

June, 14th, 2007

 POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione

Policies and Aspects for Supervision of BPEL Processes

Luciano Baresi, San Guinea, **Pierluigi Plebani**

Dipartimento di Elettronica ed Informazione

Politecnico di Milano

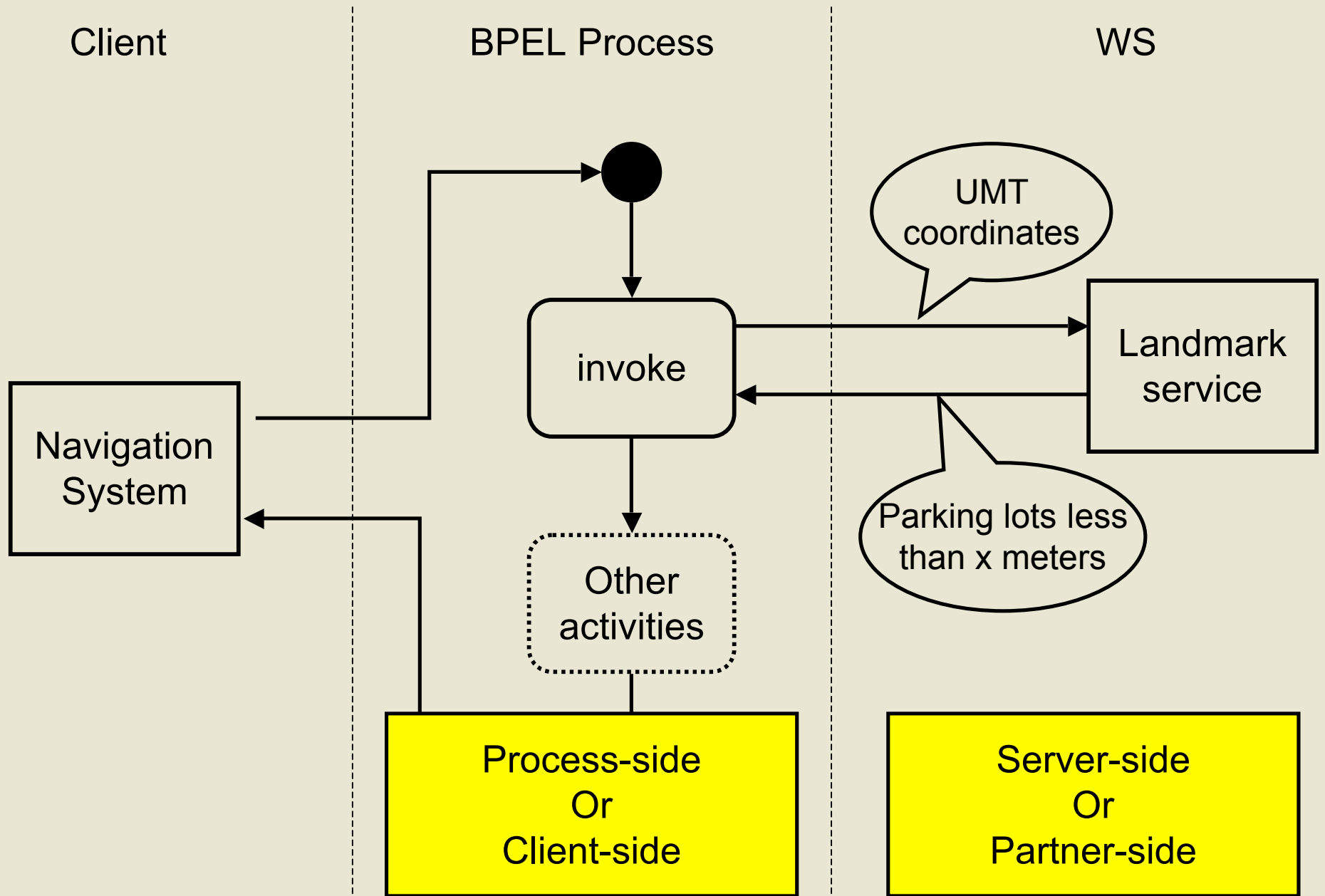
[baresi, guinea, plebani]@elet.polimi.it

- BPEL processes define a Web service based orchestration
 - A central node invokes external Web services
 - SOAP is usually adopted as communication protocol
- Most of the current solutions support BPEL at design time
 - Defining a process
 - Selecting the partners
- Few work take into account problems that may arise at run-time
 - Faults
 - Errors

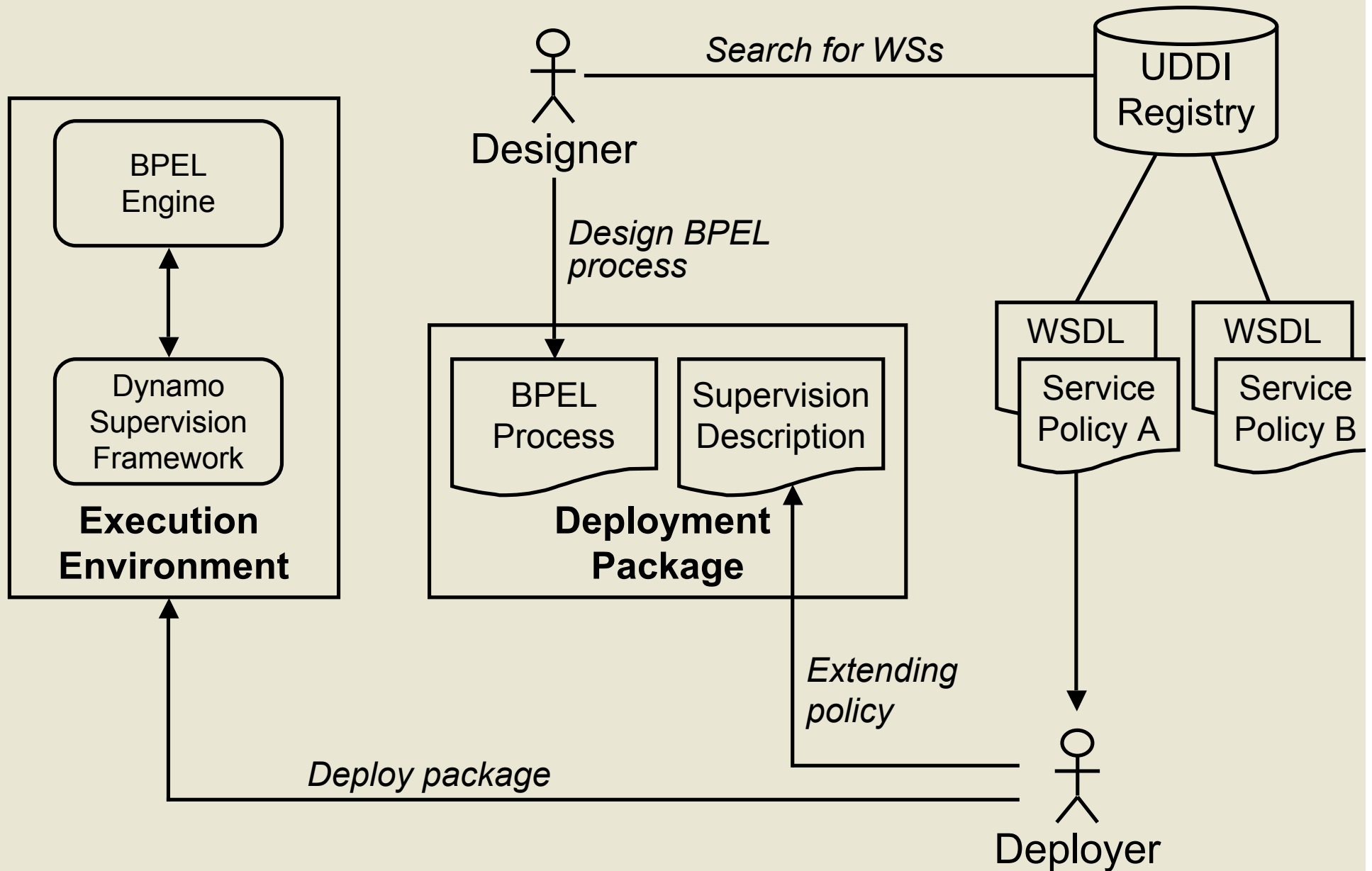
Main goals

- The correctness of the BPEL process depends on:
 - Internal process
 - External partners
- In this work we focus on the interactions with external partners
 - To supervise the process execution to discover possible anomalies
 - To perform suitable recovery strategies
- We propose:
 - A design model for BPEL processes that includes:
 - WSCoL: a language for expressing monitoring directives
 - WSReL: a language for expressing recovery strategies
 - Dynamo: an AOP-based framework for executing supervised BPEL processes

Running example



Design Process Model



- Non functional aspects of a Web Service are described by policies
- A policy is composed of a set of assertions that represent “preferences, requirements, capabilities...”
- Policy lifecycle: policies may be created by different roles during the lifecycle of a service
 - design time
 - deployment time
 - run time
- Our approach is mainly driven by policies
 - Defined at design time
 - Verified at run-time

- WS-Policy is an XML language able to define several configurations of assertions according to two main operators:
 - ExactlyOne
 - All
- Each assertion is specific to a particular quality domain:
 - Transaction
 - Security
 - Reliable messaging
- We introduce new kinds of assertions:
 - WSCoL (Web Service Constraint Language)
 - WSReL (Web Service Recovery Language)

WS-Policy example

All the assertions in the "All" scope must valid

```
<wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy/"  
  xmlns:wsat="..." xmlns:wsmr="..." xmlns:wsse="..."  
  Name="IntegratedBranch1TransactionPolicy"
```

<wsp:All>

<wsp:ExactlyOne>

Either integrity or confidentiality is allowed

```
<wsse:Integrity wsp:Usage="required">  
  <wsse:Algorithm Type="wsse:AlgSignature" URI="rsa-sha1"/>  
</wsse:Integrity>  
<wsse:Confidentiality wsp:Usage="required">  
  <wsse:Algorithm Type="wsse:AlgEncryption" URI="tripleDES-cbc"/>  
</wsse:Confidentiality>
```

The use of the TX is always allowed

<wsp:ExactlyOne>

```
<wsat:Required wsp:usage="required"/>  
<wsmr:IsReliable assurance="wsmr:AtMostOnce" inOrder="true"  
  wsp:usage="rejected"/>
```

The use of the RM is never allowed

</wsp:All>

</wsp:Policy>

- Ws-CoL is a domain independent language to express monitoring constraints
- It borrows many concepts from JML (Java Modeling Language), a behavioral interface specification language for Java:
 - Existential quantifiers, universal quantifiers
 - Logical operators
- It is based on
 - Data collection (input, output, external sources)
 - Data analysis (CLiX - Constraint Language in XML)
- Assertions in WSCoL can define:
 - Pre-conditions
 - Post-conditions

- WSSCoL is a Web service-side policy
- If a partner includes WSSCoL assertions in the Web service policy description defines:
 - Constraints for incoming messages (pre-conditions)
 - Promises for outgoing messages (post-conditions)
- When designers create the BPEL process, they must:
 - Satisfy the pre-condition before invoking
 - Rely on the post-conditions when getting the response

...

```
<wsp:All xmlns:wscol="...">
  <wscol:MonitoredItems xmlns:wscol="...">
    <wscol:MonitoredItem type="precondition"
      path="//definitions/message[@name='parkingLotRequest']">
      <wscol:Expression>
        let $zone = "//definitions/message[@name='parkingLotRequest']
          /part[@name='UTMZone']";
        let $northing = "//definitions/message[@name='parkingLotRequest']
          /part[@name='UTMNorthing']";
        let $easting = "//definitions/message[@name='parkingLotRequest']
          /part[@name='UTMEasting']";
        $zone >= 1 && $zone <= 60 &&
        $northing.ends-with("N") && $easting.ends-with("E");
      </wscol:Expression>
    </wscol:MonitoredItem>
  </wscol:MonitoredItems>
</wsp:All>
```

...

```
...
<wscol:MonitoredItem type="postcondition"
  path="//definitions/message[@name='parkingLotResponse']">
  <wscol:Expression>
    let $parkings = "//definitions/message[@name='parkingLotResponse']
      /part[@name='parking']";
    let $radius = "//definitions/message[@name='parkingLotRequest']
      /part[@name='radius']";
    (forall $parking in $parkings;
      ($parking/UTMEasting-$easting)^2 +
      ($parking/UTMNorthing-$northing)^2 <= $radius^2);
  </wscol:Expression>
</wscol:MonitoredItem>
</wscol:MonitoredItems>
</wsp:All>
...
```

- WSReL is an XML-based language compliant with WS-Policy Assertion
- An assertion expresses the recovery strategy to enact in case of condition violation
 - Condition can be expressed using WSCoL
 - Recovery strategy is defined in terms of Atomic Actions
- Process deployer
 - Reads the WSCoL assertions attached to the Web service partner
 - For each of them defines a recovery strategy

Recovery strategy

- A recovery strategy can be:
 - An atomic action
 - A multi-step process of atomic actions
- Atomic actions are:
 - Ignore
 - Notify
 - Halt
 - Retry
 - Rebind
 - ChangeSupervisionRule
 - ChangeParams
 - ChangeProcessParams
 - Call
 - ProcessCallback

```
<wssup:SupervisionRule>
  <wssup:postcondition>
    <wscol:Expression id="postcond_1">
      let $parkings =
        "//definitions/message[@name='parkingLotResponse']
        /part[@name='parking']";
      let $radius =
        "//definitions/message[@name='parkingLotRequest']
        /part[@name='radius']";
      (forall $parking in $parkings;
        ($parking/UTMEasting-$easting)^2 +
        ($parking/UTMNorthing-$northing)^2 <= $radius^2);
    </wscol:Expression>
  </wssup:postcondition>
  ...

```

```
...
<wssup:strategy>
  <wssup:strategycondition id="strategycond_1">
    <wscol:Expression>`Urgent request"</wscol:Expression>
  </wssup:strategycondition>
  <wssup:step number="1">
    <wssup:retry times="1"/>
  </wssup:step>
  <wssup:step number="2">
    <wssup:rebind url="http://..."/>
  </wssup:step>
  <wssup:step number="3">
    <wssup:notify>
      <wssup:message>...</wssup:message>
      <wssup:address>...</wssup:address>
    </wssup:notify>
    <wssup:halt/>
  </wssup:step>
</wssup:strategy>
...
```

...

```
<wssup:defaultstrategy>
```

```
  <wssup:step number="1">
```

```
    <wssup:notify>
```

```
      <wssup:message>...</wssup:message>
```

```
      <wssup:address>...</wssup:address>
```

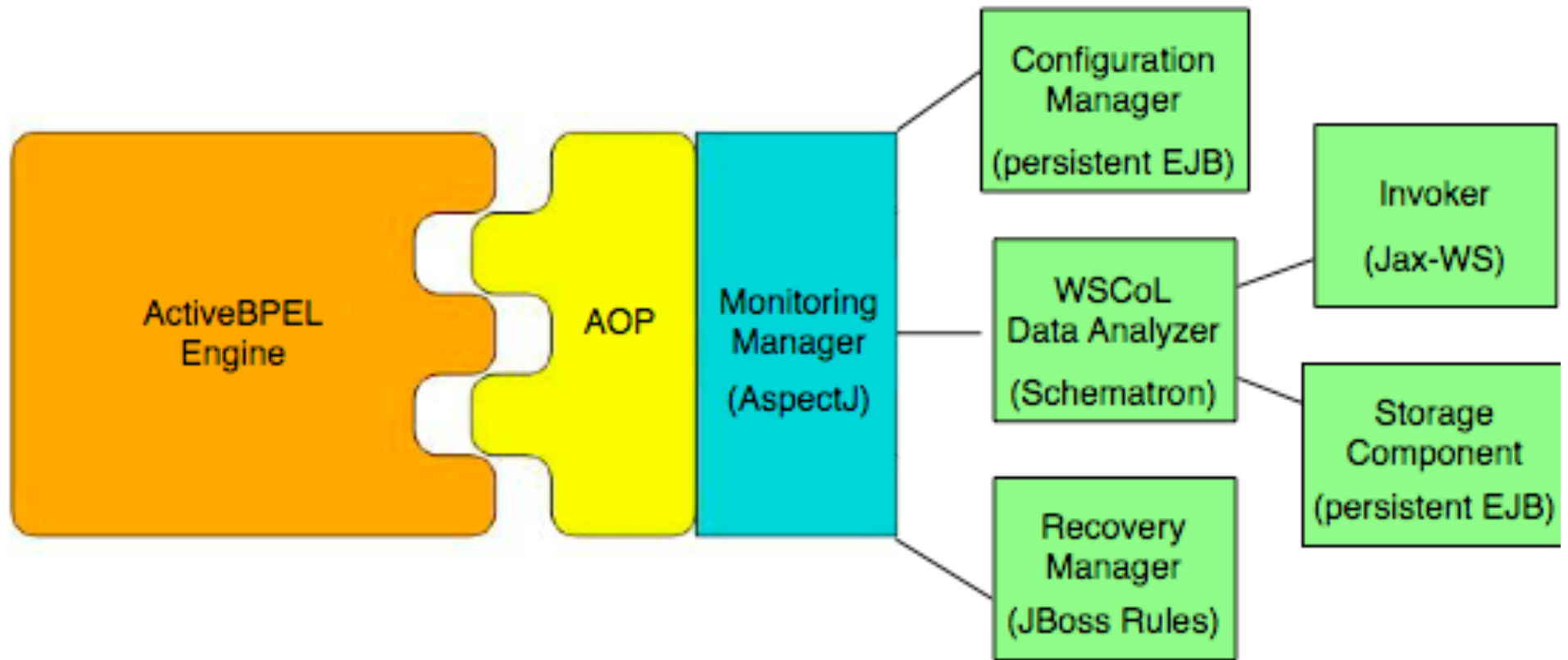
```
    </wssup:notify>
```

```
    <wssup:halt/>
```

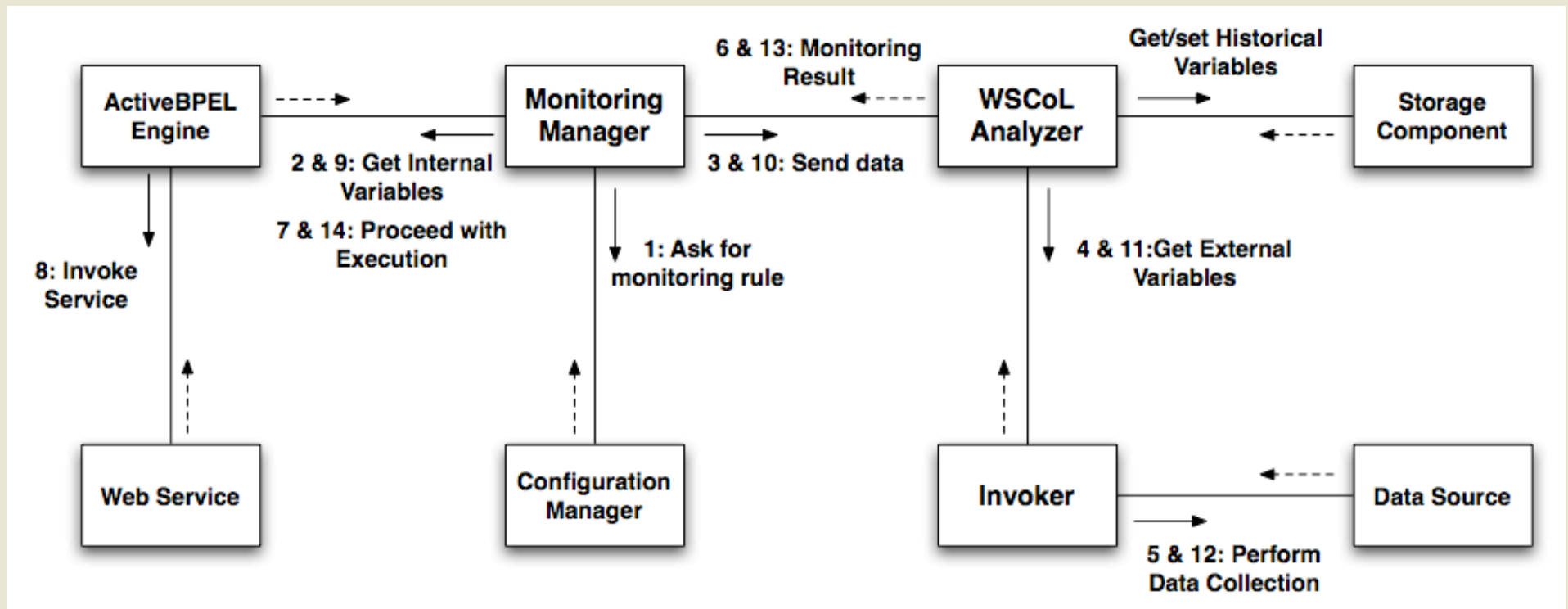
```
  </wssup:step>
```

```
</wssup:defaultstrategy>
```

```
</wssup:MonitoringRule>
```



Monitoring steps



- Some works exist about monitoring BPEL processes
 - Cremona based on WS-Agreement
 - We propose a more flexible approach
 - Spanoudakis et al. Based on event-calculus
 - We are more pervasive but with higher responsiveness
- Literature proposes some languages for expressing pre- and post-conditions
 - We need something close to Web service and WS-Policy in particular

- We have presented an approach to supervise BPEL processes by exploiting:
 - Policies
 - Aspects
- WSCoL and WS-ReL has been introduced as a WS-Policy Assertion compliant language for expressing constraints and recovery strategies
- Future work
 - Further case studies
 - Definition of domain-specific policies
 - Distribution of the supporting framework