An Integrated Rule Modeling Framework

Sergey Lukichev¹, Adrian Giurca¹ and Gerd Wagner¹

¹Institute of Informatics, Brandenburg Technical University at Cottbus, Germany
{lukichev, giurca, wagner}@tu-cottbus.de

Abstract: This paper introduces an Integrated Rule Modeling Framework, which consists of the UML-based rule modeling tool Strelka, rule interchange format R2ML and a number of services, which facilitate the development and debugging of rule-based applications and rules for the Semantic Web. We describe how the framework can be used by software engineers and how Semantic Web rule modelers can benefit from using our rule interchange services.

1 Introduction

Rules play an important role in the Semantic Web and in the early-adoption of knowledge-based techniques in e-business. But an issue of platform-independent rule modeling and knowledge integration by means of rule interchange is not yet well studied. We address the issue of platform-independent rule modeling by providing UML-based Rule Modeling Language (URML), which allows rule modeling on top of UML class diagrams (see Section 2.1). URML is supported by the Strelka tool. In order to address the issue of rule deployment into execution platforms and loss-free rule interchange, we provide REWERSE I Rule Markup Language R2ML (see Section 2.2). In addition, we have developed a number of services, which are based on top of R2ML, integrated in Strelka and perform rule visualization, verbalization and validation. The rule modeling tool, the rule markup language and the additional services define an Integrated Rule Modeling Framework, which can be used for development of rule-based applications in e-commerce and for the Semantic Web.

Almost every commercial rule platform provides its own rule modeling methodology and a tool. There are works on UML profile for rules, for instance [BHHS06], which allows modeling of derivation rules, expressed in the Semantic Web Rule Language (SWRL). The distinct point of our UML-based approach is that with a relatively small and simple extension of the UML metamodel, we implement modeling of main rule types: derivation rules, production rules, and reaction rules. Moreover, it gives rule modelers an instrument to develop rules independently from a particular rule platform. URML has forward compatibility with R2ML and the deployment of URML models is done by the Strelka tool, which uses R2ML at the background and a number of web services, which translate from the R2ML into a particular rule language (Jena 2, JBoss Rules, etc).

To the best of our knowledge, there is no standard way to perform a loss-free transfer of rules from one platform to another. To address this problem, the REWERSE Working
Group I1 has developed a rule markup language R2ML for rule interchange. Later on, W3C has initiated a rule interchange standardization process, which up to now lead to the first W3C RIF Core draft\(^1\). The current version of RIF consists of a core only and cannot be used for the rule interchange. An actual interchange is going to be provided by means of extensions. Unfortunately, the RIF working group has not yet released rules for extending the core. But R2ML is RIF-compatible in a way it follows the same design principles and fulfills the same requirements. Therefore, R2ML can be considered as a first RIF extension, which performs actual rule interchange.

The Rule Modeling Framework supports two main scenarios: i) Rules are modeled in Strelka\(^2\) ([WL06], [LW06]), using URML\(^3\) ([WGL06], Section 1), then translated into R2ML ([WGL06]), and finally put into the target rule platform i.e. Strelka $\rightarrow$ R2ML $\rightarrow$ Rule Execution Platform; ii) Rule interchange between different platforms via R2ML\(^4\). The interchange is performed by R2ML translators. Up to now we have built translators from R2ML to F-Logic, Jess, Jena 2, RuleML, SWRL and JBoss Rules. Next steps involve the Object Constraint Language (OCL), Oracle Business Rules and ILOG JRules (see Figure 1).

\section{The Structure of the Rule Modeling Framework}

The structure of the Rule Modeling Framework is depicted on Figure 1. The framework consists of three layers, which are described in the subsequent sections: rule capturing and modeling layer, rule interchange layer and rule platforms layer.

---

\(^1\)W3C RIF Core Working Draft: http://www.w3.org/TR/rif-core/
\(^2\)Strelka homepage: http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=Strelka
\(^3\)URML homepage: http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=URML
\(^4\)R2ML homepage: http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=R2ML
2.1 Rule Capturing and Modeling Layer

This layer includes two main components: rule capturing from the natural language and rules modeling using the URML. In this paper we focus on rule modeling in Strelka, which supports URML. The tool is aimed at software engineers and UML modelers, who develop rule-based applications and have a basic knowledge of UML. Let us consider a business rule example, which defines a concept of a not eligible car according to the car’s potential occupant injury rating (Figure 2): If potential occupant injury rating of the car is high then this car is not eligible car. A circle denotes the rule, an incoming arrow from the class Car denotes the rule condition and an outgoing arrow to the class NotEligibleCar denotes the rule conclusion. For more information about visual notation we refer to [LW06]. In order to answer the question about usability of the URML and its acceptance by UML modelers, we refer to the case study [WLF07], where we use Strelka to model more than 50 rules for the car insurance company. A URML evaluation among students shows that modelers quickly become familiar with the visual notation and start modeling different kind of rules. Adopting the language is easy since it is based on well-known UML and introduces just a few new visual symbols. Our experience shows that it is easier for non-experienced users to learn and to use URML for rule modeling than to write and read Jena 2 or JBoss rules directly. Another advantage of using URML is that a modeler always sees a rule vocabulary, which simplifies the modeling process.

An issue of rules capturing from the natural language involves Attempto Controlled English (ACE) and the project is currently under development [WL07]. The goal is to write rules in ACE and deploy them into a rule execution platform via R2ML.

2.2 Rule Interchange Layer

The core of this layer is the REWERSE II Rule Markup Language (R2ML). R2ML is an XML Schema-based language and it supports derivation rules, production rules, reaction rules and integrity rules. R2ML has been designed to interchange rules between different rule platforms. It uses SWRL built-ins as predicates in atoms and XQuery functions.
as built-in functions in functional terms. A set of possible built-ins can be extended by importing a corresponding schema. The semantics of such imported built-ins must be preserved by translators.

Any rule diagram, created in Strelka, can be serialized into R2ML. The first serialization step from URML to R2ML is to markup the rule vocabulary i.e. the UML elements used in the rule. Then the tool generates the markup for the rule. Due to space limitation we provide only the conclusion markup of the rule from Figure 2:

```
<r2ml:conclusion>
  <r2ml:ObjectClassificationAtom r2ml:classID="ex:NotEligibleCar">
    <r2ml:ObjectVariable r2ml:name="car"/>
  </r2ml:ObjectClassificationAtom>
</r2ml:conclusion>
```

The rule conclusion consists of the R2ML object classification atom, which classifies the object variable "car" being of class "ex:NotEligibleCar". The compositional mapping from URML to R2ML is described in [WLG+06]. Since R2ML is a rule interchange format, having R2ML code for rules obtained from Strelka, an executional code for a particular rule platform can be generated, using R2ML translators. In addition to R2ML, this layer includes a number of services. The primary one is an interchange service5, which employs translators from/to R2ML. The interchange service is publicly available and any external application may connect to it and interchange rules. A number of supplementary services on top of R2ML includes rules verbalization6, visualization7 and validation (work in progress).

### 2.3 Rule Platform Layer

This layer consists of different rule execution platforms. A platform-specific representation of rules is obtained via R2ML translators. R2ML targets rule languages from three main rule platform categories: i) Expert Systems and AI Languages: Jess8, F-Logic9; ii) Semantic Web Languages, for instance, Jena 210; iii) Object Oriented Rule Languages: JBoss Rules11, ILOG12, Oracle Business Rules13.

Interchange services, which perform rule code generation for these platforms from R2ML, support also reverse translation, which allows rule interchange between platforms. Therefore, the URML rule from Figure 2 is translated into various rule languages (supporting derivation rules):

```
// Jena 2
(idl1: (?car rdf:type ex:NotEligibleCar)← (?car rdf:Type ex:Car)
```

---

5R2ML rule interchange WS: http://oxygen.informatik.tu-cottbus.de/reverse-i1/?q=ws  
6Verbalization: http://oxygen.informatik.tu-cottbus.de/verbalization/index.jsp  
7Visualization: http://oxygen.informatik.tu-cottbus.de/reverse-i1/?q=visualization  
8Jess: http://www.jessrules.com  
10Jena 2: http://jena.sourceforge.net/inference/  
12ILOG: http://www.ilog.com  
In addition, we provide R2ML translators from/to SWRL\textsuperscript{14} and from/to OCL\textsuperscript{15}, supporting interchange of integrity constraints.

3 Conclusion

The lesson learned from RDF is that ordinary people do not know exactly what RDF is about and what its formal semantics are. Rules should be easy to model and rule-based applications may use various rule engines. Users may use rules to empower their knowledge-based applications without precise understanding of rules theoretical foundations. A visual language for rules can be used by unexperienced web developers in a way that Prolog, Mercury, Haskell, Mozart/Oz and other logic programming languages may not need to be used. Therefore, the presented framework helps developing rule-based applications and makes complex rule development transparent to users because of the simplicity of the visual language. Of course, formal semantics and easiness of use are not necessarily mutually exclusive, but they occur together in very rare cases for non-professional users.

References


\textsuperscript{[WGL06]} Gerd Wagner, Adrian Giurca, and Sergey Lukichev. Language Improvements and Extensions. Deliverable i1-d8, REWERSE IST 506779, March 2006.


\textsuperscript{14}SWRL W3C Submission: http://www.w3.org/Submission/SWRL/

\textsuperscript{15}OMG OCL: http://www.omg.org/cgi-bin/doc?formal/2006-05-01