# Rule Responder HCLS eScience Infrastructure

Adrian Paschke
Free University Berlin, Germany
paschke at inf.fu-berlin.de

## ABSTRACT

The emerging field of integrative bioinformatics provides enabling methods and technologies for transparent information integration across distributed heterogenous data sources, tools and services. The aim of this article is to evolve a flexible and expandable distributed Pragmatic Web eScience infrastructure in the domain of Health Care and Life Science (HCLS), called Rule Responder HCLS. Rule Responder HCLS is about providing information consumers with rule-based agents to transform existing information into relevant information of practical consequences, hence providing control to the end-users by enabling them to express in a declarative rule-based way how to turn existing information into personally relevant information and how to react or make automated decisions on top of it.

## Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods

## General Terms

HCLS, eScience, Multi Agent Systems, Representation Languages, Coordination, Collaboration

## Keywords

Pragmatic Agent Web, Rule Interchange Format, Reaction RuleML, Complex Event Processing, Prova

## 1. BACKGROUND

Transparent information integration across distributed and heterogeneous data sources and computational tools is a prime concern for bioinformatics. While there are systems, which integrate a number of databases and tools, such as SRS, Expasy, NCBI, etc. these systems are hard-wired, cannot adapt to changes in the underlying information resources, and it is not possible to freely customize information workflows. They also rely on a hand-crafted mapping of schemata of the different information resources. There are some efforts to tightly integrate data and computational resources such as Edit2Trembl [1], which integrates annotations of tools for transmembrane prediction, GeneWeaver [2], which integrates genome annotations, MyGrid [3], which provides personalised access to bioinformatics resources focusing on service management and discovery as part of the workflows.

Recently, the Semantic Web as a way to semantically specify data and data relationships and as a collection of specific technologies such as RDF, OWL, RIF, SPARQL, has been adopted to integrate bioinformatic knowledge from multiple publicly available databases and services (see e.g. [15]). The Semantic Web builds upon XML as the common machine-readable syntax to structure content and data, upon RDF as a simple language to express property relationships between arbitrary resources (e.g., objects or topics) identified by URIs, ontology languages such as RDFS or OWL as means to define rich vocabularies (ontologies) which are then used to precisely describe resources, their relations and their semantics, and on rules, represented e.g. in W3C RIF, which are used to specify complex decision logic, reaction logic and process workflows. This prepares an infrastructure to share the relevant meaning of content and leads to a more machine-processable and relevant Web. Besides ontologies, a Semantic Web will contain web services, which extend traditional web pages in that they allow access to remote computations and resulting data. Many bioinformatics projects, such as UniProt, Tambis, FungalWeb, YeastHub, BioPax have meanwhile adopted the Semantic Web approach (in particular the RDF standard) and large ontologies such as the Gene Ontology (GO) [4] are provided as RDFS or OWL ontologies. This has been utilized by several bioinformatics integration projects, such as W3C HCLS RDF or Bio2RDF, to solve the old problem of distributed heterogeneous data integration in health care and life sciences. The W3C HCLS RDF interest group follows a data warehousing approach to transform and store relevant knowledge from various sources (e.g. Entrez Gene, UniProt, OMIM, Enzyme) into an unified knowledge base, a RDF triple store, which provides a SPARQL query interface to extract information from it. [15] In contrast, the Bio2RDF builds a database of links between data and data sources, but not replicating any of it.

Such a Semantic Web that consists of data specified with a hierarchical common ontology and web services for computa-


$3^{rd}$ *International Conference on the Pragmatic Web* Sept 28-30, 2008, Uppsala, Sweden.

tions can be programmed by marrying concepts from declarative programming, object- oriented programming and service-oriented programming. Rules, reasoning, and logic are needed for dealing with ontologies and for specifying workflows and relationships between the data. Object- orientation is needed for reflecting the hierarchy of the ontology and for implementing services and computations. Service-oriented programming is needed to implement distributed loosely-coupled service oriented architectures (SOAs) where rule inference services are deployed on the Web. Moreover, rule-based programming has another advantage. It transparently extends the expressiveness and power of relational databases that are often underlying bioinformatics databases. Rules serve as virtual table generators that derive knowledge from elementary facts. Rule-based programming and object-orientation serve complementary purposes. Declarative programming style is very good for flexible data integration, inference, and for specifying if-then logic. Object-orientation is optimal for modeling complex systems that are relatively stable and well understood. Declarative programming is good for logic, while object-orientation is good for computation.

The Rule Responder HCLS (Health Care and Life Sciences) eScience infrastructure presented in this article, is paving a way for intelligent knowledge intensive data integration useful in knowledge discovery for biomedical research. Among many other features it allows asking queries / deriving new knowledge, getting decision support, submitting research tasks to existing services/tools, automatically triggering actions, and getting reactions. It uses and extends the Semantic Web towards a Pragmatic Web (syntax+semantic+pragmatic) as an infrastructure for collaborative and integrative networks, where rule-based agents capable of multimodal communication will provide access to existing Web-based information and tools. End-users such as researcher will instruct agents to access and transform the existing information in highly customized ways via using declarative rules. This transformation may be as simple as querying data from one single data provider, e.g. by using SQL, SPARQL or XQuery, or may be as complex as intelligently fusing the information from many different Web sources/services and inferring / computing new knowledge from it by means of declarative rules representing decision logic, or as automatically detecting actionable situations from occurred events and reacting according to a rule-based workflow-like logic possibly involving multiple other agents and services. The agents will work together to combine information from multiple data sources, access information autonomously, automatically react to occurred relevant events or are queried by human end-users via Web user interfaces to the eScience infrastructure.

The goal of this article is to show how the Rule Responder approach can be used to build a flexible, loosely-coupled and service-oriented eScience infrastructure which allows to wrap the existing web data sources, services and tools by rule-based agents which access and transform the existing information into relevant information of practical consequences for the end-user. Before we describe the architecture and implementation of the Rule Responder eScience infrastructure and demonstrate how end-user programmable agents allow users to fuse information from multiple sources into rule executable formats and hence allow to make relevant information show up at the right time (including automated reactions from the configured agents), we first introduce a concrete motivating use case.

## 2. ALZHEIMER USE CASE

Alzheimer's is a debilitating neurodegenerative disease that affects approximately 27 million people worldwide. The cause of Alzheimer's is currently unknown and no therapy is able to halt its progression. Insights into the mechanism and potential treatment of this debilitating disease may come from the integration of different existing neurological, biomedical and biological resources. By integrating e.g. information about pyramidal neurons, their genes, and gene products, signal transduction, drug target candidates that are potentially effective against Alzheimer's Disease (AD) can be provided to researchers. Moreover, further relevant information such as experts and top locations for certain elements of putative functional importance to Alzheimer's research can be discovered from statistics data from e.g. publication and patent databases.

In the following we will exemplify a typical knowledge discovery process of a researcher, who is looking for drug target candidates that are potentially effective against Alzheimer's Disease and who is looking for experts in this field who can be contacted. Table 1 lists the research questions, the scientific resources which are requested, the research results achieved, and the newly discovered knowledge which leads to further queries in the discovery process.

The research first discovers from Uniprot[1], the W3C HCLS KB[2] and the SWAN data[3] that Beta amyloid in various forms, and in particular ADDLs, are good therapeutic targets. He/she then searches the PubMed database about articles on ADDLs research for AD and ranks the results using the GoPubMed[4] statistics to find the top location, which is Evanston, and the top author, who is William Klein, for this research field. From this, the researcher makes the hypothesis that William Klein works in Evanston, and simply proves it using Google search which confirms that William Klein's affiliation is indeed Evanston. Finally, the researcher queries the EMBI-EBL patent abstract database[5] for patents addressing ADDLs as therapeutic target for AD and concludes that William Klein who also holds two patents is one of the top experts in ADDLs research.

That is, to find an AD expert the researcher accesses at least four different services and data sources as illustrated in figure 2. Implicitly the researcher executes the following rule: *IF a Person has most publications in the Field and one or more Patents in the field THEN the Person is an expert for this Field.*

In the following section we will evolve a Pragmatic Web eScience infrastructure and will demonstrate how it can automate this use case.

## 3. RULE RESPONDER HCLS ESCIENCE INFRASTRUCTURE

The Rule Responder architecture[6] extends the Semantic Web towards a Pragmatic Web infrastructure for collabo-

[1] http://beta.uniprot.org/
[2] http://sw.neurocommons.org/2007/kb.html
[3] http://swan.mindinformatics.org/
[4] http://www.gopubmed.org/
[5] http://srs.ebi.ac.uk/srsbin/cgi-bin/wgetz?-page+query+-id+3BCib1WGHAl
[6] http://responder.ruleml.org

| Research Questions | Actions | Search Results | Knowledge discovered |
|---|---|---|---|
| What is a good therapeutic target for **Alzheimer Disease** (AD)? | Search **SWAN** for **target OR therap***(citesConcept `TherapeuticTarget` - see query 1 below) | Returns 2 hypotheses [add specific statements] | Beta amyloid in various forms (ADDL, oligomer, etc.) are targets |
| Which publications are about **ADDLs**? | Search **goPubMed** for publications about **ADDLs** | Returns (currently) 29 articles | Possible therapeutic approaches aimed at lowering ADDLs in AD patients |
| What is the top location for **Alzheimer disease** reasearch addressing **ADDLs** as therapeutic target? | Use the **GoPubMed Statistics service** to compute the statics based on the found publications | Returns **Evanston** as the top location | Most publications about ADDLs are coming from Evanston |
| Who is the top author in ADDLs research? | Use the **GoPubMed Statistics service** to compute the statics based on the found publications | Returns William Klein as top author | Hypothesis: William Klein works in Evanston. Simple prove (calls Google service API): http://www.biochem.northwestern.edu/ibis/faculty/klein.htm |
| Which patents exists for ADDLs? | Search **EMBI-EBL patent abstract database** for patents addressing ADDLs | Return (currently) two patents from Klein William | AMYLOID BETA-DERIVED DIFFUSIBLE LIGANDS (ADDLS), ADDL-SURROGATES, ADDL-BINDING MOLECULES, AND USES THEREOF |
| Who is the expert in ADDLs research? | Use the expert finder rule to find the expert which combines the previous five queries 1-5 | Return (currently) Klein William as expert since he has most publications and two patents in ADDLs research | William Klein is the (current) expert in ADDLs reasearch as therapeutic target for AD |

**Figure 1: Example Research Queries about Alzheimer Disease**

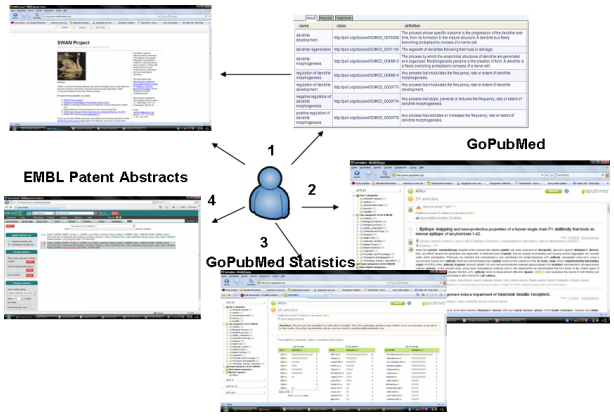


**Figure 2: Alzheimer Disease Use Case**

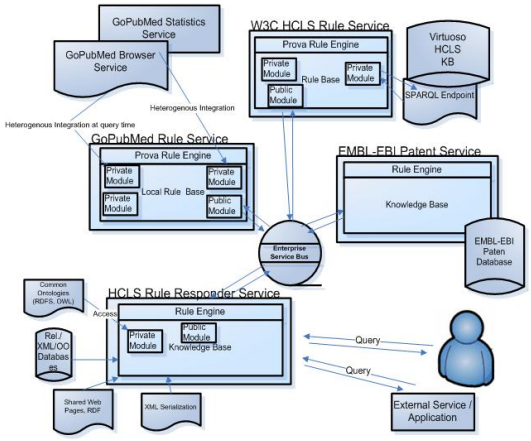


**Figure 3: Rule Responder HCLS eScience Infrastructure**

rative agent networks where independent agents engage in conversations by exchanging messages and cooperating to achieve collaborative goals. Rule Responder utilizes modern enterprise service technologies and the Semantic Web with intelligent agent services that access data and ontologies, receive and detect events (complex event processing), and make rule-based inferences and autonomous pro-active decisions and reactions based on these representations. In Rule Responder rules of various types (derivation, integrity, reaction, messaging) are first class citizens to program agent behavior and agent decision logic. Rule Responder adopts the OMG model driven architecture (MDA) approach:

1. On the computational independent level rules are engineered in a Rule Manager user interface in a natural controlled English language using blueprint templates and user-defined vocabularies and domain-specific translation rules.

2. The rules are mapped and serialized in Reaction RuleML XML which is used as platform independent rule interchange format to interchange rules between Rule Responder inference services (agents) and arbitrary other rule execution environments.

3. The Reaction RuleML rules are translated into the platform specific Prova rule language for execution, which is based on the ISO Prolog syntax.

The three core parts of the architecture (see figure 3) are (1) Reaction RuleML[7] as a common platform-independent rule interchange format to interchange rules, events and data between agent services and other Semantic Web tools, (2) a highly scalable and efficient enterprise service bus (ESB) as agent/service-broker and communication middleware ([6]), and (3) Prova rule engines[8] as platform-specific (Java) rule-based agent execution environments.

The core of a Rule Responder agent is a Prova rule engine (`http://prova.ws`) which is deployed as distributed web-based services on an ESB. The agents are distributed, have

[7] `http://ibis.in.tum.de/research/ReactionRuleML/`
[8] `http://prova.ws`

local autonomy and are decoupled via exchanging Reaction RuleML messages either asynchronously or synchronously over the ESB, or in an ad-hoc conversation directly between two agents using agent protocols such as Jade-HTTP. Arbitrary transport protocols such as MS, SMTP, JDBC, TCP, HTTP, XMPP can be used to transport rule sets, queries and answers as payload of Reaction RuleML event messages between the agent services. Each agent dynamically imports or pre-compiles and loads a possibly distributed rule base which implements the decision and (behavioral) reaction logic of the agent.

## 3.1 Prova Rule Engine

The core Prova rule language and engine is a highly expressive, hybrid, declarative and compact rule programming language which combines the declarative programming paradigm with object-oriented programming (in particular Java) and the Semantic Web approach. Prova is both a Semantic Web rule language and a highly expressive distributed Semantic Web rule engine which supports complex reaction rule-based workflows, rule-based complex event processing, distributed inference services, rule interchange, rule-based decision logic and dynamic access to external data sources, Web-based services, and Java APIs. The Prova rule engine supports different rule types:

- Derivation rules to describe the agent's decision logic
- Integrity rules to describe constraints and potential conflicts
- Normative rules to represent the agent's permissions, prohibitions and obligation policies
- Global reaction rules to define global reaction logic which are triggered on the basis of detected complex events (event patterns defined by an event algebra)
- Messaging reaction rules to define conversation-based workflow reaction and behavioral logic based on complex event processing

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

```
File Input / Output
    ..., fopen(File,Reader), ...
XML (DOM)
    document(DomTree,DocumentReader) :-  XML(DocumenReader),...
SQL
    ... ,sql_select(DB,cla,[pdb_id,"1alx"],[px,Domain]).
RDF
    ...,rdf(http://...,"rdfs",Subject,"rdf_type","gene1_Gene"),...
XQuery
 ..., XQuery = 'for $name in StatisticsURL//Author[0]/@name/text()
    return $name', xquery_select(XQuery,name(ExpertName)),...
SPARQL
    ...,sparql_select(SparqlQuery,...
```

One of the key advantages of Prova is its elegant separation of logic, data access, and computation as well as its tight integration of Java, Semantic Web technologies and enterprise service-oriented computing and complex event processing technologies. Due to the natural integration of Prova with Java, it offers a syntactically economic and compact way of specifying agents' behavior while allowing for efficient Java-based extensions to improve performance of critical operations.

```
hello(Name):-
  S = java.lang.String("Hello").
  S.append (Name),
  java.lang.System.out.println (S).
```

Via its capabilities to integrate external vocabularies such as Semantic Web ontologies and Java type systems the rule language can be custom-tailored to a specific agent vocabulary giving the used rule constructs a specific meaning with a clearly defined semantics, e.g., based on the FIPA vocabulary. Often, agents form collaborative networks where the agents interact in conversation dialogs which might follow a pre-defined negotiation or coordination protocol. This might also include negotiation and discussion about the meaning of ontological concepts, since agents might use their own micro-ontologies / vocabularies and must agree on relevant shared concepts to enable an efficient communication and knowledge interchange between the agent nodes. In addition to backward-reasoning derivation rules, which for instance are used to implement decision or planning logic, Prova supports messaging reaction rules specifying the sequences or branches of actions that an agent responds with upon detection of a matching pattern in the inbound messages which the agent receives from other agents or the environment (other tools, services). Via messaging reaction rules with constructs for asynchronously sending and receiving event messages an agent interacts with the environment. Messaging reaction rules describe (abstract) processes in terms of message-driven conversations between parties (agents or external services / APIs) and represent their associated interactions via constructs for asynchronously sending and receiving event messages.

The main language constructs of messaging reaction rules (as implemented in the rule engine Prova) are: *sendMsg* predicates to send outbound 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:

```
sendMsg(XID,Protocol,Agent,Performative,Payload |Context)
rcvMsg(XID,Protocol,From,Performative,Paylod|Context)
rcvMult(XID,Protocol,From,Performative,Paylod|Context)
```

where *XID* is the conversation identifier (conversation-id) of the conversation to which the message will belong. *Protocol* defines the communication protocol such as JMS, HTTP, SOAP, Jade etc. *Agent* denotes the target party of the message. *Performative* describes the pragmatic context in which the message is send. A standard nomenclature of performatives is, e.g. the FIPA Agents Communication Language ACL or the BPEL activity vocabulary. *Payload* represents the message content sent in the message envelope. It can be a specific query or answer or a complex interchanged rule base (set of rules and facts). Remarkably, the messaging reaction rules do not require separate threads for handling multiple conversation situations simultaneously.

For instance, the following messaging reaction rule (variables in upper case letters) waits for an inbound query.

```
% receive query and delegate it to another party
rcvMsg(CID,esb, Requester, acl_query-ref, Query) :-
```

```
  responsibleRole(Agent, Query),
  sendMsg(Sub-CID,esb,Agent,acl_query-ref, Query),
  rcvMsg(Sub-CID,esb,Agent,acl_inform-ref, Answer),
  ... (other goals)...
  sendMsg(CID,esb,Requester,acl_inform-ref,Answer).
```

When activated it first selects the responsible role for the query. This selection logic is e.g. implemented by standard derivation rules as described above. Then the rule sends the query in a new sub-conversation to the selected party and waits for the answer to the query. That is, the rule execution blocks until an answer event message is received in the inlined sub-conversation which activates the process flow again, e.g. to prove further "standard" goals, e.g. with information from the received answer which is bound to variables in the normal logic programming way including also backtracking to several variable bindings. Finally, in this example the rule sends back the answer to the original requesting party.

The corresponding messaging reaction rule of the responsible party which receives the query, derives the answer and sends it back in the sub-conversation might look like this:

```
% answers query
rcvMsg(XID, esb, From, Performative, [X|Args]):-
  derive([X|Args]),
  sendMsg(XID,esb,From, answer, [X|Args]).
```

The payload of incoming event messages is interpreted with respect to the local conversation state, denoted by the conversation id, and the pragmatic context which is defined by a possibly complex pragmatic performative(s). Depending on the pragmatic context the message payload is used e.g., to update the internal knowledge of the agent (e.g. adds new facts or rule sets), add new tasks (goals), or detect a complex event pattern (from the internal event instance sequence). Several reasoning processes might run in parallel local to their conversation flows. Inactive reactions (conversation partitions) are removed from the system. Self activations by sending a message to the receiver "self" are possible. Several expressive logic formalisms are provided for updating the knowledge base (transactional update logic), defining and detecting complex events (complex event algebra) [5], handling situations (states) (event calculus), and for reasoning (e.g. deontic logic for normative reasoning on permissions, prohibitions, obligations [6]) and planning (abductive reasoning on plans and goals [7]). These scripts can be added selectively as modules to the rule based of an agent hence providing the needed (meta) reasoning, planning and reaction capabilities to formalize the agent behavior in terms of rules.

## 3.2 Reaction RuleML Rule Interchange Format

On the platform-independent and web-based rule interchange level, Reaction RuleML is used. Via translator services the interchanged RuleML messages are translated into the platform-specific execution syntaxes of the agents' rule execution environments, i.e. Prova. Translator services translate from Prova execution syntax into Reaction RuleML and vice versa. Arbitrary transport protocols (more than 30 protocols) such as MS, SMTP, JDBC, TCP, HTTP, XMPP can be used to transport rule sets, queries and answers between distributed Prova endpoints on the ESB using e.g. Reaction RuleML as common rule interchange format. The general syntax of a (reaction) rules is:

```
<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>
```

According to the selected and omitted rule parts a rule specializes, e.g. to a derivation rule (if-then or if-then-else; reasoning style), a trigger rule (on-do), a production rule (if-do), an ECA rule (on-if-do) and special cases such as ECAP rule (on-if-do-after) or mixed rule types such as derivation rule with alternative action (if-then-elseDo), e.g. to trigger an update action (add/remove from KB) or send an event message (e.g. to a log system) in case a query on the if-then derivation rule fails.

```
Derivation Rule:    <Rule style="reasoning">
                         <if>...</if>
                         <then>...</then>
                    </Rule>

Production Rule:    <Rule style="active">
                         <if>...</if>
                         <do>...</do>
                    </Rule>

ECA Rule:           <Rule style="active">
                        <on>...</on>
                        <if>...</if>
                        <do>...</do>
                    </Rule>
```

To specify the semantics of rules, and safeguard the life cycle of rules, including their interchange between different platform-specific rule services, Prova supports test-driven development for self-validating rule bases, where rule sets are interchanged together with test suites (serialized in Reaction RuleML) which are used for verification and validation (V&V) of the interchanged rules in the target execution environment. Test cases for V&V of rule bases are particular well-suited when rule applications grow larger and more complex and are maintained (possibly distributed) by different people and are interchanged between domain and platform boundaries. They help to capture the original rule engineer's intended meaning of a rule-based program and safeguard the evolution of the intensional knowledge base, i.e. facilitate updates and extensions of the rule base in order to adapt the rule logic to changing requirements.

## 3.3 Natural Language Web Interface and Rule Manager User Interface

Online user interfaces can be setup which allow issuing queries in a controlled natural language via a web browser to the eScience rule infrastructure and receive answers from it. The translation between the used controlled English rule language (Attempto Controlled English [16]) and Reaction RuleML is based on a vocabulary template-driven approach in combination with a controlled English translator.

Figure 4 shows the architecture of the ACE to Reaction RuleML translator web service of the Rule Responder infrastructure. Queries to Rule Responder are formulated in Attempto controlled natural language. The ACE2RML trans-

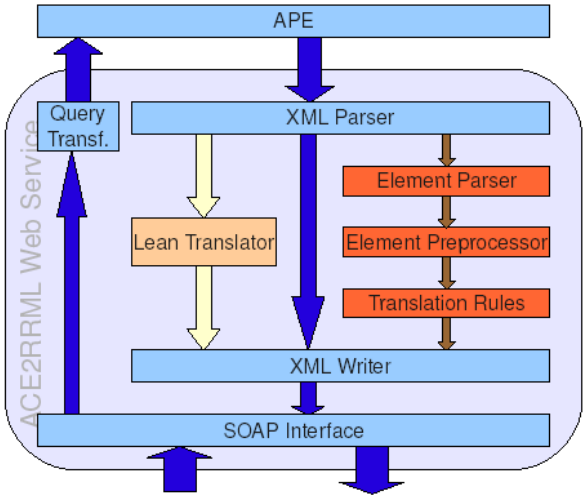


**Figure 4: ACE2RML Translator Web Service**

lator forwards the 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 to an XML parser which translates into a domain specific Reaction RuleML representation of the query. Beside parsing and processing the elements of the DRS it additionally employs domain-specific transformation rules to correctly translate the query into a public interface call of a Rule Responder HCSL agent. For instance, consider the following user question to Rule Responder in ACE format

```
Does ADDLs target Alzheimer ?
```

This is translated by the APE into the following DRS:

```
<DRS domain="A B C">
  <object ref="A" noun="ADDLs" struct="named"
        unit="na" numrel="eq" num="1" sentid="1"/>
  <object ref="B" noun="Alzheimer" struct="named"
        unit="na" numrel="eq" num="1" sentid="1"/>
  <predicate ref="C" verb="target" subj="A" obj="B" sentid="1"/>
</DRS>
```

By parsing the DRS and applying the domain-specific rules which map named objects to constants and predicate relations to atomic predicates the following Reaction RuleML query is produced:

```
<Atom>
  <Rel>target</Rel>
  <Ind>ADDLs</Ind>
  <Ind>Alzheimer</Ind>
</Atom>
```

Online translator web services using XSLT are provided which translate from Reaction RuleML into Prova. For the translation from Prova back into Reaction RuleML an Abstract Syntax Tree (AST) using ANTLR v3 is build based on the Prova grammar (`http://www.prova.ws/gram.html`). The AST is translated into RuleML XML syntax. The functionality is also wrapped into online translator services.

The translator services are configured in the transport channels of the inbound and outbound links of the deployed online Prova inference service. Incoming Reaction RuleML rule documents (received messages) are translated into Prova rule sets which can be executed by the local Prova rule engine and outgoing rule sets (send) are translated into RuleML in the outbound channels before they are transferred via a selected transport protocol such as HTTP.

Additionally, a Rule Manager tool supports collaborative role-based engineering, project-specific management and runtime execution of Prova projects serialized in Reaction RuleML. The tool splits into two integrated components, the project manager which provides development and testing of rule projects, and the runtime dashboard which enables execution of rules and visualization of results. [6]

## 3.4 Enterprise Service Bus as 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 an enterprise service bus (ESB), the Mule open-source ESB [9], is integrated as communication middleware. The ESB allows deploying the rule-based agents as highly distributable rule inference services installed as Web-based endpoints in the Mule object broker and supports the Reaction RuleML based communication between them. That is, the ESB provides a highly scalable and flexible application messaging framework to communicate synchronously but also asynchronously with external services and internal agents.

Mule is a messaging platform based on ideas from ESB architectures, but goes beyond the typical definition of an ESB as a transit system for carrying data between applications by providing a distributable object broker to manage all sorts of service components. The three processing modes of Mule are [9]:

- Asynchronously: many events can be processed by the same component at a time in various threads. When the Mule server is running asynchronously instances of a component run in various threads all accepting incoming events, though the event will only be processed by one instance of the component.
- Synchronously: when a UMO Component receives an event in this mode the whole request is executed in a single thread
- Request-Response: this allows for a UMO Component to make a specific request for an event and wait for a specified time to get a response back

The object broker follows the Staged Event Driven Architecture (SEDA) pattern [10]. The basic approach of SEDA is to decomposes a complex, event-driven application into a set of stages connected by queues. This design decouples event and thread scheduling from application logic and avoids the high overhead associated with thread-based concurrency models. That is, SEDA supports massive concurrency demands on Web-based services and provides a highly scalable approach for asynchronous communication.

Figure 5 shows a simplified breakdown of the integration of Mule into Rule Responders' Pragmatic Agent Web.

Several agent services which at their core run a rule engine are installed as Mule components which listen at configured endpoints, e.g., JMS message endpoints, HTTP ports,

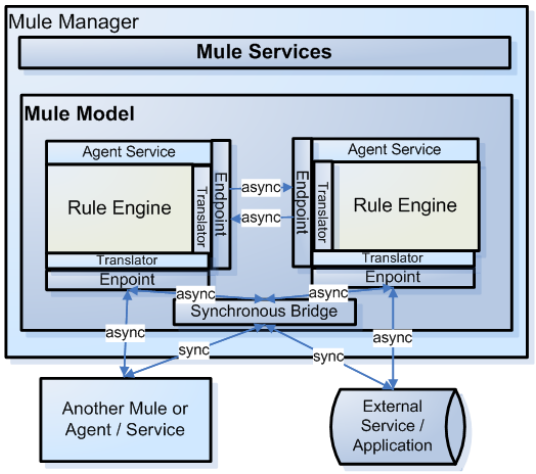


**Figure 5: Integration of Mule into RBSLM**

SOAP server/client addresses or JDBC database interfaces. 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 and/or controlled natural language into the platform-specific rule engines execution syntaxes and vice versa. 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 is used to access external Web services. The usual processing style is asynchronous using SEDA event queues. However, sometimes synchronous communication is needed. For instance, to handle communication with external synchronous HTTP clients such as Web browsers where requests, e.g. by a Web from, are send 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 send back to the still connected external service via the synchronous channel.

# 4. IMPLEMENTATION OF THE ALZHEIMER USE CASE

To illustrate how the described eScience infrastructure works we revisit the Alzheimer use case described in section 2 and implement it as shown in figure 3. The use case is available on the Rule Responder website.

The HCLS Rule Responder agent service (see figure 3) implements the main logic of the eScience infrastructure and acts as the main communication endpoint for external agents such human users which communicate with it via web interfaces (web browser) or automated external agents (e.g. web services, tools). It's rule code defines the public interfaces to receive requests (queries, tasks) to the eScience infrastructure and the logic to look-up the respective source agents and delegate requests to them in order to answer the queries and fulfil the tasks. This might possibly involve splitting of complex requests into several sub-queries which are issued to different source agents, including e.g. upload of mobile rule code for distributed problem solving to more than one agent as exemplified by the following rules:

```
% Manager
upload_mobile_code(Remote,File) :-
    Writer = java.io.StringWriter(),
    % Opening a file
    fopen(File,Reader),
    copy(Reader,Writer),
    Text = Writer.toString(),
    SB = StringBuffer(Text),
    sendMsg(XID,esb,Remote,query-ref,consult(SB)).

% Service (Contractor)
rcvMsg(XID,esb,Sender,eval,[Predicate|Args]):-
    derive([Predicate|Args]).
```

Each existing legacy data sources / service (see figure 3) is wrapped by a Rule Responder source agent which runs a Prova rule engine. The agent's rule base comprises the local rule interface descriptions, i.e. the rule functions which can be queried by other agents of the eScience infrastructure, the respective transformation rules to issue queries to the platform-specific services and access the heterogenous local data sources, and the rule logic to process incoming requests and derive answers / information from the local knowledge. For instance, the following Prova code snippet issues a query to the SPARQL endpoint of the W3C HCLS knowledge base hosted at DERI in Ireland [9].

```
...
    SparqlServiceURL =
    'http://hcls.deri.ie/sparql/?query=SPARQLQUERY&format=
    application/sparql-results+xml',
QueryString = '
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX go: <http://purl.org/obo/owl/GO#>
PREFIX obo: <http://www.geneontology.org/formats/oboInOwl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

select  ?name  ?class ?definition
from <http://purl.org/commons/hcls/20070416>
where
{   graph <http://purl.org/commons/hcls/20070416/classrelations>
       {?class rdfs:subClassOf go:GO_0008150}
?class rdfs:label ?name.
?class obo:hasDefinition ?def.
?def rdfs:label ?definition
filter(regex(?name,"[Dd]endrite"))
}
',
sparql_select(SparqlServiceURL,QueryString,
       name(Name),class(Class),definition(Def)),
...
```

Figure 6 shows the simplified activity flow for the use case. A user asks a query in controlled English to the Rule Responder eScience infrastructure. The query is translated and interpreted by the main Rule Responder HCLS agent. If the query is not understood by the HCLS agent feedback from the DRS and the shared ontologies is send to the user, who now might reformulate the query or start a new sub-conversation to clarify about used terminologies. If the query is understood it is further processed, e.g. mapped

[9] `http://hcls.deri.ie/sparql/`

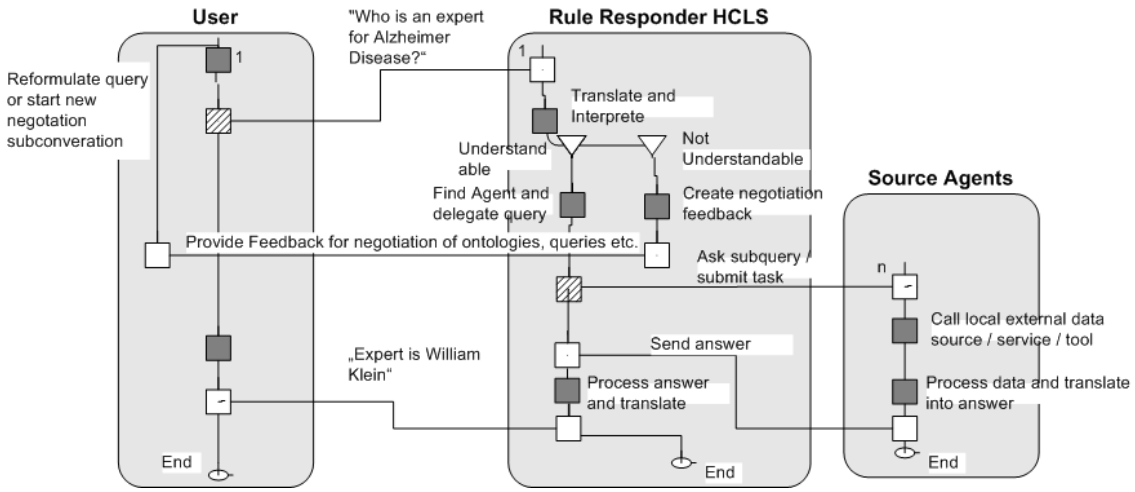


**Figure 6: Role Activity Diagram of Alzheimer Use Case in Rule Responder**

into several sub-tasks according to the agent's rule logic, and delegated to the respective Rule Responder source agents. These agents translate the requests into calls / queries to the local external data sources such as a SPARQL query to the W3C HCLS triple store or a call to the GoPubMed services. Each source agent itself might start new sub-conversations with other source agents to fulfil its tasks. The received data is processed and send back to the Rule Responder HCSL agent, who translates it into an answer in ACE format which is send to the user as response to the query.

## 5. COMPARISON AND CONCLUSION

Transparent information integration across distributed and heterogeneous data sources and computational tools is a prime concern for bioinformatics. Rules come in many guises such as database views, logic programs, constraints, description logics, ontologies, active rules, etc. They have been employed in bioinformatics in many different contexts such as model checking of biochemical networks [12], ontologies for transparent access to bioinformatics databases [13], ontologies for health applications [14], and integration of an ontology with gene expression data [14], logic programming for consistent annotation of proteins [1], constraints for structure prediction [11].

To demonstrate the use of Semantic Web technologies and practices to improve collaboration, research and development, and innovation adoption in the Health Care and Life Science domains the W3C Health Care and Life Science Interest Group (W3C HCLS IG [10]) has built a central RDF triple store database [11] which integrates a number of public datasets and ontologies into a central Virtuoso RDF triple store database hosted by Science Commons as Neurocommons Triple Store [12] which provides a SPARQL query interface [13] to answer scientific queries from the cental KB. For a list of information sources which have been integrated into the KB see [14]. More details on HCLS are available at: `http://www.w3.org/2001/sw/hcls/`. While the prevailing paradigm in the W3C HCLS approach is on a homogenous data integration approach where different data sources are mapped, translated and partially replicated in a homogenous RDF triple store as knowledge base, the aim of Rule Responder HCLS is to provide an open loosely-coupled eScience service infrastructure.

With Rule Responder HCLS we have evolved a rule-based approach which facilitates easy heterogenous systems integration and provides computation, database access, communication, web services, etc. This approach preserves local anonymity of local agent nodes including modularity and information hiding and provides much more control to users with respect to the relatively easy declarative rule-based programming techniques. The rules allow to specify where to access information, how to process information, how to invoke information access, how to present information and automatically react to it, and how to transform the general information available from existing data sources on the Web into personally relevant information accessible via the eScience infrastructure. The Rule Responder eScience infrastructure is available online at `http://ibis.in.tum.de/projects/paw/hcls/`.

## 6. REFERENCES

[1] S. Moeller, U. Leser, W. Fleischmann, and R. Apweiler. EDITtoTrEMBL: a distributed approach to high-quality automated protein sequence annotation. Bioinformatics, 15(3):219-227, 1999.

[2] K. Bryson, Michael Luck, Mike Joy, and D. T. Jones. Applying agents to bioinformatics in geneweaver. In Cooperative Information Agents, pages 60-71, 2000.

[3] Robert D. Stevens, Alan J. Robinson, and Carole A. Goble. mygrid: personalised bioinformatics on the information grid. In Eleventh International Conference on Intelligent Systems for Molecular Biology, volume 19, 2003.

[10] `http://www.w3.org/2001/sw/hcls/`
[11] `http://esw.w3.org/topic/HCLS/Banff2007Demo`
[12] `http://sw.neurocommons.org/2007/kb.html`
[13] `http://sparql.neurocommons.org:8890/nsparql/`
[14] `http://sw.neurocommons.org/2007/kb-sources/`

[4] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. Nat Genet, 25:25-29, 2000.

[5] 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.

[6] Paschke, A., Rule-Based Service Level Agreements - Knowledge Representation for Automated e-Contract, SLA and Policy Management, ISBN 978-3-88793-221-3, IDEA Verlag GmbH, München.

[7] Paschke, A. and Bichler, M.: SLA Representation, Management and Enforcement. In: Proceedings IEEE International Conference on e-Technology, e-Commerce, e-Service EEE05, Hong Kong, 2005.

[8] Wooldridge, M.: An Introduction to MultiAgent Systems. John Wiley & Sons, 2001.

[9] Mule Enterprise Service Bus, `http://mule.codehaus.org/display/MULE/Home`, 2006.

[10] M. Welsh, D. Culler, and E. Brewer. SEDA: An Architecture for WellConditioned, Scalable Internet Services. In Proceedings of Eighteeth Symposium on Operating Systems (SOSP-18), Chateau Lake Louise, Canada, 2001.

[11] Rolf Backofen, Sebastian Will, and Erich Bornberg-Bauer. Application of Constraint Programming Techniques for Structure Prediction of Lattice Proteins with Extended Alphabets. Journal of Bioinformatics, 15(3):234-242, 1999.

[12] Nathalie Chabrier and Francois Fages. Symbolic model checking of biochemical networks. In Proceedings of the First International Workshop on Computational Methods in Systems Biology CMSB'03, LNCS, Riverto, Italy, March 2003. Springer-Verlag.

[13] P. Lambrix and V. Jakoniene. Towards transparent access to multiple biological databanks. In Proceedings of the First Asia-Pacific Bioinformatics Conference, pages 53-60, Adelaide, Australia, 2003.

[14] Rolf Gruetter and Claus Eikemeier. Development of a Simple Ontology Definition Language (SOntoDL) and its Application to a Medical Information Service on the World Wide Web. In Proceedings of the First Semantic Web Working Symposium (SWWS '01), pages 587-597, Stanford University, California, July/August 2001.

[15] Alan Ruttenberg, Tim Clark, William Bug, Matthias Samwald, Olivier Bodenreider, Helen Chen, Donald Doherty, Kerstin Forsberg, Yong Gao, Vipul Kashyap, June Kinoshita, Joanne Luciano, M Scott Marshall, Chimezie Ogbuji, Jonathan Rees, Susie Stephens, Gwendolyn T Wong, Elizabeth Wu, Davide Zaccagnini, Tonya Hongsermeier, Eric Neumann, Ivan Herman and Kei- Hoi Cheung: Advancing translational research with the Semantic Web, BMC Bioinformatics 2007, 8(Suppl 3):S2, 2007.

[16] Norbert E. Fuchs, Kaarel Kaljurand, and Gerold Schneider. Attempto Controlled English Meets the Challenges of Knowledge Representation, Reasoning, Interoperability and User Interfaces. In FLAIRS 2006, 2006.