
**Project Proposal**

# Extending OO jDREW for RuleML 1.0

by

**Omar Alsaiari, Khalid F. AlMutariri, Nasser Albunian,**
**Christian Fabbricatore, Markus Zucker**

omar.alsaiari@gmail.com, k.almutairi@hotmail.com,
{nass11.s, c.fabbricatore, m.zucker}@unb.ca

| | |
|---|---|
| Course: | Semantic Web Techniques (CS6795) |
| Instructor: | Harold Boley |
| Advisors: | Bruce Spencer, Ben Craig |
| Term: | Fall 2011 |
| Date: | November 11, 2011 |

# 1 Introduction

OO jDREW is an object–oriented extension of the Java Deductive Reasoning Engine for the Web (jDREW). It also serves as a reference implementation of the RuleML web rule language. jDREW is a state–of–the–art deductive reasoning engine used for querying and processing facts and rules by using Prolog, Positional and Slotted Language (POSL), or RuleML syntax. Both, jDREW and Object Oriented jDREW (OO jDREW), are written in, and for use with, the Java language. Furthermore, (OO) jDREW provides two major execution approaches: a top–down (backward reasoning) mode for extracting information from knowledge bases (KBs) using queries, and a bottom–up (forward reasoning) mode to deduce knowledge from a specified set of facts and rules.

However, according to the OO jDREW Wiki [1] and to experimental tests executed by current authors, the software lacks usability and a number of features. These new features would make the application easier to use, enhance its web integration and may attract more individuals and businesses to make use of it.

Therefore, the authors decided to focus on the improvement of OO jDREW. For this, it is necessary to integrate support for the current RuleML 1.0 specification [2] and to improve its usability as much as possible within the given project time.

# 2 Issues

In addition to the problems listed in the RuleML Wiki [3], we have identified the following issues with OO jDREW:

- RuleML support
  - Parser does not accept valid RuleML 0.91 document[1]
  - Parser does accept invalid RuleML (for example see listing A.2)
  - System outputs RuleML which does not validate against the RuleML schema specification [2]
  - No support for RuleML 1.0
- Project management
  - No public source code control
  - No issue tracking
  - Outdated project roadmap

[1] for example: `http://ruleml.org/0.9/exa/discount.ruleml`

[2] No well-formed Extensible Markup Language (XML), does not conform to `http://ruleml.org/0.91/xsd/`

- Documentation
  - Current JavaDoc (source file comments) effort is insufficient
  - No high-level documentation for the internals of OO jDREW (data structures, reasoner, etc.)
- Overarching issues
  - Code looks rather unmaintainable and requires major refactoring
  - Dead code / useless dependencies (e.g. mysql-connector-java-5.0.4.jar)
  - Duplicated code all over the place
  - Questionable coding style[3] (use of untyped generics, magic constants, string-typing, 1000+ Lines Of Code (LOC) methods, etc.)
- Testing & Quality Assurance (QA)
  - Test infrastructure does not exist
  - Unit testing is impossible (due to current software design)
- User interface
  - User Interface (UI) mixed with business logic
  - Copy & paste issues (except on X11)
  - Does not use look & feel provided by the underlying operating system

# 3 Objectives

It should be obvious that we cannot solve every issue identified in section 2 in this project. Therefore we confine ourselves to a limited number of problems:

- Support for RuleML 1.0
  - Identify changes from RuleML 1.0
  - Rewrite parser infrastructure to support RuleML 1.0
  - Differentiate between RuleML sub–languages
- Project infrastructure
  - Move code from SourceForge to GitHub
  - Set up bug tracking and a proper project roadmap
- Refactoring & Other features
  - Code cleanup
  - Graphical User Interface (GUI) improvements
  - Definition of test cases
- Document and test changes

[3] cf. [4]

# 4 Suggested approach

According to the aspects mentioned above, the authors will use the following methodology to improve OO jDREW:

**Analysis & Design** First of all, it is necessary to understand and document the differences from the earlier version of RuleML and to get an overall understanding of OO jDREW's implementation, particularly of the RuleML parser.

Most of the changes in RuleML only affect some XML tags, e. g. the role tags `<head>` and `<body>` in version 0.91 are replaced with `<then>` and `<if>` in the current version 1.0. Moreover, there are slight changes that were made at the attribute level, e. g. the attribute in="no | semi | yes | effect | modal" has been replaced by per="copy | open | value | effect | modal".

Next, we will derive the required changes in OO jDREW's RuleML parser from the changes in the RuleML 1.0 specification. In parallel, we will check OO jDREW's current version for possible problems and derive further improvements. In order to meet our objectives regarding the RuleML parser, we will do a complete redesign of the parser infrastructure.

**Implementation & Documentation** After the analysis, we will focus on the actual implementation. We will form two sub-teams which will work on different subsets of the objectives specified in section 3. One sub-team will implement the new parser infrastructure while the other one will define test cases and handles the remaining aspects (GUI, refactoring, etc.). Of course, this also includes proper documentation for the source code, the test cases, results and possible problems.

# 5 Tools

- OO jDREW source code [4]
- Eclipse IDE / JDT [5]
- Git (distributed version control system) [6]

[4] `http://sourceforge.net/projects/oojdrew/files/oojdrew/OOjDREW96/OOjDREW961.zip`

[5] `http://eclipse.org/`

[6] `http://git-scm.com/`

# References

[1] RuleML Initiative, "OO jDREW Roadmap," http://wiki.ruleml.org/index.php/OO_jDREW:RoadMap, 2011, online, last access on November 5, 2011.

[2] RuleML Initiative, "Schema specification of ruleml 1.0," http://ruleml.org/1.0/, 2011, online, last access on November 5, 2011.

[3] RuleML Initiative, "OO jDREW Issues," http://wiki.ruleml.org/index.php/OO_jDREW:Issues, 2011, online, last access on November 5, 2011.

[4] R. C. Martin, *Clean Code: A handbook of agile software craftsmanship*. Prentice Hall, 2009.

# A Appendix

## A.1 Wiki — Roadmap update

```
== OO jDREW 1.0 ==
* Bring OO jDREW's RuleML parser up to date for the newest RuleML 1.0
  specification
** Redesign parser infrastructure
** Accept RuleML documents written according to the specification
** Differntiate between RuleML sub--languages implicitely
** Enhance parser infrastructure for RuleML 1.0 support

* Clean up, fix and enhance overall source code
** Remove duplicated and unused code
** Remove libraries wich are referenced but not used
** Improve \ac{GUI} (copy/paste, undo/redo, font sizes)
* Enhance functionality, maintain source

* Software tests
** Create test cases and test report templates

* Update project infrastructure
** Switch to improved version control system
   (i.\,e. from SourceForge to GitHub)
** Add bug--tracking functionality (for future improvements and feedback)
```

## A.2 Example of invalid RuleML 0.91 accepted by OOjDREW

```
<Assert>
<Rulebase mapClosure="universal">
<oid><Ind>Asserted content, i.e. rules and facts</Ind></oid>

<Implies>
    <And>
        <Atom>
            <op><Rel>premium</Rel></op>
            <Var>customer</Var>
        </Atom>
        <Atom>
            <op><Rel>regular</Rel></op>
            <Var>product</Var>
        </Atom>
    </And>
    <Atom>
        <op><Rel>discount</Rel></op>
        <Var>customer</Var>
        <Var>product</Var>
        <Ind>5.0 percent</Ind>
    </Atom>
</Implies>

<Implies>
    <And>
        <Atom>
            <op><Rel>premium</Rel></op>
            <Var>customer</Var>
        </Atom>
        <Atom>
            <op><Rel>luxury</Rel></op>
            <Var>product</Var>
        </Atom>
    </And>
    <Atom>
        <op><Rel>discount</Rel></op>
        <Var>customer</Var>
        <Var>product</Var>
        <Ind>7.5 percent</Ind>
    </Atom>
</Implies>

<Implies>
    <Atom>
```

```
45:         <op><Rel>spending</Rel></op>
46:         <Var>customer</Var>
47:         <Ind>min 5000 euro</Ind>
48:         <Ind>previous year</Ind>
49:     </Atom>
50:     <Atom>
51:         <op><Rel>premium</Rel></op>
52:         <Var>customer</Var>
53:     </Atom>
54: </Implies>
55:
56: <Atom>
57:     <op><Rel>luxury</Rel></op>
58:     <Ind>Porsche</Ind>
59: </Atom>
60:
61: <Atom>
62:     <op><Rel>regular</Rel></op>
63:     <Ind>Honda</Ind>
64: </Atom>
65:
66: <Atom>
67:     <op><Rel>spending</Rel></op>
68:     <Ind>Peter Miller</Ind>
69:     <Ind>min 5000 euro</Ind>
70:     <Ind>previous year</Ind>
71: </Atom>
72:
73: </Rulebase>
74: </Assert>
```