Employing WS-BPEL Design Patterns for Grid Service Orchestration using a Standard WS-BPEL Engine and a Grid Middleware

André Brinkmann, Stefan Gudenkauf, Wilhelm Hasselbring, André Höing, Holger Karl, Odej Kao, Holger Nitsche, Guido Scherp

Cracow Grid Workshop 2008
BIS-Grid

- Case study to examine the feasibility of EAI with Grid technologies
- Focus on business workflows
- Exemplary evaluation within two application scenarios with SMEs
- D-Grid project funded by the BMBF
Workflow Engine for Grid Service Orchestration

- Realised as **UNICORE 6 services**
- Using arbitrary **WS-BPEL engine**
- Provides WS-BPEL workflows as **WSRF Grid Services**
- **Open Source**
Design decisions in BIS-Grid…
- Do not modify WS-BPEL
- Do not modify a WS-BPEL engine

… led to the identification of WS-BPEL design pattern
- Invoking a Grid Service is more complex than invoking a Web Service
  -> Grid utilisation pattern
- The engine architecture requires to exchange additional information between UNICORE 6 and the WS-BPEL engine
  -> Implementation specific pattern
Invocation of WSRF Grid Services has at least three phases

- Create, Use and Destroy
- Encapsulated by pattern **Grid-Service-Invoke**
- Includes one primitive pattern for each phase
- Specific for each Grid middleware
  - Tested with **UNICORE 6** and **GT4**[1]
- Applied within Workflow Design Tool
  - Extension of **Netbeans 6**

<sequence>
  <!-- Grid-Service-Instance-Create -->
  <invoke inputVariable="GridServiceFactoryRequest" operation="create"
      outputVariable="GridServiceFactoryResponse"
      partnerLink="GridServiceFactoryPL"
      portType="gsf:GridServiceFactoryPT"/>

  <!-- Grid-Service-Instance-Use -->
  <assign>
    <!-- Fill the input variable GridServiceRequest -->
    ...
    <copy>
      <from>
        <literal>
          <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing" >
            <wsa:Address/>
            <wsa:ReferenceProperties>
              <wsa:To/>
              <wsa:Action/>
            </wsa:ReferenceProperties>
          </wsa:EndpointReference>
        </literal>
      </from>
      <to variable="DynamicEndpointReference"/>
    </copy>
  </assign>
</sequence>
<copy>
  <from part="response" variable="GridServiceFactoryResponse">
    <query>wsa:EndpointReference/wsa:Address</query>
  </from>
  <to variable="DynamicGridServiceEndpoint">
    <query>wsa:Address</query>
  </to>
</copy>

<copy>
  <from part="response" variable="GridServiceFactoryResponse">
    <query>wsa:EndpointReference/wsa:Address</query>
  </from>
  <to variable="DynamicGridServiceEndpoint">
    <query>wsa:ReferenceProperties/wsa:To</query>
  </to>
</copy>

<copy>
  <from>
    <literal>
      <wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing">
        ... <!-- insert soapAction attribute for target method from WSDL-Interface -->
      </wsa:Action>
    </literal>
  </from>
  <to variable="DynamicGridServiceEndpoint">
    <query>wsa:ReferenceProperties/wsa:Action</query>
  </to>
</copy>
<copy>
  <from variable="DynamicGridServiceEndpoint"/>
  <to partnerLink="GridServicePL"/>
</copy>
</assign>

<invoke inputVariable="GridServiceRequest" operation="use"
    outputVariable="GridServiceResponse"
    partnerLink="GridServicePL"
    portType="gsr:GridServicePT"/>

<!-- Grid-Service-Instance-Destroy -->
<assign>
  <copy>
    <from>
      <literal>
        <wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing">
          http://docs.oasis-open.org/wsrf/rlw-2/ImmediateResourceTermination/DestroyRequest
        </wsa:Action>
      </literal>
    </from>
    <to variable="DynamicGridServiceEndpoint">
      <query>wsa:ReferenceProperties/wsa:Action</query>
    </to>
  </copy>
  <copy>
    <from variable="DynamicGridServiceEndpoint"/>
    <to partnerLink="GridServicePL"/>
  </copy>
</assign>
<copy>
  <from>
    <literal/>
  </from>
  <to part="Destroy" variable="GridServiceDestroyRequest"/>
</copy>

</assign>

<invoke inputVariable="GridServiceDestroyRequest" operation="Destroy"
  outputVariable="GridServiceDestroyResponse"
  partnerLink="GridServicePL"
  portType="gsr:GridServicePT"/>

</sequence>
ID mapping problem: Two instances per workflow execution

- UNICORE 6 (resource ID) and WS-BPEL engine (process ID)
- Mapping needed to identify outgoing process messages
- Encapsulated by pattern **On-Receive-ID-Retrieve(1)** and **Pre-Invoke-ID-Absign(2)**
  - Ingoing message body is extended by resource ID\(^{(1)}\)
  - Store resource ID after instance-creating receive\(^{(1)}\)
  - Create EPR with resource ID as resource property\(^{(2)}\)
  - Outgoing invoke message header contains resource ID\(^{(2)}\)
- Pattern applied within our **UNICORE 6 services**
  - Deployment: Extension of WS-BPEL and WSDL definitions
  - Execution: Addition and Mapping of resource IDs
Monitoring problem: WS-BPEL engine-specific process-ID is needed

- Assumption: WS-BPEL engine offers a monitoring interface

- Solution 1: **Extension of** pattern **Pre-Invoke-ID-Assign**
  - Engine-specific process ID is added to EPR's resource properties
    - e.g. ActiveBPEL offers WS-BPEL extension operation `getProcessID`
  - Invoke the monitoring interface with the process ID
  - Each WS-BPEL engine adapter must apply own WS-BPEL design pattern

- Solution 2: **Reuse of** pattern **Pre-Invoke-ID-Assign**
  - Invoke the monitoring interface to get all active processes
  - Search for resource ID value in process variables
  - Effective filter, caching and polling mechanism are needed

- Applied within our **UNICORE 6 services** and within the **WS-BPEL engine adapter**
Outlook

- **Human interactions**
  - Create own solution or use existing solutions as BPEL4People, WS-HT, ...

- **Support flow-based WS-BPEL processes with links**
  - Currently only sequence-based processes are supported with link-less flows

- **Extension of Netbeans 6**
  - Grid-Service-Invoke pattern
  - Deployment

- **In principle possible**
  - BIS-Grid engine architecture including WS-BPEL design patterns can also be implemented based on Grid middlewares as GT4
To get more information

http://www.bisgrid.de -> „Dokumente“

- Deliverable 2.1 „Catalogue of WS-BPEL Design Patterns“ (DRAFT)
- Deliverable 3.1 „WS-BPEL Engine Specification“
- Deliverable 3.2 „WS-BPEL Engine Documentation“ (DRAFT)
- BIS-Grid engine prototype will be available soon