The OO jDREW Engine of Rule Responder: Naf Hornlog RuleML Query Answering Presentation

Benjamin Craig

Orlando Florida

RuleML 2007

Thursday, October 25, 2007
Outline

- Overview
- Agents
  - Personal
  - Organizational
  - External
- Rule Engines
  - Prova
  - OO jDREW
- Communication Middleware
  - Mule ESB
  - Reaction RuleML messages
- Demo
  - Use Cases
Overview of Rule Responder

- Rule Responder is an intelligent multi-agent system for collaborative teams and virtual communities
- Supports rule-based collaboration between the different members of a virtual organization
- Members of a virtual registration are represented as semi-automated rule-based agents which use rules to describe the behavioral and decision logic
- Uses RuleML subset as its Rule Markup Language, based on logic and XML
  - The member of the RuleML family employed here is Naf Hornlog
- Implemented as a Web-based service architecture
Personal Agents

- A personal agent acts on behalf of a single person of an organization
- The personal agent contains a FOAF* profile with FOAF-extended rules

*The Friend of a Friend (FOAF) project: http://www.foaf-project.org
Organizational Agents

- Organizational agents are used to represent goals and strategies shared by each person in the collaborative team.
- Organizational agents contain rule sets that describe their organizations’ policies, regulations, opportunities, etc.
External Agents

- External agents communicate with the virtual organization, exchanging messages that transport queries, answers, or complete rule sets via the public interface of the organizational agents
- HTTP interface to Rule Responder
- Support for multiple unique External Agents (end users) at a single time
- Users can use a web browser to communicate with Rule Responder (currently a test interface)
Rule Engines

- **Prova (Prolog + Java)**
  - Developed by Adrian Paschke (Germany) and Alex Kozlenkov (United Kingdom)

- **OO jDREW (Object Oriented Java Deductive Reasoning Engine for the Web)**
  - Developed by Bruce Spencer, Marcel Ball, Benjamin Craig (Canada)
Prova

- Prova is used to implement the organizational agents of Rule Responder
- Prova is also used for some personal agents
- **OO jDREW** is used for personal agents in Rule Responder
- Two modes of Rule Execution:
  - Bottom-up (forward reasoning)
  - Top-down (backward reasoning)
- Rule Responder primarily uses top-down
- Supports rules in the following formats:
  - POSL (Positional Slotted presentation syntax)
  - RuleML (XML syntax, can be generated from POSL)
Communication Middleware

- **Mule** Enterprise Service Bus (ESB)
  - Mule is used to create communication end points at each personal and organizational agent of Rule Responder
  - Mule supports various transport protocols (i.e. http, jms, soap)
  - Rule Responder uses http and jms as transport protocols
Reaction RuleML

- Reaction RuleML is a branch of the RuleML family that supports actions and events.
- When two agents need to communicate, each others’ Reaction RuleML messages are sent through the ESB.
Architecture - Overview

Use Case 4

Browser (external Agent)
- Web Browser
  - HTTP Web Form
  - Reaction RuleML Message

Personal Agent 1
- Rule Engine (Prova)
- Knowledge Base (dynamic access to external data sources)

Personal Agent 2
- Rule Engine (OO jDrew)
- Knowledge Base (with translated and replicated facts)

Organizational Agent 4
- RDF BibTex
- RDF vCard

Organizational Agent 5
- Rule Engine (Prova)
- Knowledge Base
Use Case

- **RuleML-2007 Symposium**
  - One Organizational Agent that acts as the single point of entry to the conference
    - Assists with planning, preparing, and running the Symposium
  - Personal Agents represent Chairs of the Symposium
Online Demo


- **Use Case Demo Link:**
Ex. Personal Agent’s knowledge base

% Sample rule POSL syntax
person(\texttt {?person}, \texttt {?role}, \texttt {?title}, \texttt {?email}, \texttt {?telephone}):--
  contact(\texttt {?person}, \texttt {?email}, \texttt {?telephone}),
  role(\texttt {?person}, \texttt {?role}),
  title(\texttt {?person}, \texttt {?title}).

% Sample facts that match the previous rule
contact(John, john@email.com, 1-555-555-5555).
role(John, Panel Chair).
title(John, Doctor).
Example Message to the Organizational Agent

```xml
<Message mode="outbound" directive="query">
    <oid>
        <Ind>RuleML-2007</Ind>
    </oid>
    <protocol>
        <Ind>esb</Ind>
    </protocol>
    <sender>
        <Ind>user</Ind>
    </sender>
    <content>
        <Atom>
            <Rel>getContact</Rel>
            <Ind>ruleml2007_Challenge</Ind>
            <Ind>update</Ind>
            <Var>Contact</Var>
        </Atom>
    </content>
</Message>
</RuleML>
```
RuleML-2007 Rule Responder

Use this text form to send a query in Reaction RuleML In format to the RuleML-2007 Responder.

```
<Message mode="outbound" directive="query">
  <oid>
    <Ind>RuleML-2007</Ind>
  </oid>
  <protocol>
    <Ind>getConSet</Ind>
  </protocol>
  <sender>
    <Ind>user</Ind>
  </sender>
  <content>
    <Atom>
      <Rel>getConSet</Rel>
      <Ind>ruleml2007_challenge</Ind>
      <Ind>Ind-updates</Ind>
      <Var>Content</Var>
    </Atom>
  </content>
</Message>
```

Send

Description:
RuleML-2007 Responder Use Case

Rule Interface Descriptions (Signatures)
(you might copy and paste the examples in this Rule Responder form):
- performable(performable) [example]
- Atempts(agent, Query, Descriptions) [example]
- getConSet(agent) [example]
- loc(agent) [example]
- assign(agent, Tuple, Rule) [example]
- getCommittedTuple(agent, Tuple, CommitInfo) [example]
- perAttr(Submission, submitAuthor, Submission) [example]
- submitted(Submission) [example]
- accepted(Submission) [example]
Example Message 2

- `<content>`
  - `<Atom>`
    - `<Rel>sponsor</Rel>`
    - `<Expr>`
      - `<Fun>contact</Fun>`
      - `<Ind>ben</Ind>`
      - `<Ind>nrc</Ind>`
    - `</Expr>`
    - `<Ind type="integer">500</Ind>`
    - `<Expr>`
      - `<Fun>results</Fun>`
      - `<Var>Level</Var>`
      - `<Var>Benefits</Var>`
      - `<Var>DeadlineResults</Var>`
    - `</Expr>`
    - `<Expr>`
      - `<Fun>performative</Fun>`
      - `<Var>Action</Var>`
    - `</Expr>`
  - `</Atom>`
- `</content>`
<xml version="1.0" encoding="UTF-8">

<Message mode="outbound" directive="answer">
  <oid>
    <Ind>RuleResponder@lsflscdatv6G010.iit.nsc.ge.eel34</Ind>
  </oid>
  <protocol>
    <Ind>esb</Ind>
  </protocol>
  <sender>
    <Ind>RuleResponder</Ind>
  </sender>
  <context>
    <Ind>
      <Rel sponsor="Fed">
        <Expr>
          <Fun contact="Fun">
            <Ind>Ben</Ind>
            <Ind>arc</Ind>
          </Expr>
          <Expr>
            <Ind type="integer">500</Ind>
          </Expr>
          <Expr>
            <Ind>http://www.Fun</Ind>
          </Expr>
          <Expr>
            <Ind>brontes</Ind>
          </Expr>
          <Expr>
            <Fun benefits="Fun">
              <Expr>
                <Fun logoi="Fun">
                  <Expr>
                    <Fun on="Fun">
                      <Ind>site</Ind>
                    </Expr>
                  </Expr>
                </Fun>
              </Expr>
            </Fun>
          </Expr>
          <Expr>
            <Ind>direct</Ind>
          </Expr>
          <Expr>
            <Fun acknowledgement="Fun">
              <Expr>
                <Fun loc="Fun">
                  <Ind>proceedings</Ind>
                </Expr>
              </Expr>
            </Fun>
          </Expr>
          <Expr>
            <Fun passed="Fun">
              <Ind>deadline</Ind>
            </Fun>
          </Expr>
          <Expr>
            <Fun performative="Fun">
              <Ind>email</Ind>
            </Fun>
          </Expr>
        </Expr>
      </Rel>
    </Ind>
  </context>
</Message>
</ruleML>
Example Message 3

- <content>
  - <Atom>
    - <Rel>sponsor</Rel>
    - <Expr>
      - <Fun>contact</Fun>
      - <Ind>ben</Ind>
      - <Ind>nrc</Ind>
    - </Expr>
    - <Ind type="integer">5000</Ind>
  - <Expr>
    - <Fun>results</Fun>
    - <Var>Level</Var>
    - <Var>Benefits</Var>
    - <Var>DeadlineResults</Var>
  - </Expr>
  - <Expr>
    - <Fun>performative</Fun>
    - <Var>Action</Var>
  - </Expr>
  - </Atom>
- </content>
Conclusion

- Rule Responder can be used to implement a wide range of use cases that require an intelligent, semi-automated decision layer.
- The middleware of Rule Responder allows deployment of multiple running use cases concurrently.