

WP2 Report – Integrating DR-DEVICE in Rule Responder

Table of Contents

WP2 Report – Integrating DR-DEVICE in Rule Responder	1
Table of Contents.....	1
Introduction	1
Integration Steps.....	1
DR-DEVICE Files.....	2
dr-device-0.91.xsd.....	2
sponsors-schema.rdf.....	2
sponsors.ruleml	3
dr-device-input.xslt	3
Inference Results	5
Prova Rule Base.....	5
Answers to Queries.....	7
Sequence of Actions.....	7

Introduction

This document describes the integration of the DR-DEVICE defeasible reasoner in the Rule Responder (RR) Symposium Planner application. Nevertheless, this report can guide the process of integrating DR-DEVICE in other RR applications as well.

Integration Steps

The integration of DR-DEVICE into RR Symposium Planner consists of the following subsequent steps:

1. *Development of the DR-DEVICE Personal Agent (PA):* PAs in RR are implemented as Java servlets, which, in essence, serve as wrappers for the corresponding reasoning engines. The DR-DEVICE PA, called *DRDChairRuleML2010* 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 DR-DEVICE 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\DRDChairRuleML2010, 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.

3. *Assigning Responsibilities:* The tasks assigned to the DRDChairRuleML2010 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 DRDChairRuleML2010 PA.

DR-DEVICE Files

In order for the DRDChairRuleML2010 PA to operate effectively, certain files have to be present in the host PC:

1. *DR-DEVICE Installation:* The DR-DEVICE system has to be downloaded from <http://lpis.csd.auth.gr/systems/dr-device/DR-Device%20setup.zip> and installed.
2. *"rrdemo" Folder:* Folder C:\rrdemo contains essential DR-DEVICE files for executing Symposium Planner queries. More specifically, the folder contains "dr-device-0.91.xsd", "sponsors-schema.rdf" and "sponsors.ruleml". More details regarding these files are included subsequently in this report.
3. *"drdfiles" Folder:* Folder <Tomcat folder>\webapps\drdfiles contains one core file (located inside "core" folder), which is "dr-device-input.xslt" (also described later). Additionally, all files produced by DR-DEVICE (queries, results) are placed here as well.

dr-device-0.91.xsd

Available at: <http://lpis.csd.auth.gr/systems/dr-device/dr-device-0.91.xsd>.

sponsors-schema.rdf

This file describes the RDF Schema of the query domain (sponsorships):

```
<?xml version='1.0' encoding='UTF-8'?>
  <!DOCTYPE rdf:RDF [
    <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
    <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
    <!ENTITY sp 'file:///c:/rrdemo/sponsors-schema.rdf#'>
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
  ]>
  <rdf:RDF xmlns:rdf="&rdf;"
    xmlns:sp="&sp;"
    xmlns:rdfs="&rdfs;">
    <rdfs:Class rdf:about="&sp;offered_sponsorship"
      rdfs:label="offered_sponsorship">
    </rdfs:Class>
    <rdf:Property rdf:about="&sp;sponsor" rdfs:label="sponsor">
      <rdfs:domain rdf:resource="&sp;offered_sponsorship"/>
    </rdf:Property>
  </rdf:RDF>
```

```

    <rdfs:range rdf:resource="&xsd:string"/>
  </rdf:Property>
  <rdf:Property rdf:about="&sp;amount" rdfs:label="amount">
    <rdfs:domain rdf:resource="&sp;offered_sponsorship"/>
    <rdfs:range rdf:resource="&xsd;integer"/>
  </rdf:Property>
</rdf:RDF>

```

The schema is available at: <http://emerald.csd.auth.gr:8080/ruleSetsRuleML-2010/sponsors-schema.rdf>.

sponsors.ruleml

This file contains (in RuleML format) the essential rule set for running the sponsorship use case. The file content is included here in d-POSL (defeasible-POSL) syntax that is more concise:

```

r1: suggested_sponsoring_level(sponsor->?X, level->preSponsor) :=
    offered_sponsorship(sponsor->?X, amount->?Y), ?Y >= 1.
r2: suggested_sponsoring_level(sponsor->?X, level->bronze) :=
    offered_sponsorship(sponsor->?X, amount->?Y), ?Y >= 500.
r3: suggested_sponsoring_level(sponsor->?X, level->silver) :=
    offered_sponsorship(sponsor->?X, amount->?Y), ?Y >= 1000.
r4: suggested_sponsoring_level(sponsor->?X, level->gold) :=
    offered_sponsorship(sponsor->?X, amount->?Y), ?Y >= 3000.
r5: suggested_sponsoring_level(sponsor->?X, level->platinum) :=
    offered_sponsorship(sponsor->?X, amount->?Y), ?Y >= 5000.
r6: suggested_sponsoring_level(sponsor->?X, level->emerald) :=
    offered_sponsorship(sponsor->?X, amount->?Y), ?Y >= 7500.
:= suggested_sponsoring_level(sponsor->?X, level->?Y),
    suggested_sponsoring_level(sponsor->?X, level->?Z), ?Y \= ?Z.
r2>r1. r3>r2. r3>r1. r4>r3. r4>r2. r4>r1. r5>r4. r5>r3. r5>r2.
r5>r1. r6>r5. r6>r4. r6>r3. r6>r2. r6>r1.

```

Rules r1-r6 refer to the six sponsorship levels (*preSponsor*, *bronze*, *silver*, *gold*, *platinum*, *emerald*), while the last clause is an integrity constraint, ensuring that only one *suggested_sponsoring_level* conclusion will be derived. The RuleML file is available at: <http://emerald.csd.auth.gr:8080/ruleSetsRuleML-2010/sponsors.ruleml>.

dr-device-input.xslt

Since DR-DEVICE's functionality is not oriented towards answering queries but towards running rule sets using RDF data as input data (facts). instead, the submitted queries have to be translated into RDF instances that serve as input facts to the system. The "*dr-device-input.xslt*" file performs just that via an XSL Transformation:

```

<?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:/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="file:///c:/rrdemo/sponsors-schema.rdf#"
mlns:sp1="file:///c:/rrdemo/sponsors-input.rdf#"
mlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
<sp:offered_sponsorship
  rdf:about="file:///c:/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>

```

The following figure (Fig. 1) displays a translation example via the XSL transformation described above.

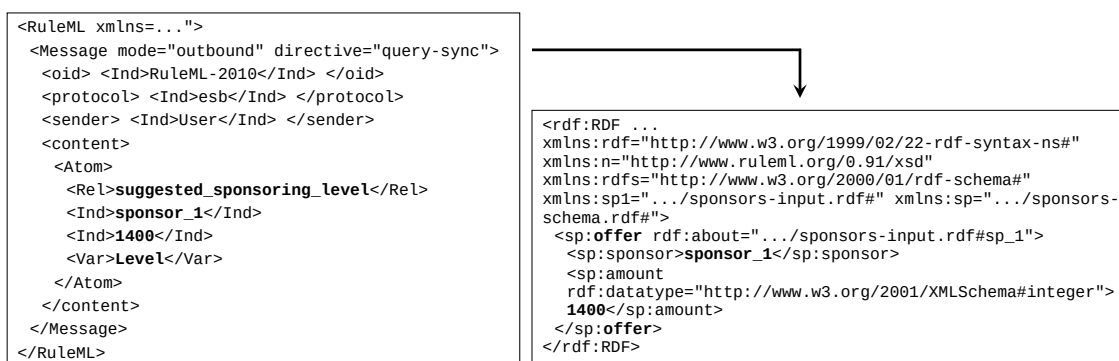


Fig. 1. Translation of the input query from a Reaction RuleML atom into an RDF instance.

All translated queries are placed inside <Tomcat folder>\webapps\drdfiles\queries.

Inference Results

Inference results are placed inside <Tomcat folder>\webapps\drdfiles\results. Fig. 2 displays a sample file containing DR-DEVICE inference results in RDF syntax.

```
<rdf:RDF xmlns:sponsors-output=...>

<rdfs:Class rdf:about='&sponsors-output;suggested_sponsoring_level'>
  <rdfs:label rdf:resource='suggested_sponsoring_level'/>
  <rdfs:subClassOf rdf:resource='&defeasible;DefeasibleObject'/>
</rdfs:Class>

<rdf:Property rdf:about='&sponsors-output;sponsor'>
  <rdfs:domain rdf:resource='&sponsors-output;suggested_sponsoring_level'/>
</rdf:Property>

<rdf:Property rdf:about='&sponsors-output;level'>
  <rdfs:domain rdf:resource='&sponsors-output;suggested_sponsoring_level'/>
</rdf:Property>

<sponsors-output:suggested_sponsoring_level
rdf:about='&sponsors-output;suggested_sponsoring_level1'>
  <sponsors-output:sponsor>sponsor_1</sponsors-output:sponsor>
  <sponsors-output:level>silver</sponsors-output:level>
  <defeasible:truthStatus>defeasibly-proven-positive</defeasible:truthStatus>
  <defeasible:proof rdf:datatype='&xsd:anyURI'>
    &proof-sponsors-output;proof1</defeasible:proof>
</sponsors-output:suggested_sponsoring_level>

</rdf:RDF>
```

Fig. 2. DR-DEVICE inference results in RDF syntax.

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. More specifically, Fig. 3 displays the Prova rule that is responsible for the communication between the agents OA and PA.

```
% Rule responsible for the communication between agents OA-PA
suggested_sponsoring_level(XID,Sponsor,Amount,Level) :-

  % Retrieve the responsible PA
  assigned(XID,Agent,ruleml2010_DRDSponsoring,ruleml2010_responsible),

  % Send query to the PA
  sendMsg(XID,esb,Agent, "query", suggested_sponsoring_level(Sponsor,Amount)),

  % Receive the answer
  rcvMsg(XID,esb,Agent,"answer", URL),

  % Parse the result (the result is in RDF syntax)
  retrieve_results(URL,sponsors_output,suggested_sponsoring_level,
    [pv(sponsor,Sponsor),pv(level,Level)]), !.
```

Fig. 3. Prova rule for handling queries delegated to the DR-DEVICE PA.

The query gives as input the Sponsor name and the Amount of money that the user wants to sponsor and the PA returns as a result the suggested sponsoring level, namely the highest level that fits into the offered amount. The process is as follows:

- 1) The OA sends to the DRDChairRuleML2010 PA the query with the input values, and waits for the result.
- 2) The result is not a binding of variables (as in the case of OO-JDrew) but a URL or a Path where the answer resides. The answer is an RDF file with the conclusions of the defeasible RuleML rule set.
- 3) The RDF file is given as input to the `retrieve_results/4` predicate, which is responsible for parsing the file and returning the appropriate results as bindings of the Level variable.

The description of the `retrieve_results/4` predicate, which is part of the OA's interface API with DRDChairRuleML2010 PA, is described in Fig. 4 below:

```
retrieve_results("url",Path,Namespace,Class,PropertyValueList) :-
    assertOWL(rdfs_simple, ["rdf_triple","subject","predicate","object"], Path),
    concat([Namespace,"_",Class],NSClass),
    rdf_triple(Resource,rdf_type,NSClass),
    rdf_triple(Resource,defeasible_truthStatus,defeasibly_proven_positive),
    retrieve_results_aux(Resource,Namespace,PropertyValueList).

% Auxiliary predicate for retrieve_results/5
% Iterates over all property-value pairs of a given Resource
retrieve_results_aux(_,_, []).
retrieve_results_aux(Resource,Namespace,[pv(Property,Value)|Rest]) :-
    concat([Namespace,"_",Property],NSProperty),
    rdf_triple(Resource,NSProperty,Value),
    retrieve_results_aux(Resource,Namespace,Rest).
```

Fig. 4. Prova rule for handling queries delegated to the DR-DEVICE PA.

The predicate parses an RDF file at location URL and queries the triplets found in it, according to the following: Retrieve all Resources of class Namespace:Class, which have been defeasibly proven positive and whose properties found in the PropertyValueList can match the corresponding values found in the same list. The elements of this list are in the form `pv(Property,Value)`, where Value can be either a constant or a variable. Being constant means that the corresponding Property of the Resource MUST have that value for that property. Being a variable means that the value of that property is retrieved by this predicate.

Answers to Queries

Fig. 5 displays the final answer to the initial query (see Fig. 1) in Reaction RuleML syntax. The answer is sent from the OA to the EA that initially submitted the query.

```
<RuleML xmlns=...>
  <Message mode="outbound" directive="answer">
    <oid> <Ind>RuleResponder@kalliopi-6b34b73</Ind> </oid>
    <protocol> <Ind>esb</Ind> </protocol>
    <sender> <Ind>RuleResponder</Ind> </sender>
    <content>
      <Atom>
        <Rel>suggested_sponsoring_level</Rel>
        <Ind>sponsor_1</Ind>
        <Ind>1400</Ind>
        <Ind>silver</Ind>
      </Atom>
    </content>
  </Message>
</RuleML>
```

Fig. 5. Answer to initial query in Reaction RuleML syntax.

Sequence of Actions

Fig. 6 displays the sequence of actions for deploying the DR-DEVICE PA in a Symposium Planner query execution scenario.

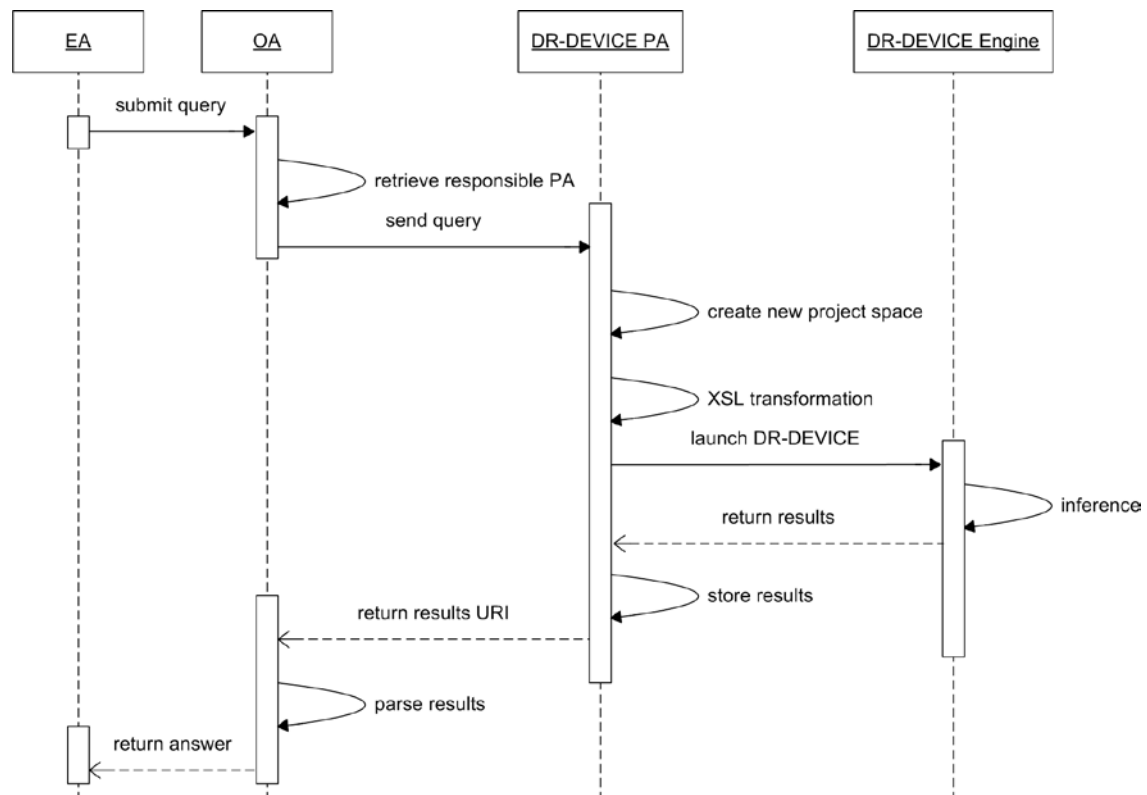


Fig. 6. Answer to initial query in Reaction RuleML syntax.

