A Rule-based Middleware for
Business Process Execution

2008-02-28

Biotec / Technical University Dresden
+49 351 463 40074
http://biotec.tu-dresden.de

A Rule-based Middleware for Business Process Execution

Agenda

- RuleResponder Approach
- Reaction RuleML
- Prova Semantic Web Rule Engine
- Use Cases
- Summary

A. Paschke[1], A. Kozlenkov[2]
[1] BioTec TU Dresden, Germany [2] Betfair Ltd.

Multi-Konferenz Wirtschaftsinformatik (MKWI 2008)

Chair of Bioinformatics
Dept. of Artificial Intelligence, TU Dresden

TECHNISCHE UNIVERSITÄT DRESDEN

Motivation

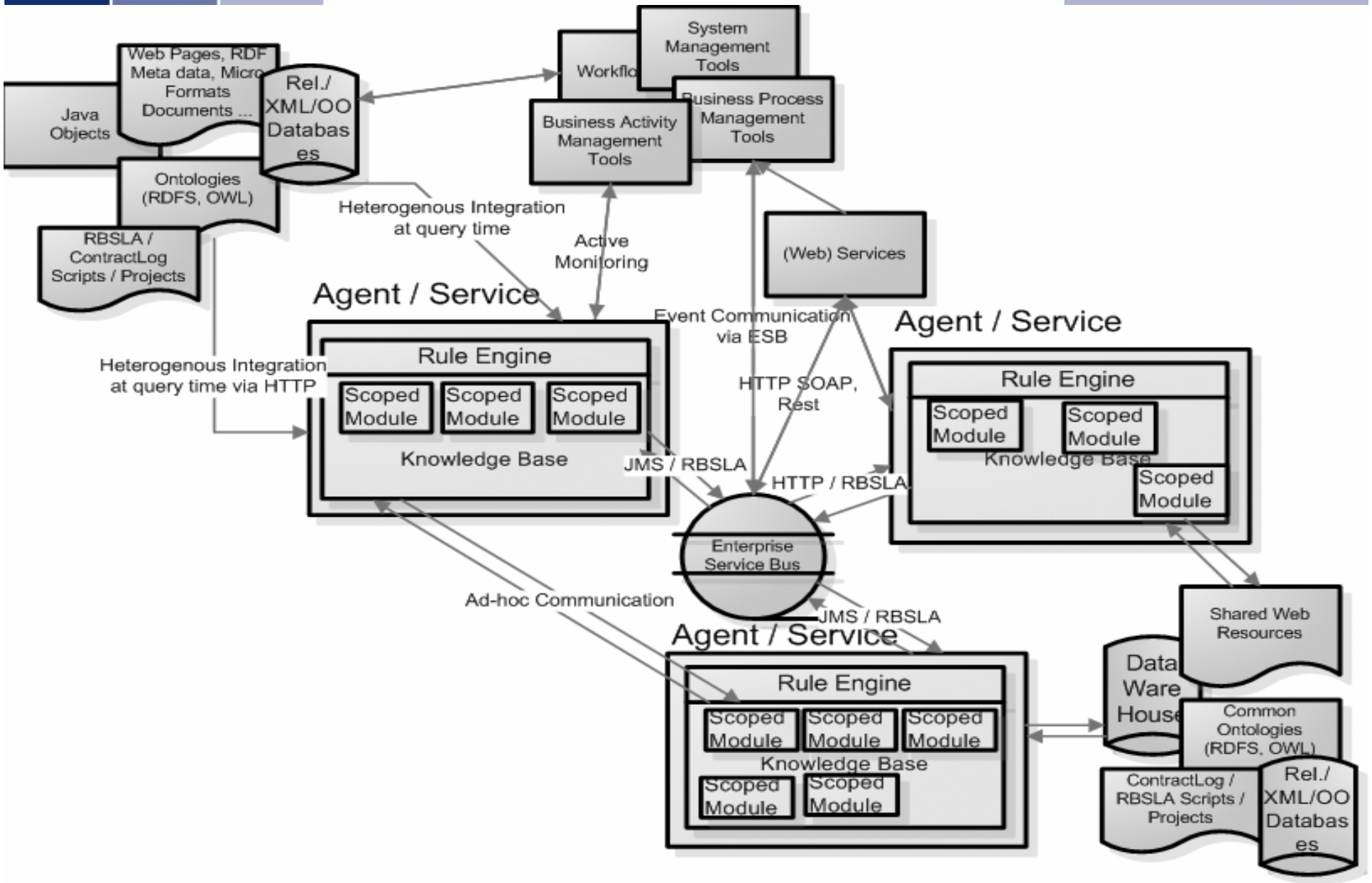
- Main deficits of activity-centered business process languages (e.g. BPEL, XPDL, BPML)
  - Limited support for descriptions of complex declarative rule logic, e.g. restrictions, decision logic, transformations
  - Activity-centered; not event-centered
    Intermediate Reactions, Compensations, Exceptions Human interactions, Peoplelinks, Partnerlinks

- Rule-based Approach

1. Compact declarative representation of rules
   - Clear semantics (e.g. logic programming)
   - Different rule types: derivation rules, reaction rules, normative rules
   - High expressiveness, e.g. for business rules, contract rules, workflow-style reaction rules
   - Dynamic extensibility and adaptation of rule bases
2. Efficient generic interpreters
   - Rule engines supporting rule chaining and execution of large rule sets
3. Automated conflict detection and resolution

- <u>But:</u>
  - Modularization of rules for business process specification and integration with local entities (services, human interaction points, partnerlinks)
  - Local rule execution state, specific and intern of the current process instance
  - Orchestration vs. Choreography

Rule-based Agent / Inference Service Approach



Rule Responder Architektur

1. **Computational independent model (CIM)** with rules, processes, conversational flows (e.g. in a natural or visual language)

2. **Platform independent model (PIM)** which represents the rules, events and ontologies in a common (standardized) interchange format (e.g. a markup language)

3. **Platform specific model (PSM)** which encodes the rule statements in the language of a specific execution environment (e.g. a rule engine / inference service or compiled code)

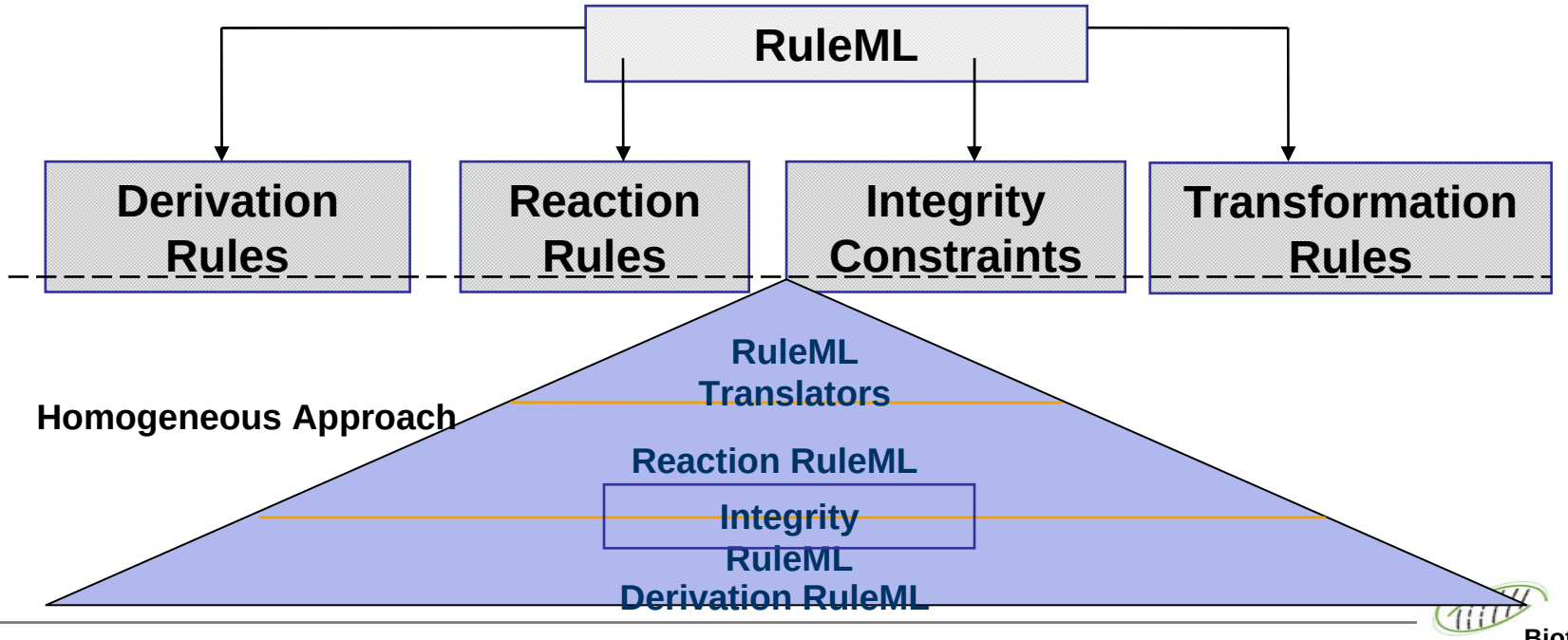
Reaction RuleML

Platform Independent Rule Interchange Format

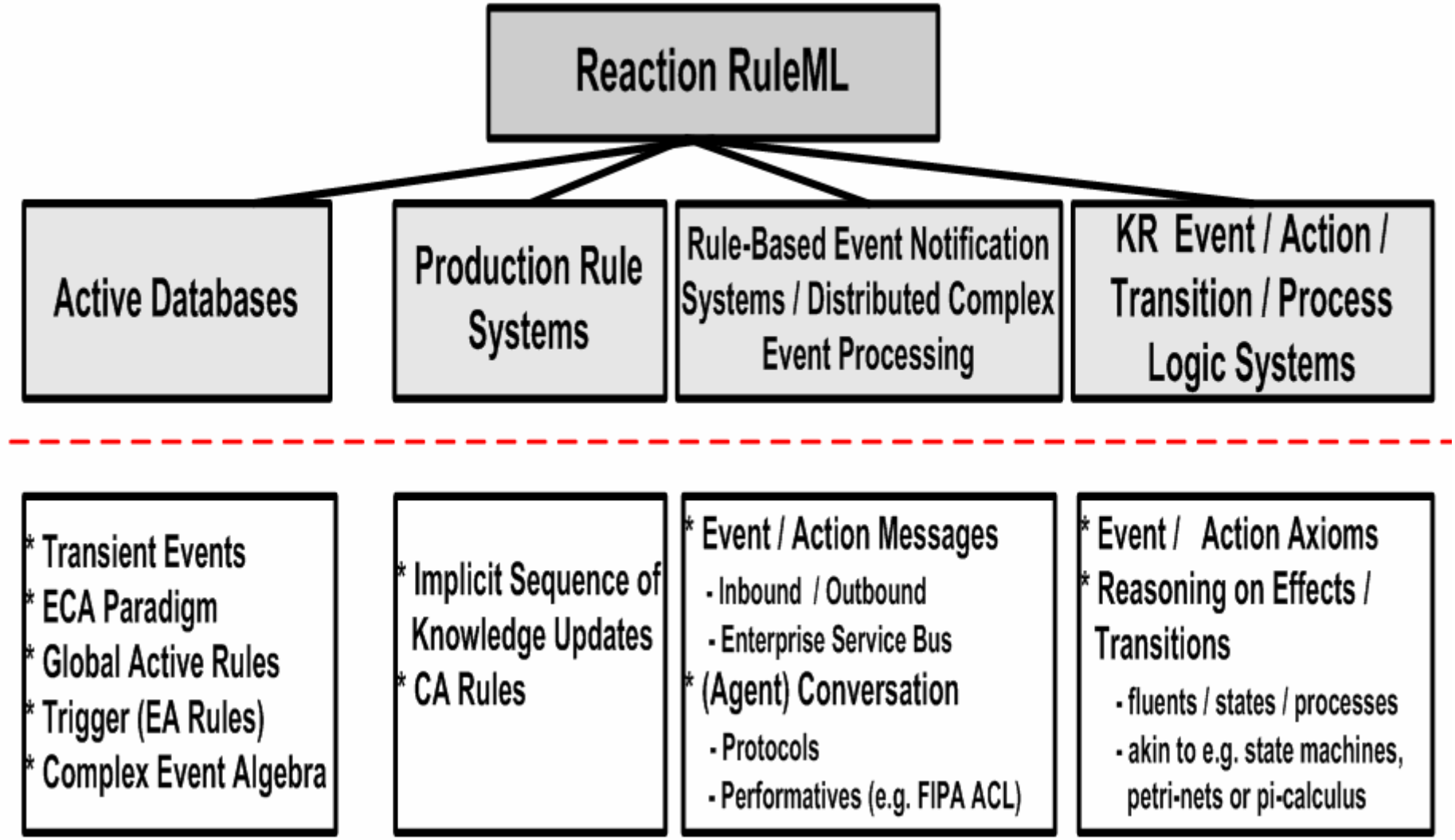
http://ibis.in.tum.de/research/ReactionRuleML/

RuleML as Common Rule Interchange Format

- Rule Markup and Modeling Initiative (RuleML) (www.ruleml.org)
  - Standardization effort for a rule markup and modelling language, tools and applications
- RuleML is the de facto open language standard for rule interchange/ rule markup on the Web
- Reaction RuleML (http://ibis.in.tum.de/research/ReactionRuleML/)
  - Language family for reaction rules and complex event messaging / processing



Scope of Reaction RuleML



Biotec

General Syntax for Reaction Rules (Reaction RuleML 0.2)

```
<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>
```

Impl.: Messages in Reaction RuleML (PIM Layer)

```
<Message mode="outbound" directive="ACL:inform">
        <oid> <!-- conversation ID--> </oid>
        <protocol> <!-- transport protocol --> </protocol>
        <sender> <!-- sender agent/service --> </sender>
        <content> <!-- message payload --> </content>
</Message>
```

- **@mode** = inbound|outbound – attribute defining the type of a message
- **@directive** – attribute defining the pragmatic context of the message, e.g. one or more FIPA ACL performatives, KQML, OWL-QL, Standard Deontic Logic norms, …
- **< oid >** – the conversation id used to distinguish multiple conversations and conversation states
- **< protocol >** – a transport protocol such as HTTP, JMS, SOAP, Jade, Enterprise Service Bus (ESB) ...
- **< sender >< receiver >** – the sender/receiver agent/service of the message
- **< content >** – message payload transporting a RuleML / Reaction RuleML query, answer or rule base

Example: Request / Query

```
...
<Message mode="outbound" directive="ACL:query-ref">
   <oid> <Ind>RuleML-2008</Ind> </oid>
   <protocol> <Ind>esb</Ind> </protocol>
   <sender> <Ind>User</Ind> </sender>
   <content>
      <Atom>
            <Rel>getContact</Rel>
            <Ind>Sponsoring</Ind>
            <Var>Contact</Var>
      </Atom>
   </content>
</Message>
...
```

FIPA ACL directive

- Event Message is local to the conversation state (oid) and pragmatic context (directive)

Prova

a

Distributed Semantic Web Rule Engine

Prova – Selected Expressive Features

- Backward-reasoning Derivation rules + ECA-style rules
- Messaging Reaction Rules and Complex Event Processing
- External Data and Object Integration + Query Built-Ins
  - Java Integration
  - XML Integration
  - SQL Integration
  - RDF Integration
- External Type Systems: Order-Sorted Polymorphic Typed Logic
  - Java Class Hierarchies
  - Semantic Web Ontologies
- Input/Output Mode Declarations
- Module Import and Integration: Order Modularized Logic Programs
- Meta Data Labels and Scopes (constructive views)
- Integrity Constraints and Test Cases for Verification and Validation
- Dynamic Transactional Updates

Impl. Messaging Reaction Rules in Prova (PSM)

- Send a message

sendMsg(XID,Protocol,Agent,Performative,[Predicate|Args]|Context)

- Receive a message

rcvMsg(XID,Protocol,Agent,Performative,[Predicate|Args]|Context)

- Receive multiple messages

rcvMult(XID,Protocol,Agent,Performative,[Predicate|Args]|Context)

- Description:
  - *XID is the conversation identifier*
  - *Protocol: transport protocol e.g. self*, *jade*, jms, esb
  - *Agent:* denotes the target or sender of the message
  - *Performative*: pragmatic context, e.g. FIPA ACL
  - *[Predicate|Args]* or Predicate(Arg_1,..,Arg_n): Message payload

Example: Request – Response" Flow with Delegation (PSM)

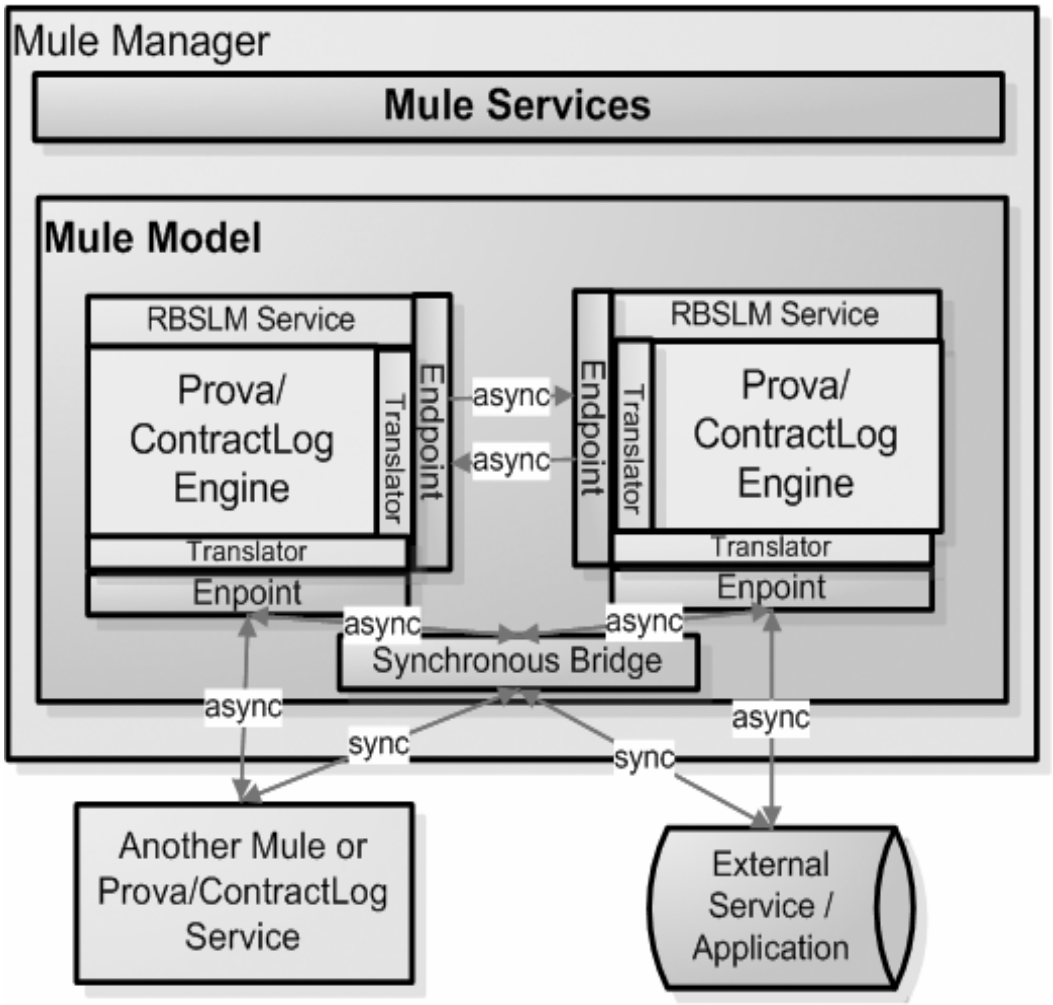
```
% receive query and delegate it to another agent
rcvMsg(CID,esb, Requester, acl_query-ref, Query) :-
  ... (other goals) ...
  sendMsg(Sub-CID,esb,Agent,acl_query-ref, Query),
  rcvMsg(Sub-CID,esb,Agent,acl_inform-ref, Answer),
  ... (other goals)...
  sendMsg(CID,esb,Requester,acl_inform-ref,Answer).
```

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

Note: Local conversations (and sub-conversations) with local state

Enterprise Service Bus

Mule Enterprise Service Bus

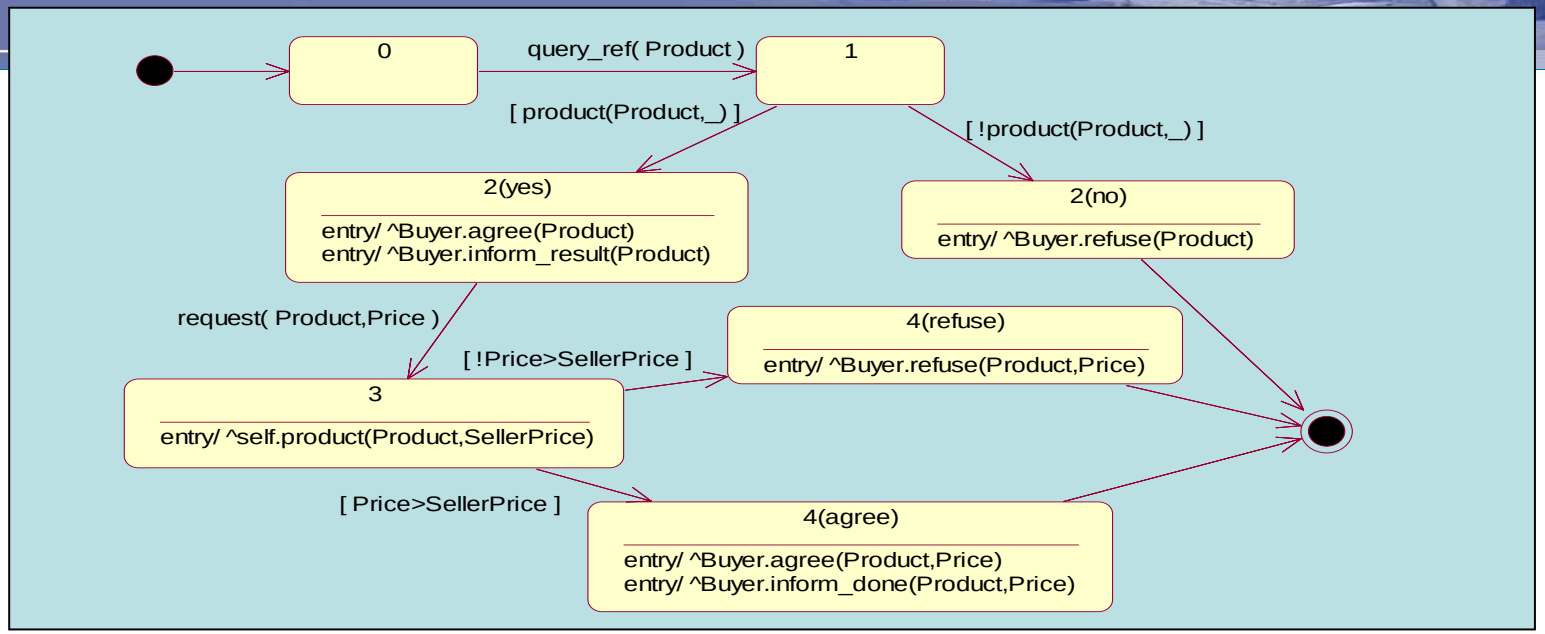


- Mule ESB Open Source
- Message Platform and distributed Object Broker
- Staged Event Driven Architecture (SEDA)
- > 30 Protocols (JMS, HTTP, SOAP …)
- Synchronous and Asynchronous Communication
- Complex Message-driven Event Processing (CEP)

Biotec

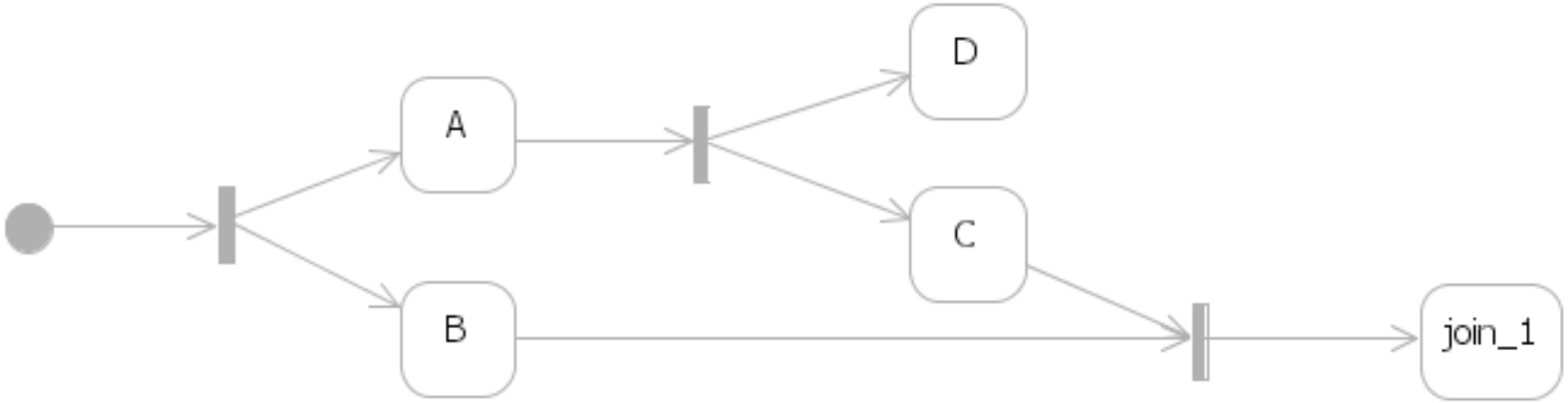
Examples

State machines based conversations




```
directbuy_seller_1(XID,Protocol,From,Product) :-
     product(Product|_),
     !,
     sendMsg(XID,Protocol,From,agree,Product,seller),
     sendMsg(XID,Protocol,From,inform_result,Product,seller),
     directbuy_seller_2(yes,XID,Protocol,From,Product).
directbuy_seller_1(XID,Protocol,From,Product) :-
     sendMsg(XID,Protocol,From,refuse,Product,seller),
     directbuy_seller_2(no,XID,Protocol,From,Product).
directbuy_seller_2(yes,XID,Protocol,From,Product) :-
     !,
     rcvMsg(XID,Protocol,From,request,[Product,Price],buyer),
     product(Product,SellerPrice),
     directbuy_seller_3(XID,Protocol,From,Product,Price,SellerPrice).

directbuy_seller_2(no,XID,Protocol,From,Product).
```




```
process_join() :-
    iam(Me),
    init_join(XID,join_1,[c(_),b(_)]),
    fork_a_b(Me,XID).
fork_a_b(Me,XID) :-
    rcvMsg(XID,self,Me,reply,a(1)),
    fork_c_d(Me,XID).
fork_a_b(Me,XID) :-
    rcvMsg(XID,self,Me,reply,b(1)),
    join(Me,XID,join_1,b(1)).
fork_c_d(Me,XID) :-
    rcvMsg(XID,self,Me,reply,c(1)),
    % Tell the join join_1 that a new pattern is ready
    join(Me,XID,join_1,c(1)).

% The following rule is invoked by join once all the inputs are assembled.
join_1(Me,XID,Inputs) :-
    println(["Joined for XID=",XID," with inputs: ",Inputs]).

% Prints
% Joined for XID=agent@hostname001 with inputs [[b,1],[c,1]]
```

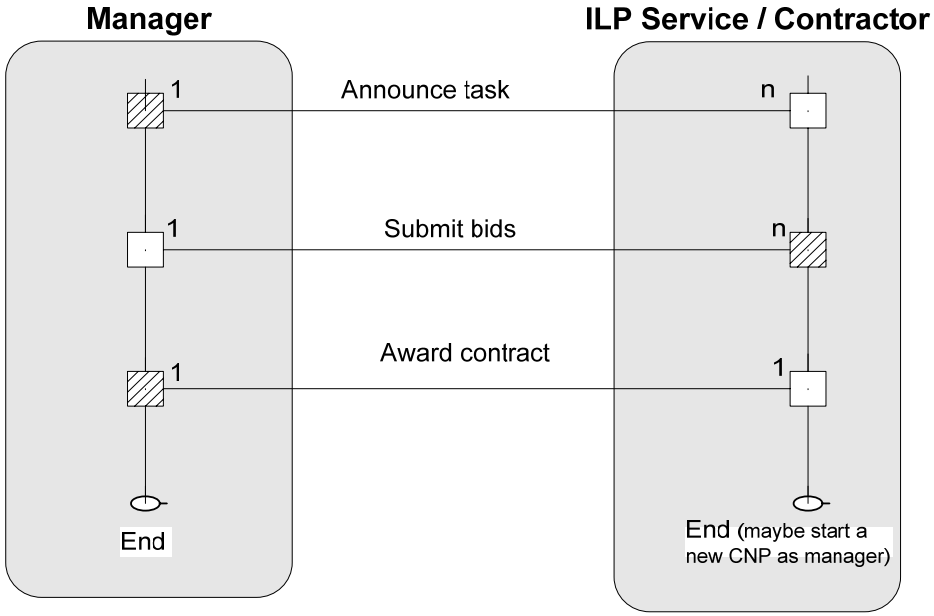
Mobile Code

```
% 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,eval,consult(SB)).

% Worker Service (Contractor)

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



Contract Net Protocol

Biotec

Conclusion

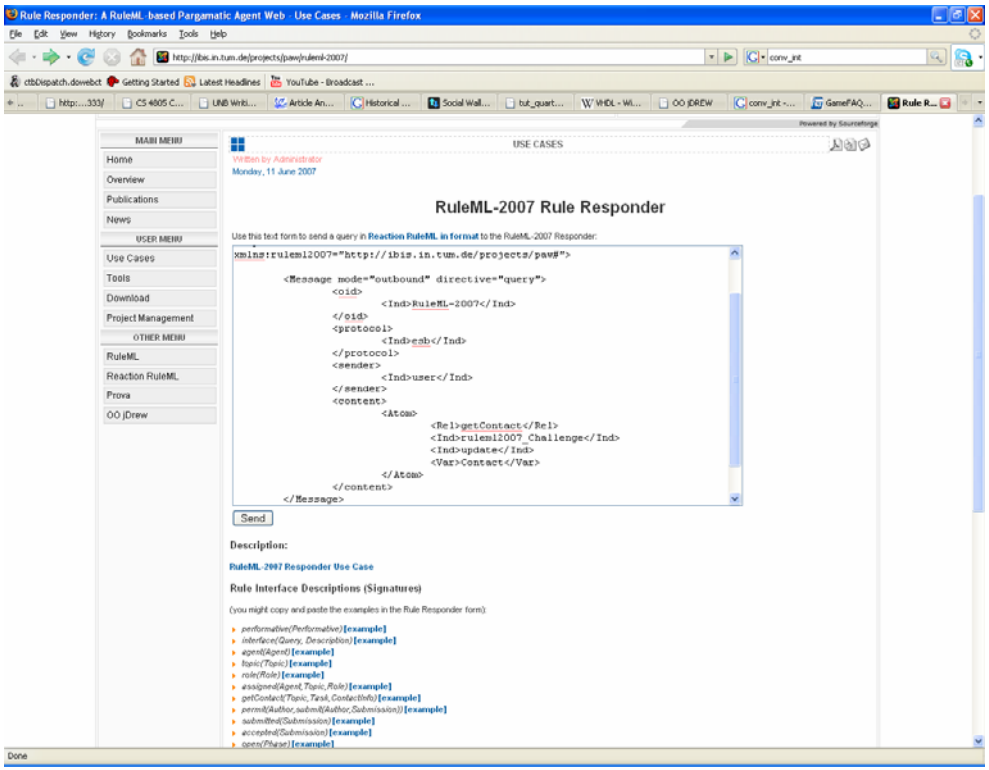
- Blends and tightly combines the ideas of multi-agent systems, distributed rule management systems, and service oriented and event driven architectures
- Applies distributed coordination mechanisms of rule-based complex event processing and rule-based workflow like reaction rule patterns for business process execution
- Develops an effective methodology and an efficient infrastructure to interchange and reuse knowledge on the Web and communicate contextual events and actions
- Allows orchestration or choreography models
- Demonstrates the interoperation of various distributed platform-specific rule execution environments based on Reaction RuleML as a platform-independent rule interchange format interchanged over an enterprise service bus as transport middleware
- Adds a Pragmatic Rule-based Layer (Pragmatic Web),
  - defines the rules for using information resources and ontologies to support human agents in their decisions and react partially self-autonomously by means of automated agents or services

Applications and Next Steps

- Rule Responder / Reaction RuleML
  - Rule Responder: http://responder.ruleml.org/
  - Reaction RuleML: http://ibis.in.tum.de/research/ReactionRuleML/
  - Prova Agent Architecture: http://www.prova.ws/
  - **Prova Workflow Patterns**: http://www.prova.ws/csp/?q=taxonomy/term/11
  - Rule Based Service Level Agreements: http://ibis.in.tum.de/projects/rbsla/
- Projects
  - W3C Rule Interchange Format
  - RuleML / Reaction RuleML Standardization Initiative
  - EU Integrated Project about Complex Event Processing
  - Rule Responder
    - **RBSLA** (Rule Based Service Level Agreements)
    - **DILP** (Distributed Inductive Logic Programming) for Mining Multi-Relation Data
    - **Pragmatic Agent Web** – Virtual Organization
    - **W3C HCLS RuleResponder** - eScience Service Infrastructure for HCLS
    - **Expert Finder** – Semantic Web

Rule Responder: A RuleML-based Pragmatic Agent Web

http://responder.ruleml.org



Sourceforge SVN:

https://mandarax.svn.sourceforge.net/svnroot/mandarax.

Sub-project "pragmatic-agent-web"

Biotec