RULE LANGUAGES FOR INTEROPERABILITY

Bob McWhirter (bob@openxource.com), OpenXource, LLC

LANGUAGES VERSUS SYNTAXES

Languages and their concrete syntaxes should be considered separate and different. Rule-engines that use domain-specific languages and other rule conceptualization constructs particularly may demonstrate the disjunction between the two. A standardized syntax without a standardized language provides little value in meeting the goals of this workshop.

PRIOR ART & PRECEDENT

Being a W3C workshop, it is assumed that an XML-based approach is a given. XML is pervasive and many existing tools and libraries support the basic techniques of document manipulation.

Looking at the specifications that surround XML itself, I suggest that we evaluate following a model similar to that of the XML, XML-InfoSet and XML-Namespaces Recommendations. The formal syntax should be defined in XSD.

RULE-INFOSET

XML-InfoSet helps to provide an abstract way of viewing the core data within a given concrete XML document. XPath and XML-Canonicalization Recommendations are based around InfoSet manipulation. InfoSet is ostensibly the language of XML, while octect streams containing pointy-brackets represent usages of the syntax of XML.

In this line, before defining a syntax for a rule language, the rule language itself should be created. We should aim to define the language in terms of entities, components and relationships. The business rule community already has an established to discuss rules from which we can draw. Sub-communities, such as inference vs. ECA rules, forward-chaining vs. backwards-chaining, each have their own additions and changes to the basic language of rules.

Since each community has a history and a valid reason for their own dialects of the general Platonic rule language, their sub-languages should be supportable. Many times the semantic differences in seemingly equivalent terms across dialects truly reflect a significant feature. Maintaining this goal also helps avoid "design by committee" where all viewpoints are compromised resulting in a mediocre product. Supporting sub-language dialects should also allow easy support for domain-specific language constructs.

RULE-EXPRESSION

In addition to simply expressing the model of a rule, the semantic content within each entity may be expressible in a common fashion. General forms of relationships can expressed in all languages. Operators such as "is less than," "is equal to," and "matches"

are easily to genericize across rule systems. Pulling from perhaps the XPath expression language or SQL, a rule-centric relational language should be defined against the Rule-InfoSet.

Rule Syntax

After the Rule-InfoSet and Rule-Expression languages are defined, mapping them onto a pointy-bracket XML-based syntax should be relatively straightforward. The structure of the InfoSet model will lend itself to a particular XML structure.

Weaknesses

A single language and syntax that can be read by any system may be a false Holy Grail. Each rule-system has its own strengths and weaknesses. Having a completely compatible language would neutralize these differences. Each system should be encouraged to shine in its own feature-set. The resulting language and syntax defined will probably contain several opaque portions not easily or commonly handled by all systems. The result may be simply attribute values that have no semantic meaning outside of a specific system. It may result in entire sub-trees of the XML document are engine-specific. This should not be viewed as a failing.

Benefits

By creating an InfoSet for rule languages, generic tools can be created to perform high-level functions such as acting as a repository/deployer, performing general rule relationship analysis, and transformations can be written in a system-independent manner.

InfoSet bindings for different programming languages can be created (similar to the DOM), allowing for reusable generic rule-manipulation libraries.