Implement gateways between Rule Responder and Emerald

Authors:
Kalliopi Kravari
Nick Bassiliades
Release Date:
15-12-2010

Table of Contents

Introduction

This document describes the implementation of gateways between Rule Responder (RR) and EMERALD. Nevertheless, this report can guide the process of implementing new gateways between agent-based platforms as well. EMERALD is a JADE-based implementation framework for interoperable reasoning among agents in the Semantic Web, by using third-party trusted reasoning services. Rule Responder and EMERALD are to be compared regarding their agent-connection topologies, their interchange principles (performative wrapper and content language), and their used subsets of RuleML.

Conceptual Comparison between Rule Responder and EMERALD

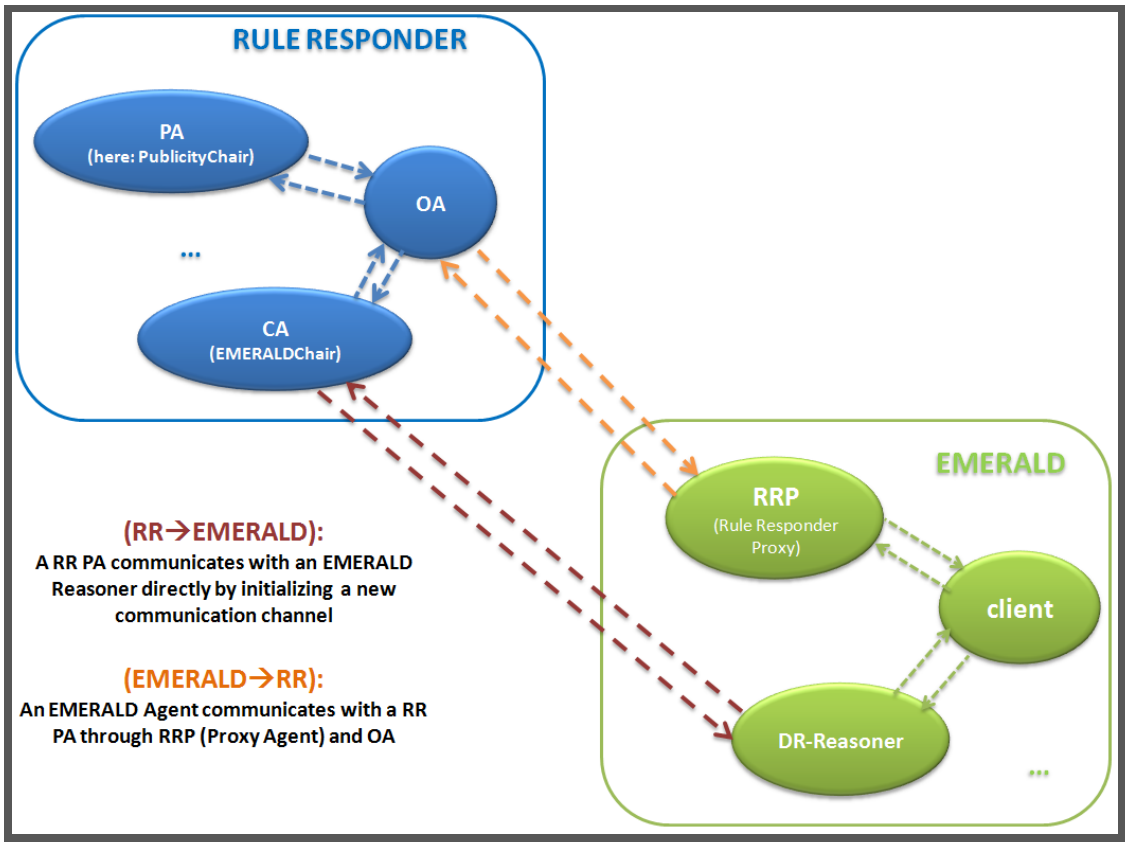
	Rule Responder	***EMERALD***
Agent technology	Java servlets / Mule	Java (JADE) agents
Interchange principles	Mule middleware	JADE (ACL)
RuleML	Reaction RuleML	(D)R-DEVICE RuleML
Agent knowledge	Internal rule base Internal & External data-knowledge base	External rule base External data-knowledge base
Reasoning	Multiple reasoning engines and instances of reasoning engines	Multiple reasoning engines (independent external services)

	Rule Responder	***EMERALD***
Directory service	NO	AYPS
Use of Prova	OAs always written in Prova, PAs and CAs optionally	A Prova Reasoner has been developed (one of the reasoning agents) prova 3 not supported because it does not support JADE yet
Use	Use cases can be obtained as instantiations of the Rule Responder framework	Use cases can be obtained by using different reasoners and different agent behavior KBs

EMERALD Rule Responder Architecture

Based on this comparison and analysis, (bidirectional) RuleML gateways between Rule Responder and EMERALD were designed and implemented. The resulting EMERALD RuleML Responder was tested both by extending the Rule Responder use case SymposiumPlanner (WP1 and WP2) with an EMERALD bridge to additional/external agents and, conversely, by extending an EMERALD use case with a Rule Responder bridge.

The Rule Responder EMERALD (RR→EMERALD) Gateway was implemented as a new CA that handles an appropriate communication channel. On the other hand, the EMERALD Rule Responder (EMERALD→RR) Gateway was implemented as a new proxy agent in EMERALD, communicating directly with RR OA.



EMERALD → Rule Responder Gateway

The implementation of this Gateway consists of the following subsequent steps:

1. *Development of the Rule Responder Proxy Agent (RRP)*: The RRP Agent is an EMERALD agent, acting as a gateway to Rule Responder. This RRP agent is flexible and reusable and, thus, not hardwired, meaning that it can receive any (RuleML) query, connect to Rule Responder, forward the query by invoking the proper Rule Responder agent and finally receive the result. Thus, RRP was developed as a Java (EMERALD) agent class that integrates API methods for interacting both with EMERALD agents and Rule Responder's PAs.

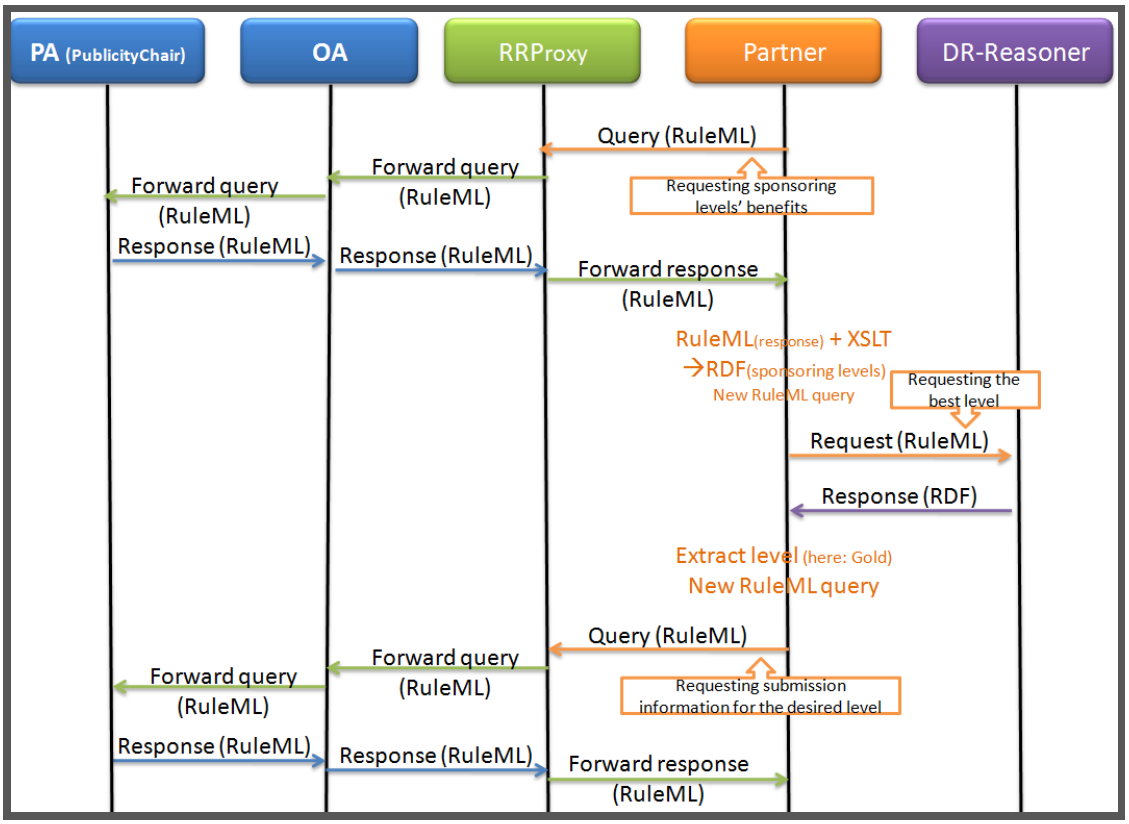
2. *Placement of the new agent in EMERALD directory*: the newly developed agent (its class file) has to be appropriately placed inside EMERALD's file directory. For convenience, it has to be placed in [*C:\jade\EMERALD*] folder. Mention that EMERALD itself has to be already downloaded from http://lpis.csd.auth.gr/systems/emerald/resource.html and installed.

3. *Registering the new agent in EMERALD*: A new agent has to be created either inside the GUI (Agent name: RRProxy; Agent class:RRPAgent) or in command line [*java jade.Boot RRProxy:RRPAgent*].

EMERALD → Rule Responder Use Case

In order to demonstrate the EMERALD-RR gateway, we have devised a scenario where an external-to-SymposiumPlanner partner (an EMERALD agent) would like to sponsor the RuleML-YYYY Symposium. The decision on the sponsoring level will be based on its personal preferences, related to the benefits of each level. The latter can be obtained from the corresponding RuleML Symposium-Planner chair. So, the EMERALD agent has to communicate with the PublicityChair in the Symposium Planner application. First of all, it sends its query (requesting the sponsoring levels and their benefits) to the RRP, the proxy agent, in order to forward it to the PA. RRP forwards the query, receives the response and returns it back to the partner. The decision making of the EMERALD agent is based on rules, and more specifically on defeasible logic rules. EMERALD hosts a defeasible logic reasoner, which is actually DR-Device, wrapped up as an agent. For details regarding the EMERALD architecture and philosophy, see [1].

The partner transforms the received RuleML message to RDF, in order to be used as a fact base for the rule base, which is formed in a defeasible RuleML dialect. The rule base contains its personal preferences and a link to the data that will be used (the RDF file) and it is sent to the defeasible logic reasoner (DR-Reasoner) in order to find out the best sponsoring level. Afterwards, the partner receives back DR-Reasoner's response (in this case the decision was the Gold sponsoring level) and sends a new query to the PublicityChair (through RRP) requesting the appropriate submission information for that level (e.g. to contact Sponsor chair by e-mail or phone).



Explanation for the documents and program files involved in this Use Case

RRP (Rule Responder Proxy)

The proxy (RRP) has just a Java file: ***RRPAgent.java*** (see above)

Partner (EMERALD agent)

In order for the partner to operate, certain files have to be present in the host PC: "WP3" folder in [C:\WP3] containing

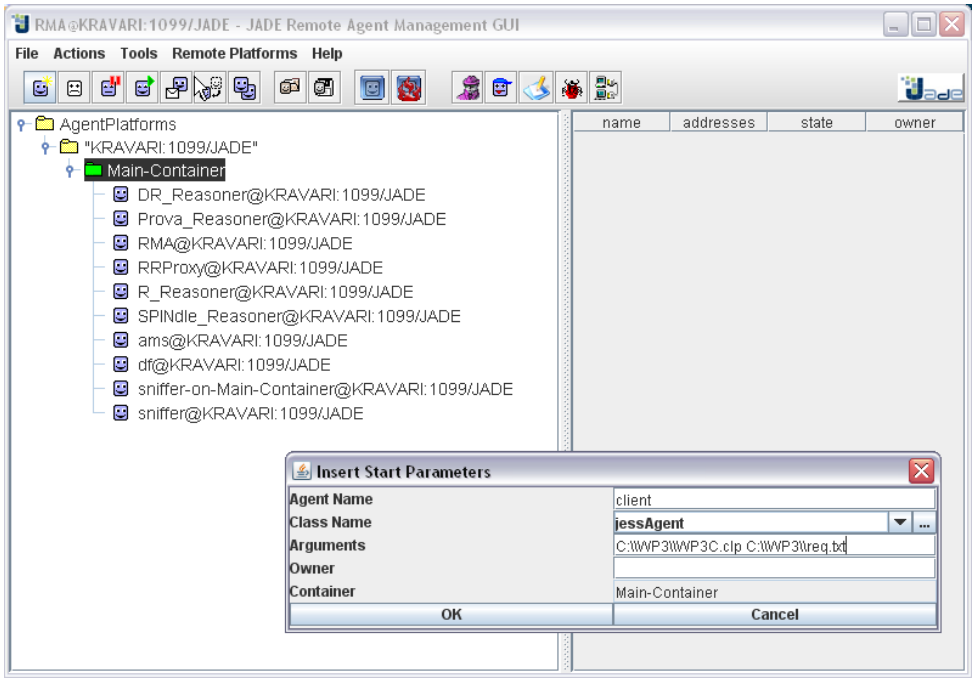
1. ***WP3C.clp***: This file describes the partner's personal strategy. It is written in Jess, part of this file can be found below (for details see [2])).

```
• • •
(deftemplate triple
        (slot subject)
        (slot predicate)
        (slot object))

(deffacts AN "agent's knowledge"
(query C:/WP3/query.ruleml)
(request C:/WP3/sponsors.ruleml)
(xslt C:/WP3/dr-device-input.xslt)
(path C:/WP3/sponsor-levels.rdf)
(query2path C:/WP3/query2.ruleml)
(detailspath C:/WP3/details.ruleml)
(attribute Level))

(defrule sendQuery "send a query to RRProxy"
        (MyAgent (name ?n))
        (query ?c)
        (Rule_Responder_Proxy (provider ?r))
        =>
        (send (assert (ACLMessage (communicative-act REQUEST) (sender ?n) (receiver ?r) (content ?c)))))

(defrule getAnswer "receive answer from RRProxy"
        ?z<-(ACLMessage (communicative-act INFORM) (sender ?r)(content ?answer))
        (Rule_Responder_Proxy (provider ?r))
        (xslt ?xslt)
        (path ?path)
        =>
        (bind ?bb (new Basic))
        (bind ?rr (?bb xslTransformation ?answer ?xslt ?path))
        (assert (file transformed))
        (retract ?z))

  • • •
```

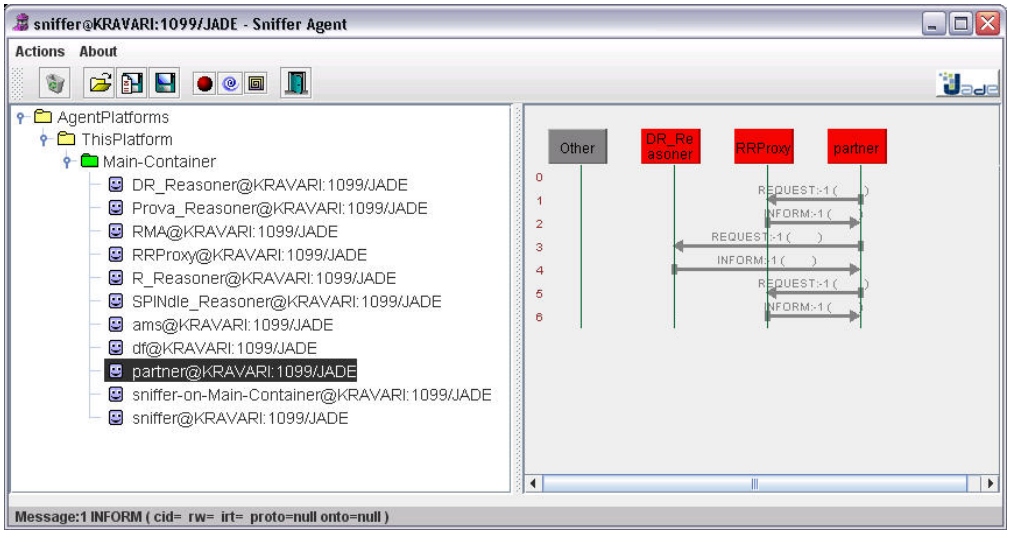
2. ***req.txt***: This file contains necessary information for the partner. It actually contains the types of the required services provided in EMERALD (also described in [2]).

```
Rule_Responder_Proxy
Defeasible_Reasoning_Service
```

Note that the above files have to be specified as arguments when initializing the agent in EMERALD: (Agent class: jessAgent; Agent Arguments: C:\\WP3\\WP3C.clp C:\\WP3\\req.txt).



Tip: In order to run the use case scenario in EMERALD either place the WP3e.bat file in C:\jade\EMERALD and execute it or use command line (at C:\jade\EMERALD) ***java jade.Boot -gui DR_Reasoner:ReasoningAgent RRProxy:RRPAgent sniffer:jade.tools.sniffer.Sniffer(DR_Reasoner;RRProxy)*** and then start a new GUI agent as shown above. The procedure steps, regarding the EMERALD side, are displayed (see below) by Sniffer, a special JADE agent that tracks message exchanges in the (EMERALD) environment.



3. ***dr-device-input.xslt***: This file contains the essential XSL transformation rules in order to transform the Reaction RuleML document that contains the answer of the PublicityChair agent to an RDF document (***sponsors-level.rdf***), which is required as input data for the DR-Reasoner

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:sp="file:///c:/WP3/sponsor-levels.rdfs#" xmlns: . . .
	<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
	<xsl:template match="/">
		<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:sp="file:///c:/WP3/sponsor-levels.rdfs#" xmlns: . . .
			<xsl:apply-templates select="//n:Atom"/>
		</rdf:RDF>
	</xsl:template>
	<xsl:template match="n:Atom">
		<sp:SponsorLevel rdf:about="{concat('file:///c:/WP3/sponsor-levels.rdfs#sp_',n:Ind[1])}">
			<sp:level>
				<xsl:value-of select="n:Ind[1]"/>
			</sp:level>
			<sp:amount rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
				<xsl:value-of select="n:Ind[2]"/>
			</sp:amount>
			<xsl:choose>
				<xsl:when test=".//n:Expr[n:Fun='logo' and n:Expr/n:Fun='on' and n:Expr/n:Ind='site']">
					<sp:logo-on-site rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">true</sp:logo-on-site>
				</xsl:when>
				<xsl:otherwise>
					<sp:logo-on-site rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">false</sp:logo-on-site>
				</xsl:otherwise>
			</xsl:choose>

			. . .

				<xsl:otherwise>
					<sp:free-registration rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">0</sp:free-registration>
				</xsl:otherwise>
			</xsl:choose>
		</sp:SponsorLevel>
	</xsl:template>
</xsl:stylesheet>
```

Notice that this file is domain-dependent, so in a different use case must be changed to reflect the class and properties of the input RDF facts for the defeasible logic rule base.

4. ***query.ruleml***: This file contains the query (in Reaction RuleML) for requesting (from the PublicityChair PA) the sponsoring levels and their benefits.

```
<RuleML
    xmlns="http://www.ruleml.org/0.91/xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.ruleml.org/0.91/xsd
    http://ibis.in.tum.de/research/
    ReactionRuleML/0.2/rr.xsd"
    xmlns:ruleml2007="http://ibis.in.tum.de/projects/paw#">

    <Message mode="outbound" directive="query-sync">
        <oid>
            <Ind>RuleML-2010</Ind>
        </oid>
        <protocol>
            <Ind>esb</Ind>
        </protocol>
        <sender>
            <Ind>User</Ind>
        </sender>
        <content>
            <Atom>
                <Rel>getBenefits</Rel>
                <Var>Level</Var>
                <Var>Amount</Var>
                <Var>Benefits</Var>
            </Atom>
        </content>
    </Message>

</RuleML>
```

5. ***sponsors2.ruleml***: This file contains the rule base (in the defeasible logic RuleML dialect of DR-Device) for deciding upon the best sponsoring level according to the potential sponsor's personal preferences. The file content is presented here in d-POSL (defeasible-POSL) syntax that is more concise:

```
r1: possibleOffer(level->?x) :=
        sponsorLevel(level->?x).
r2: ~possibleOffer(level->?x) :=
        sponsorLevel(level->?x, demo->false).
r3: ~possibleOffer(level->?x) :=
        sponsorLevel(level->?x, amount->?y),
        ?y > 5000.
r4: ~possibleOffer(level->?x) :=
        sponsorLevel(level->?x, free-registration->?y),
        ?y < 1.
r2 > r1.
r3 > r1.
r4 > r1.
r5: makeOffer(level->?x) :=
        possibleOffer(level->?x),
        sponsorLevel(level->?x, amount->?z),
        \+ ( possibleOffer(level->?y), ?y \= ?x,
           sponsorLevel(level->?y,amount->?w), ?w < ?z ).
```

The above rules indicate that the partner is looking for a sponsoring level that offers him/her the possibility to present a demo at the symposium, and at least one free

registration, without spending more than 5000$. If there are more than one such sponsoring levels, the lowest one will be preferred.

The corresponding RuleML file is available at: http://emerald.csd.auth.gr:8080/ruleSetsRuleML-2010/sponsors2.ruleml.

6. ***query2.ruleml***: This file contains the query (in Reaction RuleML) for requesting (from the PublicityChair PA) the appropriate sponsor offer submission procedure according to the desired sponsoring level.

```
<RuleML
    xmlns="http://www.ruleml.org/0.91/xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.ruleml.org/0.91/xsd
    http://ibis.in.tum.de/research/
    ReactionRuleML/0.2/rr.xsd"
    xmlns:ruleml2007="http://ibis.in.tum.de/projects/paw#">

    <Message mode="outbound" directive="query-sync">
        <oid>
            <Ind>RuleML-2010</Ind>
        </oid>
        <protocol>
            <Ind>esb</Ind>
        </protocol>
        <sender>
            <Ind>User</Ind>
        </sender>
        <content>
            <Atom>
                <Rel>askInfo</Rel>
                <Ind>Level</Ind>
                <Var>Action</Var>
                <Var>Info</Var>
            </Atom>
        </content>
    </Message>

</RuleML>
```

Prova Rule Base

OA's Prova rule base (available at: http://emerald.csd.auth.gr:8080/ruleSetsRuleML-2010/RuleML-2010-Responder.prova) has also been modified.

```
%Rule responsible for retrieving the sponsoring levels' benefits
getBenefits(XID, Level, Amount, Benefits):-
        %Retrieve the responsible PA
        assigned(XID, Agent, ruleml2010_PublicityChair, ruleml2010_responsible),
        %Send query to the PA
        sendMsg(XID, esb, Agent, "query", requestSponsoringLevel(Level, Amount, Benefits)),
        %Receive the answer
        rcvMult(XID, esb, Agent, "answer", substitutions(Level, Amount, Benefits)).

%Rule responsible for retrieving the appropriate contact information for a potential sponsor
askInfo(XID, Level, Action, Info) :-
        assigned(XID, Agent, ruleml2010_PublicityChair, ruleml2010_responsible),
        sendMsg(XID, esb, Agent, "query", sponsor_action(Level, Action, Info)),
        rcvMult(XID, esb, Agent, "answer", substitutions(Action, Info)).
```

The first new query returns the sponsoring levels and their benefits. The second new query just returns info for all the sponsoring levels from the PublicityChair PA.

POSL Rule Base

PublicityChair PA's knowledge base (Posl file) has also been modified. (available at: http://emerald.csd.auth.gr:8080/ruleSetsRuleML-2010/publicityChairRuleML-2010-Responder.posl)

```
% request sponsoring levels
requestSponsoringLevel(?Level, ?Amount, ?Benefits):-
        benefits(?Level, ?Benefits),
        sponsoringLevel(?Rank,?Level, us$[?Amount:integer]).

sponsor_action(?Level, ?Action, ?Info) :-
        actionPerformed(?Action,?Level,?),
        get_info(?Action, ?Info).

% request submission information
get_info(email, person[?Name,?Email]) :-
        person(
          symposiumChair[ruleML_2010,publicity],
          ?Name,?Title,?Email,?Phones).

% request submission information
get_info(phone, person[?Name,?Phones]) :-
        person(
          symposiumChair[ruleML_2010,publicity],
          ?Name,?Title,?Email,?Phones).
```

There are two new queries. The first query returns the sponsoring levels and their benefits and the second query just returns info for all the sponsoring levels.

Rule Responder → EMERALD Gateway

The implementation of this Gateway consists of the following subsequent steps:

1. *Development of a new Personal Agent (PA)*: PAs in RR are implemented as Java servlets, which, in essence, serve as wrappers for the corresponding reasoning engines. The PA, called *EMERALDChairRuleML2010* in the Symposium Planner application follows exactly this principle. The PA was developed as a Java servlet class that integrates API methods for interacting with EMERALD as well as core RR methods for exchanging messages with the Organizational Agent (OA).
2. *Placement of the new PA in Apache Tomcat*: The newly developed PA has to be appropriately placed inside Apache Tomcat's file directory. Apache Tomcat currently serves as RR's servlet container and Web server. A new folder has to be created, at the location: <Tomcat folder>\webapps\EMERALDChairRuleML2010, where a file structure has to be created that is similar to the tree of files in the other agents' directories. The compiled Java servlet class is placed appropriately inside this sub-tree.

For convenience, a .zip file containing the most up-to-date version of the agent's directory structure is available at http://lpis.csd.auth.gr/systems/emerald/resource.html.

3. *Assigning Responsibilities*: The tasks assigned to the EMERALDChairRuleML2010 PA are described in the role assignment matrix (OWL Lite ontology), available at: http://emerald.csd.auth.gr:8080/ruleSetsRuleML-2010/RuleML-2010.owl. This way, the OA is aware each time of the responsible agents, where the related messages are forwarded.

4. *Registering the new agent with Mule*: A new endpoint identifier has to be created inside Mule's "*mule-all-config.xml*" file, similarly to the endpoints for the rest of the PAs.

5. *Creating appropriate queries*: Finally, the website serving as the External Agent (EA) endpoint has to be "equipped" with appropriate queries for the EMERALDChairRuleML2010 PA.

6. *Development of the BlackBoard – JadeGateWay class*: BlackBoard is an object created by the EMERALDChairRuleML2010 PA and used as a communication channel. Actually, it will be the message channel MyGatewayAgent (EMERALD) and EMERALDChairRuleML2010 PA (Rule Responder).

7. *Development of the MyGatewayAgent*: MyGatewayAgent is also created by the EMERALDChairRuleML2010 PA and it behaves as a dispatcher for EMERALD. It gets the dashboard object described (and created) previously extracts who is the recipient and what's the message and forwards it accordingly. Moreover, it also packs the reply and sends it via BlackBoard to the EMERALDChairRuleML2010 PA

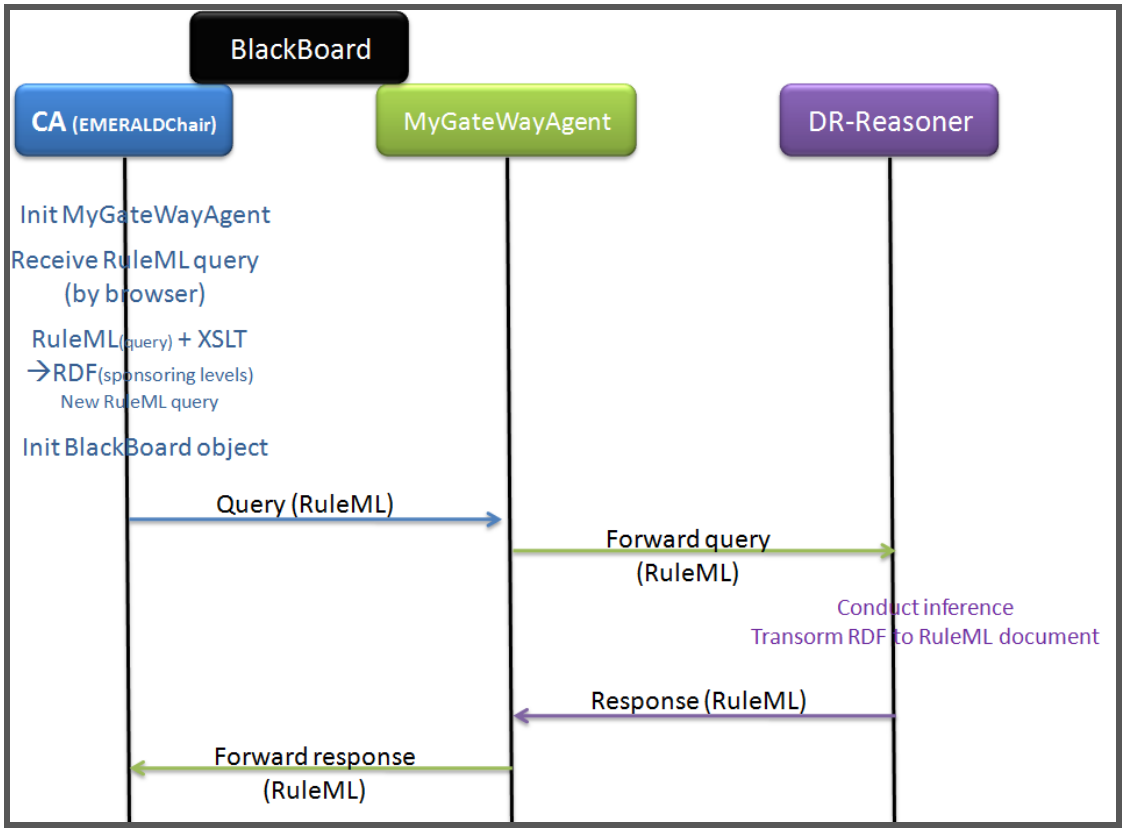
8. *Placement of the BlackBoard and the* MyGatewayAgent *in Apache Tomcat:* They have to be appropriately placed inside Apache Tomcat's file directory. A new folder has to be created, at the location: <Tomcat folder>\webapps\EMERALDChairRuleML2010, where a folder structure has to be created as follows: \classes\solarforce\agent\ for the MyGatewayAgent classes and \classes\solarforce\bean for the BlackBoard. The compiled Java classes are placed appropriately inside this sub-tree. For convenience, a .zip file containing the most up-to-date version of the agent's directory structure is available.

9. *Placement of the jade.jar file in Apache Tomcat:* In order for the gateway to operate efficiently, the jade.jar file has to be present in the Apache lib directory and in the Apache EMERALDChairRuleML2010 webapp's lib directory. This file is provided with JADE distribution, it can also be found at C:\jade\lib folder.

10. *Upgrade DR-Reasoner in EMERALD*: DR-Reasoner has to be modified in order to handle net queries and be able to provide not only RDF documents but also RuleML documents. The newly upgraded reasoner is available at http://lpis.csd.auth.gr/systems/emerald/ website.

Rule Responder → EMERALD Use Case

In order to demonstrate the RR – EMERALD gateway, a new Symposium Planner query is provided (Suggest Sponsoring Level [EMERALD/DR-DEVICE/Publicity Chair Agent]).This query implements a part of the publicity chair PA. The query gives as input the Sponsor name and the Amount of money that the users wants to sponsor and the PA returns as a result the suggested sponsoring level, namely the largest level that fits into the offered amount. It is important to mention that the answer is provided by EMERALD and its DR-Reasoner (DR-DEVICE) in particular.



Explanation for the documents and program files involved in this Use Case

DR-Reasoner

It is an EMERALD agent that provides the DR-Device functionality, for more details see [1 , 2]. Note that it's an update version, provided for RR. It is available at http://lpis.csd.auth.gr/systems/EMERALDRR/rrGateway.html

dr-device-input.xslt: This file contains the essential XSL transformation rules in order to transform the Reaction RuleML document that contains the initial query to an RDF document, which is required as input data for the DR-Reasoner

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
	xmlns:sp="http://155.207.113.24:8080/EMERALDfiles/rrdemo/sponsors-schema.rdf#"
	xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:n="http://www.ruleml.org/0.91/xsd">
	<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
	<xsl:template match="/">
		<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
			xmlns:sp="http://155.207.113.24:8080/EMERALDfiles/rrdemo/sponsors-schema.rdf#"
			xmlns:sp1="http://155.207.113.24:8080/EMERALDfiles/rrdemo/sponsors_input.rdf#"
			xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
			<sp:offered_sponsorship rdf:about="http://155.207.113.24:8080/EMERALDfiles/rrdemo/sponsors_input.rdf#sp_1">
				<sp:sponsor>
					<xsl:value-of select="//n:content//n:Ind[1]"/>
				</sp:sponsor>
				<sp:amount rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
					<xsl:value-of select="//n:content//n:Ind[2]"/>
				</sp:amount>
			</sp:offered_sponsorship>
		</rdf:RDF>
	</xsl:template>
</xsl:stylesheet>
```

Notice that this file is domain-dependent, so in a different use case must be changed to reflect the class and properties of the input RDF facts for the defeasible logic rule base.

dr-device-output.xslt: This file contains the essential XSL transformation rules in order to transform the RDF document that contains the answer to an Reaction Ruleml document, which will be sent back to Rule Responder.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
	<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
	<xsl:template match="/">
		<RuleML xmlns="http://www.ruleml.org/0.91/xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.ruleml.
    http://ibis.in.tum.de/research/
    ReactionRuleML/0.2/rr.xsd" xmlns:export="http://startrek.csd.auth.gr/dr-device/export/export.rdf#" xmlns:ruleml2007="http://ibis.in.tum.de/projects/paw
			<Message mode="outbound" directive="answer">
				<oid>
					<Ind>RuleResponder@kalliopi-6b34b73</Ind>
				</oid>
				<protocol>
					<Ind>esb</Ind>
				</protocol>
				<sender>
					<Ind>ruleml2010_EMERALDChair</Ind>
				</sender>
				<content>
					<Atom>
						<Rel>suggested_sponsoring_level</Rel>
						<Ind>
							<xsl:value-of select="//export:sponsor"/>
						</Ind>
						<Ind>
							<xsl:value-of select="//export:amount"/>
						</Ind>
						<Ind>
							<xsl:value-of select="//export:level"/>
						</Ind>
					</Atom>
				</content>
			</Message>
		</RuleML>
	</xsl:template>
</xsl:stylesheet>
```

Notice that this file is domain-dependent, so in a different use case must be changed to reflect the class and properties of the input RDF facts for the defeasible logic rule base.

query.ruleml: This file contains the query (in Reaction RuleML) for requesting (from the PublicityChair PA) the sponsoring level.

```
<RuleML
    xmlns="http://www.ruleml.org/0.91/xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.ruleml.org/0.91/xsd
    http://ibis.in.tum.de/research/
    ReactionRuleML/0.2/rr.xsd"
    xmlns:ruleml2007="http://ibis.in.tum.de/projects/paw#">

    <Message mode="outbound" directive="query-sync">
        <oid>
            <Ind>RuleML-2010</Ind>
        </oid>
        <protocol>
            <Ind>esb</Ind>
        </protocol>
        <sender>
            <Ind>User</Ind>
        </sender>
        <content>
            <Atom>
                <Rel>suggest_sponsoring_level</Rel>
                <Ind>sponsor_1</Ind>
                <Ind>1400</Ind>
                <Var>Level</Var>
            </Atom>
        </content>
    </Message>

</RuleML>
```

The answer to this query (in Reaction RuleML) is:

```
<?xml version="1.0" encoding="UTF-8"?>
<RuleML xmlns="http://www.ruleml.org/0.91/xsd" xmlns:xsi="http://www.w
http://ibis.in.tum.de/research/ReactionRuleML/0.2/rr.xsd">
                                                                  . . .

<Message mode="outbound" directive="answer">
  <oid>
     <Ind>RuleResponder@iitfrdextdev02.iit-iti.priv27</Ind>
  </oid>
  <protocol>
     <Ind>esb</Ind>
  </protocol>
  <sender>
     <Ind>RuleResponder</Ind>
  </sender>
  <content>
     <Atom>
        <Rel>noPublicInterface</Rel>
        <Expr>
           <Fun>interface</Fun>
           <Expr>
              <Fun>suggest_sponsoring_level</Fun>
              <Ind>sponsor_1</Ind>
              <Ind>1400</Ind>
           </Expr>
        </Expr>
     </Atom>
  </content>
</Message>



</RuleML>
```

Prova Rule Base

OA's Prova rule base (available at: http://emerald.csd.auth.gr:8080/ruleSetsRuleML-2010/RuleML-2010-Responder.prova) has also been modified.

```
processMessage(XID,From,Primitive,suggested_sponsoring_level(Sponsor,Amount,Level)) :-
        assigned(XID,Agent,ruleml2010_EMERALDSponsoring,ruleml2010_responsible),
        sendMsg(XID,esb,Agent, "query",
        suggested_sponsoring_level(Sponsor,Amount,Level)),
        println(["sent message to: ",Agent]),
        rcvMsg(XID,esb,Agent,"answer",
        suggested_sponsoring_level(Sponsor2,Amount2,Level2)),
        println(["message recevied: "]),
        sendMsg(XID,esb,From, "answer",
        suggested_sponsoring_level(Sponsor2,Amount2,Level2)),
        println(["message sent to browser: ",From]).
```

The new query returns, from the PublicityChair PA, the largest level that fits into the offered amount.

References

[1] K. Kravari, E. Kontopoulos, N. Bassiliades, "Trusted Reasoning Services for Semantic Web Agents", *Informatica: International Journal of Computing and Informatics*, Slovenian Society Informatika, 34(4), pp. 429-440, 2010. (available at: http://www.informatica.si/PDF/34-4/04_Kravari%20-%20Trusted%20Reasoning%20Services%20for%20Semantic%20Web.pdf)

[2] K. Kravari, E. Kontopoulos, N. Bassiliades, "EMERALD: A Multi-Agent System for Knowledge-based Reasoning Interoperability in the Semantic Web", 6th Hellenic Conference on Artificial Intelligence (SETN 2010), Springer Berlin / Heidelberg, LNCS, Vol. 6040/2010, pp. 173-182, Athens, Greece, 4-7 May, 2010. (available at: http://www.springerlink.com/content/1w3g635l4n1p152u/)