

Standard Business Rules Language: why and how? ICAI'06

M. Diouf K. Musumbu S. Maabout
LaBRI (UMR 5800 du CNRS),
351, cours de la Libération, F-33.405 TALENCE Cedex
e-mail: {diouf, musumbu, maabout}@labri.fr

Abstract

In each business (enterprise) a decisional mechanism exists. Traditionally, when we want this mechanism to be automated, we have to hardly code it into applications. This makes maintenance cumbersome and expensive, in time and money. So the code become fixed and interdependent so that an evolutionary maintenance is difficult, even impossible. As business rules represent business's logic and not system's logic, the great challenge is to transfer the control of business rules from IT's persons (developers) to the business experts.

Keywords: *business rules, rule engine, JSR 94, RuleML.*

1 Introduction

Whatever is the technique used to visualize business rules (Decision table or tree viewer), it is necessary to have a language which makes it possible to specify the syntax and the semantics of the rules. A rule's language must be intuitive for business users, so that they can take part in the development, with test and modification's process. This can reduce implementation's duration and mistakes of interpretation between the intention (will) of the business and its realization.

To specify business rules, we must respect several preconditions. The first and most important one is to be completely independent of the mechanisms used to handle the data and to carry out the actions. Currently the limitation of business rules occurs when a company wishes to use another rule engine, the rules have to be rewritten in the new system's suitable implementation. The cost to pay justifies the need to have a standard language.

2 Why a rule standard language is it needed?

Let us suppose that we use Drools [1], a rule engine with the drools rule language formalism (DRL). Let us name it r_1 . Imagine we have 3000 rules in our knowledge base in DRL format. Again suppose that one day, reaching the limitation of Drools we have no other solution than changing the rules engine. The technical team chooses Jrules [3] with the Ilog rule language formalism (IRL). For the migration we need a system mapping M , such that $r_2 = M(r_1)$, transforming DRL to IRL. Now we have to answer to the following questions:

- Does such direct mapping available?
- By combining other mappings, can we get such mapping? E.g. does exist mappings M_1 and M_2 so that: $r_2 = M_1(r'_1)$, $r'_1 = M_2(r_1)$?
- We know that such mapping doesn't exist and no combination of other available mappings will give it. In this case, can we create it?

Presently there exist many rules engines and no mapping systems exist for any pairs of them. In the case of IRL for Jrules, like all commercial rule engines, this format is proprietary. So a customer cannot have access to metadata for creating a mapping. Thus the solution for using Jrules with our rules is to rewrite all of them from DRL to IRL. Suppose that we are in an integration case of many rules sources using different rules languages, the problem is harder. Indeed, if we have n rules engines with resp. r_1, r_2, \dots, r_n as rules languages, we will need $n(n - 1)$ translations because every one must have one way to be translated to the whole other formats. This is not acceptable regarding the complexity point of view.

One other solution would be to have a standard language, to create mapping from this one to the other formalisms and vice versa. The complexity falls from $n(n - 1)$ to $2n$. The Java Community Process has created an API allowing moving in a very transparent way from one rule engine to another by simply changing a line in the code. However that API doesn't tell anything about business rules interchange.

3 The JSR 94

The JSR 94 [4] allows moving in a very transparent way from one rule engine towards another one by simply changing one line in the code. The JSR 94 makes it possible to harmonize “the populating” of the working memory, rule's insertion in the agenda, rule's execution and the interrogation of the working memory state. However the JSR 94 does not say absolutely anything concerning the formalism of rules themselves, which is the hard core of the problem. Most of rules engines are JSR 94 compliance (moreover when a company thinks of investing in an engine this must be a characteristic to be sought) and all engine vendors are agreed with the utility of implementing a standard language for modeling business rules.

For having a standard language it is necessary that a known and recognized organization, like the W3C or the OMG, drives discussion. For a long time the OMG harnesses

to problem with at the key a document trying to put everyone agree on the definition of terms used in business rules. The OMG is in a advanced stage for proposing a rule language known as SBVR for Semantic Business Vocabulary and business Rules [5]. The W3C also cares about this especially with the organization of a meeting from the April 27 to 28, 2005 with the participation of many actors of the domain. After this conference a workgroup in this topic [7] was born. These organisms must work with other companies and industries for facilitating the future formalism's adoption. Many other tentatives was born like SBRL, CommonRules, RuleML, etc without success.

4 The RuleML initiative

The RuleML's initiative [6] goal is to propose a standard language for business rule's representation to consortium W3C. With times other very important issues and directions were born. In itself, RuleML is a representation based on XML for rule, which was to be interoperable on the major part of the commercial rule's system. It supports 4 types of rule's system: SQL (relational data bases), Prolog, production rules (OPS5, CLIPS, Jess, Jrules, Blaze, Mandarax, Drools) and (event-condition-action) rules ECA. It's important to know that RuleML supports both backward and forward chaining. Along time the XSD and DTD of RuleML evolved much to allow more expressiveness and so became more complex.

4.1 The initiative's advantages

While attempting to solve the task of rule's interoperability, the RuleML's designers were interested closely in much principles and problems in this field. They acquired great experience and extremely useful know-how. RuleML is very flexible with the use of XML and progressively much functionalities were added: Or predicates, the two negations (classical/strong, as failure), term's typing.

RuleML is more and more used in rule systems and the team works in collaboration with the W3C although it is not recognized yet as standard. RuleML is not limited only to propose a language but also "translators" for some targeted rules engines. (ex: RuleML → JESS). RuleML is available in several formats, DTD for the first versions, XSD and RDF.

4.2 The initiative's disadvantages

RuleML is extremely complex. That is the result of wanting to do many things at the same time. Handling backward and forward chaining is not a small matter. It should be known that this complexity is growing with the language's functionalities. The problem with RuleML is really not at this level, it is rather the fact that there is no large realization around it, in this we mean, a very good rule's editor and much of exploitation's tools behind. Admittedly there is Mandarax with its graphic editor which allows storing rules directly in RuleML also some others. However the RuleML's

version which is inside (Oryx 3.7) is far from being up to date (no management of the negation nor disjunction). It's very simple to lose itself among the various formats and versions.

5 The characteristics which must have a future standard language

A standard business rule's modeling language should have the following characteristics:

- Based on an XML like language. All integrated architectures of Information systems now use this language systematically. The definition of a language in XML is then essential to communicate and exchange business information in order to be correctly interpreted by several systems (rule engine). Moreover all the power around XML could be made profitable (parser, binding). XML is very flexible for referring data models.
- The language should be as simple as possible for easy integration and to be adopted. For that the language must not have the claim to do everything at the same time (formalizing both forward and backward rules).
- The language should be based on what exists: for example most rule engines implement the strong negation and it is very interesting thus the future language have to integrate it.
- Allow a limitation/extension i.e. companies have the flexibility to use or not the full functionality proposed by the standard and possibly they can extend it to achieve their own needs.
- Integrate the concept of rules packaging for a better management and switch of the ruleflow.
- Have a readable syntax for good comprehension: to understand and assimilate more quickly the language, since an understanding syntax by the common people is necessary. Use of semantics Web for the semantics of syntax definition.
- And for accelerating the rules language adoption, good editors in natural languages must be proposed early.

6 Proposed Solution

Due to the lack of homogeneous and interoperable solution and in order to keep their autonomy, companies have to think about their own language's implementation and use translators from a such language towards a targeted engine, waiting a standard rules language's birth.

The Java/XML couple offers serious options for having something in a correct time.

We used XML schema (XSD) for modeling the rule language (an other same kind technology could be used). XSD offers much flexibility for grammar's implementation and a whole arsenal of tools exists to make it's treatment (compilation in java, validation, parsing). After this we used XSLT technology for making mappings ("translators").

Example:

- OSRML (Own Standard Rule Markup Language) to Drools.
- OSRML to JRules.
- OSRML to Drools.
- OSRML to any other engine.

Processing XML document manually is very cumbersome so, we used JAXB [2] for binding XML model in Object model. Manipulating Java objects is less heavy and less dangerous than manipulating XML document which can be large. It is much simpler to write a "translator" than to rewrite thousand of rules from the IRL's format (JRules) towards DRL's format (Drools). Thus the coupling between the enterprize and a rule engine's vendor will be strongly decreased. When a standard business rule language will be accepted, it will be quite simply to make a "translator" OSRML towards it. The enterprize, which will proceed like this, will strongly gain in autonomy. The rule's creation and edition graphical interface will store all rules in the own format (here for example OSRML) and it's only when rules will be executed that will be determined which "translator" to use (see Figure 1).

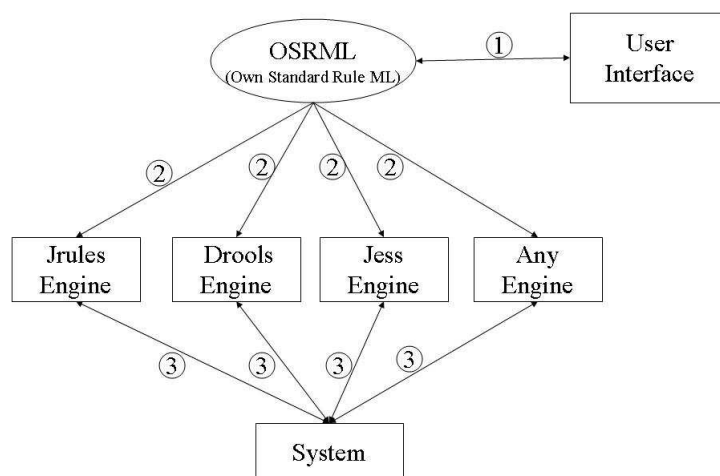


Figure 1: Architecture of our solution

At 1. Business experts use the editor for creating rules which will be stoked in OSRML format.

At 2. At runtime, rules in OSWRL format will be transformed in targeted rule engine format.

At 3. Application use the rule engine for processing knowledge management.

7 Summary

A standard language must be quickly established to be adopted because with the flowering of rule engines, more the time will be long, more there will be engines and more consensus work will be difficult. It is clear that this language should be based on what already exists and standards. The experience that RuleML has acquired along the initiative will be very useful as they said during their participation at the conference organized by the W3C in April 2005. It could also be possible to be based on other attempts like CommonRules where, although having not succeeded in being adopted as standard, good things had followed from. Waiting this language to be born, the users of business rules oriented systems may have their own rule's formalism and create "translators" according to the target engine for a complete autonomy from any vendors.

The business rules approach becomes more and more used because the principle of separating business logic and system logic. Furthermore, it provides a way for business experts to access directly the system's behavior in a zero-development environment (natural language), facilitates maintenance, extensions and reduces legacy code.

References

- [1] Drools. Drools rule engine, <http://www.drools.org>.
- [2] Sun JAXB. Jaxb, <http://java.sun.com/webservices/jaxb/>.
- [3] Ilog Jrules. Ilog jrules, <http://www.ilog.com>.
- [4] Java Community Process. JSR94. Jsr94 api for business rules, <http://www.jcp.org/en/jsr/detail?id=94>.
- [5] OMG. Business semantics of business rules rfp br/2003-06-03. Object Management Group, 2003.
- [6] RuleML. The ruleml initiative, <http://www.ruleml.org>.
- [7] W3C. Rule interchange format workgroup, <http://www.w3.org/2005/rules/>.