Expert Querying and Redirection with Rule Responder

FEWS-2007, 12 Nov 2007

Harold Boley\textsuperscript{1}, Adrian Paschke\textsuperscript{2}

\textsuperscript{1} National Research Council of Canada
University of New Brunswick, Canada

\textsuperscript{2} Bioinformatics, BIOTEC Center,
Technical University Dresden, Germany
Introduction

- Earlier expert finding (discovery) focus: Static expert description using agent-centric metadata profiles
- The current focus: Dynamic expert interaction using agent-communication infrastructure
- Rules employed for semi-automated decision support
  - delegation/coordination
  - negotiation flow
  - reaction logic
Introduction

- Research builds on our previous work in
  - **AgentMatcher**: Similarity match-making of weighted trees describing (learning) objects
  - **FindXpRT** (Find an eXpert via Rules and Taxonomies): (RDF or RuleML) FOAF facts and (RuleML) FOAF rules as well as (RDFS) classes for expert finding
  - **Rule Responder**: Extending Semantic Web towards a Pragmatic Web infrastructure for distributed rule-based human-computer collaboration
RuleML FOAF

- Previous system – RDF FOAF: Person-centric metadata **facts** only
- FindXpRT contribution – RuleML FOAF: Person-centric metadata **rules** added
  - Enables XML-based
    - Rule Formalization
    - Rule Interchange
    - Rule Execution
  - Can derive new FOAF facts
  - Can often be ‘normalized’ into RDF FOAF
RuleML FOAF: Rules

- Make **implicit** properties and relationships **explicit**:

  \[
  \text{watchMovie} (Peter, ?X) \leftarrow \text{recommendMovie} (Firefly, ?X).
  \]

- Constitute person-centric metadata properties **conditional** on the time, place, ...:

  \[
  \begin{align*}
  &\text{phonePreference} (Peter, \text{office}) \leftarrow \text{time} (9-12) \text{ OR } \text{time} (13-17). \\
  &\text{phonePreference} (Peter, \text{cell}) \leftarrow \text{time} (12-13) \text{ OR } \text{time} (17-18). \\
  &\text{phonePreference} (Peter, \text{home}) \leftarrow \text{time} (18-21). \\
  &\text{phonePreference} (Peter, \text{voicemail}) \leftarrow \text{time} (21-9).
  \end{align*}
  \]
Current Scenario

- An entity that seeks expertise in some area, hence called the (expertise) consumer consults

- Another entity that mediates, or directly offers, expertise, hence called the (expertise) provider

- Both consumers and providers of expertise can be individuals, organizations, or software systems

- Interested in systems as expertise providers for individuals/organizations as expertise consumers

- Based on Semantic (FOAF) & Pragmatic (Agent) Web
Expertise Providers

Can be

- *Expert finders*
  - Systems that find experts who then solve problems

or

- *Problem solvers*
  - Systems that directly solve problems

The same system can be an expert finder and a problem solver:

Duality of expert finding and problem solving
In its pure form, expert finding uses a description of human expertise, skills, and traits to search for a matching expert by

- Broadcasting the description to candidates, comparing it to candidates' self-descriptive (FOAF or SIOC) profiles, or
- Via a match-maker that mediates between both

Once an expert is engaged by an organization, he or she starts with problem (analysis and) solving
Continuum Between Extremes of Pure Expert Finding and Pure Problem Solving (2)

- When a consumer performs expert finding, this is part of problem solving in broad sense:

- An expert is sought — e.g. by advertising a generalized problem description — to solve temporary or continuing problems of the consumer

- If the problems can be solved via ‘outsourcing’, the found expert(s) can perform problem solving remotely
Continuum Between Extremes of Pure Expert Finding and Pure Problem Solving (3)

- In some cases, not even the identity of individuals within a team of remote experts will need to be revealed. A remote team that solves problems of a consumer can become increasingly ‘virtual’:
  - The focus can shift to communication protocols between the problem solving provider and the consumer.
  - The experts no longer need to form a fixed team or consist of human experts only: Problem-solving systems can perform some of the tasks.
  - The internal structure of the providing system need not be known to the consumer.
If, for all problems, the consumer has to deal with only one provider without being revealed the expert(s) or system(s) behind it, the provider turns into a pure problem solver.
Interplay Between Expert Finding and Problem Solving

- Even if a consumer primarily plans to engage an expert, their decision on, say, hiring him or her for a job may be (partially) based on his or her performance when solving a trial problem as part of the selection process.

- Conversely, even if a consumer primarily plans to get a problem solved, their trust in a proposed solution may be (partially) based on the reputation of the expert(s) involved in problem solving.
Problem Solving Specialized to Query Answering

- Some problems themselves consist in answering a query, and various other problems, X, are basically solved when the query “How to solve problem X?” is answered.

- Therefore, and since query answering is more amenable to current software systems than, say, sensor/effector-intensive problem solving, we will focus on the important special case where problem solving equals query answering.
Overlap of Expert Finding and Query Answering (1)

- A **specific query** is normally posed to directly receive an answer from a given expert (it is too narrow to define an area of expertise)

- A **general query** is normally posed to find an expert in the whole area of expertise it defines (it is too broad to lend itself to a direct answer)

- A **typical query** can result in both a (partial) answer and a (partial) identification of the expert(s) involved in finding it
Overlap of Expert Finding and Query Answering (2)

**Horizontal axis:** specificity/generality of queries

**Vertical axis:** degree to which queries are answered by a given expert
degree to which queries are used to find experts
Expert Redirection as Expert Referral or Expert Delegation

- **Expert referral**: provider refers consumer to other providers inside or outside his/her organization

- **Expert delegation**: provider delegates query to such providers, maybe without telling the consumer
Expert Chaining (1)

- Redirection of an expertise consumer C from an expertise provider P_j to the ‘next best’ provider P_{j+1} can be stored \textit{statically} on the profile of P_j, e.g. using the renowned \texttt{foaf:knows} property.

- Can also be computed \textit{dynamically} by P_j, e.g. based on the knowledge P_j has gained so far about the query of consumer C and the partial answer P_j may have given to C.

- \textit{Combinations} of static and dynamic redirections are also possible.
Expert Chaining (2)

- Leads to chaining, e.g. a redirection path P1, P2, ..., Pj, Pj+1, ..., Pn
- In more general case becomes a redirection tree
  - with branches from a provider P to k other providers, who
    - besides performing own referrals and/or ‘downward’ delegations
  - give partial answers ‘upward’ to P for answer integration
- In most general case, the redirection tree becomes a general redirection graph, where providers perform networking
Find an Expert with Referrals

Lucy

Offers Expertise: Pop Music
Seeks Expertise: Logic Programming
Rating Threshold: 4.5
Degree Threshold: 1

Julia

Offers Expertise: Logic Programming
Seeks Expertise: Pop Music
Overall Rating: 4.5
Status: Busy

Hart

Offers Expertise: Logic Programming
Seeks Expertise: Pop Music
Overall Rating: 4.0


Variable Bindings:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>?CSExpert</td>
<td>Julia</td>
</tr>
<tr>
<td>?RatingSought</td>
<td>4.0:Real</td>
</tr>
<tr>
<td>?PeerSought</td>
<td>Hart</td>
</tr>
</tbody>
</table>
Towards a Pragmatic Web

1. Explicit Meta-data
   - vCard, PICS, Dublin Core, RDF, IEEE LOM (Learning Objects Metadata), Micro Formats, FOAF, SIOC …

2. Ontologies
   - RDFS, OWL Lite|DL|Full

3. Logic and Inference
   - e.g. Logic Programming Rule/Inference Engines

4. Software Agents and Web Services
   - FIPA, Semantic Web Services, RBSLA, …

Syntactic Layer:
- Information/Data Resources, e.g. individual and shared HTML Webpages, Databases, data/object representation, XML data

Semantic Layer:
- Meta Data Resources (e.g. DC, vCard, iCal, Bibtex RDF, FOAF, SIOC)
- Individual Ontologies (RDFS, OWL), e.g. role ontology, indiv. conceptual specification
- Shared vocabularies, concepts, ontologies

Pragmatic Rule Layer:
- Individual and common context, norms and values (e.g. Deontic norms, needs, avoidance)
- Individual and shared goals/purposes (wants/desires)
- Individual and shared strategies and decision logic (e.g. deductive logic and abductive plans)
- Individual and collaborative reaction logic, negotiation and coordination situation (e.g. event/actions, situative temporal states)
Pragmatic Web with Two Kinds of Collaborative Agents

Interaction

Personal Agent
- Role 1 – Personal Information and Context
- Syntax
  - Syntactic Layer:
    - Syntactic Information Resources, e.g. HTML Webpages, rel. SQL Databases, OO representations such as EJBs etc.
- Semantic
  - Semantic Layer:
    - Semantic Resources:
      - Meta Data Resources (e.g. DC, vCard, iCal, Bibiex RDF, FOAF)
      - Individual Ontologies (RDFS, OWL), e.g. role ontology, incl. Conceptual specification
  - Pragmatic Rule Layer:
    - Individual context, norms and values (e.g. Deontic norms, needs, avoidance)
    - Individual goals/purposes (wants/desires)
    - Individual strategies and decision logic (deductive logic and abductive plans)
    - Individual reaction logic, negotiation and coordination situation (e.g. event/actions, situative temporal states)

Organizational Agent
- Syntactic Layer:
- Semantic Layer:
- Pragmatic Rule Layer:

Personal Agent
- Role 1
- Role 2

Interaction

Organizational Agent
- Syntactic Layer:
- Semantic Layer:
- Pragmatic Rule Layer:

Personal Agent
- Role 1
- Role 2

Interaction
Rule Responder Project:
http://www.responder.ruleml.org/
Benefits of Rule-Based Decision and Reaction Logic

1. Compact declarative representation of rules
   - Clear semantics
   - Global rules which might apply in several contexts (reusability)
   - Separation of contract rules from the application code
   - Extensibility of the rule base (without changing the interpreter)

2. Efficient, generic interpreters (rule engines) for automated rule chaining and execution of reaction rules

3. Automated conflict detection of rule conflicts
   - Traceable and verifiable results
   - Integrity constraints are possible
   - Automated conflict resolution by rule prioritization

Rules play an important role to automatically transform contextual data, derive new conclusions and decisions from existing knowledge and behaviourally act according to changed conditions or occurred events.
Rule Responder Architecture (MDA)

1. **Computational independent model (CIM)** with rules, processes, conversational flows (e.g. in a natural or visual language)

2. **Platform independent model (PIM)** which represents the rules, events and ontologies in a common (standardized) interchange format (e.g. a markup language)

3. **Platform specific model (PSM)** which encodes the rule statements in the language of a specific execution environment (e.g. a rule engine / inference service or compiled code)
General Syntax for Reaction Rules
(Reaction RuleML 0.2)

```xml
<Rule style="active" evaluation="strong">
  <label> <!-- metadata --> </label>
  <scope> <!-- scope --> </scope>
  <qualification> <!-- qualifications --> </qualification>
  <oid> <!-- object identifier --> </oid>
  <on> <!-- event --> </on>
  <if> <!-- condition --> </if>
  <then> <!-- conclusion --> </then>
  <do> <!-- action --> </do>
  <after> <!-- postcondition --> </after>
  <else> <!-- else conclusion --> </else>
  <elseDo> <!-- else/alternative action --> </elseDo>
  <elseAfter> <!-- else postcondition --> </elseAfter>
</Rule>
```
Messages in Reaction RuleML

<Message mode="outbound" directive="ACL:inform">
  <oid> <!-- conversation ID--> </oid>
  <protocol> <!-- transport protocol --> </protocol>
  <sender> <!-- sender agent/service --> </sender>
  <content> <!-- message payload --> </content>
</Message>

- **@mode** = inbound|outbound – attribute defining the type of a message
- **@directive** – attribute defining the pragmatic context of the message, e.g. one or more FIPA ACL performatives, KQML, OWL-QL, Standard Deontic Logic norms, …
- **<oid>** – the conversation id used to distinguish multiple conversations and conversation states
- **<protocol>** – a transport protocol such as HTTP, JMS, SOAP, Jade, Enterprise Service Bus (ESB) …
- **<sender><receiver>** – the sender/receiver agent/service of the message
- **<content>** – message payload transporting a RuleML / Reaction RuleML query, answer or rule base
Example: Request / Query

...<Message mode="outbound" directive="ACL:query-ref">
  <oid> <Ind>RuleML-2007</Ind> </oid>
  <protocol> <Ind>esb</Ind> </protocol>
  <sender> <Ind>User</Ind> </sender>
  <content>
    <Atom>
      <Rel>getContact</Rel>
      <Ind>Website</Ind>
      <Var>Contact</Var>
    </Atom>
  </content>
</Message>...

- Message is local to the conversation state (oid) and pragmatic context (directive)
Example: Response / Answer

<Message ...
...
$content>
<Atom>
  <Rel>getContact</Rel>
  <Ind>Website</Ind>
<Expr>
  <Fun>person</Fun>
  <Ind>Adrian Paschke</Ind>
  <Ind>Program Co-Chair RuleML-2007</Ind>
  <Ind>Technical University Munich</Ind>
  <Ind>paschke@in.tum.de</Ind>
  <Ind>+49 89 289 17504</Ind>
</Expr>
</Atom>
</content>
...
</Message>
Enterprise Service Bus
Communication Middleware
+ Service Object Broker
Mule Enterprise Service Bus

- Mule ESB Open Source
- Message Platform and distributed Object Broker
- Staged Event Driven Architecture (SEDA)
- > 30 Protocols (JMS, HTTP, SOAP …)
- Synchronous and Asynchronous Communication
- Complex Message-driven Event Processing (CEP)
Mule Enterprise Service Bus
Use Case: Expert Finding (1)
Use Case: Expert Finding (2)
Advanced Performatives

- **Query Answering**: Should the query be answered directly, only be used for expert finding, or both?
- **Expert Identification**: Do experts answering a query inside or outside the provider's organization need to be identified to the consumer?
- **Query Confidentiality**: Must the query be treated as confidential inside or outside the provider's organization?
- **Expert Referral**: Should the provider refer the consumer to other providers inside or outside his/her organization if no (complete) answer can be found (the consumer himself/herself would then need to contact that next provider, perhaps showing them the referral message from the previous provider)?
- **Expert Delegation**: Is the provider entitled to delegate the query (perhaps anonymized?) to other providers inside or outside his/her organization?
- **Delegation Chains**: Should delegation be stopped after a maximum length of delegation chains is reached (special cases: no delegation, 1-step delegation)?
- **Query Decomposition**: Should the provider decompose the query into subqueries if it cannot be answered or delegated as a whole?
- **Answer Returning**: Should providers to whom a (sub)query was delegated return answers to the delegating provider, to the consumer, or to both?
- **Answer Integration**: Should the provider integrate answers returned to it for subqueries or just collect them and hand them back unchanged to the consumer?
Use Case: Symposium Question-Answering (by its Virtual Organization)
Use Case: Symposium Question-Answering (by its Virtual Organization)
Example: Request Registration Fee (1)

Query from External Agent (e.g. Web form)

```
<RuleML ...>
    <Message mode="outbound" directive="query">
        ...
        <content>
            <Atom>
                <Rel>computeFee</Rel>
                <Ind>Peter Pan</Ind>
                <Var>Fee</Var>
            </Atom>
        </content>
    </Message>
</RuleML>
```

Delegation Logic of Organizational Agent (Prova)

```
fee(Phase, Fee):-
    % look-up responsible agent
    assigned(Agent, ruleml2007_Submission, ruleml2007_responsible),
    % query permission from responsible agent
    sendMsg(XID, esb, Agent, "query", fee(Phase, Fee)),
    % receive answers multiple times
    rcvMult(XID, esb, Agent, "answer", fee(Phase, Fee)).
```
Example: Request Registration Fee (2)

Rule of Program Chair Agent: (Prova)

```
computeFee(Participant, Fee, T):-
    holdsAt(open(RegistrationPhase), T),
    fee(RegistrationPhase, BasicFee),
    sendMsg(XID, esb, ruleml2007_PublicityChair, query,
            registrationDiscount(Participant, BasicFee, 1, Discount, Cost)),
    rcvMsg(XID, esb, ruleml2007_PublicityChair, answer,
            registrationDiscount(Participant, BasicFee, 1, Discount, Cost)),
    Fee = Cost - (Cost * Discount).

    fee(earlyRegistration, currency_Dollar:500).
    fee(regularRegistration, currency_Dollar:600).
```

Rule of Publicity Chair Agent: (O O jDREW)

```
%This rule is used to calculate the discount of registration if
%the organization is a collaboration partner.
    multiply(?IndividualDiscountCost:real, ?RegistrationCost:real, 0.9:real),
```
Conclusions (1)

- Explained duality of expert finding and query answering, and introduced new approach to both
- Rule Responder was employed as the Web-based communication infrastructure for expert querying and redirection
- As our use case, we discussed the organization of the RuleML-2007 Symposium
Conclusions (2)

Future work includes

- Formal foundations of expert finding (e.g., expert and conversation id's as $\pi$-calculus channel names)
- Refinement of the system of performatives supporting Rule Responder expert finding
- Further development of Reaction RuleML for serializing the required event and rule types
- Continuation of the RuleML-2007 use case and exploration of similar ones