Expert Querying and Redirection with Rule Responder

Harold Boley[1], Adrian Paschke[2]

[1] Institute for Information Technology – e-Business
National Research Council of Canada
Fredericton, NB, Canada
harold.boley AT nrc.gc.ca
[2] Bioinformatics Group – Biotec Center
TU Dresden, Germany
adrian.paschke AT biotec.tu-dresden.de

Abstract. The duality of expert finding and query answering is characterized. Expert querying and redirection are realized in the Rule Responder platform. The use case of the RuleML-2007 organization is discussed.

1 Introduction

Finding experts on the Web with semantics has recently received increased attention [AMBB+07]. While earlier work in Semantic Web expert finding focussed on (static) expert description using agent-centric metadata profiles, the current paper focusses on (dynamic) expert interaction using a Web-based agent-communication infrastructure. Rules are employed here for semi-automated decision support, delegation and reaction logic, and conversational coordination and negotiation flows.

This paper is based on our work in FOAF-extending social networking and expert finding, as realized, e.g., with RuleML FOAF in FindXpRT [LBBM06] and more recently in Rule Responder [PBKC07,Cra07]. FindXpRT (Find an eXpert via Rules and Taxonomies) combines FOAF facts and RuleML [Bol06] rules. This implemented system[1] allows users to derive FOAF data by deploying person-centric rules, either before FOAF publication or, on demand, from published (RuleML FOAF) pages. The natural match between Rule Responder and expert finding is the theme of the current paper.

The Rule Responder project extends the Semantic Web towards a Pragmatic Web infrastructure [PBKC07] for collaborative human-computer networks. These allow semi-automated agents – with their individual (semantic and pragmatic) contexts, decisions and actions – to form corporate, not-for-profit, educational, or other virtual teams or virtual organizations. The project develops an effective methodology and an efficient infrastructure to interchange and reuse knowledge (ontologies and rules), e.g. data from Web resources such as vCards, FOAF or SIOC profiles. Such knowledge plays an important role for

[1] http://www.ruleml.org/usecases/foaf/JieLiMCSThesis.pdf

(semi-automatically and contextually) transforming data, deriving new conclusions and decisions from existing knowledge, and acting according to changed situations or occurred (complex) events. Ultimately, this might put AI theories on distributed multi-agent systems into larger-scale practice and might form the basis for highly flexible and adaptive Web-based service-oriented architectures.

In this paper, we will first characterize the duality of expert finding and problem solving (especially query answering), and exhibit their synergy. We will then show how Rule Responder can be used for expert querying and redirection based on its use of Semantic Web rules and Pragmatic Web performatives. Finally, we will discuss expert finding with the use case of the RuleML-2007 symposium organization.

2 Expert Finding and Query Answering

Consider the following scenario. An entity that seeks expertise in some area, hence called the *(expertise) consumer*, consults another entity that mediates or directly offers expertise in that area, hence called the *(expertise) provider*. Both consumers and providers of expertise can in principle be individuals, organizations, or software systems. We are interested here in software systems that are expertise providers for individuals or organizations as expertise consumers.

Expertise providers can be *expert finders* – systems that find experts who then solve problems – or *problem solvers* – systems that directly solve problems. At the first glance, expert finders might seem to be hardly related to problem solvers. But actually, as explained from several angles below, the same system can be an expert finder and a problem solver, based on what we will call the *duality* of expert finding and problem solving.

Continuum: There are various intermediate forms in the continuum between the extremes of pure expert finding and pure 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[2] or SIOC[3]) 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.
- However, when a consumer performs expert finding, this is usually part of an effort at problem solving in the 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 expert(s) will be able to perform problem solving remotely.
- In some cases, not even the identity of individuals within a team of remote experts will need to be revealed (extreme example: help desks). A remote

[2] http://www.foaf-project.org
[3] http://www.sioc-project.org

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 software 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 has turned into a pure problem solver.

Interplay: The interplay between expert finding and problem solving is highlighted by the following observations.

- 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.

Specialization: Many *problem solving* tasks can be rendered as *query answering* tasks: 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: The overlap of expert finding and query answering can be visualized via a coordinate system as in Fig. 1. The horizontal axis depicts the specificity/generality of queries. Two sample graphs are plotted on the vertical axis: The first shows the degree to which queries are answered by a given expert. The second shows the degree to which queries are used to identify experts.
Variations of these plots would reflect different combinations of expert finding and query answering without changing the qualitative observation of Fig. 1: At one extreme, 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). At the other extreme, 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). However, in the normal case, a typical query can result in both a (partial) answer and a (partial) identification of the expert(s) involved in finding it.
Thus, one can pose a query to a given expert and also use it (or a generalized version) to find a new expert. Actually, if an (external) consumer's query cannot be answered by an expertise provider directly, he/she can himself/herself assume the role of an (internal) consumer, either delegating the query to another provider or – if none is known or available – even trying to find a new expert, e.g. by generalizing the query to a whole area of expertise and using that for expert

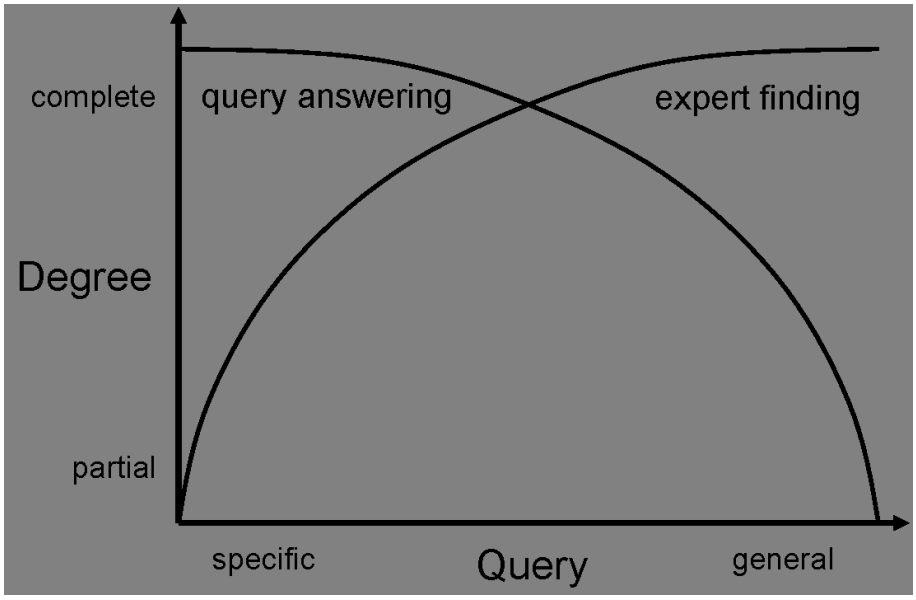

Fig. 1. Overlapping queries for direct answering and for expert finding

finding. If a new expert is found, the original query can be delegated to him/her, as well as can subsequent queries in that area of expertise.

Redirection: A crucial aspect of integrated expert finding and query answering is *expert redirection*, which can be expert referral or expert delegation: *Expert referral* means that the provider refers the consumer to other providers inside or outside his/her organization. *Expert delegation* means that the provider delegates the query to such providers, maybe without telling the consumer.

Chaining: Both referral and delegation may occur when an expert cannot (completely) answer a query but knows or finds another expert who should be able to answer the (open parts of the) query or who should again know such an expert. Such a redirection of an expertise consumer C from an expertise provider P_j to the 'next best' provider P_{j+1} can be stored statically on the profile of P_j, e.g. using the renowned `foaf:knows` property. It can also be computed 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. Combinations of static and dynamic redirections are also possible. With either method, this leads to chaining, e.g. a *redirection path* P_1, P_2, ..., P_j, P_{j+1}, ..., P_n. In a more general case, this redirection path for a consumer C becomes a *redirection tree* with branches from a provider $P_{i_1,...,i_m}$ to k other providers $P_{i_1,...,i_m,1}$, $P_{i_1,...,i_m,2}$, ..., $P_{i_1,...,i_m,k}$. Here, the knowledge and partial answer of $P_{i_1,...,i_m}$ leads to referrals and/or delegations to the k other providers, who – besides performing their own referrals and/or 'downward' delegations – may give their partial answers to C

and/or, for answer integration, 'upward' to $P_{i_1,\ldots,i_m}$. In the most general case, the redirection tree becomes a general *redirection graph*, where providers perform networking – via repeated (circular) referrals and/or delegations as well as partial answers – to (incrementally or ultimately) generate and possibly integrate their joint answer to the consumer's query.

The Rule Responder platform [PBKC07,Cra07] is an implemented system that is suitable for dual expert finding and query answering on the Web. The following sections will demonstrate this by discussing Rule Responder for query answering and delegation, and then applying it to the use case of a symposium organization.

3 The Rule Responder Platform

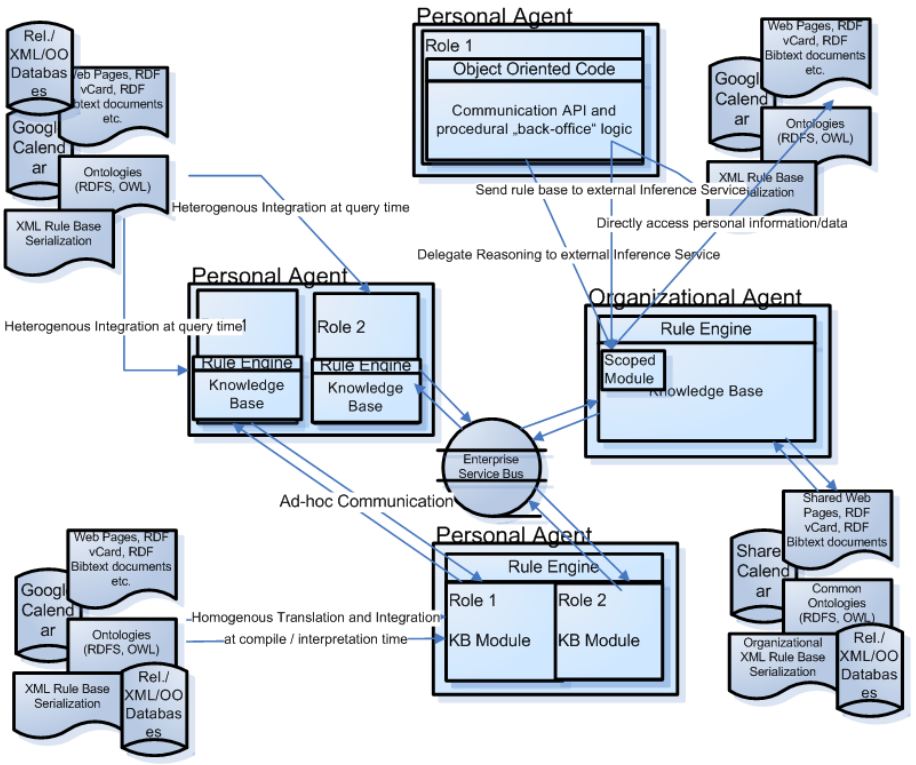

Fig. 2. Distributed Rule Responder Services

The three core technological components of the Rule Responder architecture are (1) a common platform-independent rule interchange format to interchange

queries and answers as well as complete rule sets between arbitrary inference services (rule engines), (2) platform-specific rule engines and execution environments that execute the rule-based expert delegation and referral logic in combination with further expert decision logic and expert knowledge to automatically answer queries (akin to rule-based agent systems), and (3) a highly scalable and efficient agent/service-broker and communication middleware, an enterprise service bus (ESB), on which the distributed execution environments are deployed and which is used to interchange messages, e.g., queries and answers, via discretionary transport protocols. Fig. 2 exemplifies this technical design.

Multiple rule engines may be deployed as distributed Web-based inference services on the ESB. Each inference service serves as a platform-specific execution environment for the rule-based logic and might dynamically import or pre-compile and load distributed rule bases. The rule bases declaratively implement the selection, referral and delegation logic as well as the possible expert decision and behavioral logic of Rule Responder agents/services. External data from data sources such as Web resources (e.g. FOAF, SIOC or vCard documents) or relational databases (person databases), or external application tools, Web services or programmatic object representations (e.g., EJBs) might be integrated into the rule execution at runtime or by translation at compile time. The ESB acts as the synchronous or asynchronous communication and messaging middleware between the different rule inference services, and is also used to call and interchange data with external components such as Web services or Web-based data sources. Different transport protocols such as JMS, HTTP or SOAP (our ESB supports more than 30 protocols) can be applied to transport queries, answers, and complete rule and data sets as payloads of event messages between the platform-specific inference services that are deployed on the ESB. Reaction RuleML [PKB07] is used as common platform-independent rule interchange format to serialize the interchanged event messages, which are then translated into the respective platform-specific execution languages at the inference service endpoints. The following paragraphs detail the technical components.

The open-source enterprise service bus Mule[4] is used as the communication middleware to seamlessly handle message-based interactions between the responder agents/services and with other applications and services using disparate complex event processing (CEP) technologies, transports and protocols. This ESB allows deploying the rule-based agents as highly distributed rule inference services installed as Web-based endpoints in the Mule object broker and supports Reaction RuleML based communication between them. That is, Mule provides a highly scalable and flexible application messaging framework to communicate synchronously but also asynchronously with external services and internal agents.

Each agent service may run one or more arbitrary rule engines, such as Prova[5] [Pas07] or OO jDREW [BBH+05], to execute the interchanged queries, rules

[4] http://mule.codehaus.org/display/MULE/Home
[5] http://prova.ws

and events and derive answers on requests. The agents comprise the rule-based expert search, delegation and referral logic which executes on top of the personal information data of the experts, e.g. FOAF and SIOC profiles or vCard documents. An agent might also implement the decision and behavioral logic of a real expert, i.e. act as a autonomous rule-automated agent.

Reaction RuleML [PKB07] is a general, practical, compact and user-friendly XML-serialized sublanguage of RuleML [Bol06] for the family of reaction rules. It incorporates various kinds of reaction rules as well as (complex) event/action messages into the native RuleML syntax using a system of step-wise extensions. The building blocks of Reaction RuleML (version 0.2) are:

- One general (reaction) rule form (< *Rule* >) that can be specialized to e.g. production rules, trigger rules, ECA rules, messaging rules ...
- Three execution styles defined by the attribute @*style*
  - *Active*: 'actively' polls/detects occurred events in global ECA style, e.g. by a ping on a service/system or a query on an internal or external event database
  - *Messaging*: waits for incoming complex event message and sends outbound messages as actions
  - *Reasoning*: Knowledge representation derivation and event/action logic reasoning and transitions (as e.g. in Event Calculus, Situation Calculus, TAL formalizations)
- Messages < $Message$ > define inbound or outbound event message

Inbound and outbound messages < $Message$ > are used to interchange events (e.g. queries and answers) and rule bases (aka modules) between the inference nodes:

```
<Message mode="outbound" directive="pragmatic performative">
  <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. a FIPA ACL performative
- < *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

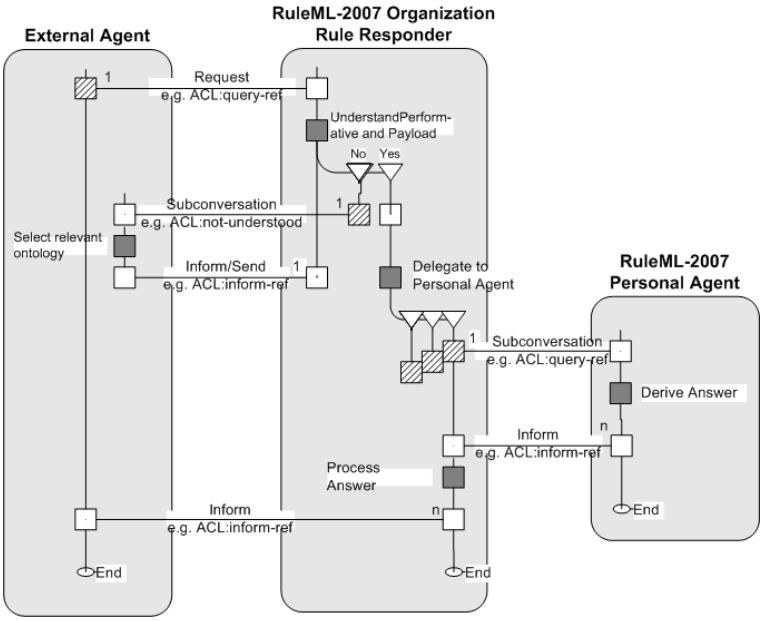

Fig. 3. Role Activity Diagram for a simple Query-Answer Conversation

The *directive* attribute corresponds to the pragmatic instruction, i.e. the pragmatic characterization of the message context. External vocabularies defining pragmatic performatives might be used by pointing to their conceptual descriptions. The typed logic approach of RuleML enables the integration of external type systems such as Semantic Web ontologies or XML vocabularies. A standard nomenclature of pragmatic performatives is defined by the Knowledge Query Manipulation Language (KQML) and the FIPA Agent Communication Language (ACL)[6], which defines several speech act theory-based communicative acts. Other vocabularies such as OWL-QL or the normative concepts of Standard Deontic Logic (SDL) to define, e.g., action obligations or permissions and prohibitions, might be used as well.

The *conversation identifier* is used to distinguish multiple conversations and conversation states. This allows to associate messages as follow-up to previously existing conversations, e.g. to start sub-conversations and to implement complex coordination and negotiation protocols, message-oriented workflows and complex event processing situations. Via sub-conversations it is possible to delegate queries to other expert agents and collect and aggregate multiple answers which are then used to create a consolidated answer to the original requestor's query.

[6] http://www.fipa.org/

The *protocol* might define lower-level ad-hoc or enterprise service bus transport protocols such as HTTP, JMS, and higher-level agent-oriented communication protocols such as Jade or Web Service protocols such as SOAP. More than 30 different transport protocols are supported by the Mule ESB, which is the main communication backbone in our implementation.

The *content* of a message might be a query or answer following a simple request-response communication pattern or it might follow complex negotiation or coordination protocols where complete rule sets, complex events or fact bases serialized in RuleML / Reaction RuleML are interchanged.

As an example, the Role Activity Diagram (RAD) shown in Fig. 3 describes a simple query-answer (request-response) pattern. An external agent requests some information from a responder agent. A possible query might be as follows (in Reaction RuleML syntax):

```
<RuleML ...>
    <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>ruleml2007_Website</Ind>
                <Ind>update</Ind>
                <Var>Contact</Var>
            </Atom>
        </content>
    </Message>
</RuleML>
```

The responder agent (expertise provider) tries to understand the query and starts a sub-conversation notifying the requester if the pragmatic context or the message content was not understood. In this case, the requester agent (expertise consumer) gives the responder agent further relevant information that is need to understand the query, e.g. references to relevant ontology parts or synonyms. If the message is understood by the responder agent it may delegate the query (possibly executing some further preprocessing) in a new sub-conversation to a personal expert agent of the same virtual provider organization. Other expert agents might be informed or consulted in parallel (not shown here). The expert agent derives the answers and sends them back to the responder agent which might simply forward them to the original requesting agent or further refine them according to the internal rule-based logic to answer the original query. A possible answer which is queried from an external FOAF profile and possibly further processed by a rule set might look like this:

```
...
  <content>
     <Atom>
```

```
        <Rel>getContact</Rel>
          <Ind>ruleml2007_Website</Ind>
          <Ind>update</Ind>
          <Expr>
              <Fun>person</Fun>
              <Ind>Adrian</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>
...
```

For advanced expert finding, the general FIPA-like performatives of Rule Responder could be enriched as shown below, e.g. to specify a "redirection protocol". Besides the content of a query and basic performatives, the consumer would specify pragmatic properties such as the following (most of which could also be used to specify a sender's expectations in email exchange between people, e.g. about whether – external, anonymized – redirection of a question is allowed).

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 or to the consumer?

Answer Integration: Should the provider integrate answers returned to it for subqueries or just collect them and hand them back unchanged to the consumer?

Accordingly, each conversation flow is local to the conversation id, including the current state of the conversation, the sender and receiver addresses, the pragmatic context as defined above, and the event message payload that should be executed.

4 The Symposium Organization Use Case

The use case chosen in this paper to demonstrate the Rule Responder capabilities is the organization of a symposium such as the RuleML-2007 Symposium[7], which is an example of a virtual organization that requires online collaboration within a team of experts with specialized roles to answer request from external requestors, e.g. sponsors, authors, participants see Fig. 4). In other words, it requires finding the responsible experts in a virtual organization to answer questions and fulfil tasks.

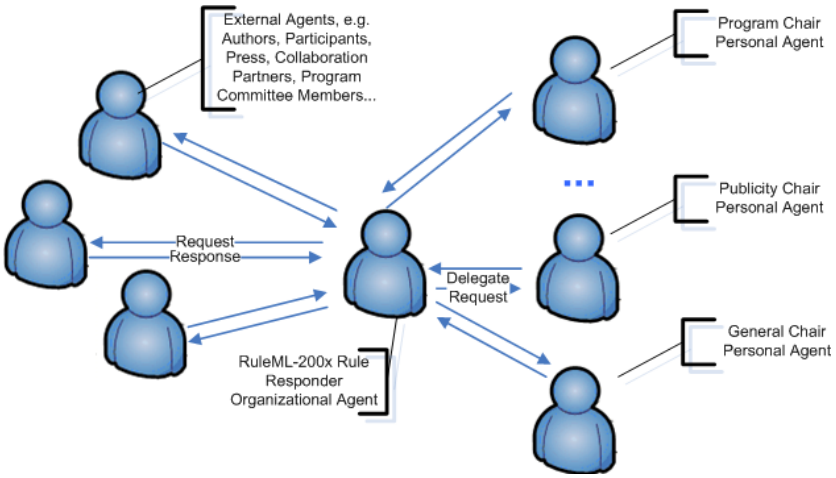

Fig. 4. RuleML-200x Use Case

The use case is publicly available[8] [PBKC07,Cra07]. Further use cases from other domains such as Life Sciences [PS07] can be found on the Rule Responder web page[9].

Rule Responder has experimentally supported the Organizing Committee of the RuleML-2007 Symposium by formalizing and proposing responsibility assignment and task delegation, automating query answering for information regarding the symposium, and screening of incoming submissions based on metadata (e.g., to determine if submitted paper topics fit the symposium or not). It can also aid with other issues associated with the internal management of the symposium, including e.g., 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 to the responsible committee chairs who are the

[7] http://2007.ruleml.org
[8] http://ibis.in.tum.de/projects/paw/ruleml-2007/
[9] http://responder.ruleml.org

experts for these tasks. Each committee chair has a personal agent that acts in a rule-based 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, e.g. about the RuleML-2007 sponsoring levels on behalf of the publicity chair[10]. 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 **personal agents** in this use case contain profiles for each person of the organizational team and these profiles can be stored as, e.g., RDF (BibTeX, vCard, iCard, Dublin Core), RuleML FOAF profiles, and RDFS/OWL role and responsibility models. For instance, this is a typical query whose RuleML version can be answered by a personal agent: "What is the best (fastest response time) method to contact this committee member?" For answering it, the 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 and the delegation and organizational decision logic. The following is a query whose RuleML version an organizational agent is expected to answer: "Who should be contacted about the symposium's panel discussion?"

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 Get/Post or SOAP is used for communication with external agents, such as Web services or HTTP clients (Web browsers).

The execution flow is as described in the role activity diagram in Fig. 5.

When the incoming query from an external agent is received by the organizational agent, the agent must first determine (from the responsibility assignment matrix) who the correct contact person is, e.g. for the panel discussion. When the contact person has been confirmed, the organizational agent sends a subquery in a sub-conversation 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 is on vacation or currently busy, then the personal expert agent would respond back to the organizational responder agent that the contact person is unavailable and possibly directly refer to another expert (e.g., according to the alternative FOAF referral / 'knows' information in the FOAF profile) or leave this task of finding another expert within the organization to the organizational agent. That is, if the first-line contact person cannot be reached, then the organizational agent will use the responsibility matrix (i.e., which committee

[10] http://www.jdrew.org/oojdrew/rulesets/publicityChair.ruleml

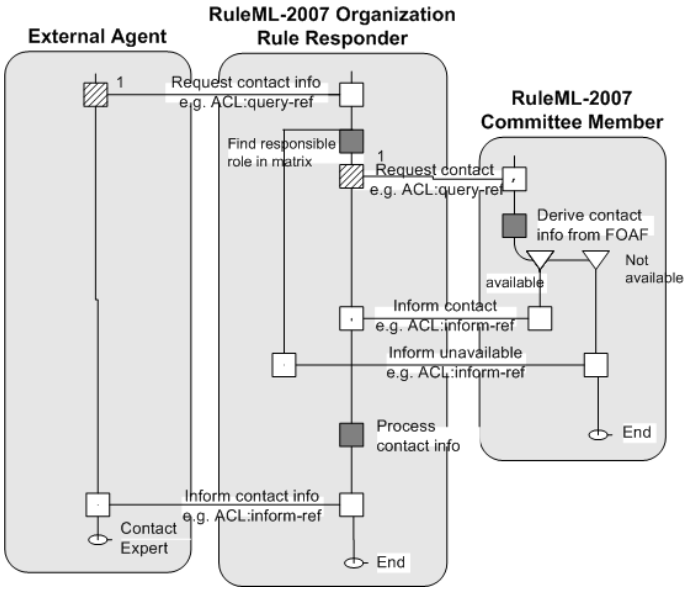

Fig. 5. Role Activity Diagram for Requesting Contact Information

members are responsible for certain tasks, and what members can fill their role (supportive role) of if they are unavailable) to try to contact the next available expert agent. This is one way Rule Responder can act in an automatic process by cascading subqueries that find the best contact person at the time the original query is requested.

From a technical perspective, all agents are deployed at inbound and outbound endpoints on the Mule ESB object broker, e.g. a HTTP port. Any received RuleML message is translated by a translator service (via XSLT) into the execution language of the agents' rule engine. The derived answers or newly created subqueries are translated back into the common rule interchange format and sent to the original requestor, or are delegated to another expert agent, typically in an asynchronous manner via the ESB.

5 Conclusion

We explained the duality of expert finding and query answering and introduced a 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.

Future work includes the formal foundations of expert finding (e.g., expert and conversation id's as π-calculus channel names), the refinement of the system of performatives supporting Rule Responder expert finding, the further devel-

opment of Reaction RuleML for serializing the required event and rule types, as well as the continuation of the RuleML-2007 use case and the exploration of similar ones.

Acknowledgements

Rule Responder is joint work with Alexander Kozlenkov and Benjamin Craig. Special thanks go to Alexander Kozlenkov for discovering the potential of the Mule ESB and for advising us about its practical use. Special thanks also go to Benjamin Craig for his Rule Responder extensions of OO jDREW, and for exploring the RuleML-2007 Symposium query answering use case.

References

[AMBB+07] Boanerges Aleman-Meza, Uldis Bojars, Harold Boley, John G. Breslin, Malgorzata Mochol, Lyndon J. B. Nixon, Axel Polleres, and Anna V. Zhdanova. Combining RDF vocabularies for expert finding. In Enrico Franconi, Michael Kifer, and Wolfgang May, editors, *ESWC*, volume 4519 of *Lecture Notes in Computer Science*, pages 235–250. Springer, 2007.

[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.

[Bol06] Harold Boley. The RuleML Family of Web Rule Languages. In José Júlio Alferes, James Bailey, Wolfgang May, and Uta Schwertel, editors, *PPSWR*, volume 4187 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2006.

[Cra07] Benjamin Craig. The OO jDREW Engine of Rule Responder: Naf Hornlog RuleML Query Answering. In Adrian Paschke and Yevgen Biletskiy, editors, *RuleML-2007*, volume 4824 of *Lecture Notes in Computer Science*. Springer, 2007.

[LBBM06] Jie Li, Harold Boley, Virendrakumar C. Bhavsar, and Jing Mei. Expert Finding for eCollaboration Using FOAF with RuleML Rules. In *Montreal Conference of eTechnologies 2006*, pages 53–65, 2006.

[Pas07] Adrian Paschke. *Rule-Based Service Level Agreements – Knowledge Representation for Automated e-Contract, SLA and Policy Management.* IDEA Verlag GmbH, Munich, forthcoming 2007.

[PBKC07] Adrian Paschke, Harold Boley, Alexander Kozlenkov, and Benjamin Craig. Rule Responder: RuleML-Based Agents for Distributed Collaboration on the Pragmatic Web. In *2nd ACM Pragmatic Web Conference 2007*. ACM, 2007.

[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.

[PS07] Adrian Paschke and Michale Schroeder. Inductive Logic Programming for Bio-Informatics in Prova. In *VLDB DBM 2007*, VLDB proceedings. VLDB, 2007.