Principles of the SymposiumPlanner Instantiations of Rule Responder

Zhili Zhao[1], Adrian Paschke[1], Chaudhry Usman Ali[2], and Harold Boley[2,3]

[1] Computer Science Department, Freie Universität Berlin, Germany
{zhili.zhao,paschke}@inf.fu-berlin.de
[2] Faculty of Computer Science, University of New Brunswick, Canada
maniali@gmail.com
[3] Institute for Information Technology, National Research Council of Canada
harold.boley@nrc-cnrc.gc.ca

Abstract. The Rule Responder SymposiumPlanner system supports topic-oriented collaboration between the distributed members of a virtual organization. Each member (or small team of members) is assisted by a semi-autonomous rule-based personal agent, which uses Semantic Web rules to capture aspects of the member's (or team's) derivation and reaction logic. SymposiumPlanner is a series of Rule Responder use cases for supporting the RuleML Symposia (2007-2011) by coordinating personal agents that assist the symposium chairs, intelligently answering questions from people interested in the symposium. In this paper, we introduce principles of SymposiumPlanner and make suggestions about its future development, mainly for RuleML-2012, and about further Rule Responder use cases.

1 Introduction

Rule Responder[1] is a tool for creating virtual organizations as multi-agent systems that support collaborative teams on the Semantic Web. It thus extends the Semantic Web towards a Pragmatic Web infrastructure with collaborative rule-based agent networks realizing distributed inference services, where independent agents engage in conversations by exchanging messages and cooperate to achieve shared goals [3]. Rule Responder's architecture realizes a system of personal agents (PAs), computational agents (CAs), and organizational agents (OAs), accessed via external agents (EAs), on top of an Enterprise Service Bus (ESB) communication middleware. These agents together process events, queries, and requests according to their rule-based decision and behavioral reaction logic. An agent can also delegate subtasks to other agents, collect partial answers, and send the completed answer(s) back to the requester. Since the Rule Responder framework has been conceived, many instantiations of it have been developed

[1] http://responder.ruleml.org

F. Olken et al. (Eds.): RuleML 2011 - America, LNCS 7018, pp. 97–111, 2011.

such as the Health Care and Life Sciences eScience infrastructure [13], Rule-based IT Service Level Management, Semantic Business Process Management (BPM) [14,15], WellnessRules(2) [2], PatientSupporter, and SymposiumPlanner systems.

SymposiumPlanner is a series of Rule Responder instantiations for the Questions&Answers (Q&A) sections of the official websites of the RuleML Symposia. Since 2007 [5], SymposiumPlanner has continued to support the organizing committee of the RuleML Symposium and was continuously developed to support this annual meeting (in 2011, it supports two RuleML Symposia). Symposium organization typically involves organization partner coordination, sponsoring correspondence, panel participants management, etc. Through a collaboration between the organizational agent, personal agents, and the external agent, SymposiumPlanner has assisted the symposium committee with structuring the meeting, and answered various kinds of questions by people interested in the symposium.

In this paper, we introduce the general architecture of the SymposiumPlanner system and present how it is used for symposium organization. In our latest version of SymposiumPlanner-2011, we introduce a user friendlier Rule Responder interface and information integration from external Web data repositories. The new Rule Responder web interface allows users to issue formal queries via web forms and in controlled natural language. Meanwhile, there are large semantic knowledge repositories on the Internet, such as: DBpedia (Deutschland)[2], Freebase[3], YAGO[4], Semantic Web Dog Food[5]. By reusing and integrating existing fact information on the Internet, we avoid redundancy in the knowledge bases of the SymposiumPlanner's agents.

The paper is organized as follows. Section 2 discusses related work. Section 3 describes issues in symposium organization where Rule Responder can help the chairs, and considers what needs to be taken into account when using Rule Responder in symposium organization. Section 4 introduces the conceptual architecture to address these issues. Section 5 presents the Rule Responder implementation architecture. Section 6 concludes the work on Rule Responder SymposiumPlanner and discusses the proof-of concept instantiations.

2 Related Work

To the best of our knowledge, there are no rule-based agent systems focusing on supporting conference planning, but there are a lot of other applications for rule-based multi agent systems. Rule-based agent systems make intelligent decisions quickly and in repeatable form based on their rule based and ontology based knowledge bases. They run specialized rule engines for executing the agent logic. The agent behaviour specification is mainly represented by programming

[2] http://de.dbpedia.org/
[3] http://www.freebase.com/
[4] http://www.mpi-inf.mpg.de/yago-naga/yago/
[5] http://data.semanticweb.org/

it in logical rules. Two typical representatives for this category are RC++ [19] and SOAR [11]. Another kind of rule-based agent architectures encompasses approaches that aim at introducing abstract mentalistic notions as agent programming language constructs. These approaches propose specific concerted sets of mental state components and introduce agent programming languages with specific types of rules to operate on the agents components. Prominent representatives of this category are the agent-oriented programming (AOP) [17] and the 3APL/2APL language families. 3APL ("An Abstract Agent Programming Language") and its successor 2APL ("A Practical Agent Programming Language") are developed at the University of Utrecht [9]. Closely related to Rule Responder are agent architectures which directly use expressive rule languages and rule engines as basis for the agent behavior control. Using this kind of architecture basically requires that the rule base is properly connected with the agent's sensors and effectors in order to allow an agent to receive percepts and execute actions. Examples of this domain are e.g. JADE/Jess agent [4], Vivid Agents [16], OPAL Agents [18] and Emerald [10]. While these approaches use their own proprietary agent runtime environments and rule engines, Rule Responder aims at a more general approach using established and highly efficient enterprise service and messaging technologies based on standard Internet transport technologies. For representing the agents knowledge and behaviour it applies standards such as RuleML rules and W3C RDF/OWL ontologies so that the agents logic is declaratively described in a platform-independent manner and can be translated and executed in different platform-specific rule engines deployed as distributed (Web) inference services.

3 Rule Responder for Symposium Organization

Rule Responder has been successfully employed as a distributed query answering framework instantiated for many areas. Two main benefits are gained by alleviating the burden of repetitive tasks and by enabling the automation of rule based organizational processes. SymposiumPlanner assists organizers in managing the meeting and answering queries about it. As Rule Responder uses Semantic Web rules to describe aspects of their owners derivation and reaction logic, it is necessary to extend the SymposiumPlanner system for users who may not have deep knowledge about semantic technologies.

3.1 Issues in Symposium Organization

We are not concerned here with paper submission and reviewing, which are well supported by existing conference management systems such as EasyChair[6] and WitanWeb[7]. Even without those processes, symposium organization involves lots of procedures, and it consumes much energy and time of organizers. Although none of these procedures seems inherently complex, taken together they are non-trivial to manage for meeting chairs. These procedures include:

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

[7] http://witanweb.ca/cascon2010/WitanWebFAQ.jsp

- coordinating chair responsibilities (responsibility assignment),
- finding contact information about selected chairs of the symposium,
- helping the program and track chairs with mapping planned paper topics to program and track themes,
- helping the program chair to monitor and possibly move important dates,
- helping the liaison chair with special events by symposium partners,
- helping the panel chair with managing panel participants,
- helping the publicity chair with sponsoring correspondence,
- and answering questions of participants about the conference such as important dates, topics addresses, program schedule etc.

3.2 Interaction with Users

For deploying agents on the Web and enabling communication in agent networks, Rule Responder uses an ESB middleware, and utilizes messaging from Reaction RuleML[8] for communication between the distributed agent inference services. While Reaction RuleML/XML has adequate expressiveness for the communication between heterogeneous rule agents in the virtual organization, there are two human-oriented methods to support the interaction between users and Rule Responder.

One method is creating dynamic HTML forms as Web user interface of an organizational agent. A Rule Responder interface description file – itself specified in XML – contains information about the interface name, parameters, and types, which is used to describe the interfaces of the organizational agent. Users are guided to select the appropriate interface and fill in the query parameters. Queries are translated based on the Rule Responder interface description file and the values that users give in the HTML form, as shown in Figure 1. However, it may be difficult for users to identify the desired interface when there are a plenty of interfaces available.

The other method is based on controlled natural language, which allows users to describe queries in controlled English, where a translator maps the controlled English to Reaction RuleML. The benefit is that users do not need to know the available interfaces for various queries. However, the controlled English (e.g. Attempto Controlled English) usually has more types of declarative sentences and some of which are difficult to translate into Reaction RuleML, such as: commands and sentence subordination. In the SymposiumPlanner-2011 system, we thus use both solutions to improve user experience.

3.3 Communication between Distributed Agents

Both centralized rule system and distributed rule system can be used to support the collaboration of distributed members of an organization. A centralized

[8] `http://reaction.ruleml.org`

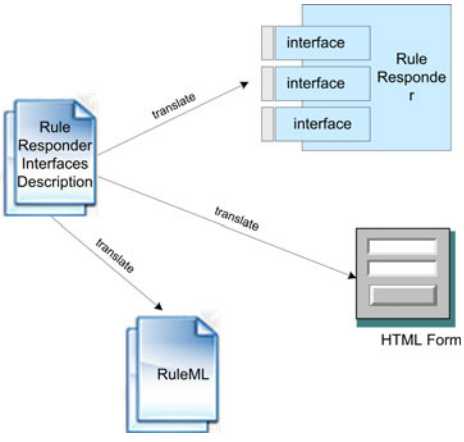

Fig. 1. Rule Responder Interface Description

rule system would contain all of the facts and rules in one knowledge base or in one centralized location. In contrast, the advantages of a distributed rule system over a centralized system include e.g., achieving a fault-tolerant system by using distribution for redundancy, and improved efficiency through distributed processing.

Distributed maintenance allows agents to update their rules and facts without affecting the rule bases of other agents (their consistency, completeness, etc.). If all of the knowledge was stored in one central rule base, problems introduced by one agent would affect the entire system.

Also, if an agent is offline, i.e. a PA is not responding when an OA delegates a query to it, a timeout would be received and either the OA would try another PA that may be able to answer the query or would respond back that the PA is currently offline.

Rule Responder and SymposiumPlanner are implemented as a distributed rule system. It connects OAs and PAs so that they can share knowledge and external agents can query this knowledge. Each PA and OA has its own set of rules and facts. The rules of the PAs correspond to their expert owners while the OA's knowledge describes the virtual organization as a whole. All of the PAs with their rule bases are stored at distributed locations.

In a distributed rule system the knowledge is spread over many different physical locations and communication overhead can become a problem, but the overhead may not be noticeable to external agents. Rules execute faster when there are less clauses for the engine to process, so our distributed approach improves efficiency because we have multiple rule engines working on smaller knowledge modules instead of one rule engine working on a large knowledge base as in a centralized approach.

3.4 Integration with External Information

Rule Responder acts as a virtual organization which consists of many autonomous rule-based agents. Each uses Semantic Web rules to describe aspects of their owners' derivation and reaction logic. With the development of semantic technologies in last decade, many huge semantic knowledge bases have been published, e.g. in the linked open data cloud, which can be utilized. We reduce redundancy of SymposiumPlanner system by integrating selected external semantic knowledge on the Internet, as shown in Figure 2. In SymposiumPlanner 2011 each agent manages personal information, such as a Friend of a Friend (FOAF)-like profile containing a layer of facts about the committee members as well as FOAF-extending rules.

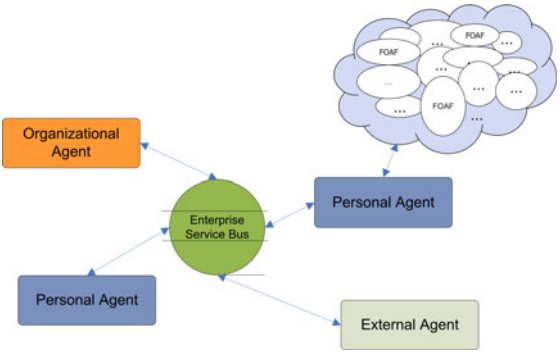

Fig. 2. Integration with External Information

3.5 Role Assignment

Personal agents in Rule Responder are usually loosely coupled and implemented based on their different functionalities. As one possible way for coordination in a virtual organization the Rule Responder framework uses a 'pluggable' Responsibility Assignment Matrix (RAM) to support the OA in its selection of a PA and its optional participating profiles underneath. A RAM describes the responsibility of agent roles in completing certain tasks or deliverables in a virtual organization. A standard RAM is a matrix which describes four key responsibilities most typically used: Responsible, Accountable, Consulted, and Informed.

In the Rule Responder agent topology, a single RAM matrix can be used in the OA to map an incoming query to the PA whose local knowledge base is deemed to be best suited for answering it. The RAM matrix is represented as an OWL ontology (OWL Lite) and can be used by a Rule Responder agent via querying it with the Semantic Web built-ins of Prova, binding the respective roles and their responsibilities to typed variables in the agent's rule logic. Many variants of the RAM with different role distinctions are possible such as RACI (with Consulted agents), RASCI (with Supporting agents) etc. - see, e.g., Table 1.

Table 1. Responsibility Assignment Matrix

	General Chair	Program Chair	Publicity Chair
Symposium	responsible	consulted	supportive
Website	accountable	responsible	
Sponsoring	informed, signs	verifies	responsible
Submission	informed	responsible	

4 Conceptual Architecture

So far, we have presented the major issues addressed in Symposium organization and several critical theoretical considerations in our SymposiumPlanner system. This section introduces the conceptual architecture as a novel design artefact (following the design science research methodology) (see Figure 3). Each committee chair acts as a personal agent. As users usually initialize the queries via a web browser and the SymposiumPlanner client assists users to construct queries and get the answers with web pages. OAs, CAs, PAs and EAs are composed of distributed agent topologies and coordinate with each other to complete users' objectives.

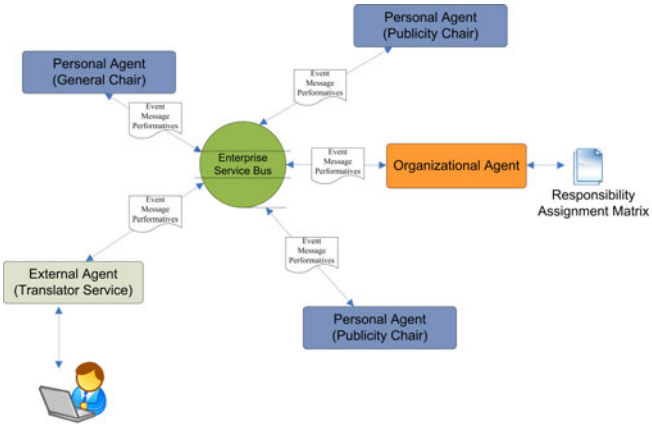

Fig. 3. Conceptual Architecture of SymposiumPlanner

4.1 Organizational Agent

An organizational agent is used to describe the goals shared by its symposium as a whole and contains a knowledge base that describes the symposium's policies, regulations, and opportunities. This knowledge base contains condition/action/event rules as well as derivation rules. An OA manages its local Personal Agents (PAs), providing control of their life cycles and ensuring overall

goals and policies of the organization and its semiotic structures. OAs can act as a single point of entry to the managed sets of local PAs to which requests by EAs are disseminated. This allows for efficient implementation of various mechanisms of making sure the PAs functionalities are not abused (security mechanisms) and making sure privacy of entities, personal data, and computation resources are respected (privacy & information hiding mechanisms) [3]. The selection logic for the dissemination of queries to PAs is described by RAM and OA selects responsible agents with a SPARQL query.

4.2 Personal Agents

In the SymposiumPlanner system, each organization committee chair is designed as a personal agent, which contains a knowledge base that represents its chair's responsibilities to answer corresponding queries. Personal agents are chairs' roles in the symposium organization. But, they might also be services or applications in, e.g. a service oriented architecture. A PA runs a rule engine which accesses different sources of local data and computes answers according to the local rule-based decision logic of the PA. Depending on the required expressiveness to represent the personal agents rule logic, arbitrary rule engines can be used as long as they provide an interface to ask queries and receive answers which are translated into the common interchange format in order to communicate with other agents.

Query Delegation to Personal Agents. Query delegation is done by the organizational agent, but the personal agents can help the OA in this responsibility. Currently, the task responsibility in SymposiumPlanner is managed through a RAM, which defines the tasks that committee members are responsible for. The matrix, defined by an OWL Lite Ontology, assigns roles to topics within the virtual organization. Should there be still no unique PA to delegate a query to, the OA needs to make a heuristic delegation decision and send the query to the PA that most likely would be able to answer the query.

Performatives. Rule Responder is multiple distributed rule system, where each rule agent can run a different rule engine having its own proprietary syntax to access different sources of local data. Usually these distributed agents connect and communicate with each other based on a common rule interchange language, which carries pragmatic performatives. These performatives can be used by the receiver agents to understand the pragmatic context of the message.

Query Answering for Personal Agents. In some cases, the OA can try to solve a query from an external agent by itself, but in the following we consider only cases where it delegates queries to PAs. When a PA receives a query, it is responsible for its answering. If there are multiple solutions to a query, the PA attempts to send an enumeration of as many of the solutions to the OA as possible (it is of course impossible when there are infinitely many solutions).

There are different methods for processing multiple solutions to a query. A naive method of the PAs would be to first compute all of the solutions and then send all of the answers back to the OA, one at a time. After the last answer message is sent, an `end-of-transmission` message is sent to let the OA know that there will be no more messages. The main problem with computing all of the answers before sending any of them is obvious: in case of an infinite enumeration of solutions the OA will not receive any answer. The way our implementation addresses the infinite solutions problem is to interleave backtracking with transmission. When a solution is found, the PA immediately sends the answer, and then begins to compute the next solution while the earlier answer is being transferred. When the OA has received enough answers from such a (possibly infinite) enumeration, it can send a `no-more` message to the PA, stopping its computation of further solutions. Once all solutions have been found in a *finite* interleaved enumeration, the PA can send an `end-of-transmission` message.

If a PA receives a query and the agent does not have any solutions for it, a `failure` message is sent right away back to the OA. If this situation or a timeout occurs (i.e., the PA is offline and did not respond back to the OA within the preset time period), then the OA can try to delegate the query to another PA to see if *it* is able to solve the query. If no solution can be found in any of these ways, a `failure` message is sent back to the external agent that states that the OA (representing the entire organization) cannot solve the query.

Communication between Personal Agent and Expert Owner. One problem that can arise when a personal agent works on a query is that the PA may require help or confirmation from its human 'owner'. The PA may not be able to (fully) answer the query until it has communicated with its human owner. The way we approach this problem is to allow PAs to send messages to their owners and vice versa, e.g. in the form of emails. When the owner receives a PA email, he or she can respond to help finding the answer to the external agent. When the personal agent has received the answer from its owner, the PA can use it to complete the answer to the original query.

Agent Communication Protocols. Rule Responder implements different communication protocols which our agents can utilize. The protocols vary by the number of steps involved in the communication. We try to follow message patterns similar to Web Service communication [8]. For example, there can be **in-only**, **request-response**, and **request-response-acknowledge** protocols, as well as entire **workflow** protocols [6]. Most of the instantiations of SymposiumPlanner primarily focus on the request-response protocol.

Translation between the Interchange Language and Proprietary Languages. Having an interchange language is a key aspect in a distributed rule system. Each agent must be able to understand one common language that every other agent can interpret. The interchange language carries performatives

that each agent is able to understand and react to. Agents can understand the content of the interchange language by interpreting its semantics and pragmatic performative. Each rule engine can have its own platform specific syntax and, in order to run different rule engines in the Rule Responder agents, there must exist a translator between the platform-independent interchange language and the execution syntax of that rule engine.

5 SymposiumPlanner System

Since 2007, we have implemented five instantiations which support the organizing committee of the RuleML Symposium. We implemented the presented Rule Responder agent architecture using the ESB Mule[9]. We mainly use two representative rule engines, namely Prova[10] and OOjDrew [1] (but furhter extended SymposiumPlanner in 2010 to other engine such as Emerald). The developed prototype proves the applicability of the concept in practice. Figure 4 illustrates the general architecture of SymposiumPlanner instantiations that coordinate symposium chairs and the people who are interested in the meeting.

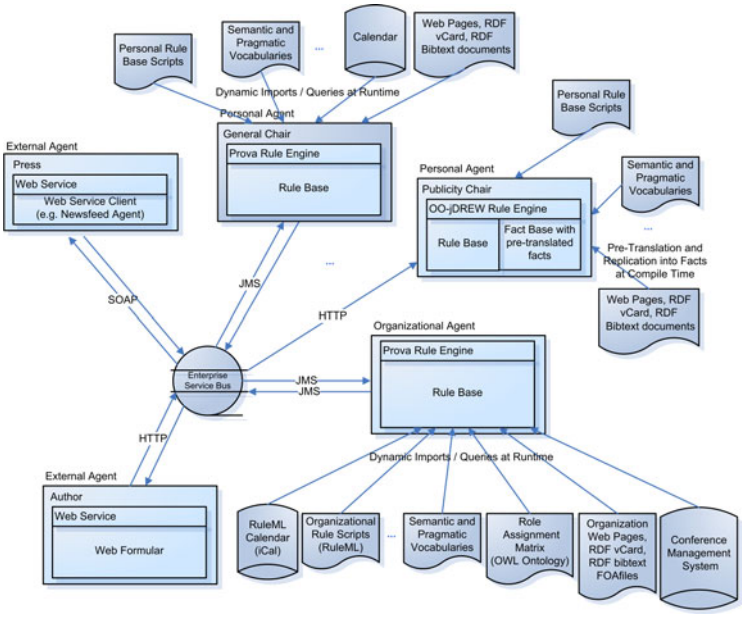

Fig. 4. Main Components of SymposiumPlanner System

[9] http://www.mulesoft.org
[10] http://prova.ws

5.1 Mule Enterprise Service Bus

To seamlessly handle message-based interactions between the Rule Responder agents/services and other agents/services using disparate complex event processing (CEP) technologies, transports, and protocols, the Mule open-source ESB is used in Rule Responder as the communication middleware. This ESB allows deploying the rule-based agents as highly distributed rule inference services installed as Web-based endpoints on the Mule object broker and supports the communication in this rule-based agent processing network via a multitude of transport protocols. That is, the ESB provides a highly scalable and flexible application messaging framework to communicate synchronously or asynchronously amongst the ESB-local agents and with agents/services on the Web.

Distributed agent services, which at their core run a rule engine, are deployed as Mule components which listen at configured endpoints, e.g., JMS message endpoints, HTTP ports, SOAP server/client addresses or JDBC database interfaces, etc. Reaction RuleML is used as a common platform-independent rule interchange format between the agents (and possible other rule execution/inference services). Translator services are used to translate inbound and outbound messages from platform-independent Reaction RuleML into the platform-specific execution syntaxes of rule engines, and vice versa. Extensible Stylesheet Transformations(XSLT) and ANTLR based translator services are provided as Web forms, HTTP services and SOAP Web services on the Reaction RuleML Web page.

The large variety of transport protocols provided by Mule can be used to transport the messages to the registered endpoints or external applications/tools. Usually, JMS is used for the internal communication between distributed agent instances, while HTTP and SOAP are used to access external Web services. The usual processing style is asynchronous using Staged Event Driven Architecture (SEDA) event queues. However, sometimes synchronous communication is needed. For instance, to handle the communication with external synchronous HTTP clients such as Web browsers where requests, e.g. by a Web from, are sent through a synchronous channel. In this case, a synchronous bridge component dispatches the requests into the asynchronous messaging framework and collects all answers from the internal service nodes, while keeping the synchronous channel with the external service open. After all asynchronous answers have been collected, they are sent back to the still connected external service via the HTTP-synchronous channel.

5.2 Platform-Specific Rule Responder Agents

Each agent service might run one or more arbitrary rule engines to execute the interchanged queries, rules and events and derive answers on requests. Prova is a highly expressive Semantic Web rule engine which we used in our reference implementation for agents with complex reaction workflows, decision logic and dynamic access to external Semantic Web data sources. Another rule engine which we applied was the OOjDrew rule engine [1] in order to demonstrate

rule interchange between various rule engines. Further rule engines and event correlation engines (CEP engines) are used in the Rule Responder project in other applications.

Prova follows the spirit and design of the recent W3C Semantic Web initiative and combines declarative rules, ontologies and inference with dynamic object-oriented Java API calls and access to external data sources via query languages such as SQL, SPARQL and XQuery.

One of the key advantages of Prova is its separation of logic, data access, and computation and its tight integration of Java and Semantic Web technologies. Due to the natural integration of Prova with Java, it offers a syntactically economic and compact way of specifying agents' behaviour while allowing for efficient Java-based extensions to improve performance of critical operations.

The main language constructs of messaging reaction rules in Prova are: sendMsg predicates to send messages, reaction rcvMsg rules which react to inbound messages, and rcvMsg or rcvMult inline reactions in the body of messaging reaction rules to receive one or more context-dependent multiple inbound event messages.

5.3 Reaction RuleML

For SymposiumPlanner System we use Reaction RuleML as our interchange language between agents. Reaction RuleML [12] is a general, practical, compact and user-friendly XML-serialized sublanguage of RuleML for the family of reaction rules. It incorporates various kinds of production, action, reaction, and KR temporal/event/action logic rules as well as (complex) event/action messages into the native RuleML syntax using a system of step-wise extensions.

Rule Responder permits agents to use local languages and engines, only requiring that all rule bases, queries, and answers be translated to RuleML for transmitting them to other agents over the Mule ESB. The RuleML Interface Description Language (RuleML IDL) as sublanguage of Reaction RuleML describes the signatures of public rule functions together with their mode and type declarations and narrative human-oriented meta descriptions.

Reaction RuleML provides a translator service framework with Web form interfaces accepting controlled natural language input or predefined selection-based rule templates for the communication with external (human) agents on the computational independent level, as well as Servlet HTTP interfaces, and Web service SOAP interfaces, which can be used for translation into and from platform-specific rule languages such as Prova. On the platform-independent and platform-specific level, the translator services are using different translation technologies such as XSLT stylesheet, Java Architecture for XML Binding (JAXB), etc. to translate from and to Reaction RuleML as a general rule interchange format.

5.4 SymposiumPlanner User Client

One of the main advantages of SymposiumPalnner is that it answers users queries promptly and reduces users' burden of finding the interested information by

themselves. The queries include the information about the symposia and the procures mentioned in section 3.1. For its usability, the SymposiumPlanner user client provides an interface to distributed personal agents, allowing users to query the available interfaces, describe and submit the queries, and retrieve the answers from a standard web browser.

SymposiumPlanner user client allows users to query the SymposiumPlanner agents via the SymposiumPlanner interface either by HTML forms or by a controlled natural language.

The first solution uses the XML based Rule Responder interfaces description file to create HTML forms which present users with the information of interface in detail, such as interface name, parameter and its descriptions, and etc. The translator service will combine the structure of Reaction RuleML message from the Rule Responder interfaces description file with values which users initialize to construct the Reaction RuleML message.

The translation between the used controlled English language and Reaction RuleML is based on domain-specific language translation rules in combination with a controlled English translator service. In SymposiumPlanner, we use Attempto Controlled English [7] which is a rich subset of standard English designed to serve as knowledge representation language. Queries to Rule Responder are formulated in Attempto Controlled English and the ACE2RML translator forwards the text to the Attempto Parsing Engine (APE), which translates the text into a discourse representation structure (DRS) and/or advices to correct malformed input. The DRS gives a logical/structural representation of the text. It is fed into an XML parser which translates it into a domain-specific Reaction RuleML representation of the query. Besides parsing and processing the elements of the DRS, the parser additionally employs domain-specific transformation rules to correctly translate the query into a public interface call of a Rule Responder OA.

6 Conclusion

Following a design science research methodology, we introduced the design principles, conceptual model and component architecture of the Rule Responder SymposiumPlanner agent system. SymposiumPlanner allows users to issue queries to the RuleML conference organisation committee members which are represented by their SymposiumPlanner agents.

SymposiumPlanner is a Semantic Web infrastructure for distributed rule-based event processing multi-agent eco-systems. Based on modern enterprise service technologies and Semantic Web technologies for implementing intelligent rule-based agent services that access data and ontologies, receive and detect events (e.g., for complex event processing in event processing agent networks), and make rule-based inferences and (semi-) autonomous pro-active decisions for reactions based on these representations. The core rule agents of SymposiumPlanner implement the decision and behavioural reaction logic of the agents roles and manage the symposium organization effectively.

SymposiumPlanner instantiations span various implementations from initial state in 07,08,09 to Emerald based instantiation in 2010 to the 2011 double-instantiation using the latest Mule and Prova with a more user friendly interface involving 3 OA's instead of just one for the sake of clarity (although future implementation could see reunification of business logics into one OA as in the initial instantiations).

In SymposiumPlanner, distributed agent instances follow SEDA style, which decomposes a complex, event-driven application into a set of stages connected by event queues. This design decouples event and thread scheduling from application logic and avoids the high overhead associated with thread-based concurrency models. However, although event queues decouples the execution of distributed components, it increases response time correspondingly.

Future implementation will focus on increased automation of processes as well as better support from human operators to add flexibility (e.g. the system already contacts human operator in case the query is not solvable by the agent itself). Further improvements will be also in the form of better support through systems in order to achieve automation in terms of responses from agents as they replace actual human users and try to respond to increasingly complex queries. This might also lead to the need of communication between personal agents in order to help each other in answering queries posed by external agents. However, a single PA might not be able to answer a query as a whole. Another extension to query delegation would thus be query decomposition, followed by delegation of its decomposed parts to multiple PAs, and finally re-integration of the PAs' answers before being sent back to the OA.

References

1. Ball, M., Craig, B.: Object Oriented java Deductive Reasoning Engine for the Web, http://www.jdrew.org/oojdrew/
2. Boley, H., Osmun, T., Craig, B.: Social semantic rule sharing and querying in wellness communities. In: Gómez-Pérez, A., Yu, Y., Ding, Y. (eds.) ASWC 2009. LNCS, vol. 5926, pp. 347–361. Springer, Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-10871-6_24
3. Boley, H., Paschke, A.: Rule responder agents framework and instantiations. In: Eli, A., Kon, M., Orgun, M. (eds.) Semantic Agent Systems. SCI, vol. 344, pp. 3–23. Springer, Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-18308-9_1
4. Cardoso, H.L.: Integrating jade and jess (2007), http://jade.tilab.com/doc/tutorials/jade-jess/jade_jess.html
5. Craig, B.L.: The OO jDREW Engine of Rule Responder: Naf Hornlog RuleML Query Answering. In: Paschke, A., Biletskiy, Y. (eds.) RuleML 2007. LNCS, vol. 4824, pp. 149–154. Springer, Heidelberg (2007)
6. Craig, B.L., Boley, H.: Personal agents in the rule responder architecture. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) RuleML 2008. LNCS, vol. 5321, pp. 150–165. Springer, Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-88808-6_17

7. Fuchs, N.E., Kaljurand, K., Schneider, G.: Attempto controlled english meets the challenges of knowledge representation, reasoning, interoperability and user interfaces. In: Sutcliffe, G., Goebel, R. (eds.) FLAIRS Conference, pp. 664–669. AAAI Press (2006), http://dblp.uni-trier.de/db/conf/flairs/flairs2006.html#FuchsKS06
8. Gudgin, M., Lewis, A., Schlimmer, J.: Web Services Description Language (WSDL) Version 1.2 Part 2: Message Patterns, http://www.w3.org/TR/2003/WD-wsdl12-patterns-20030611/
9. Hindriks, K.V., De Boer, F.S., Van Der Hoek, W., Meyer, J.-J.C.: Agent programming in 3apl. Autonomous Agents and Multi-Agent Systems 2, 357–401 (1999)
10. Kravari, K., Kontopoulos, E., Bassiliades, N.: Emerald: A multi-agent system for knowledge-based reasoning interoperability in the semantic web. In: Konstantopoulos, S., Perantonis, S., Karkaletsis, V., Spyropoulos, C.D., Vouros, G. (eds.) SETN 2010. LNCS, vol. 6040, pp. 173–182. Springer, Heidelberg (2010)
11. Lehman, J.F., Laird, J., Rosenbloom, P.: A gentle introduction to soar, an architecture for human cognition. In: Sternberg, S., Scarborough, D. (eds.) Invitation to Cognitive Science. MIT Press (1996)
12. Paschke, A., Kozlenkov, A., Boley, H.: A homogenous reaction rules language for complex event processing. In: International Workshop on Event Drive Architecture for Complex Event Process (2007), http://ibis.in.tum.de/staff/paschke/
13. Paschke, A.: Rule responder hcls escience infrastructure. In: Proceedings of the 3rd International Conference on the Pragmatic Web: Innovating the Interactive Society, ICPW 2008, pp. 59–67. ACM, New York (2008), http://doi.acm.org/10.1145/1479190.1479199
14. Paschke, A., Bichler, M.: Knowledge representation concepts for automated sla management. CoRR abs/cs/0611122 (2006)
15. Paschke, A., Kozlenkov, A.: A rule-based middleware for business process execution. In: Multikonferenz Wirtschaftsinformatik (2008)
16. Schroeder, M., Wagner, G.: Vivid agents: Theory, architecture, and applications. Applied Artificial Intelligence 14(7), 645–675 (2000)
17. Shoham, Y.: Agent-oriented programming. Artif. Intell. 60(1), 51–92 (1993)
18. Wang, M., Purvis, M., Nowostawski, M.: An internal agent architecture incorporating standard reasoning components and standards-based agent communication. In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2005, pp. 58–64. IEEE Computer Society, Washington, DC (2005), http://dx.doi.org/10.1109/IAT.2005.43
19. Wright, I., Marshall, J.: The execution kernel of rc++: Rete*, a faster rete with treat as a special case. International Journal of Intelligent Games and Simulation 2 (2003)