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

Benjamin Larry Craig (ben.craig AT unb DOT ca)

Faculty of Computer Science, University of New Brunswick
Frederiction New Brunswick, Canada

Abstract. Rule Responder is an intelligent multi-agent system for collaborative teams and virtual communities that uses RuleML as its rule interchange format. The system allows these virtual organizations to collaborate in an effective semi-automatic manner, and is implemented as a Web-based application on top of the Enterprise Service Bus Mule. It supports rule execution environments (rule/inference engines) such as the Prova distributed Semantic Web rule engine and OO jDREW (Object Oriented java Deductive Reasoning Engine for the Web). The paper describes the role of OO jDREW for answering queries against rule bases in the Naf Hornlog RuleML sublanguage for the real-world use case of a symposium organization.

1 Introduction

Person-centered and organization-centered profile descriptions using Semantic Web techniques such as Resource Description Framework (RDF), Friend of a Friend (FOAF)[1], Semantically-Interlinked Online Communities (SIOC) and ExpertFinder has become increasingly popular. Recent work on FindXpRT [BLB+] generalized fact-based to rule-based profiles to capture conditional person-centered metadata such as the right phone number to call a person depending on the time, the topic, or the urgency. Rule Responder is a service-oriented middleware tool that can be used by virtual organizations for automated rule-based collaboration [PKB07]. Distributed users (human or automated agents) can interact with Rule Responder by simple query-answer conversations or complex negotiation and coordination protocols. The Rule Responder agents will process the events, queries, and requests according to their rule-based decision and behavioral logic, delegate certain subtasks to other agents, collect partial answers, and send the completed answer(s) back to the requestor. The communication protocol used between the architectural components of Rule Responder (e.g., external, personal, and organizational agents) is Reaction RuleML [PKB07]. The Rule Responder team is focused on implementing use cases that require the interchange of rule sets and the need for a querying service. The use cases demonstrate rule-based collaboration in a virtual organization. One specific use case that will be explained in detail throughout the paper is the organization of the RuleML-2007 Symposium. Beside the contribution of an intelligent autonomous agent layer on top of the current Semantic Web, the use case demonstrates rule interchange between rule inference services using a common rule exchange format (RuleML/Reaction RuleML).

[1] http://www.foaf-project.org/

Section 2 of the paper explains the details of the RuleML-2007 use case. Section 3 of the paper contains details on the architectural structure of Rule Responder. In Section 4, an explanation of how Rule Responder and OO jDREW [BC] function is demonstrated. Section 5 will conclude the paper.

2 RuleML-2007 Use Case Description

The initial use case created to demonstrate Rule Responder is the organization of a symposium such as the RuleML-2007 Symposium, which is an example of a virtual organization that requires online collaboration within a team. Rule Responder can support the Organizing Committee of the RuleML-2007 Symposium by embodying responsibility assignment, automating first-level contacts for information regarding the symposium, and screening of incoming submissions based on metadata (e.g., to see if the paper topics fit the symposium or not). It can also aid with other issues associated with the organization of a symposium, including presentation scheduling, room allocation, and special event planning.

The use case utilizes a single organizational agent to handle the filtering and delegation of incoming queries. Each committee chair has a personal agent that acts in a rule-constrained manner on behalf of the committee member. Each agent manages personal information, such as a FOAF profile containing a layer of personal information about the committee member as well as FOAF-extending rules. These rules allow the personal agent to automatically respond to requests concerning the RuleML-2007 Symposium. Task responsibility in the organization is managed through a responsibility matrix, which defines the tasks committee members are responsible for. The matrix and the roles assigned within the virtual organization are defined by an OWL Lite Ontology. The Pellet [HPSM] reasoner is used to infer subclasses and properties from the ontology.

3 Architecture and Workflow of Rule Responder

Rule Responder's architecture is implemented by the use of personal agents, organizational agents, communication middleware and external agents. The rule-based personal agents represent different members of the RuleML-2007 Organizing Committee. These agents are implemented by a rule engine, which acts as an inference and execution environment for the rule-based decision and behavioral logic of the semi-autonomous agents. The organizational agents constitute an intelligent filtering and dispatching system, using a rule engine execution environment, for blocking incoming queries or delegating them to other agents. The communication middleware implements an Enterprise Service Bus (ESB) supporting various transmission protocols (e.g., JMS, HTTP, SOAP). The ESB implementation for Rule Responder is Mule [BCC$^+$], an efficient open source communication middleware. External agents can interact with the Rule Responder-enabled virtual organization via its public communication interface (e.g., an HTTP endpoint interface of an organizational agent as single point of entry).

The **personal agents** used by Rule Responder contain FOAF-extending profiles for each person of the organizational team. Beyond FOAF-like facts, person-centric rules are used. All clauses (facts and rules) are serialized in Naf Hornlog RuleML [HBG$^+$], the RuleML sublanguage for Horn logic (allowing complex terms) enriched by Naf (Negation as failure). These FOAF-extending profiles have access to RDF (BibTeX, vCard, iCard, Dublin Core) and RDFS/OWL (role and responsibility models). The following is a query that a personal agent can be expected to answer: "What is the best (fastest response time) method to contact a committee member?" For answering it, the corresponding personal agent's FOAF profile must be queried to find the most convenient contact method.

The **organizational agent** contains a rule set that describes the organization. The following is a query that an organizational agent can be expected to answer: "Who should be contacted about the symposium's panel discussion?" When this incoming query is received by the organizational agent, the agent must first determine who the correct contact person is for the panel discussion. When the contact person has been confirmed, the organizational agent sends a subquery to that committee member's personal agent. The personal agent could then respond with a contact method (e.g., email or telephone number, depending on contact preferences in their FOAF profile). Alternatively, if that contact person was on vacation or currently busy, then the personal agent would respond back to the organizational agent that the contact person is unavailable. If the first-line contact person cannot be reached, then the organizational agent will use the responsibility matrix (i.e., which committee members are responsible for certain tasks, and what members can fill their role of if they are unavailable) to try to contact the next personal agent.

External agents can communicate with the virtual organization by sending messages that transport queries, answers, or complete rule sets to the public interface of the organizational agent (e.g., an HTTP port to which `post` and `get` requests can be sent from a Web form). The standard protocol for intra-transport of Reaction RuleML messages between Rule Responder agents is JMS. HTTP SOAP is used for communication with external agents, such as Web services or HTTP clients (Web browsers).

4 OO jDREW Execution for the RuleML-2007 Use Case

The internal agents of Rule Responder are currently implemented using the reasoning engines Prova [KPS] and OO jDREW[BBH+05]; although further rule engines can be adapted for use by Rule Responder (which basically requires a translator between RuleML and the execution syntax of the rule engine). OO jDREW is used for certain personal agents, while Prova is used for other personal agents and for the organizational agent. OO jDREW has two main modes of operation: top-down and bottom-up. Bottom-up execution is used to infer all derivable knowledge from a set of clauses (forward reasoning). Top-down execution is used to solve a query on the knowledge base (backward reasoning). Rule Responder primarily uses top-down execution due to the nature of query-answering services.

The query-answering service begins when a message from an external agent is received by an organizational agent. To obtain a query from the exchanged message, the OO jDREW engine must first parse the message and translate the message payload (i.e. the RuleML query) into the executable syntax of OO jDREW. OO jDREW then loads that person's FOAF profile (stored as POSL [Bol04] or RuleML), and according to the rules defined therein, derives the answer, which is sent back to the requester as a Reaction RuleML event message. Once the FOAF profile has been loaded, an RDF Schema file of user classes will be required if the FOAF profile requires a user defined taxonomy. When the FOAF profile and RDF Schema has been parsed and the query was successfully parsed, then OO jDREW will run the query against the knowledge base. After solutions have been derived, they must be serialized into a Reaction RuleML message, to be sent back to the requesting agent. The following is an example of such a query-answering service being executed by Rule Responder. As mentioned in section 3, we want to know: "Who should be contacted about the symposium's panel discussion?" When the person to be contacted is determined a subquery is used to get the contact information of that person. The messages below show the interchange between the agents.

```
Incoming Message to Organizational Agent from the Rule Responder External Agent (shown left):
Incoming Message (subquery) to OO jDREW from the Organizational Agent (shown right):
<RuleML>
  <Message mode="outbound" directive="query">
    <oid><Ind>RuleML-2007</Ind></oid>
    <protocol><Ind>esb</Ind></protocol>
    <sender><Ind>Ben Craig</Ind></sender>
    <content>
      <Atom>
        <Rel>getContact</Rel>
        <Ind>ruleml2007_Panel</Ind>
        <Ind>update</Ind>
        <Var>Contact</Var>
      </Atom>
    </content>
  </Message>
</RuleML>
```

```
. . .

<content>
  <Atom>
    <Rel>person</Rel>
    <Var>Name</Var>
    <Var>Role</Var
    <Var>Title</Var>
    <Var>EMail</Var>
    <Var>Telephone</Var>
  </Atom>
</content>

. . .
```

The subquery corresponds to the head of a rule (in POSL), executed in OO jDREW, whose body premises could be directly retrieved as facts or further delegated to other agents. This is just a sample rule, for a full FOAF profile please refer to this reference [Cra].

```
person(?Person, ?Role, ?Title, ?Email, ?Telephone) :-
  contact(?Person, ?Email, ?Telephone),
  role(?Person, ?Role),
  title(?Person, ?Title).
```

Each message contains a performative wrapper that is used by Rule Responder to understand how to interpret and use the message. The information contained in the message is comprised of the following: the *sender* of the message, the transport *protocol* used by the ESB, the object identifier (*oid*), the *content* of the message, and the attributes *mode* and *directive*. The content of the incoming messages are the queries that are to be answered by either a personal or organizational agent. The content of the outgoing messages are the query results. The *oid* is the conversation id of the current message, and is used to distinguish between multiple conversations. The *mode* attribute indicates whether the message is an inbound or outbound message. The *directive* attribute distinguishes whether the message's content is a query or an answer.

When this message (above, left) is received by the RuleML-2007 organizational agent, the agent can determine which Organizing Committee members are to be contacted about the panel discussion. The agent will then send a subquery (above, right) to the personal agent (OO jDREW servlet). Once the query has been successfully executed, OO jDREW will send the following message (below, left) back to the RuleML-2007 organizational agent.

```
Outgoing content from the OO jDREW Servlet to the Organizational Agent (shown left):
Outgoing content from Organizational Agent to the External Agent (shown right):
    <content>
      <Atom>
        <Rel>person</Rel>
        <Ind>John</Ind>
        <Ind>PanelChair</Ind>
        <Ind>PHD</Ind>
        <Ind>John@email.com</Ind>
        <Ind>1-555-555-555</Ind>
      </Atom>
    </content>
```

```
<content>
  <Atom>
   <Rel>getContact</Rel>
     <Ind>ruleml2007_Challenge</Ind>
     <Ind>update</Ind>
     <Expr>
       <Fun>person</Fun>
       <Ind>John</Ind>
       <Ind>PanelChair</Ind>
       <Ind>PHD</Ind>
       <Ind>John@email.com</Ind>
       <Ind>1-555-555-555</Ind>
     </Expr>
  </Atom>
</content>
```

The organizational agent will then parse the subquery solution and send one final message (above, right) back to the external agent. As can be seen in the example, the organizational agent starts a subconversation with the personal agent by asking a new query and uses the received results to answer the original query of the external agent. As the answer to a query, the query will be instantiated, i.e. each variable within the query will be bound to the found individual constant or complex term.

5 Conclusion

Rule Responder can be used to implement a wide range of use cases that require an intelligent, semi-automated decision layer (e.g. to collect data, infer new knowledge, answer queries, and solve problems). The middleware of Rule Responder allows deployment of multiple running use cases concurrently. The ESB provides the communication backbone to synchronously or asynchronously interchange messages between multiple agents. RuleML is a descriptive rule interchange language that can implement all logical structures that are necessary for Rule Responder. For more information about Rule Responder and a use case demo, readers are referred to [PBKC] (section "Use Cases"). Rule Responder is an open source project and its current rule engines, Prova [KPS] and OO jDREW [BC], are available open source as well.

References

[BBH+05] Marcel Ball, Harold Boley, David Hirtle, Jing Mei, and Bruce Spencer. The OO jDREW reference implementation of ruleML. In Asaf Adi, Suzette Stoutenburg, and Said Tabet, editors, *RuleML*, volume 3791 of *Lecture Notes in Computer Science*, pages 218–223. Springer, 2005.

[BC] Marcel Ball and Benjamin Craig. Object Oriented java Deductive Reasoning Engine for the Web. http://www.jdrew.org/oojdrew/.

[BCC+] Antoine Borg, Travis Carlson, Alan Cassar, Andrew Cookeand, Stephen Fenech, and More. Mule. http://mule.codehaus.org/display/MULE/Home.

[BLB+] Harold Boley, Jie Li, Virendrakumar C. Bhavsar, David Hirtle, and Jing Mei. FindXpRT: Find an eXpert via Rules and Taxonomies. http://www.ruleml.org/usecases/foaf/findxprt.

[Bol04] Harold Boley. POSL: An Integrated Positional-Slotted Language for Semantic Web Knowledge. http://www.ruleml.org/submission/ruleml-shortation.html, May 2004.

[Cra] Benjamin Craig. A RuleML-2007 Publicity Chair FOAF Profile. http://www.jdrew.org/oojdrew/rulesets/RuleML-2007publicityChair.posl.

[HBG+] David Hirtle, Harold Boley, Benjamin Grosof, Michael Kifer, Michael Sintek, Said Tabet, and Gerd Wagner. Naf Hornlog XSD. http://www.ruleml.org/0.91/xsd/nafhornlog.xsd.

[HPSM] James Hendler, Bijan Parsia, Evren Sirin, and More. Pellet: The Open Source OWL DL Reasoner. http://pellet.owldl.com/.

[KPS] Alex Kozlenkov, Adrian Paschke, and Michael Schroeder. Prova: A Language for Rule Based Java Scripting, Information Integration, and Agent Programming. http://www.prova.ws/.

[PBKC] Adrian Paschke, Harold Boley, Alexander Kozlenkov, and Benjamin Craig. Rule Responder: A RuleML-Based Pragmatic Agent Web for Collaborative Teams and Virtual Organizations. http://www.responder.ruleml.org.

[PKB07] Adrian Paschke, Alexander Kozlenkov, and Harold Boley. A Homogenous Reaction Rule Language for Complex Event Processing. In *Proc. 2nd International Workshop on Event Drive Architecture and Event Processing Systems (EDA-PS 2007)*. Vienna, Austria, September 2007.