Chapter XXIV
Mapping Policies to Web Rules:
A Case of the KAoS Policy Language

Nima Kaviani
University of British Columbia, Canada

Dragan Gašević
Athabasca University, Canada

Marek Hatala
Simon Fraser University, Canada

ABSTRACT

Web rule languages have recently emerged to enable different parties with different business rules and policy languages to exchange their rules and policies. Describing the concepts of a domain through using vocabularies is another feature supported by Web rule languages. Combination of these two properties makes web rule languages appropriate mediums to make a hybrid representation of both context and rules of a policy-aware system. On the other hand, policies in the domain of autonomous computing are enablers to dynamically regulate the behaviour of a system without any need to interfere with the internal code of the system. Knowing that policies are also defined through rules and facts, Web rules and policy languages come to a point of agreement, where policies can be defined through using web rules. This chapter focuses on analyzing some of the most known policy languages (especially, KAoS policy language) and describes the mappings from the concepts for KAoS policy language to those of REWERSE Rule Markup Language (R2ML), one of the two proposals to Web rule languages.

INTRODUCTION AND MOTIVATION

Rules are among the most frequently used knowledge representation techniques. They can generally be categorized as *reaction rules* (event-condition-actions), *integrity rules* (rules of consistency checking), *derivation rules* (implicitly, derived rules), and *production rules* (Boley, Tabet, & Wagner, 2001). *Facts* can also be regarded as derivation rules with no premises.

Recently, rule markup languages have started to be considered as the vehicle for using rules on the Web and in other distributed systems, forming a new category of rule languages, referred to as

Web rule languages (Wagner, Giurca, & Lukichev, 2006). The main strength of markup languages is their machine readability combined with their inherent potentials to easily be linked to distributed knowledge sources also represented in the form of markup languages (e.g., Data Type Definition (DTD), XML Schema (Fallside & Walmsley, 2004), Resource Description Framework (RDF) (Lassila & Swick, 1999), and Web Ontology Language (OWL) (Smith, Welty, & McGuinness, 2004)). Rule markup languages allow for the reuse, interchange, and publication of rules as well as their communication and execution in a distributed environment. More specifically, rule markup languages allow for specifying business rules as modular, standalone, units in a declarative way, and publishing and interchanging them between different systems and tools (Wagner, Giurca, & Lukichev, 2006). Rule markup languages play an important role in facilitating business-to-customer (B2C) and business-to-business (B2B) interactions over the Internet by enabling information exchange across various stakeholders. For example, they may be used to express derivation rules for enriching Web ontologies by adding definitions of derived concepts, or for defining data access permissions; to describe and publish the reactive behaviour of a system in the form of reaction rules; and to provide a complete XML-based specification of a software agent (Wagner, Giurca, & Lukichev, 2005).

RuleML and *REWERSE Rule Markup Language (R2ML)* are two newly and rapidly emerging Web rule languages that help with interchanging various types of rules from one rule domain to another. RuleML represents an initiative for creating a general rule markup language that will support different types of rules and different semantics (Boley, Tabet, & Wagner, 2001). R2ML more or less follows a similar idea to RuleML; however, its design follows Model Driven Architecture (MDA) defined by the Object Management Group (OMG) (Miller & Mukerji, 2003). Moreover, R2ML is a trial to cover a more comprehensive set of atoms and elements required to develop and define rules, thus bringing more flexibility to rule definition. These rule languages are designed to be conformant and compatible with the guidelines and use cases defined by the Rule Interchange Format (RIF) working group (Ginsberg, Hirtle, McCabe, & Patranjan, 2006).

Policies in the domain of autonomous computing are guiding plans that restrict the behaviour of autonomous agents in accessing the resources (Toninelli, Bradshaw, Kagal, & Montanari, 2005). The main advantage in using policies is the possibility to dynamically change the behaviour of the system by adjusting the policies without interfering with the internal code of the system. They can authorize/oblige the users to, or prohibit/dispense them from, accessing the resources or taking particular actions in the system. Policy languages can be considered as instructional sets that enable phrasing and putting systematic guidelines in place for a target agent. KAoS (Uszok, et al., 2003) is one of the most known policy languages that goes beyond the traditional policy systems by giving special care to the context to which the policies are applied. This is done by enabling these policy languages to use domain knowledge (a.k.a, vocabularies) readily available on the Internet and represented in knowledge representation markup languages such as XML-Schemas, RDF, and OWL.

Knowing that policies are also defined through the use of rules and facts that can share online knowledge bases, Web rules and policy languages come to a point of agreement. Web rules and policies get even closer bringing it into the consideration that most of the newly emerging policy languages also have chosen markup syntax for their specifications (e.g., KAoS). Following the hierarchy of rules represented earlier, a policy rule can be considered a reaction rule which generates an action (possibly permission or denial) upon the occurrence of an event and satisfaction of a series of conditions. On the other hand, a policy rule can be considered a derivation rule that