
On Interchange between JBoss Rules and Jess

Oana Nicolae Adrian Giurca and Gerd Wagner

Department of Internet Technology
Institute of Informatics
Brandenburg Technical University at Cottbus, Germany
{nicolae, giurca, G.Wagner}@tu-cottbus.de

Summary. There is a growing demand for research to provide insights into challenges, and solutions based on business rules related to target PSM (Platform Specific Model in OMG's MDA terms - Implementation Model). As an answer to this needs, this paper argues on the relevance of business rules target platforms for the actual IT and business context, emphasizing the important role of business rules inter-operability initiatives, so that rule-system developers can do their work without concern about a vendor-specific format and in particular without concern about the compatibility between the technologies. This paper provides a description of the business rules translation from a particular rule-system such as JBoss Rules to another rule-system representation as Jess, using R2ML as interchange language.

Key words: : Business rules, JBoss Rules, Jess, Rete, ReteOO, R2ML, RIF.

1 Business rules and inter-operability initiatives

There is a growing request for business rules technology standardization from both UML modelers and ontology architects communities. For these reasons, business rules aim to express rules in a platform independent syntax.

A number of initiatives on rules inter-operability have been started. They include the RuleML [2], OMG Production Rules Representation [6], RIF [1], and the REVERSE I1 Rule Markup Language (R2ML¹) [8]. We mention here the efforts to establish some standards for expressing business rules and their vocabularies in natural language such as OMG's SBVR [7] and Attempto Controlled English (ACE) [4]. SBVR, this human readable format of business rules comes under OMG's Model Driven Architecture (MDA²) standards and is defined as Computation-Independent Model (CIM³). The second layer in

¹ <http://oxygen.informatik.tu-cottbus.de/reverse-i1/?q=node/6>

² MDA - Model Driver Architecture is a framework for distinguishing different abstraction levels defined by the Object Management Group.

³ CIM - Computational Independent Model is most frequently used in the context of the Model Driven Architecture (MDA) approach which corresponds the Object

OMG's MDA is Platform-Independent Model (PIM)⁴ where rule interchange formats (i.e. RuleML, RIF, R2ML) try to accomplish their general purpose: a PSM to PSM business rules migration through the PIM level. The third MDA level is Platform-Specific Model (PSM⁵) containing rule specific languages together with their specific engines/platforms like: F-Logic [5], JRules(ILOG⁶), Jess⁷ or JBoss Rules⁸.

Our rule interchange work address JBoss Rules as source platform and Jess as a target platform, because those languages are actually in business market interest as popular business logic frameworks, used by Java developers to create complex rule-based applications by combining Java platform and business rule technology.

The REVERSE II Rules Framework developed R2ML as an interchange language for deploying and sharing rules between different rule systems and tools (e.g. Object Oriented rule languages, Semantic Web rule languages, Artificial Intelligence rule languages). Actually, R2ML (now at version 0.4) is a mature and experienced enough rule interchange language to provide a concrete interchange format for different rule systems and languages (i.e. <http://oxygen.informatik.tu-cottbus.de/reverse-i1/?q=node/15>).

R2ML has a rich syntax, so it could represent business rules from both JBoss Rules and Jess languages, providing this way the interchange possibility. As an interchange language, R2ML addresses the PIM level. The main idea is to use a model transformation language (MTL), or an application language to transform a PIM model into a PSM as in the Figure 1.



Fig. 1. Interchanging between JBoss Rules and Jess

2 Mapping JBoss Rules to R2ML

In this section we describe the JBoss business rules transformation into R2ML interchange language.

Management Group (OMG) vision of Model Driven Engineering (MDE). The Meta-Object Facility (MOF), is the OMG standard for Model Driven Engineering.

⁴ PIM - Platform Independent Model

⁵ PSM - Platform Specific Model

⁶ <http://www.ilog.com>

⁷ *Jess*, Sandia Lab., <http://herzberg.ca.sandia.gov/jess/>

⁸ *JBoss Rules*, <http://labs.jboss.com/jbossrules/docs>

Business rules are built following a business model representation. In many cases, a business model is first represented in a natural language descriptions based on core ontological concepts like classes and variables (CIM). At this stage, we can identify all objects referenced in the rules, and for each object we identify all referenced attributes. For each attribute, we identify all its constraints. Consider the following production rules examples provided in the RIF Public Mailing List ⁹:

1. Rule "Credit Score Adjustments 1": *If the creditScore attribute is between 580 and 679 , for "ACMEPowerBuyerGroup" as program group, "FIRST_TD;SECOND_TD" as link type and "Wholesale" division, then lower the score attribute with the 0.3 value.*
2. Rule "Occupancy Adjustments 1": *For any investor of "Wholesale" division, grow the occupancy value with 0.95.*

2.1 Mapping rules vocabularies

Object oriented rules as JBoss Rules and ILOG JRules are build on top of Java vocabularies. JBoss business rules vocabulary consists of Java bean classes. This vocabulary is used by the rules through the `import` declarations, which are specified inside of the rules file (`dr1` files or `xml` files). For example, the before JBoss rules use `CreditScore` and `Occupancy` Java beans classes in order to describe the JBoss business rules vocabulary.

An R2ML rule always refers to a vocabulary which can be R2ML own vocabulary or an imported one (RDF(S) and OWL). R2ML vocabulary is a serialization of an UML fragment of class diagrams. Below is an excerpt from rule "Occupancy Adjustments 1" 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="Occupancy">
    <r2mlv:Attribute r2mlv:ID="division">
      <r2mlv:range><r2mlv:Datatype r2mlv:ID="xs:string"/></r2mlv:range>
    </r2mlv:Attribute>
    <r2mlv:Attribute r2mlv:ID="occupancy">
      <r2mlv:range><r2mlv:Datatype r2mlv:ID="xs:string"/></r2mlv:range>
    </r2mlv:Attribute>
    <!-- ... -->
  </r2mlv:Class>
</r2mlv:Vocabulary>
</r2mlv:RuleBase>
```

Any Java qualified class names will be translated into (`xs:QName`) together with the corresponding names declarations. For example, if we assume the JBoss declaration: `org.drools.examples2.Occupancy`, this is translated into the namespace declaration `xmlns:ex="http://drools.org/examples2"` and the qualified name: `ex:Occupancy` for the reference to the class name.

⁹ <http://lists.w3.org/Archives/Public/public-rif-wg/2007Mar/0083.html>

2.2 Rule mapping

Consider the rule "Credit Score Adjustments 1":

```
package examplesRuleSet;
import example1.CreditScore;
rule "Credit Score Adjustments 1"
date-effective "25-OCT-2001 17:26:14"
when
  cs : CreditScore(programGroup == "ACMEPowerBuyerGroup",
                  lienType == "FIRST_TD; SECOND_TD",
                  division == "Wholesale", creditScore >= 580 & <= 679 )
then
  cs(score = cs.score - 0.3);
end
```

The mapping principles to translate a JBoss rule into an R2ML production rule are:

- A R2ML `r2ml:ruleID` is generated using the JBoss rule ID value. For example rule "Credit Score Adjustments 1", generates the value "Credit Score Adjustements 1" for the `r2ml:ruleID` attribute.
- A JBoss rule has a conditions part (the `when` part) and an action part (the `then` part). The condition part of a JBoss rule is mapped into the content of `r2ml:conditions` role element. The RHS part of a JBoss rule containing multiple actions, maps into the content of `r2ml:producedAction` role element.
- Relational operations from JBoss Rules (e.g. `>=`) translates into `r2ml:DatatypePredicateAtom` using SWRL builtins as predicate names (e.g. "swrlb:greaterThanOrEqual").
- In many cases JBoss column with field constraints translates into `r2ml:ObjectDescriptionAtom` or into a conjunction of atoms. The attribute `r2ml:classID` corresponds to the column name from JBoss implementation.

```
cs : CreditScore( ....)
```

translates into:

```
<r2ml:ObjectDescriptionAtom r2ml:classID="CreditScore">
  <r2ml:subject>
    <r2ml:ObjectVariable r2ml:name="cs"
                       r2ml:classID="CreditScore"/>
  </r2ml:subject>
  <!-- ... -->
</r2ml:ObjectDescriptionAtom>
```

- Any JBoss variables (i.e. *declarations* in the JBoss terminology) translates into R2ML variables. The JBoss *fact variable* (e.g. `cs:CreditScore`) is mapped into `r2ml:ObjectVariable` using the value of `r2ml:name` attribute to represent the variable name (i.e. `cs`). The optional `r2ml:classID` attribute specifies the type of the object variable (i.e. `CreditScore`). An `r2ml:ObjectVariable` is a variable that can be only instantiated by objects.
- The rule field constraints i.e. `programGroup=="ACMEPowerBuyerGroup"` translates into `r2ml:DataSlot/r2ml:ObjectSlot`, depending on the type

of the involved property. Since `programGroup` is a string value, it translates into `r2ml:DataSlot`, having the name of the attribute as `r2ml:attributeID="programGroup"` and its value translates into `r2ml:TypedLiteral`.

- A JBoss column may contain many field constraints all of them referring to the same context variable. In the example below there is a context variable `cs` referring the column `CreditScore` together with its field constraints.

```
cs : CreditScore(programGroup == "ACMEPowerBuyerGroup",
                 lienType == "FIRST_TD; SECOND_TD",
                 division == "Wholesale", creditScore >= 580 & <= 679 )
```

- The comma is the implicit connective between constraints and corresponds to the logical `and`. In the above example the field constraints are equalities such as `division == "Wholesale"` or a kind of relational expressions (*literal numeric constraints* in JBoss terminology) such as `creditScore >= 580 & <= 679` which corresponds to the relational `creditScore >= 580 & creditScore <= 679`.

The above `CreditScore` column may be written as a conjunction of the following columns:

```
cs : CreditScore(programGroup == "ACMEPowerBuyerGroup")
cs : CreditScore(lienType == "FIRST_TD; SECOND_TD")
cs : CreditScore(division == "Wholesale")
cs : CreditScore(creditScore >= 580)
cs : CreditScore(creditScore <= 679)
```

Below is depicted the corresponding R2ML serialization of a literal numeric constraint from the rule condition:

```
<!-- cs:CreditScore(programGroup == "ACMEPowerBuyerGroup")-->
<r2ml:ObjectDescriptionAtom r2ml:classID="CreditScore">
  <r2ml:subject>
    <r2ml:ObjectVariable r2ml:name="cs"/>
  </r2ml:subject>
  <r2ml:DataSlot r2ml:attributeID="CreditScore.programGroup">
    <r2ml:value>
      <r2ml:TypedLiteral r2ml:lexicalValue="ACMEPowerBuyerGroup"
                       r2ml:datatypeID="xs:string"/>
    </r2ml:value>
  </r2ml:DataSlot>
<!-- ... -->
</r2ml:ObjectDescriptionAtom>
```

The R2ML markup of equality constraints is an enumeration of `r2ml:DataSlot(s)` inside an `r2ml:ObjectDescriptionAtom`, while the other relational constraints maps into `r2ml:DatatypePredicateAtom(s)` as in example below:

```
<!-- cs:CreditScore(creditScore >= 580) -->
<r2ml:DatatypePredicateAtom r2ml:datatypePredicateID="swrlb:greaterThanOrEqual">
  <r2ml:dataArguments>
    <r2ml:AttributeFunctionTerm r2ml:attributeID="CreditScore.creditScore">
      <r2ml:contextArgument>
        <r2ml:ObjectVariable r2ml:name="cs" r2ml:classID="CreditScore"/>
      </r2ml:contextArgument>
    </r2ml:AttributeFunctionTerm>
    <r2ml:TypedLiteral r2ml:lexicalValue="580" r2ml:datatypeID="xs:decimal"/>
  </r2ml:dataArguments>
</r2ml:DatatypePredicateAtom>
```

- R2ML actions are built according with the OMG PRR Specification [6].
 1. InvokeActionExpression - invokes an operation with a list of parameter arguments and changes the state of the context object.
 2. AssignActionExpression - assigns a value to the attribute of the object context.
 3. CreateActionExpression - creates an object from the list of property-value pairs.
 4. DeleteActionExpression - deletes a specific object.

The JBoss action part of the rule is written in MVEL ¹⁰:

`cs(score=cs.score - 0.3)` which corresponds to the following JBoss standard syntax:`cs.setScore(score - 0.3); modify(cs);` and translates into a `r2ml:AssignActionExpression` as below:

```
<r2ml:producedAction>
<r2ml:AssignActionExpression r2ml:propertyID="score">
  <r2ml:contextArgument>
    <r2ml:ObjectVariable r2ml:name="cs" r2ml:classID="CreditScore"/>
  </r2ml:contextArgument>
  <r2ml:DatatypeFunctionTerm r2ml:datatypeFunctionID="op:numeric-subtract">
    <r2ml:dataArguments>
      <r2ml:AttributeFunctionTerm r2ml:attributeID="CreditScore.score">
        <r2ml:contextArgument>
          <r2ml:ObjectVariable r2ml:name="cs" r2ml:classID="CreditScore"/>
        </r2ml:contextArgument>
      </r2ml:AttributeFunctionTerm>
      <r2ml:TypedLiteral r2ml:lexicalValue="0.3"
        r2ml:datatypeID="xs:decimal"/>
    </r2ml:dataArguments>
  </r2ml:DatatypeFunctionTerm>
</r2ml:AssignActionExpression>
</r2ml:producedAction>
```

The second rule "Occupancy Adjustments 1" translates in a similar manner.

3 Mapping R2ML production rules to Jess language

In this section we describe how R2ML rules are translated into Jess rule language. In Jess, rules are defined using the `defrule` construct. They are in fact written as lists where the head is the special symbol `defrule`.

Jess provides two main categories of rules: forward-chaining rules and backward-chaining rules. Forward-chaining rules are the most common and used rules in Jess, and our translation will obtain Jess forward-rules. Jess is a forward-chaining reasoning engine, backward-chaining rules being simulated in terms of forward chaining [3].

To translate a R2ML rule into a Jess rule we will use Jess `unordered facts`. Any R2ML rule will translate into a Jess rule which use `unordered facts`, because they are better emulating the internal structure of a R2ML rule (which includes the rule vocabulary).

¹⁰ <http://wiki.mvflex.org/>

3.1 Mapping rules vocabulary

Jess business rules vocabulary consists of `deftemplate(s)` structures. A `deftemplate` describes a fact, in the same way as a Java class describes an object. In particular, a `deftemplate` is a Jess concept which includes a name, an optional documentation string, an optional "extends" clause, an optional list of declarations, and a list of zero or more member variables (called slot descriptions) with a type qualifier. Each slot description can optionally include a type qualifier or a default value qualifier.

Using vocabulary classes from R2ML¹¹, corresponding Jess `deftemplates` are generated. The `deftemplate`¹² structure corresponds to the `Occupancy` class description from the R2ML vocabulary (i.e. SubSection 2.1) is depicted below i.e.:

```
(deftemplate occupancy
 (slot division (type INTEGER))
 (slot occupancy (type STRING))
 ...)
```

They can be placed in the same file as the rules, or in a separate file, which need to be imported into the rules file, using the `import` keyword.

3.2 Rule mapping

Consider the following Jess rule "Credit Score Adjustments 1":

```
(defrule CreditScoreAdjustments1
 ?cs <- (CreditScore{programGroup == "ACMEPowerBuyerGroup" &&
 lienType == "FIRST_TD; SECOND_TD" &&
 division == "Wholesale" &&
 (creditScore >= 580 && creditScore <= 679)}(score ?oldScore))
 =>
 (modify ?cs (score (- ?oldScore 0.3))) )
```

- The `r2ml:ruleID` attribute value is used to obtain a Jess rule ID. If `r2ml:ruleID=Credit Score Adjustments 1` then, the `defrule CreditScoreAdjustments1` is generated for the corresponding Jess rule.
- An `r2ml:ObjectDescriptionAtom` will translate into a Jess pattern structure, which contains all R2ML's Object/Data Slots translated into Jess constraints i.e.:

```
?cs <- (CreditScore{programGroup == "ACMEPowerBuyerGroup" &&
 lienType == "FIRST_TD; SECOND_TD" && division == "Wholesale"})
```

- All R2ML variables are mapped into Jess variables: `fact variables` and `variables`. The `r2ml:ObjectVariable` is mapped into the Jess `fact variable` using the value of the `r2ml:name` attribute as the variable name. The `r2ml:classID` attribute specifies the type of the object variable.(e.g. `?cs <- (CreditScore{ })`)

¹¹ `xmlns:r2mlv="http://www.rewerse.net/11/2006/R2ML/R2MLV"`

¹² `herzberg.ca.sandia.gov/docs/71/api/jess/Deftemplate.html`

- Each `r2ml:DataSlot` from our R2ML rule representation will translate into appropriate Jess rule constraint. Internally, all Jess values (symbols, numbers, strings etc) are represented by instances of the class `jess.Value`. Its possible values are enumerated by a set of constants in the `jess.RU` class.

```
<r2ml:DataSlot r2ml:attributeID="CreditScore.programGroup">
  <r2ml:value>
    <r2ml:TypedLiteral r2ml:lexicalValue="ACMEPowerBuyerGroup"
                      r2ml:datatypeID="xs:string"/>
  </r2ml:value>
</r2ml:DataSlot>
```

will translate into:

```
programGroup == "ACMEPowerBuyerGroup"
```

- Our R2ML rule contains an enumeration of `r2ml:DataSlots` inside the `r2ml:ObjectDescriptionAtom`. They will translate into a conjunction of field constraints inside a Jess LHS's pattern, all having the same context: the Jess fact variable `cs`. A Jess conjunction of field constraints is represented using the `&&` operator i.e.:

```
?cs<-(CreditScore{programGroup == "ACMEPowerBuyerGroup" &&
lienType == "FIRST_TD; SECOND_TD" && division == "Wholesale"})
```

- All R2ML built-in predicates and operators translate into corresponding Jess constraints. The `r2ml:DatatypePredicateAtoms` having the attributes: `r2ml:datatypePredicateID="swrlb:greaterThanOrEqual"` and respectively `r2ml:datatypePredicateID="swrlb:lessThanOrEqual"` translate into the following conjunction of Jess patterns i.e.:

```
?cs <- (CreditScore{creditScore >= 580 && creditScore <= 679})
```

- The possible actions of a R2ML production rule are defined using OMG's PRR Proposal (the content of `r2ml:producedAction` role element). AssignActionExpression from our R2ML rule example corresponds to a `modify()` function invocation from Jess language, which updates an unordered fact from working memory. Notice that `?oldScore` is a bound variable to the `score` field value i.e.:

```
(modify ?cs (score (- ?oldScore 0.3)))
```

The second R2ML rule to detail its Jess implementation is rule "Occupancy Adjustments 1":

```
(defrule OccupancyAdjustments1
?o <- (Occupancy{ division== "Wholesale" && occupancy == "Investor" }
      (value ?oldValue))
=>
(modify ?o( value (+ ?oldValue 0.95))) )
```

The conditions part of the R2ML rule `Occupancy Adjustments 1` (the content of `r2ml:conditions` role element) is an `r2ml:ObjectDescriptionAtom` and translates into the LHS pattern with field constraints (the `Occupancy` pattern with appropriate field constraints).

The action part translates into Jess as a function call i.e.:

```
(modify ?o( value (+ ?oldValue 0.95)))
```

An Excerpt of mapping from JBossRules to Jess		
JBoss Rules	R2ML	Jess
<pre>rule "ruleName" when //conditions then //actions end</pre>	<pre><r2ml:ProductionRule r2ml:ruleID="ruleName"> <r2ml:conditions> ... </r2ml:conditions> <r2ml:producedAction> ... </r2ml:producedAction> </r2ml:ProductionRule></pre>	<pre>(defrule ruleName patterns => functions calls)</pre>
import JavaBeans files	declare the vocabulary, appropriate namespaces and qualified names (xs:QName)	import deftemplate(s) structures files
fact variable (optionally \$ symbol) \$var:ColumnName()	<r2ml:ObjectVariable r2ml:name="var" r2ml:classID="ColumnName"/>	fact variable ?var<-(ColumnName{ })
field variable (optionally \$ symbol) ColumnName(\$var:prop)	<r2ml:DataVariable r2ml:name="var"/> <r2ml:ObjectVariable r2ml:name="var"/>	field variable (?var)
column without field constraints	r2ml:ObjectClassificationAtom	LHS's pattern
column with field constraints	r2ml:ObjectDescriptionAtom	LHS's pattern with field constraints
data constraints inside a column	r2ml:DataSlot	pattern constraints
object constraints inside a column	r2ml:ObjectSlot	pattern constraints
setter call	r2ml:AssignAcionExpression	modify() function invocation

4 Limitations of the translation

The problems regarding the translation process refer the PSM systems: JBoss-Rules and Jess.

The translation process from PSM to PSM using intermediate R2ML (PIM) level demands also the mapping of rules vocabulary. The JBossRules to R2ML translator is a Java application that requests access to the JBoss-Rules vocabulary (Java compiled classes) in order to establish the types of objects and primitives. As R2ML supports Production Rules format[6], the JBossRules to R2ML translation of rule conditions part relies naturally.

The problems appear in the translation of the actions part of a JBoss rule, which may contain any Java valid code: variable declaration, non-declarative structures like `if...then` structures or cycling structures(i.e. while, for).

R2ML intends to solve this problem by providing in its future version an `<r2ml:OpaqueExpression>` with the role to encapsulate the code which don't find its semantic equivalent into R2ML `<r2ml:producedAction>` role element.

The purpose of R2ML, as PIM level markup language is to provide PSM to PSM rules translation, by sharing a vocabulary model (actually R2ML vocabulary), which can easily be mapped into Jess vocabulary (deftemplate(s) structures). Therefore, it makes possible the translation from R2ML to Jess language, as the business rules expressed in R2ML format don't require any

conceptual changes in order to be implemented in PSM target platforms (e.g. Jess, F-Logic, RuleML, OCL, SWRL, JBossRules, Jena).

The reverse translation, from Jess to JBoss Rules is also possible, as Jess supports a new richer syntax¹³ which offers the capability to represent object types using *deftemplate* structures. One limitation of this syntax is that it can only be used with unordered facts.

5 Conclusion and future work

The paper provides a description of rule translation from JBoss Rules, an Object Oriented rule language, into Jess, an Artificial Intelligence rule language using R2ML as interchange format.

Our future work intends to finalize the implementation of all transformation proposed in the paper, which will be followed by the detailed report on the experience.

References

1. H. Boley, M. Kifer (2007). *RIF Core Design*, W3C Working Draft, March 30, 2007 <http://www.w3.org/TR/rif-core/>
2. H. Boley, S. Tabet and G. Wagner (2001) *Design Rationale of RuleML: A Markup Language for Semantic Web Rules*, In Proc. of Int. Semantic Web Working Symposium (SWWS), July 30 - August 1, 2001, Stanford University, California, USA. <http://www.ruleml.org>
3. E. Friedman-Hill (2003). *Jess in Action - Rule-Based Systems in Java*, Manning Publications Co., 2003.
4. N. E. Fuchs, U. Schwertel, R. Schwitter (1997). *Attempto Englisch als (formale) Spezifikationssprache*, In: F. Bry, B. Freitag, D. Seipel (eds.), Proceedings of the Twelfth Workshop on Logic Programming WLP '97, Munich, September 1997.
5. M. Kifer, G. Lausen and J. Wu, Logical Foundations of Object-Oriented and Frame-Based Languages, *Journal of ACM*, May 1995 <ftp://ftp.cs.sunysb.edu/pub/TechReports/kifer/flogic.pdf>
6. OMG (2007). *Production Rule Representation Ver. 1.0*, March 5, 2007 <http://www.omg.org/docs/bmi/07-03-05.pdf>
7. OMG (2006). *Semantics of Business Vocabulary and Business Rules (SBVR)* <http://www.omg.org/docs/dtc/06-03-02.pdf>
8. G. Wagner, A. Giurca and S. Lukichev (2005). *R2ML: A General Approach for Marking up Rules*, Dagstuhl Seminar Proceedings 05371, In F. Bry, F. Fages, M. Marchiori, H. Ohlbach (Eds.) Principles and Practices of Semantic Web Reasoning, ISSN:1862-4405, <http://drops.dagstuhl.de/opus/volltexte/2006/479/pdf/05371.GiurcaAdrian.Paper.479.pdf>

¹³ <http://herzberg.ca.sandia.gov/docs/71/memory.html>