Self-Healing BPEL Processes with Dynamo and the JBoss Rule Engine

L. Baresi, S. Guinea, L. Pasquale

[baresi|guinea]@elet.polimi.it
liliana.pasquale@gmail.com
Outline

- The challenge - *our goals*
- A case study - *a treasure hunt*
- The solution - *self-healing processes*
- WSCoL/WSReL
- The Dynamo Framework
  - Architecture
  - The Translation Process
  - Performance Evaluation
- Future Work
The challenge

- **Composition**: build intrinsically distributed and dynamic systems by leveraging remote services

  Dynamism/Flexibility/Distributed Ownership/Open World

  pre-deployment validation is not possible!

  unexpected and catastrophic events can arise!

- **Self-Healing Processes (monitoring/recovery)** [1]
  - detect faults/errors instantly
  - contain the effects
  - recover and proceed if possible

A case study - a treasure hunt

- **Goal**: find the treasure in the less time possible
- **How**: answer questions to obtain new maps
- **Loose** if you consume all your points
- **Points** can be used to buy suggestions

- Things that could go wrong:
  - **Advice Service**: does not answer within 5 seconds
  - **GPS Service**: the coordinates are in the wrong format
  - **Quiz Service**: the questions are in the wrong language
  - **Advice service**: the advice is too long
A case study - *a treasure hunt*

- **Goal**: find the treasure in the least time possible
- **How**: answer questions to obtain new maps
- **Loose** if you consume all your points
- Points can be used to buy suggestions

- **Things that could go wrong**:  
  - **Advice Service**: does not answer within 5 seconds  
  - **GPS Service**: the coordinates are in the wrong format  
  - **Quiz Service**: the questions are in the wrong language  
  - **Advice service**: the advice is too long
The solution

- Provide a holistic approach that does NOT only consider the process but also:
  - who is running it
  - when it is being run
  - the environment in which it is being run
The solution

- Provide a holistic approach that does NOT only consider the process but also:
  - who is running it
  - when it is being run
  - the environment in which it is being run

- Complete the process design with
  - a declarative indication of the functionalities and QoS that should be guaranteed at run time
    - pre-/post-conditions on the interactions the process has with the outside world or invariants - specified using WSCoL (Web Service Constraint Language)
  - an indication of the recovery strategies to be used to keep things on track
    - a set of strategies built from a set of atomic recovery actions we provide to the designer - specified using WSReL (Web Service Recovery Language)
The solution

- Provide a holistic approach that does NOT only consider the process but also:
  - who is running it
  - when it is being run
  - the environment in which it is being run

- Complete the process design with:
  - a declarative indication of the functionalities and QoS that should be guaranteed at run time
    - pre-/post-conditions on the interactions the process has with the outside world or invariants - specified using WSCoL (*Web Service Constraint Language*)
  - an indication of the recovery strategies to be used to keep things on track
    - a set of strategies built from a set of atomic recovery actions we provide to the designer - specified using WSReL (*Web Service Recovery Language*)

- Implement a framework (*Dynamo*) that augments a BPEL engine with self-healing capabilities and allows for separation of concerns
WSCoL/WSReL

WSCoL

- Mixes JML and XML technologies
- Characteristics:
  - internal/external/historical variables and aliasing
  - boolean/relational/mathematical operators
  - universal/existential quantifiers
  - aggregate functions
  - data type related functions
WSCoL/WSReL

WSCoL
- Mixes JML and XML technologies
- Characteristics:
  - internal/external/historical variables and aliasing
  - boolean/relational/mathematical operators
  - universal/existential quantifiers
  - aggregate functions
  - data type related functions

WSReL
- mix atomic actions into recovery steps
  - take a step and check if the problem is fixed
  - proceed to next step
- Some actions:
  - ignore/retry/rebind/substitute/call/callback/...
Self-Healing Capabilities are added to the ActiveBPEL Execution Engine using AspectJ

Main run-time components:

- **Monitor Manager**
  - collects the internal/external/historical variables
  - gets the properties to be checked and the strategies to be executed

- **JBoss Rule Engine**
  - checks the properties
  - activates recovery

- **Recovery Manager**
  - executes the atomic actions in a recovery step
Self-Healing Capabilities are added to the ActiveBPEL Execution Engine using AspectJ

Main run-time components:

- **Monitor Manager**
  - collects the internal/external/historical variables
  - gets the properties to be checked and the strategies to be executed

- **JBoss Rule Engine**
  - checks the properties
  - activates recovery

- **Recovery Manager**
  - executes the atomic actions in a recovery step
Self-Healing Capabilities are added to the ActiveBPEL Execution Engine using AspectJ

Main run-time components:

- **Monitor Manager**
  - collects the internal/external/historical variables
  - gets the properties to be checked and the strategies to be executed
- **JBoss Rule Engine**
  - checks the properties
  - activates recovery
- **Recovery Manager**
  - executes the atomic actions in a recovery step
Self-Healing Capabilities are added to the ActiveBPEL Execution Engine using AspectJ

Main run-time components:

- **Monitor Manager**
  - collects the internal/external/historical variables
  - gets the properties to be checked and the strategies to be executed

- **JBoss Rule Engine**
  - checks the properties
  - activates recovery

- **Recovery Manager**
  - executes the atomic actions in a recovery step
Self-Healing Capabilities are added to the ActiveBPEL Execution Engine using AspectJ

Main run-time components:

- **Monitor Manager**
  - collects the internal/external/historical variables
  - gets the properties to be checked and the strategies to be executed

- **JBoss Rule Engine**
  - checks the properties
  - activates recovery

- **Recovery Manager**
  - executes the atomic actions in a recovery step
Self-Healing Capabilities are added to the ActiveBPEL Execution Engine using AspectJ

Main run-time components:
- **Monitor Manager**
  - collects the internal/external/historical variables
  - gets the properties to be checked and the strategies to be executed
- **JBoss Rule Engine**
  - checks the properties
  - activates recovery
- **Recovery Manager**
  - executes the atomic actions in a recovery step

But how do the rules get into the Rule DB? and in what format are they?
The Translation Process

Rule:
- 1 set of Drools rules (.drl) - *will be later translated into a rule package by JBoss*
  - use of agenda-groups, activation-groups, and salience to ensure correct semantics
- Rule managers - *Java*
  - responsible for asserting data and/or managing aggregate functions

Two possibilities
- Simple Rule
  - 1 rule + 1 RuleManager
The Translation Process

Rule:
- 1 set of Drools rules (.drl) - will be later translated into a rule package by JBoss
  - use of agenda-groups, activation-groups, and salience to ensure correct semantics
- Rule managers - Java
  - responsible for asserting data and/or managing aggregate functions

Two possibilities
- Simple Rule
  - 1 rule + 1 RuleManager

```java
RULE $internalData/player/points >0;
rule "rule name"
agenda-group "manager_id"
when
  uuid : XMLWrapper(id == "uuid") &&
  eval (uuid.getDoubleValue() >0)
then
  end
```
The Translation Process

- Rule with aggregate functions
  - set of rules
  - set of managers
The Translation Process

- Rule with aggregate functions
  - set of rules
  - set of managers

```plaintext
forall($p2 in $internalData/player, $p2/points >= 0);

rule "0"
agenda-group "manager_id"
when
  forall_uuid : ForallManager (id="forall_uuid")
  eval(forall_uuid.getValue());
then
end

rule "1"
agenda-group "forall_uuid"
when
  forall_uuid : ForallManager (id="forall_uuid")
  uuid : XmlWrapper(id="uuid")
  eval(uuid.getDoubleValue() >= 0);
then
  forall_uuid.execute();
end

rule "2"
agenda-group "forall_uuid"
when
  forall_uuid : ForallManager (id="forall_uuid")
  uuid : XmlWrapper(id="uuid")
  eval(!uuid.getDoubleValue() >= 0);
then
end
```
Performance Evaluation

- Over 1000 measurements on 5 supervision rules
- Translation times
  - (WSCoL/WSReL → .drl file) + (.drl file → rule package)
  - %RE - amount due to JBoss
  - Entirely achieved off line - *una tantum*
- Execution times
  - Data collection + Data Analysis
  - %EV - amount spent collecting external/historical data
    - calling external services can lead to high variance!

<table>
<thead>
<tr>
<th></th>
<th>Average [s]</th>
<th>Median [s]</th>
<th>Variance</th>
<th>%RE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Store</strong></td>
<td>0.004</td>
<td>0.002</td>
<td>0</td>
<td>27.05%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>0.120</td>
<td>0.089</td>
<td>0.002</td>
<td>95.31%</td>
</tr>
<tr>
<td><strong>Length</strong></td>
<td>0.092</td>
<td>0.075</td>
<td>0.002</td>
<td>89.98%</td>
</tr>
<tr>
<td><strong>Quiz</strong></td>
<td>0.091</td>
<td>0.072</td>
<td>0.002</td>
<td>92.71%</td>
</tr>
<tr>
<td><strong>Point</strong></td>
<td>0.209</td>
<td>0.219</td>
<td>0.006</td>
<td>93.35%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Average [s]</th>
<th>Median [s]</th>
<th>Variance</th>
<th>%EV</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Store</strong></td>
<td>0.100</td>
<td>0.079</td>
<td>0.004</td>
<td>85.01%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>0.245</td>
<td>0.237</td>
<td>0.005</td>
<td>29.64%</td>
</tr>
<tr>
<td><strong>Length</strong></td>
<td>0.049</td>
<td>0.039</td>
<td>0.001</td>
<td>-</td>
</tr>
<tr>
<td><strong>Quiz</strong></td>
<td>0.205</td>
<td>0.133</td>
<td>0.042</td>
<td>72.82%</td>
</tr>
<tr>
<td><strong>Point</strong></td>
<td>0.118</td>
<td>0.106</td>
<td>0.002</td>
<td>-</td>
</tr>
</tbody>
</table>
Future Work

➢ Our research
  – Take advantage of level of technical expertise to improve recovery approach
  – AOP gives us access to the run-time rep of the process
    • backward recovery vs. forward recovery
    • process re-organization
    • monitoring may also lead to changes in the process definition and to changes in other processes

➢ The community
  – A lot of research is not taking advantage of the intrinsic distributed nature of web services (think of BPEL) -> should we try to move to a more distributed scenario?
An example

- The game can be played by people speaking different languages
  - they will be moving throughout europe
  - the language should always be the correct one (contained in their preferences)

**post-condition:**

```plaintext
let $question = $internalData/quiz_service/question;
returnString(LanguageVerifierService_WSDL, "getLanguage", input + $question + input, output) ==
$internalData/player[codepoint-equal(id/text(),"playerID")]/favouriteLanguage;
```

**recovery strategy:**

```plaintext
retry(1) ||
rebind(QuizBackup_WSDL) && notify(messageRebind, email) ||
halt() && notify(email, messageHalt)
```