

Towards a Financial Service rule-based implementation using Jena and JBoss

Oana Nicolae¹ Ion Mircea Diaconescu¹ Adrian Giurca¹
Gerd Wagner¹

¹Brandenburgische Technische Universität Cottbus, Germany



The 7th International Conference AIDC' 2007
Craiova, Romania

Outline

1

Introduction

- About UServ rule-based Implementations
- UServ Product Derby Use Case 2005

2

UServ rule-based Implementation Steps

- Natural Language Representation
- Visual Rule Modeling using Strelka
- R2ML Serialization
- JBossRules Experience
- JenaRules Experience

3

Conclusions and Future Work

The Goals

The paper intends to be a practical guide for those interested on the serialization and deployment of a **Business Rules Model**, from **Natural Language description** based on core ontological concepts like classes and variables, to a target **Rule-Based Platform/Engine**.

UServ Product Derby Use Case

http://www.businessrulesforum.com/2005_Product_Derby.pdf

... and Achieved Results

The paper covers the basics steps followed by **I1 Rules Framework Model**, in order to achieve the UServ-2005 Business Rules Model translation into the **JBossRules v 3.0.6** and **JenaRules v 2.5.3** implementations, via **R2ML v 0.4** interchange language.

UServ Online Tutorial

<http://oxygen.informatik.tu-cottbus.de/reverse-11/?q=node/33>

Steps in UServ Implementation

To illustrate the translation process, we provide a business rule code example for each of next steps:

- 1 A **natural language description** from UServ Product Derby Use Case.
- 2 The **URML model** and **R2ML serialization** using Strelka.
- 3 The **JBossRules** and **JenaRules implementations** using R2MLToJBossRules translation and R2MLToJenaRules translators, respectively.

UServ Business Rules Model

- UServ's Business Rules are expressed in a natural English language.
- UServ enables a scenario which emulates a **complete vehicle insurance service**.
- UServ computes the UServ **annual premium for a vehicle insurance policy**, which belongs to an **eligible client**.
- UServ provides a concrete Business Rules Sets separation on distinct categories of rules.

UServ Business Rules Sets

- **Automobile Eligibility** - sets the eligibility category for a car.
- **Driver Eligibility** - sets the eligibility category for a driver.
- **Eligibility Scoring** - determines the client's eligibility category.
- **Automobile Premiums** - calculates the car premium.
- **Driver Premiums** - calculates the premium for particular driver.
- **Automobile Discounts** - lowers the car premium with specific percents, if the car has or not airbags or alarm system.
- **Market Discounts** - lowers the total premium based on client segmentation (elite or preferred).

Natural Language Representation

- From the natural language description view of the rule, we can identify all the objects referenced in the rule, its properties and their constraints.
- The natural language description of the rule also provide the rule general structure. It comprises the **rule name**, the **rule conditions** and the **rule conclusion**.
- The rule fits well into the typical paradigm of rule representation: **if conditions then action**.

Some Rule Example

Rule AE_PTC04

Rule AE_PTC04: If all of the following are true, then the car's Potential Theft Rating is moderate:

- car's price is between \$20000 and \$45000.
 - car model is not on the list of High Theft Probability Auto.
-
- The objects: car and car model.
 - Car object properties: car model - an object type property and price - a data type property.
 - Car model object properties: highTheftProbability - expresses the absence of the car model object from the High Theft Probability Auto List.
 - **Conclusion: UServ Business Rules Model needs a rule language, based on vocabulary, in order to be represented.**

View Rules Browser

<http://oxygen.informatik.tu-cottbus.de/userv/index.html>

URML Model

The UServ Product Derby Use Case rules are modeled as Production Rules.

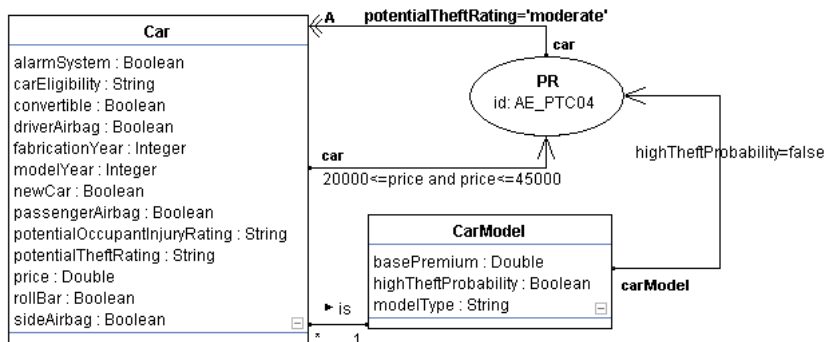


Figure: UServ Product Derby - rule AE_PTC04.

R2ML as interchange language

- Expresses rules which don't require any conceptual changes in order to be implemented in PSM target platforms (i.e JBossRules and JenaRules).
- Complies Web naming concepts like (URI and Namespaces), datatype concepts of RDF and ontological distinction between objects and data.
- Being an interchange language, R2ML addresses the PIM level (in OMG' MDA).

R2ML - REVERSE I1 Rule Markup Language

<http://oxygen.informatik.tu-cottbus.de/reverse-i1/?q=R2ML>

RIF - Rule Interchange Format

<http://www.w3.org/2005/rules/>

R2ML Vocabulary

Excerpt from Rule AE_PTC04 Vocabulary

```
<r2ml:RuleBase xmlns:r2ml="http://www.rewerse.net/I1/2006/R2ML"
xmlns:r2mlv="http://www.rewerse.net/I1/2006/R2ML/R2MLV"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://oxygen.informatik.tu-cottbus.de/R2ML/0.4/R2ML.xsd">
  <r2mlv:Vocabulary>
    <r2mlv:Class r2mlv:ID="Car">
      <r2mlv:Attribute r2mlv:ID="price">
        <r2mlv:range><r2mlv:Datatype r2mlv:ID="xs:integer"/></r2mlv:range>
      </r2mlv:Attribute>
      <r2mlv:Attribute r2mlv:ID="potentialTheftRating">
        <r2mlv:range><r2mlv:Datatype r2mlv:ID="xs:string"/></r2mlv:range>
      </r2mlv:Attribute>
    </r2mlv:Class>
  </r2mlv:Vocabulary>
  <!-- ... -->
</r2mlv:RuleBase>
```

R2ML - Mapping Conditions

Code Snippet I

```
<r2ml:DatatypePredicateAtom r2ml:datatypePredicateID="swrlb:lessThanOrEqual">
  <r2ml:dataArguments>
    <r2ml:TypedLiteral r2ml:datatypeID="xs:integer" r2ml:lexicalValue="20000"/>
    <r2ml:AttributeFunctionTerm r2ml:attributeID="userv:Car.price">
      <r2ml:contextArgument>
        <r2ml:ObjectVariable r2ml:name="car" r2ml:classID="userv:Car"/>
      </r2ml:contextArgument>
    </r2ml:AttributeFunctionTerm>
  </r2ml:dataArguments>
</r2ml:DatatypePredicateAtom>
```

Code Snippet II

```
<r2ml:AttributionAtom r2ml:attributeID="userv:CarModel.highTheftProbability">
  <r2ml:subject>
    <r2ml:ObjectVariable r2ml:name="carModel" r2ml:classID="userv:CarModel"/>
  </r2ml:subject>
  <r2ml:dataValue>
    <r2ml:TypedLiteral r2ml:lexicalValue="false" r2ml:datatypeID="xs:boolean"/>
  </r2ml:dataValue>
</r2ml:AttributionAtom>
```

R2ML Production Rules Actions

R2ML complies OMG PRR Specifications

- InvokeActionExpression - invokes an operation with a list of parameter arguments and changes the state of the context object.
- AssignActionExpression - assigns a value to the attribute of the object context.
- CreateActionExpression - creates an object from the list of property-value pairs.
- DeleteActionExpression - deletes a specific object.

Code Snippet

```
<r2ml:producedAction>
  <r2ml:AssignActionExpression r2ml:propertyID="userv:Car.potentialTheftRating">
    <r2ml:contextArgument>
      <r2ml:ObjectVariable r2ml:name="car" r2ml:classID="userv:Car"/>
    </r2ml:contextArgument>
    <r2ml:TypedLiteral r2ml:lexicalValue="moderate" r2ml:datatypeID="xs:string"/>
  </r2ml:AssignActionExpression>
</r2ml:producedAction>
```

Why JBossRules?

- Rule engine based on an enhanced version of Rete algorithm.
- Open-source, object-oriented production rules system written entirely in Java language.
- Separation of the business logic (the rules) from business data (the facts).
- The runtime provides dynamic assertion and remove of rules.
- Employs Conflict Resolutions like salience rule attribute and LIFO.
- Light and easy to understand syntax (i.e.DRL syntax) uses Java to express field constraints, functions and consequences.
- Easy to integrate with the mainstream JEE5 technologies.
- Collect complex decision-making logic and work with large data sets.

JBossRules

<http://labs.jboss.com/jbosrules/>

JBossRules Implementation

Code Snippet

```
rule "AE_PTC04"  
  when  
    $carModel:CarModel(highTheftProbability == false)  
    $car:Car(carModel == $carModel, price >= 20000, price <= 45000)  
  then  
    $car.setPotentialTheftRating("moderate");  
    modify($car);  
end
```

UServ Product Derby - JBossRules Implementation

<http://oxygen.informatik.tu-cottbus.de/userserv/jsp/jboss/index.jsp>

R2ML to JBossRules translator

<http://oxygen.informatik.tu-cottbus.de/translator/R2MLtoJBossRules/>

Why JenaRules?

- Open-source, Java framework for building Semantic Web applications
- Use RDF as data (desirable in Semantic Web since people works with Atom, Microformats, FOAF etc)
- Reasoner supports rule-based inference over RDF graphs and provides forward chaining (RETE), backward chaining (Datalog-based) and a hybrid execution model

JenaRules

<http://www.jena.sourceforge.net>

JenaRules Implementation

Code Snippet

```
[AE_PTC04:(?car rdf:type Car)
  (?car carModel ?carModel)
  (?carModel highTheftProbability 'false'^^xs:boolean)
  (?car price ?price)
  ge(?price,20000)
  le(?price,45000)
  ->
  (?car potentialTheftRating 'moderate')]
```

UServ Product Derby - JenaRules Implementation

<http://oxygen.informatik.tu-cottbus.de/userserv/jsp/jena/index.jsp>

R2ML to JenaRules translator

<http://oxygen.informatik.tu-cottbus.de/translator/Jena2/>

Conclusions and Future Work

Conclusions

- Business Rule Approach - provides the ability to respond and to quickly adapt to changes.
- Business Rules Vendors (i.e. Jess, ILog) have powerful competitors: open-source, Rete-based Rule Engines/Platforms (i.e. JBossRules, JenaRules)
- OMG's MDA (PRR for Standardisation), W3C (RIF), REVERSE (R2ML) - fight for Business Rules Standardization.
- R2ML is powerful enough to provide a platform independent syntax in order to further express the UServ production rules in two rule-based languages that came from distinct areas: Object-Oriented Rule Systems (i.e. JBossRules) and Semantic Web Rule Systems (i.e. JenaRules).

Future Work

- UServ 2006 Implementation based on R2ML 0.5, JenaRules 2.5.x and JBossRules 4.0.x

Thank you!

Questions?