Automated Generation of Enforcement Mechanisms for Semantically-rich Security Policies in Java-based Multi-Agent Systems

Gianluca Tonti 1,2, Rebecca Montanari1, Jeffrey M. Bradshaw2, Larry Bunch2, Renia Jeffers2, Niranjan Suri2, Andrzej Uszok2

1 Dipartimento di Elettronica, Informatica e Sistemistica (DEIS)
University of Bologna
Viale Risorgimento 2, 40136 Bologna - ITALY
{gtonti, rmontanari}@deis.unibo.it

2 Institute for Human and Machine Cognition (IHMC)
40 S. Alcaniz Street, Pensacola, FL 32502 - USA
{jbradshaw, rjeffers, lbunch, nsuri, auszok}@ihmc.us

Abstract

Policies are being increasingly used for controlling the behavior of complex systems (including agent systems). The use of policies allows administrators to specify agent permissions and obligations without changing source code or requiring the consent or cooperation of the entities being governed. Past approaches to policy representation have been restrictive in many ways. By way of contrast, semantically-rich policy representations can reduce human error, simplify policy analysis, reduce policy conflicts, and facilitate interoperability. However, semantically-rich policies increase the complexity of fielding policy-governed multi-agent systems. This paper discusses some technical challenges to automatically enforce semantically-rich security policies in Java-based multi-agent systems and presents an engineering approach for addressing some of these challenges. We have developed a first implementation that allows to enforce OWL policies represented using the KAoS policy framework into multi-agent systems built on top of the JDK1.4. The proposed solution allows to control the behavior of agents at a high level of abstraction and exploits the security mechanisms provided by the Java Authentication and Authorization Service (JAAS) to enforce OWL policies.

1. Introduction

The multi-agent paradigm offers a promising software engineering approach for the development of applications in complex environments [3; 12]. By their ability to operate autonomously without constant human supervision, agents can perform tasks that would be impractical or impossible using traditional software techniques [22; 1]. On the other hand, this additional autonomy, if unchecked, also has the potential of causing severe damage if agents are poorly designed, buggy, or malicious. The technical challenge is to assure that agents will always operate within the bounds of any behavioral constraints currently in force while remaining responsive to human control [4].

Explicit policies can help in dynamically regulating the behavior of agents and in maintaining an adequate level of security, predictability, and responsiveness to human control. By changing policies, the levels of agent autonomy can be continuously adjusted to accommodate variations in externally imposed constraints and environmental conditions without modifying the agent code or requiring the cooperation of the agents being governed [6].

A policy-based approach calls for a policy model specifying how agent permissions and obligations can be expressed and for an enforcement model supporting dynamic control of agent behavior according to desired policies. A few research activities have emerged that propose semantically-rich policy-based approaches to the control of agent systems [8; 14]. Most proposals focus on the problem of policy definition by recognizing the need for the adoption of semantically-rich policy representations [24]. In contrast, relatively little attention has been paid to building general infrastructure-based mechanisms that can monitor and govern the behavior of agent systems.

The development of enforcement mechanisms for semantically-rich policy in agent systems raises several challenges. Semantically-rich policy specifications can be difficult to implement because their high-level descriptions can be far from the concrete implementation details required by policy enforcement components. The gap between specification and implementation of policies has to be resolved to a greater or lesser degree by human programmers, consistently with the capabilities and features of each platform. The mapping of specification to implementation usually requires ad-hoc platform-specific solutions developed each time from scratch and hardly reusable.