3 Good Reasons to Adopt the W3C Rule Interchange Format (RIF)

Christian de Sainte Marie

IBM (ILOG)
What is the W3C rule interchange format (RIF)?

- A standard XML serialization for rules
  - with a standard semantics
  - allowing rules written for one application to be published, shared, and re-used in other applications and other rule engines (between rule languages with compatible semantics)
  - Encouraging interoperability
  - Compatible with relevant standards (PRR, XSD, RDF, OWL, …)

- A rule is (just another) data item
  - that is produced by (or for) one application and that can be re-used in other applications
  - that can be interchanged, published, shared and executed across multiple platforms and technologies

- A W3C recommendation
  - [http://www.w3.org/standards/techs/rif#w3c_all](http://www.w3.org/standards/techs/rif#w3c_all)
  - An extensible framework
W3C RIF 101
W3C RIF design principles

- Translation paradigm
  - No intrusion in covered rule languages and rule sets

- Same semantics ⇔ same syntax
  - Share constructs across dialects wherever they agree on the semantics
  - Different constructs where semantics do not agree

- Alternating normal form XML
  - Alternating <Class> and <role> tags
  - Metadata can be attached to any class element
  - Except...

- Only XML schema is normative
  - Presentation syntax for specification’s readability (examples, semantics etc)
RIF-Core syntax
Metadata

Metadata can be attached to instances of any class element

Arbitrary metadata as (object, attribute, value) triples

An instance identifier as a RIF constant
Humans are mortal
Mortal(x) :- Human(x)
XML syntax (attributes)

Literal [@lang] ^symbol_spaceID

?X["human" -> ?Y]

<Frame>
  <object>
    <Var>?X</Var>
  </object>
  <slot ordered="yes">
    <Const type="&xs;string">
      human
    </Const>
    <Var>?Y</Var>
  </slot>
</Frame>
Syntax examples

- Look at
  - http://www.w3.org/2005/rules/wiki/Category:Test_Case
- Select test case
- Click on « view XML »
Built-in data types and externals

- All XML Schema 1.1 data types
  - Except anyType, anySimpleType, anyAtomicType, normalizedString and derived types, NOTATION, precisionDecimal, QName, ENTITIES, IDREFS, NMTOKENS
  - Plus rdf:XMLLiteral and rdf:PlainLiteral

- Symbol spaces without value spaces
  - rif:local and rif:iri

- RIF built-in functions and predicates are external in the sense that their semantics is defined outside of – and independently from – the rules where they are used
  - As opposed to logic functions and predicates
  - Include:
    - Data type comparison, guards, conversion and casting
    - Numeric, boolean, string, date, time and duration operators and comparison predicates
    - rdf:XMLLiteral and rdf:plainLiteral specific predicates and operators
    - Lists predicates and operators
    - Dialect-specific built-in functions
      - act:print (RIF-PRD)

- Many RIF built-in functions and predicates are adapted from XQuery 1.0 and XPath 2.0 Functions and Operators
  - The differences from the original XF&O include the handling of errors, the differentiation between predicates and functions, and a few other specific differences
  - If an argument value is outside of its domain, the value of the function or predicate is left unspecified
RIF semantics

- **RIF-BLD**: standard first-order model-theoretic semantics with functions and equality
- **RIF-PRD**: standard RETE-style operational semantics
  - Default standard conflict resolution strategy `rif:forwardChaining`
    - Refaction
    - Priority
    - Recency
    - Implementation dependent selection
  - Caveat: **Modify** is not an atomic action
    - It is a shortcut for `Retract+Assert`
- **RIF-Core**: essentially Datalog
  - Model-theoretic semantics of RIF-BLD, restricted to RIF-Core syntax
    - No subclass, no logic functions, no named argument UNITERMS
    - Only atoms and frames in the head
  - Operational semantics of RIF-PRD, restricted to RIF-Core syntax
    - No negation
    - Pattern2Condition equivalence
    - Only Asserts in the RHS: Do2And equivalence
    - With those restriction, the operational semantics of RIF-PRD reduces to Datalog inflationary semantics, equivalent to minimal Herbrand model semantics
- **RIF Core and RIF-PRD require safeness** (RIF-BLD does not)
  - All variables must be safely bound in the condition before being used in the conclusion
Implementation
Disclaimer

- **Disclaimer:** Information regarding potential future products is intended to outline IBM’s general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for IBM’s products remains at IBM’s sole discretion.
Implementations

- 13 reference implementations
  - Including open source tools such as
    - Java library of (most) RIF built-ins functions and predicates
    - RIF-BLD converter to TPTP syntax
      - [http://www.freewebs.com/riazanov/software.htm](http://www.freewebs.com/riazanov/software.htm)

- More commercial and open source implementations to come (including for IBM Websphere ILOG JRules)
Example: implementing RIF based on model to model transform

- **Assumption:** rule languages the same family, as covered by a RIF dialect, as likely to have very similar abstract structures, even if they have very different concrete syntaxes

- **Conclusion:** it should be relatively easy to do the translation at the level of the abstract, rather than the concrete, syntaxes

![Diagram]

- Abstract syntax of the <partner's> document
  - Parse
  - Transform
  - Abstract syntax of the RIF document
    - Parse
    - Serialize
    - Transform
    - Abstract syntax of the document in <YPRL>
  - Translate
  - Your Prefered Rule Language
Example: RIF to CLIPS using KM3, TCS and ATL
KM3, TCS and ATL

- TCS, KM3 and ATL are Eclipse component
  - TCS (Textual Concrete Syntax) enables the specification of textual concrete syntaxes for Domain-Specific Languages (DSLs)
    - generates parsers (text-to-model) and serializers (model-to-text) for DSLs
    - provides an Eclipse editor with syntax highlighting, outline, etc
  - KM3 (Kernel Meta Meta Model) is an implementation-independent language to define abstract syntaxes of DSLs
    - provides an Eclipse editor with syntax highlighting, outline, etc
  - ATL (ATLAS Transformation Language) is a model transformation language and toolkit
    - the ATL Integrated Environnement (IDE) provides a number of standard development tools (syntax highlighting, debugger, etc.)
    - [http://www.eclipse.org/m2m/atl/](http://www.eclipse.org/m2m/atl/)

- The TCS specifications of RIF-PRD/XML and CLIPS textual concrete syntaxes, the KM3 definitions of RIF-PRD and CLIPS abstract syntaxes, and the ATL specifications of the RIF-PRD to CLIPS and CLIPS to RIF-PRD transforms are freely available
  - @@URL
  - Notice that the RIF-PRD TCS and KM3 specifications can be re-used to specify transforms to and from other rule languages
Demonstration
Use cases and usage issues
Rules standards are good for you

- **Standardisation**: an agreement to support a common interface or representation (with cooperation of the vendors)
  - Standard semantics
  - Platform neutral persistence
  - Use across different platforms

- **Users benefits**
  - Avoid vendor’s lock-in
    - Commoditization
  - Allows cooperative development/deployment
    - Share rules/policies/documentation
  - Improves communication
    - Improves quality, traceability, maintainability
  - Preserve investment, increases ROI
    - re-use rules across applications and tools
    - skill sets transferable

- **Vendors point of view**
  - Ease integration with applications, process servers, events servers, business process modeling environments, and business process management systems
  - Extend reach
  - Increase market confidence
Use cases

- Publication
  - Enforcement down the line
  - Mismo, regulatory compliance
- Design once, deploy everywhere
  - ArcelorMittal
- Collaborative design
  - UC1
- Migration
- Negotiation
- Persistence
Business Rules interchange

Compliance Use Case

Policy setting organisation (regulator)
  specifies rules (CIM level)

Compliance organisation
  • Specifies operational interpretation of rules (PIM level)
  • Merges rules from relevant sources

Policy compliant organisation (regulated)
  Implements rules (PSM level)
  Implements rules
Rule interchange use case

- The expanding effect of changing rules throughout the mortgage process
  - As rules change early in the process their affect can be more significant as it move through the mortgage process.

A relatively small change in interest rates...

(borrowed from Doug Doedens, FirstAm and MISMO)

...can cause the amount of risk to balloon as it moves through the service chain.
Use case: Mortgage industry

- Rate Sheet Distribution Today
  - Wholesale Lenders fax rate sheets to brokers two times per day.

- Processes created by brokers to limit risk
  - All loans must be submitted to a single “approval manager” who looks at current rates and makes sure loans submitted by brokers are based off of the current rates (faxed documents).
  - No loans can be locked after 3pm.

(borrowed from Doug Doedens, FirstAm and MISMO)
Current Rate sheet distribution process

Wholesale Lenders watch the market to determine current rates. Rate sheets are then faxed to brokerage houses. Brokers use the rate sheets to price loans to the consumer.

(borrowed from Doug Doedens, FirstAm and MISMO)
Future distribution process with BR interchange

Wholesale lender watches the market to determine current rates. The Rate Sheet is converted to BREW XML format. Brokers quote rates based on real-time rate sheets. Rate sheet information is now available to systems and brokers within the brokerage house.

(borrowed from Doug Doedens, FirstAm and MISMO)
Sample Rate Sheet filled with RULES
(borrowed from Doug Doedens, FirstAm and MISMO)
Business rules interchange using RIF

Wholesale lender

Application A

Rules

Rule system 1

Data

Rate sheet

Agreed on vocabulary (MISMO 2.4 XSD)

<XML doc>

maps

maps

e.g. loan application

Application B

Rule system 2

Data

Rate sheet

<XML doc>

maps

maps

Broker

Wholesale lender

Application A

Rules

Rule system 1

Data

Rate sheet

Agreed on vocabulary (MISMO 2.4 XSD)

<XML doc>

maps

maps

e.g. loan application

Application B

Rule system 2

Data

Rate sheet

<XML doc>

maps

maps

Broker

Wholesale lender

Application A

Rules

Rule system 1

Data

Rate sheet

Agreed on vocabulary (MISMO 2.4 XSD)

<XML doc>

maps

maps

e.g. loan application

Application B

Rule system 2

Data

Rate sheet

<XML doc>

maps

maps

Broker
ArcelorMittal

Line

S-Block Temper Mill S-Block S-Block S-Block

Temper Mill S-Block

S-Block Tension Leveller S-Block

Tension Leveller S-Block

S-Block Temper Mill S-Block

Temper Mill

Main Engine Elongation Engine

Main Engine Elongation Engine

V1 V1 V1 V1 V2 V2

V1 V1 V1 V1 V2 V2

V2 = V1 + \Delta V

V2 = V1 + \Delta V

Product

Customer

Alternate Customer

Expert Review

Scrap

Repair

Cut
ArcelorMittal

Evaluate process data -> extract events and individual defects

Piece together individual events and defects to characterise the product (types, severity, extension of the defects)

Decide the appropriate path for the product: send to client, downgrade, repair, cut, scrap, to be examined by experts
W3C RIF is not...

- ...a data format (except for rules)
  - Not designed for facts interchange
- ...a data modeling language
  - Not designed for data models interchange
- But rules are about data and data obeys a data model...
- RIF specifies the combination with data and data modeling standards
  - RIF combination with RDF and OWL
  - RIF combination with XML data and XML schemas
The Import directive

- Import RIF documents
  - Semantically equivalent to merging the importing and imported document, with a caveat:
    - rif:local constants are local to a document
    - Two rif:local constants with the same literal that occur in two different documents are not equal

- Import RDF and OWL graphs (schema and/or data)
  - RDF triple $s p o$ mapped to frame $s'[p']->o'$
    - Conditions on data types alignment
  - OWL 2 Full compatibility is straightforward extension of RDF compatibility
  - OWL 2 DL requires syntactic restrictions and semantic extension of RIF frames

- Import XML Schema and/or data
  - Element names and schema types mapped to class names
  - Sub-element and attribute names mapped to slot names
    
    ```
    Forall ?c such ?c # "Customer"
    Forall ?a such that And( ?a # "Customer/Address"
    ?c["Address" -> ?a]
    If Or( ?c [@xml:lang -> "en" ^^xs:language]
    ?a["City" -> "London"] )
    Then Do( Assert( speakEnglish(?n) ) )
    ```

```xml
<Import>
  <location> xs:anyURI </location>
  <profile> xs:anyURI </profile>?
</Import>

<Customer xml:lang="en">
  <Name> John </Name>
  <Age> 39 </Age>
  <Address>
    <City> London </City>
    <Street> ... </Street>
  </Address>
</Customer>
```
Conclusion

- **What we have now, with W3C RIF**
  - Consistent foundations for standard XML-based rule interchange
  - Covers the two main families of rule languages
  - Combines with XML, XSD, RDF, OWL
  - Ready for deployment

- **Why you should implement, use and/or deploy it now**
  - « Because it’s there! » (George Mallory)
  - XYZ
  - @@@

- **What we need next**
  - Atomic Modify for RIF-PRD
  - A standard Java RIF API?
    - An obvious complement to the Java Rule Engine API (JSR-94)
  - Events? Aggregates?
Thank you!

Questions?
Back-up slides
Specification of the syntax of a language

Language

Abstract syntax of the language (meta-model)

represents

Concrete syntax of the language

specifies

Concrete syntax of the language

conforms to

Concrete syntax of the language

uses

Abstract syntax of the document (model)

Document

Instance

specifies

Specification

specifies

Speciation in SL1*

according to

Speciation in SL*

according to

* SL1, SL2: specification languages
NB: SL1 and SL2 are languages

SL2

Abstract syntax of SL2

represents

Concrete syntax of SL2

SL1

Abstract syntax of SL1

represents

Concrete syntax of SL1

uses

Specification of a language

Specification in SL1

uses

Specification in SL2
Using the specification for parsing and serialization
Using the specification for translating documents

The diagram illustrates the process of translating documents using the M2M Transformation Language. The process involves the following steps:

1. **Specification of the source language**
   - **Specification of the abstract syntax**
   - **Specification of the concrete syntax**

2. **Source instance**
   - **Abstract syntax of the source document**
   - Uses **Parser**

3. **Target instance**
   - **Abstract syntax of the target document**
   - Uses **Transformer**

4. **Target document**
   - Uses **Serializer**

5. **Specification of the target language**
   - **Specification of the abstract syntax**
   - **Specification of the concrete syntax**

The diagram shows the flow of information from the source to the target document, with transformation rules applied using the M2M Transformation Language (SL1 and SL2).
Rule standards
Standards in the rules life cycle

Requirements analysis

Model business rules and business vocabularies in a verifiable and communicable way

OMG SBVR defines the vocabulary and rules for documenting the semantics of business vocabularies, business facts, and business rules

Application design

Include rules as 1st class components in UML models

OMG PRR is the UML extension that permits to associate each rule with the classes they check and the classes they impact

Deployment and operation

Write rules once, use them everywhere
Share rules with business partners
Publish/retrieve rules

W3C RIF is the standard XML serialization for rules

Embed rule engines in your Java applications

JSR-94 is the Java API for rule engines

W3C RIF

JSR-94

OMG PRR

OMG SBVR

Business rule engine
Rule standards and surrounding

- Requirements analysis
  - OMG BMM
  - OMG SBVR
  - W3C SKOS
  - OMG ODM
  - OMG BPMN

- Application design
  - OMG PRR
  - OMG DMN
  - W3C OWL
  - ISO CL
  - OMG OCL
  - PMML
  - W3C XSD

- Deployment and operation
  - JSR-94
  - JSR-331
  - XBRL
  - W3C RIF
  - W3C RDF
  - W3C XML
  - W3C RDF

- Other standards
  - OMG BPMN
  - OMG BMM
  - OMG PRR
  - OMG OCL
  - PMML
  - XBRL
  - JSR-94
  - JSR-331
  - W3C RDF
  - W3C XML

Arrows indicate:
- : concrete syntax for (possibly, part of)
- : uses
- : combines with
What’s next? Convergence

Requirements analysis

Application design

Deployment and operation

OMG SBVR

OMG PRR

Concrete syntax

W3C RIF

JSR-94

Rule/ruleset syntax

Transform

Transform
What’s next? Convergence

Requirements analysis

Application design

Deployment and operation

OMG SBVR

Cannot transform

OMG PRR

Cannot transform

Rule/ruleset syntax

Application context must be added

Cannot generate

No concrete syntax

W3C RIF

JSR-94

© 2010 IBM Corporation
What’s next? Challenges

Requirements analysis

- OMG SBVR
- Transform

Application design

- OMG PRR
- Business notation
- Decision trees
- Decision tables
- Constraints
- Decision Model and Notation (DMN)

Deployment and operation

- W3C RIF
- JSR-94

Transform
Generate