide policy based control of robotic systems.

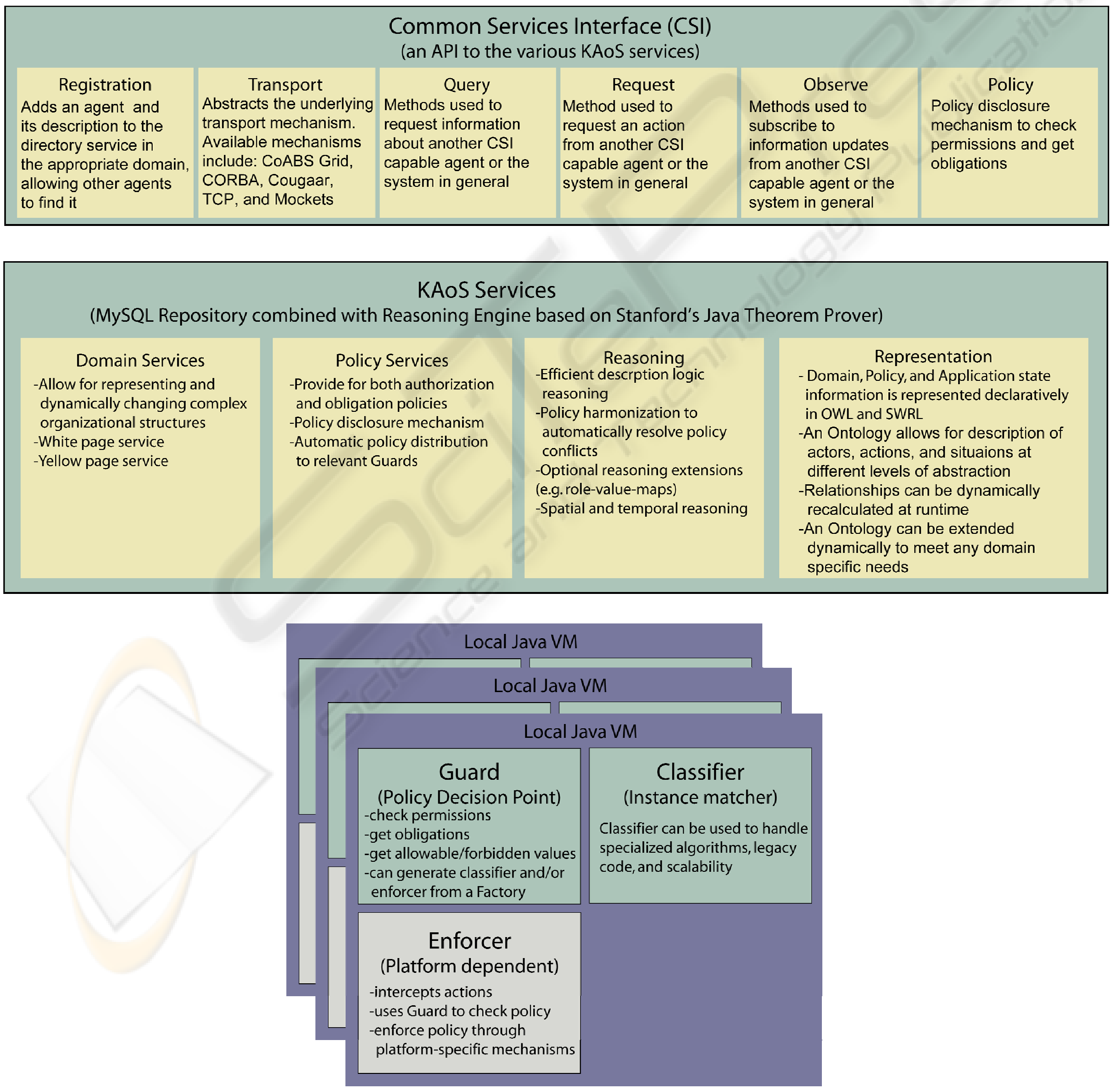
Common Services Interface (CSI)
(an API to the various KAoS services)
Registration
Adds an agent and its description to the directory service in the appropriate domain, allowing other agents to find it
Transport
Abstracts the underlying transport mechanism. Available mechanisms include: CoABS Grid, CORBA, Cougaar, TCP, and Mockets
Query
Methods used to request information about another CSI capable agent or the system in general
Request
Method used to request an action from another CSI capable agent or the system in general
Observe
Methods used to subscribe to information updates from another CSI capable agent or the system in general
Policy
Policy disclosure mechanism to check permissions and get obligations
KAoS Services
(MySQL Repository combined with Reasoning Engine based on Stanford's Java Theorem Prover)
Domain Services
-Allow for representing and dynamically changing complex organizational structures
-White page service
-Yellow page service
Policy Services
-Provide for both authorization and obligation policies
-Policy disclosure mechanism
-Automatic policy distribution to relevant Guards
Reasoning
-Efficient descrption logic reasoning
-Policy harmonization to automatically resolve policy conflicts
-Optional reasoning extensions (e.g. role-value-maps)
-Spatial and temporal reasoning
Representation
- Domain, Policy, and Application state information is represented declaratively in OWL and SWRL
-An Ontology allows for description of actors, actions, and situaions at different levels of abstraction
-Relationships can be dynamically recalculated at runtime
-An Ontology can be extended dynamically to meet any domain specific needs
Local Java VM
Local Java VM
Local Java VM
Guard
(Policy Decision Point)
-check permissions
-get obligations
-get allowable/forbidden values
-can generate classifier and/or enforcer from a Factory
Classifier
(Instance matcher)
Classifier can be used to handle specialized algorithms, legacy code, and scalability
Enforcer
(Platform dependent)
-intercepts actions
-uses Guard to check policy
-enforce policy through platform-specific mechanisms

Figure 3: KAoS Architecture.

3.1 KAoS Background

KAoS is a collection of generic domain and policy services that have been successful in providing policy based management for various platforms (Suri, 2003) (Tonti, 2003) (Johnson, 2003). KAoS also provides access to a distributed communication mechanism. It also has an extension that allows external systems to control multiple heterogeneous robots. Figure 3 depicts the main services provided by KAoS. Domain services allow for the grouping of entities to facilitate policy creation, making the use of policies scalable and efficient. Policy services enable the creation, management, and disclosure of policies. In addition to storing policy data centrally, KAoS performs automatic policy distribution to the local robot platforms. Each platform contains a *Guard*, which is a KAoS component designed to cache policies locally and to handle local policy checking.

The *Common Services Interface (CSI)* is a software interface to various services including registration, transport, query, request, subscribe, and policy disclosure as shown in Figure 3. These services are available to any entities desiring to interact with the robots including internal components like a deliberative layer and other separate, external agent/robotic systems. The CSI layer provides a consistent approach for accessing a robot regardless of whether the code is running locally with respect to the robot hardware or remotely. Our *KAoS Robot Interface* provides the back-end implementation to enforce policy constraints on robotic systems that have a native adapter available.

Each robot typically comes with its own proprietary and/or unique native software interface for controlling it. Therefore, all heterogeneous robot control architectures will require a translation layer between the common language and the native programming language, and our system is no different. Our mobile robot extension to the KAoS Policy Ontology is the common language. The robot developers will need a thin translation layer in the native implementation. This layer is where the ontological strings are converted to the various platform specific calls. In order to provide hardware abstraction, KAoS has defined a set of interfaces for various components, such as a mobile base, sonar, and a camera, similar to Player (Gerkey, 2003). By implementing an interface a robotic system can be accessed through CSI. Several popular robotic platforms are supported as shown in figure 3, including the Pioneer, Amigo, Evolution Robotics ER-1, and Player. Building algorithms on top of these interfaces makes the algorithm available to all robots that implement them. It also allows policy checking at multiple levels of abstraction. This thin native adapter enables *KAoS Robot Interface* to provide policy enforcement on the given platform.
KAoS makes use of the communication layer in order to provide policy enforcement. It is as minimally invasive as possible and only screens actions that relate to current policies. Actions which do not relate to any current policies pass through unaffected and without any significant delay. Actions that require policy checks incur a small delay on the order of ten milliseconds. Once a robot has its native translation layer, it can be subject to policy constraints without any further modification.

Lastly, we briefly mention Kaa (KAoS adjustable autonomy). Kaa is software component used to perform limited automatic adjustments of autonomy consistent with human-defined policy. Kaa is designed to use influence-diagram-based decision-theoretic algorithms to determine what if any changes should be made to agent autonomy.

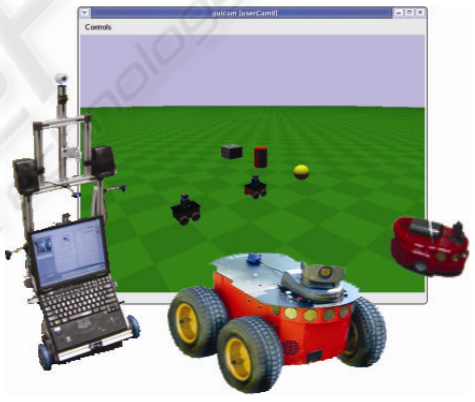

Figure 4: Platforms with policy implementations.

4 APPLYING POLICIES TO ROBOT SYSTEMS

We have demonstrated the use of KAoS policies in multi-robot systems in three different contexts; a NASA Human-Robot Teamwork (HRT) project, an Office of Naval Research (ONR) Naval Automation and Information management project and a Transportation Security Robots (TSR) project. A brief description of each is provided next since they will be referenced as examples.

4.1 NASA Application

The NASA HRT work involves a MARS exploration scenario including an astronaut on a simulated Extra-Vehicular Activity who had two robots nearby. The focus of this project was human-robot teamwork. As such, the policies implemented reflect teamwork issues like task switching, delegation and notification. The scenario began with the astronaut trying to task one of the robots to take a picture of him.

Our first policy prohibited the tasked robot to move due to some previous tasking. This caused the "take a picture" action to fail because the astronaut was remote to the robot and the task would require the robot to move into range before taking the picture. This example demonstrates a negative authorization policy being imposed on real robots. It also highlighted translation across levels of abstraction, since the restricted action is a subtask of the high level action. The failure to take a picture triggered an obligation policy that required notification of the astronaut about the failure. The failure also triggered a second obligation policy. This policy obligated the robot to try to delegate the "take a picture" action to another entity with the necessary capabilities, in this case the second robot. The obligation included getting verification from the astronaut that delegation was acceptable. We also included the simple example of obligating robots to provide a warning before moving. When the second robot was tasked to take the picture through delegation, its attempt to move triggered an obligation to beep before moving. In this scenario, our robots were only required to implement some very rudimentary behaviors of moving, beeping, and taking a picture. All of the teamwork issues were handled externally through policies created by the operator.

4.2 ONR Application

Our project for the Office of Naval Research (ONR) was based on finding a clear lane of approach in preparation for an amphibious landing. The scenario involved a single human managing four remote Pioneer 3-AT robots. The human interacted with the robots through a multi-modal dialogue system (Chambers, 2005). The robots coordinated with each other throughout the mission to divide their task and update status. This scenario also demonstrated the breadth of our system, including matchmaking based on capability, simultaneous multi-robot control, authorization and obligation enforcement, robot-robot collaboration, mixed initiative interaction, and dynamic policy updates.

Policies first came into play when a robot found an object and was unable to sufficiently classify it. The robot was obligated to obtain classification assistance. We also used policies to determine if robots were allowed to change roles. This project also demonstrated a new infrastructure for proactive ad hoc network maintenance. The robots were all initially assigned to find a clear lane, but if communications were lost, the network infrastructure had the ability to move the robots to regain communications. Since the change in behavior might alarm a supervisor or be detrimental to completing the overall task, we used policy to constrain the behavior and required approval from the human supervisor. We also demonstrated dynamic assignment of an area where the robots were prohibited from operating. The scenario was defined such that after a clear lane was found, a staging area for the landing craft would be established. For safety reasons, the returning robots were not allowed in this area. This policy was put in force dynamically during the demonstration. The resultant robot behavior successfully navigated around the restricted area while returning. Figure 5 shows the behavior during our demo. The restricted area, shown as a shaded red square, was graphically added to the picture to aid in visualization and the blue arrows indicate the paths chosen to avoid the area. The change in behavior was completely handled through the policy infrastructure and external to any robot implementation.

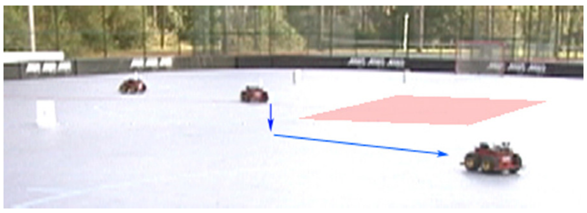

Figure 5: Restricted Area Policy.

4.3 TSR Application

For the Transportation Security Robots (TSR) project we focused on two potential uses for Kaa (KAoS adjustable autonomy). The first was overriding policy based on context. As in the ONR application, the operator could create a restricted area forbidding robot movement in a certain zone. However, there are certain exemptions that may apply, for example, "don't enter this area, unless there is an emergency". We used Kaa to monitor the system and determine when an emergency existed. When Kaa observed an emergency situation inside the restricted area, a fire for our example, it overrode the policy to allow robots capable of providing assistance to enter the area. The second use of Kaa was to adjust permissions based on context. Policy sets were established for the various Homeland Security Threat Levels, as posted by the U. S. government. Kaa was in charge of monitoring the threat levels and adjusting the policies accordingly. As the threat level moved from green to yellow to red, the actions the robots were authorized to take would vary accordingly.

5 ADVANTAGES OF USING POLICIES

Some of the characteristics of KAoS policies that make them useful for robot control are their powerful expressiveness, external nature, transparency, and flexibility.

5.1 Expressive Power

The choice of policy language directly impacts the expressiveness available in policies. We use OWL to provide declarative specification of policies at a broad range of levels. By combining this with JTP, we are able to reason about relationships and produce complex context sensitive policies. They can address the entire system, groups within the system or individual instances within the system. They can refer to actions at any level of abstraction and translate between levels. One example of this is in our NASA application, where the constraint was on movement, which was an optional subtask of the original “take a picture” action.” Most importantly, policies allow for a context to be explicitly defined, which helps to prevent over (or under) restricting the autonomy of the system. Anyone who has worked with robots has run into a constraint that required a “work-around”. Policies provide a mechanism to explicitly define the “work-around” solution based on context. Context can be any information, including things that the robot was never programmed to consider, such as time of day or outside temperature. As we have seen, one example occurred in our TSR work, where robots were prohibited from entering a restricted area, but the context of an emergency overrode the general policy.

5.2 External Nature of Policy

There are many ways to apply constraints to a robotic system. Typically they are coded into the behaviors or deliberative layer, hidden from the operator. Policies can be used to separate the behavioral constraints and preferences of operators from the underlying functionality. This is an idea that has been successful in many other areas such as database and web design. Because these different aspects of knowledge are decoupled, KAoS policies can be easily reused across different robots and in different situations. By putting the burden for policy analysis and enforcement on the infrastructure, rather than having to build such knowledge into each of the robots themselves, we minimize the implementation burden on developers and ensure that all robots operate within the bounds of policy constraints. This is demonstrated by our simple “Beep before moving” example discussed earlier and demonstrated in our NASA application. The obligation applied to all robots that were capable of moving and warning without needing to include the code for this behaviour in any individual robot.
Since policy enforcement is handled separate from the robot implementation, externally imposed policies provide an additional layer of controllability that is independent of the robot developer’s code. Poorly performing robots can be immediately brought into compliance with externally applied policy constraints, improving both controllability and safety. There have been instances in which an operator has had difficulty in overriding a faulty system that refused to yield control (Wolff, 1999). An accident (Williams, 2004) involving an Unmanned Aerial Vehicle (UAV) trying to taxi is a good example. Although the operators were very aware of taxi speed restrictions, the UAV had no such information, and it blindly followed an accidental command to taxi at 155 knots. Policies could be used in this situation to provide an additional layer of safety checks.

5.3 Transparency

The use of KAoS policies can also help to make the robot behavior more transparent. Again, constraints are made explicit, instead of being scattered and buried in the robot’s code. As robots move from the lab to the real world, it is unlikely that the operator of the robot will be the developer. Although developers have the know-how to tweak robot programs, operators are typically not as familiar with such inner workings. They should not have to be. Regardless of how the robot has been programmed, they would like guarantees that certain behaviors will be executed in ways that are consistent with their personal and organizational constraints. Often operators themselves lack sufficient situation awareness to properly control the system (Yanco, 2004). Transparency can help improve situational awareness. Our NASA example demonstrated how policies can be used to notify humans about changes and thus maintain situational awareness.

The benefits of transparency are not restricted to humans. Deliberative systems are also free to take advantage of the information available through policy disclosure mechanisms. Such information can be used to reason about the implication of policies and generate a more accurate model of behavior for the robot itself as well as for other robots or even humans. The transparency of policies can be used for planning purposes, resulting in more efficient plans by considering constraints. This can both reduce the search space and prevent futile actions from being attempted. For example in our TSR demo, robots where restricted from entering an area, however, an emergency situation presented itself. Kaa reasoned about the current policies and the current context and authorized a one time override of the policy for the given situation.

Finally, policies are viewable and verifiable. As systems grow, multiple constraints in complex systems can lead to unexpected (and possibly undetected) conflict. Often these oversights surface at very inopportune moments. Polices can be screened for conflicts prior to activation and in some cases can be automatically harmonized. More importantly, the policy creators can be informed of the problem, so they may take the best course of action.

5.4 Flexibility

One of the main advantages of using KAoS policies is that they are a means to dynamically regulate the behavior of a system without changing code. New constraints can be imposed at runtime and can be dynamically changed and updated as the environment or domain changes. Robotic behaviors are typically tuned to a specific domain application and must be re-tuned when the domain changes. Policies can be used to make these adjustments more transparent. For example, in open terrain, the maximum speed can be high, where as in wooded environments the maximum speed might be less. With policies, this could be accomplished by manually changing the policy, or more elegantly by specifying the appropriate context for each constraint. For example, the classification threshold for identifying an object in our ONR application could be varied based on the environment.

Policy flexibility can also be used to suit the system to the human, instead of solely training the human to the system. Through policy, people can precisely express bounds on autonomous behavior in a way that is consistent with their appraisal of an agent's competence in a given context. This provides a broad range of controllability, as well as allowing individuals to tailor the system to their needs. As trust increases, policy can be altered to allow greater autonomy. Again using our ONR example, policy provided a way to limit the robot's authority to classify objects based on the operator's assessment of its competence in the given context. As the operator's confidence in the robot's abilities increased, policies could be adjusted accordingly.

As a final note, policies need not be applied uniformly across all robots. If a particular robot is performing worse then others, a policy can be tailored for that individual robot. Policies can be applied within any organizational boundary defined in the ontology.

6 CONCLUSION

We have presented our work toward developing a viable approach to the application of semantically rich policy in robotic systems operating in the real world. Our next step is to develop quantitative metrics to evaluate the benefits of policy based control. There is research to indicate that a core set of teamwork policies exists and is reusable across domains (Tambe, 1997). Accordingly, we will seek to define reusable policy sets for a variety of areas.

Like any technology, policies can be misused or poorly implemented. An unsophisticated robot, insufficiently monitored and recklessly endowed with unbounded freedom, may pose a danger both to others and to itself. On the other hand, a capable robot shackled with too many constraints will never realize its full potential. The key to effective policy usage is experience in writing and testing policies. Fortunately, for this purpose policies provide a more transparent mechanism than typical robot control techniques. We can reason about policies, verify them, determine conflicts, and provide automated fixes in some circumstances. We believe that policy-based approaches hold much promise in improving robot control. Polices can also play a vital role in expressing and enforcing various social norms including those regarding human robot interaction (Feltovich, 2004) which will be increasingly important as robots begin to work along side humans. Semantically-rich policy representations like those used in KAoS provide expressive power needed for context sensitive policy expression and enforcement. There external nature of KAoS policies decouples the constraint specification from the underlying robotic implementation and as such is more transparent and flexible than standard approaches.

Policies of this type are not well suited for all aspects of robot control. Low level control with real time constraints is an example in which policy usage would not be appropriate. Our policies are most effectively applied in areas of human interest. People do not generally consider the low level control aspects when trying to work with a robot, but instead focus on the higher level aspects that generally do not have such sever real-time constraints. Policy mechanisms do not replace sound control theory or robot behavioral schemas, but supplement them and provide a much more intuitive way for operators to interact with and manage multiple robots. Our goal is to explore the conditions under which the use of policy services can be most beneficial in promoting effective humane-machine interaction.

REFERENCES

Allen, J. F., et al 2001. D.K. Byron, M. Dzikovska, G. Ferguson, L. Galescu, A. Stent, "Towards conversational human-computer interaction", *AI Magazine*, 22(4), 27-35.

Bradshaw, J. M., et al, 2004. "Making agents acceptable to people", *Intelligent Technologies for Information Analysis: Advances in Agents, Data Mining, and Statistical Learning*. (pp. 355-400). Berlin: Springer Verlag.

Carvalho, M., et al 2002. M. R. Breedy, "Supporting Flexible Data Feeds in Dynamic Sensor Grids through Mobile Agents", *Mobile Agent*, pp.171-185.

Chambers, N., 2005.James Allen, Lucian Galescu, and Hyuckchul Jung. A Dialogue-Based Approach to Multi-Robot Team Control. In Proceedings 3rd International Multi-Robot Systems Workshop. Washington, DC. March 14-16.

Feltovich, P.J., et al 2004. Bradshaw, J.M., Jeffers, R., Suri, N. & Uszok, A. (2004). Social order and adaptability in animal and human cultures as analogues for agent communities: Toward a policy-based approach. In A. Omacini, P. Petta, & J. Pitt (Eds.), Engineering societies for the agents world IV (pp.21-48). Lecture Notes in Computer Science Series. Heidelberg, Germany: Springer-Verlag.

Gerkey, B. et al. 2003. R. T. Vaughan and A. Howard. "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems" *Proceedings of the 11th International Conference on Advanced Robotics*, pages 317-323, Coimbra, Portugal, June 2003

Johnson, M., et al, 2003. "KAoS semantic policy and domain services: An application of DAML to Web services-based grid architectures", *Proceedings of the AAMAS 03 Workshop on Web Services and Agent-Based Engineering*, Melbourne, Australia.

Mataric, M., 2001. "Learning in behavior-based multi-robot systems: Policies, models, and other agents." Cognitive Systems Research, special issue on Multidisciplinary studies of multi-agent learning, 2(1):81--93, April.

Sierhuis, M., et al 2003. W. J. Clancey, and R. v. Hoof, "Brahms: a multi-agent modelling environment for simulating social phenomena", presented at *First conference of the European Social Simulation Association* (SIMSOC VI), Groningen, The Netherlands.

Suri, N. et al, 2003. "DAML-based policy enforcement for semantic data transformation and filtering in multi-agent systems", *Proceedings of the Autonomous Agents and Multi-Agent Systems Conference* Melbourne, Australia, New York, NY: ACM Press.

Tambe, M. 1997 Towards Flexible Teamwork Journal of Artificial Intelligence Research, Volume 7, Pages 83-124.

Tonti, G. et al, 2003. "Semantic Web languages for policy representation and reasoning: A comparison of KAoS, Rei, and Ponder", *International Semantic Web Conference*, Sanibel Island, Florida.

Williams, K. 2004. "Summary of Unmanned Aircraft Accident/Incident Data: Human Factors Implications", Technical Report.

Wolff, M., 1999 "The Automation debate revisited: Computer Clash", *Flight Safety Australia,* September-October.

Yanco, H., 2004. J. Drury, J. Scholtz, "Beyond Usability Evaluation: Analysis of Human-Robot Interaction at a Major Robotics Competition", *Journal of Human-Computer Interaction*.