Using UNICORE and WS-BPEL for Scientific Workflow Execution in Grid Environments

Guido Scherp (OFFIS), André Höing (OFFIS), Stefan Gudenkauf, Wilhelm Hasselbring (OFFIS), Odej Kao (TU Berlin)

GEFÖRDERT VOM

Bundesministerium für Bildung und Forschung

UNICORE Summit 2009

August 25, 2009

BIS-Grid

- „Business Information Systems: Grid-based Integration and Orchestration“
- Project Funded by german Federal Ministry of Education and Research in the context of the German Grid initiative (D-Grid)

- Prove the feasibility of Enterprise Application Integration using Grid technologies
- Provide Grid utilization and (D-)Grid participation for SMEs
- Provide a generic Grid-compatible orchestration solution

- Focus: originally business workflows, but also regards scientific workflows
- Application scenarios: call center, machine construction, flood simulation (new, provided by GDI-Grid)

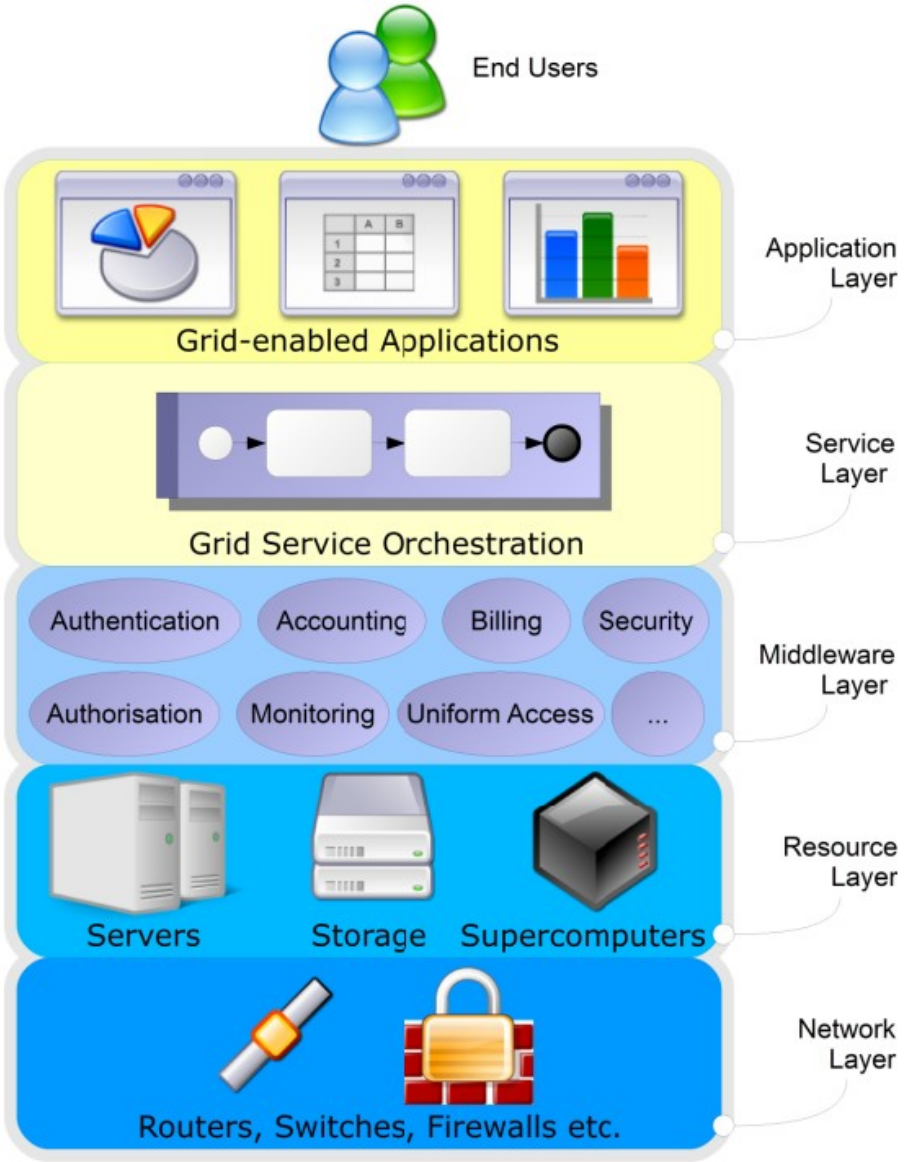
- Trustworthy collaboration and business models for Grid Providing
- Orchestration-as-a-Service (OaaS)
- Utilization of industry standards (e.g., BPMN, WS-BPEL)
- Integration with widespread technologies (e.g., ActiveBPEL)

Overview

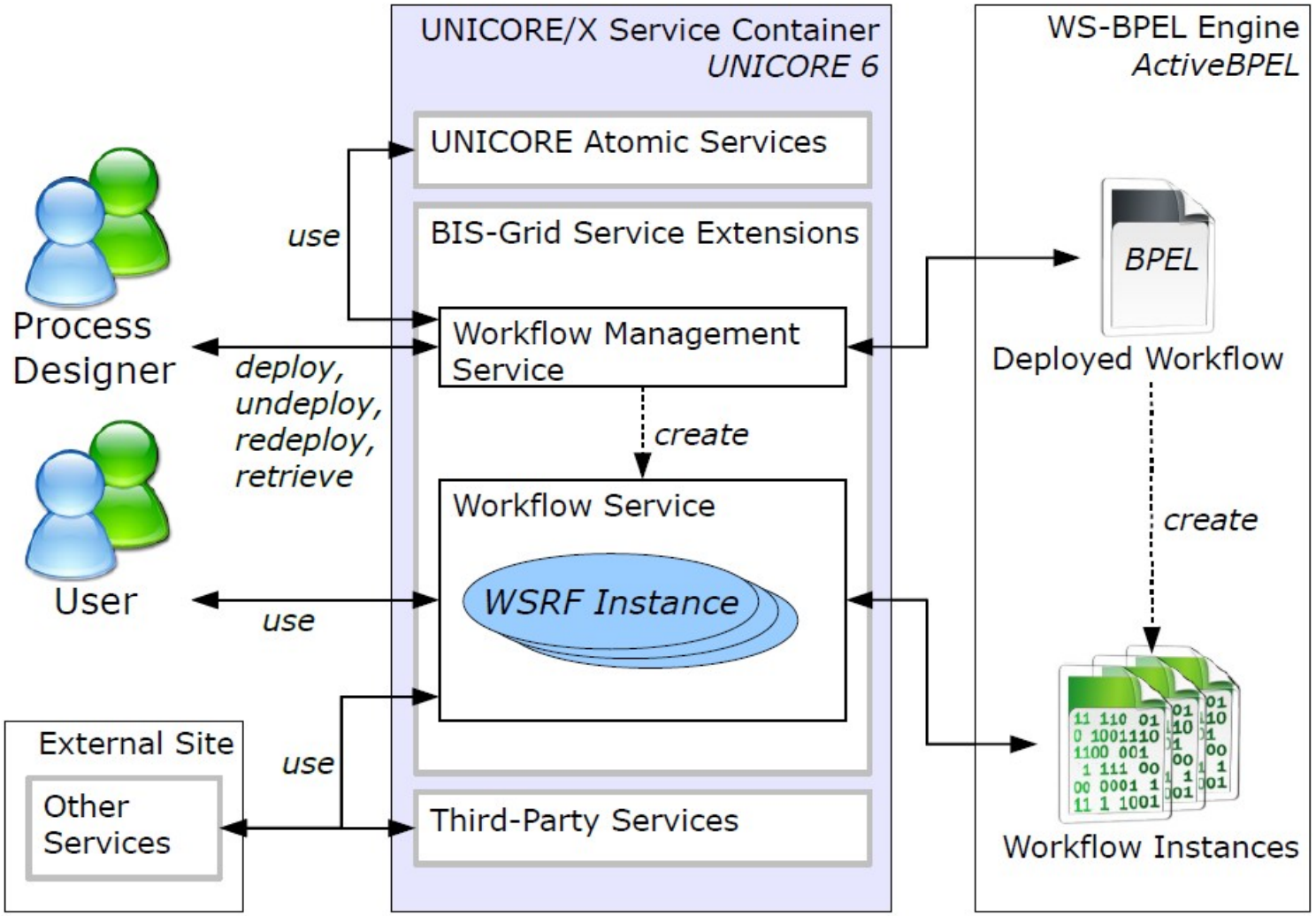
- **Grid Workflow Management**
- **BIS-Grid Engine Architecture**
- **Scientific vs. Business Workflows**
- **WS-BPEL for Scientific Grid Workflows**
- **WS-BPEL Patterns and Job Submission**
- **Future Work**

Grid Workflow Management

- Coordinate multiple job submissions over heterogeneous resources
- Provide high-level abstractions over distributed computations in the form of service orchestration
- Let users concentrate on their (scientific) application
- Abstract from technical non-domain-specific aspects
  - Resource management
  - Job scheduling
  - Details of data transfers



BIS-Grid Engine Architecture



Realized as UNICORE 6 Services, uses an arbitrary WS-BPEL engine (ActiveBPEL), provides workflows as WSRF-Grid services, supports RBAC (SAML, XACML), open source

Scientific and Business Workflows

Business workflows

- Control-driven
- Role-centric (RBAC, elaborate role models)
- Handling of information (small workflow data stored in process variables during workflow execution)
- Service provisioning (guaranteed completion, provided results as advertized)

Scientific workflows

- Data-driven (implicit control flow, activity starts when data available)
- User-centric (rights are often associated to persons directly)
- Handling of voluminous data (also third party transfers)
- Experiment implementation (importance of monitoring and workflow evolution caused by knowledge aquisition)

Similarities

- High-level view on the overall behavior of complex distributed applications
- Abstraction from technical details
- Capture and reuse domain knowledge/expertise
- Automate execution as far as possible

WS-BPEL for Scientific Workflows #1

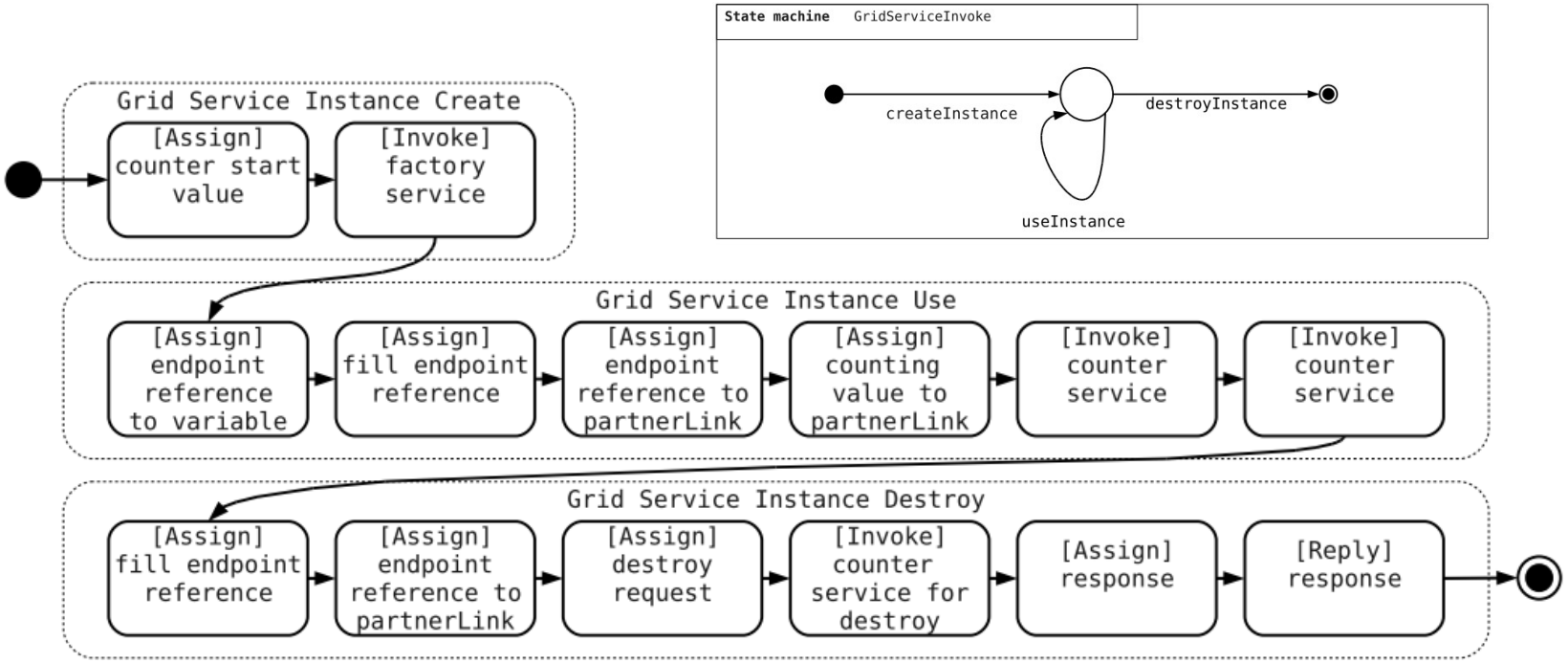
- WS-BPEL provides a single high-level abstraction for complex distributed applications in the form of reusable orchestrations of services (*programming in the large*)
- WS-BPEL descriptions represent and preserve reusable domain knowledge from a service orchestration view
- As a *de facto* industry standard WS-BPEL is well-supported by various tools for modeling and execution, and can also integrate other WS-* standards
  - Netbeans and BPMN for workflow design
  - ActiveBPEL as workflow engine
  - Possibly WS-HumanTask and BPEL4People for human interaction
- WS-BPEL addresses many of the general requirements of scientific workflow execution (next slide)

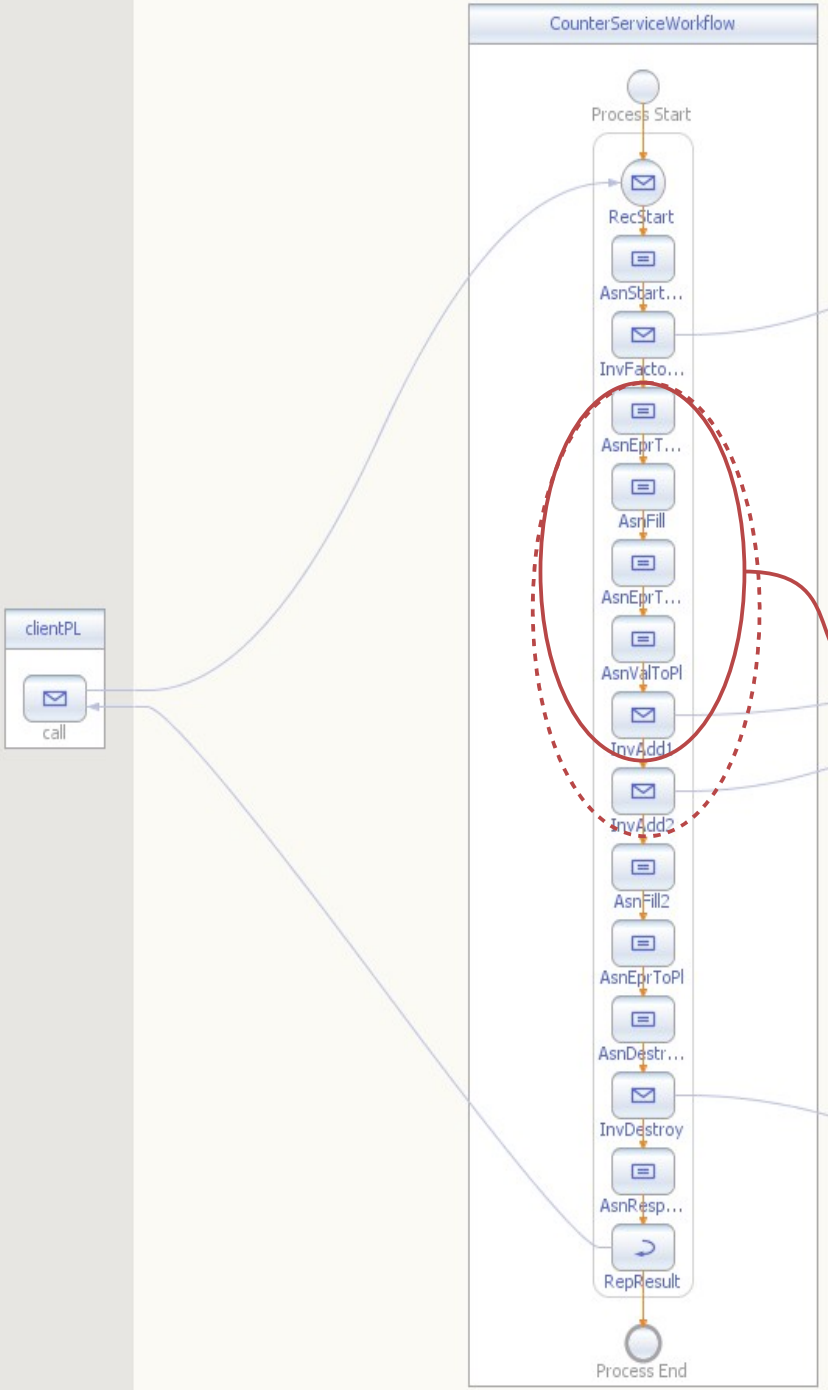
Requirements of Scientific Workflows

- *Modularity and composability* (e.g., to provide sub-workflows as standard activities on a higher level of abstraction) — *WS-BPEL*
- *Monitoring* (e.g., inspection of intermediate results of long-running processes) — *Engine, WS-BPEL Event Handlers*
- *Error handling and fault tolerance at* design-time and run-time (e.g., when sub-experiments fail) — *WS-BPEL (at design-time)*
- *Adaptability* at design-time and run-time (e.g., when the underlying resource infrastructure changes) — *WS-BPEL (at design-time)*
- *Domain-specificity* (e.g., modeling must respect that users of scientific workflows often are also their designers) — *Modeling Tools*
- *Voluminous data transfers* (e.g., large amounts of data should not be passed through the workflow engine in order to transfer it between two sites) — *WS-BPEL + JSDL-compliant Grid Middleware*

WS-BPEL for Scientific Workflows #2

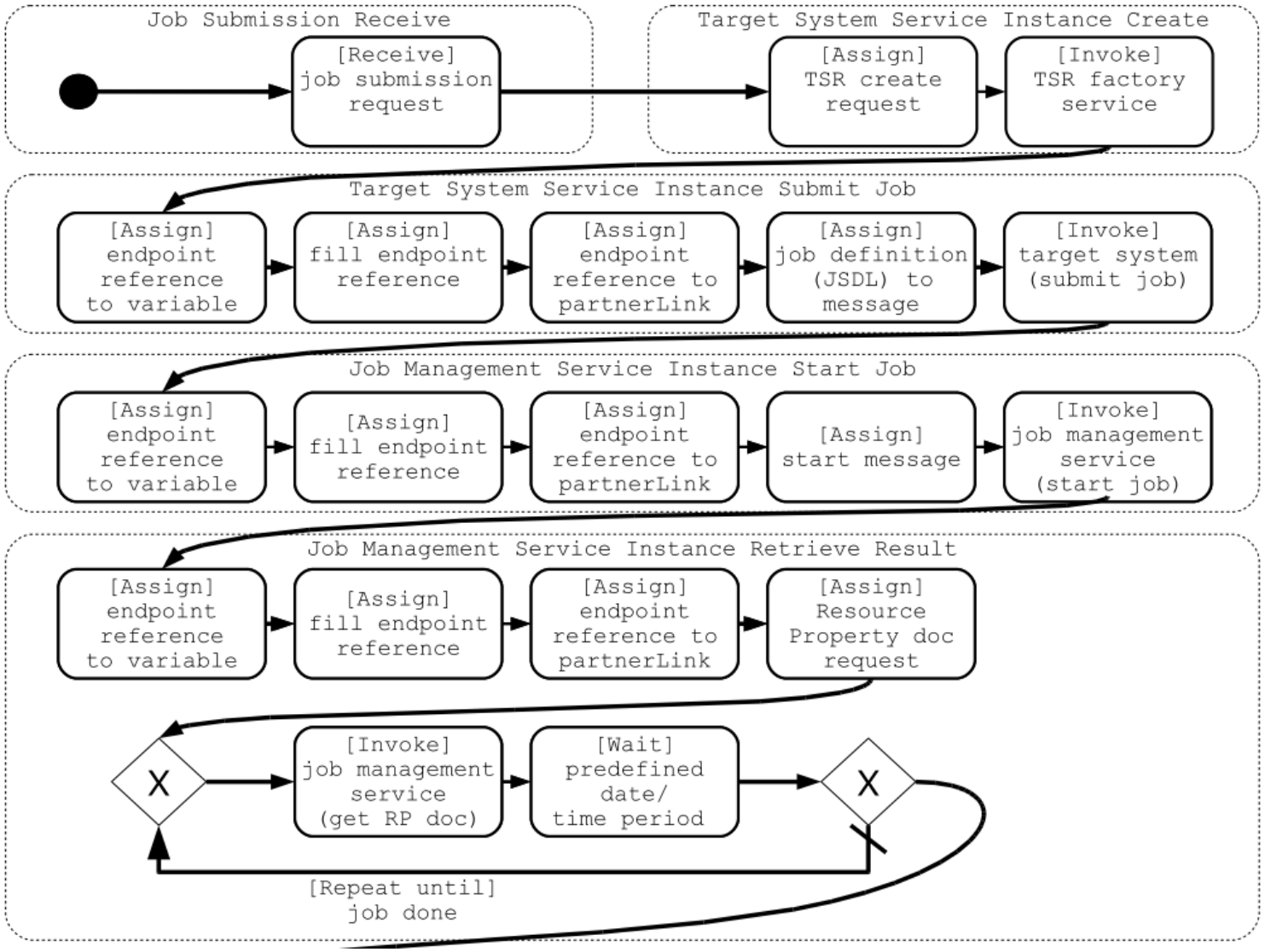
- It is bothersome to write WS-BPEL code without proper modeling support
  - Adequate modeling abstractions needed
- WS-BPEL does not support the use of complex stateful services natively, for example, regarding WSRF-based Grid services
- Grid Services can be orchstrated by standard WS-BPEL, resulting in a complex service invocation pattern to be reused as a simple service in higher-level orchestrations





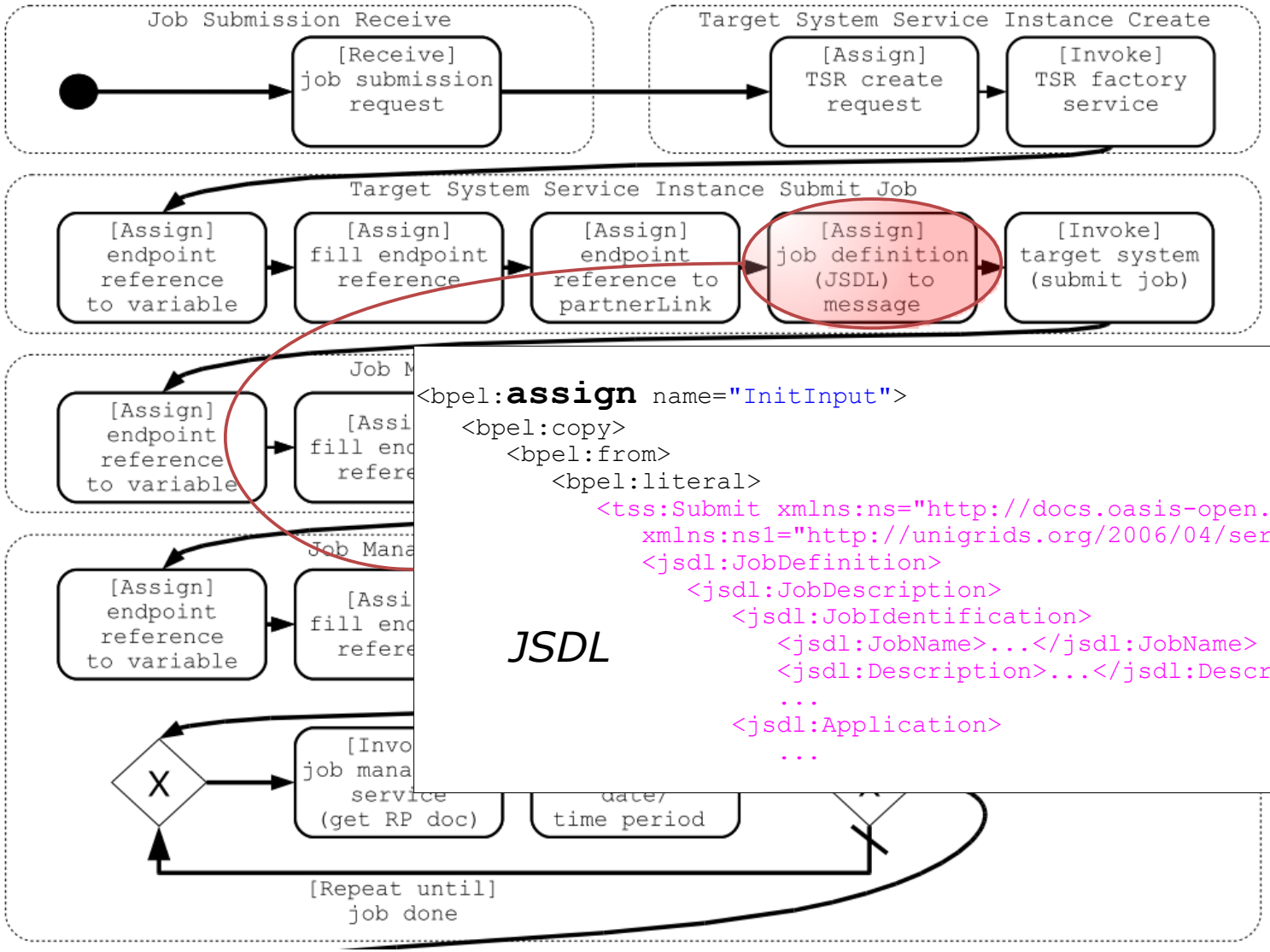

```
<bpel:assign name="AsnEprToVar">
   <bpel:copy>
      <bpel:from><bpel:literal>
            <wsa:EndpointReference
                  xmlns:wsa="http://www.w3.org/2005/08/addressing">
               <wsa:Address/>
               <wsa:ReferenceProperties>
                  <wsa:To/>
                  <wsa:Action/>
               </wsa:ReferenceProperties>
            </wsa:EndpointReference></bpel:literal>
      </bpel:from>
      <bpel:to variable="EndpointReference"/>
   </bpel:copy>
</bpel:assign>

<bpel:assign name="AsnFill">
   <bpel:copy>
      <bpel:from part="createCounterResponse" variable="createResponse">
         <bpel:query>ns6:EndpointReference/ns6:Address</bpel:query>
      </bpel:from>
      <bpel:to variable="EndpointReference">
         <bpel:query>ns6:Address</bpel:query>
      </bpel:to>
   </bpel:copy>
   <!-- SOME MORE COPIES HERE -->
</bpel:assign>

<bpel:assign name="AsnEprToPL">
   <bpel:copy>
      <bpel:from variable="EndpointReference"/>
      <bpel:to partnerLink="counterservicePL"/>
   </bpel:copy>
</bpel:assign>

<bpel:assign name="AsnValToPl">
   <bpel:copy>
      <bpel:from><bpel:literal>
            <mes:addRequest
                  xmlns:mes="http://bisgrid.dgrid.de/
                        services/counterservice/messages">
               <mes:value
                     xmlns:mes="http://bisgrid.dgrid.de/
                        services/counterservice/messages">
                  10
               </mes:value>
            </mes:addRequest></bpel:literal>
      </bpel:from>
      <bpel:to part="addRequest" variable="requestRequest"/>
   </bpel:copy>
</bpel:assign>

<bpel:invoke name="InvAdd1" inputVariable="requestRequest"
      operation="request" outputVariable="requestResponse"
      partnerLink="counterservicePL" portType="ns5:CounterServicePort"/>
```

Job Submission



to be cont'd ...

Job Submission

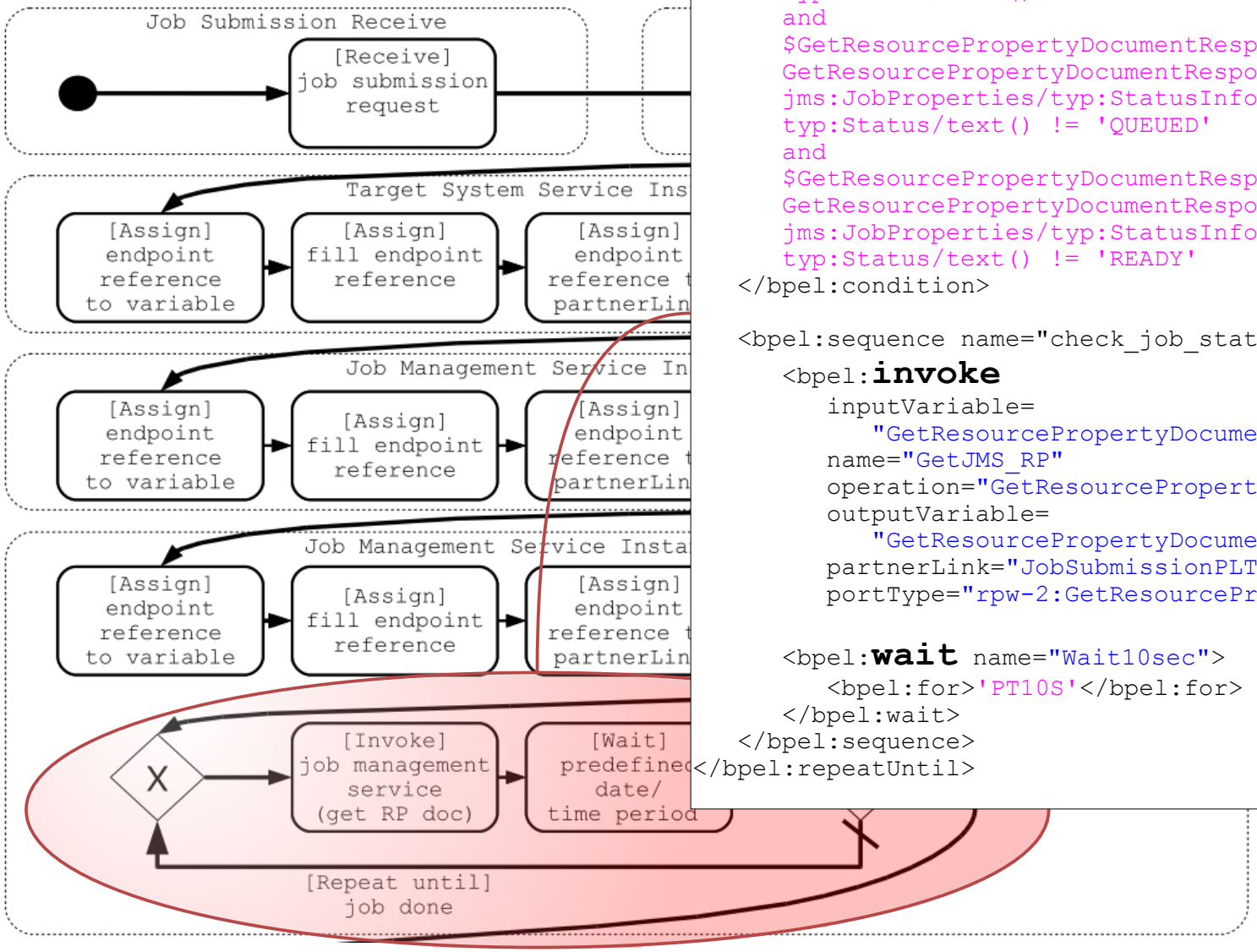



```
<bpel:assign name="InitInput">
   <bpel:copy>
      <bpel:from>
         <bpel:literal>
            <tss:Submit xmlns:ns="http://docs.oasis-open.org/wsrf/rl-2"
               xmlns:ns1="http://unigrids.org/2006/04/services/jms">
               <jsdl:JobDefinition>
                  <jsdl:JobDescription>
                     <jsdl:JobIdentification>
                        <jsdl:JobName>...</jsdl:JobName>
                        <jsdl:Description>...</jsdl:Description>
                        ...
                     <jsdl:Application>
                        ...
```

JSDL

to be cont'd ...

Job Submission




```
<bpel:repeatUntil name="job_done">
   <bpel:condition>
      $GetResourcePropertyDocumentResponse.
      GetResourcePropertyDocumentResponse/
      jms:JobProperties/typ:StatusInfo/
      typ:Status/text() != 'RUNNING'
      and
      $GetResourcePropertyDocumentResponse.
      GetResourcePropertyDocumentResponse/
      jms:JobProperties/typ:StatusInfo/
      typ:Status/text() != 'QUEUED'
      and
      $GetResourcePropertyDocumentResponse.
      GetResourcePropertyDocumentResponse/
      jms:JobProperties/typ:StatusInfo/
      typ:Status/text() != 'READY'
   </bpel:condition>

   <bpel:sequence name="check_job_state">
      <bpel:invoke
         inputVariable=
            "GetResourcePropertyDocumentRequest"
         name="GetJMS_RP"
         operation="GetResourcePropertyDocument"
         outputVariable=
            "GetResourcePropertyDocumentResponse"
         partnerLink="JobSubmissionPLT"
         portType="rpw-2:GetResourcePropertyDocument"/>

      <bpel:wait name="Wait10sec">
         <bpel:for>'PT10S'</bpel:for>
      </bpel:wait>
   </bpel:sequence>
</bpel:repeatUntil>
```

to be cont'd ...

Job Submission

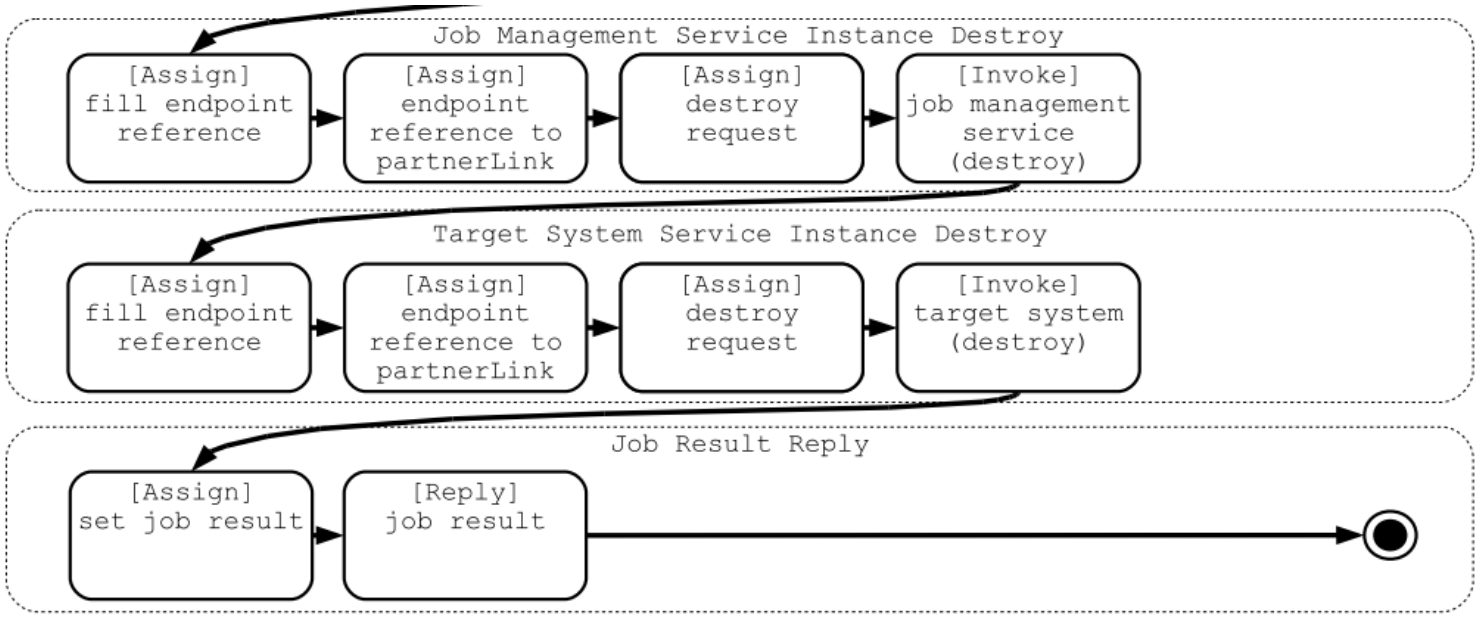
Current signature of the Job submission workflow

- *Input parameters*
  - Target System (`wsa: EndpointReferenceType`) *mandatory*
  - JSDL (`jsdl:JobDefinition_Type`) *mandatory*
  - LifetimeIntervall (`xsd:duration`) *optional*
  - WaitIntervall (`xsd:duration`) *optional*
- *Output parameters*
  - Result (`xsd:string`, *failed* or *successful*)

to be cont'd ...

Job Submission

... cont'd



- Job submission planned to be tested and evaluated in more detail in September
- Job submission workflow can be used in higher-level workflows in which, for example, a resource broker is invoked previously to choose a target system

Future Work

File transfers

- By using JSDL, the job submission pattern supports *simple* file transfers for *single* job executions
- However, the execution of *several* jobs using the *same* (voluminous) data and so-called *third party transfers* require special workflows
  - WS-BPEL engines are not designed for these purposes
  - File transfer as a single reusable domain-specific workflow activity to be configured properly

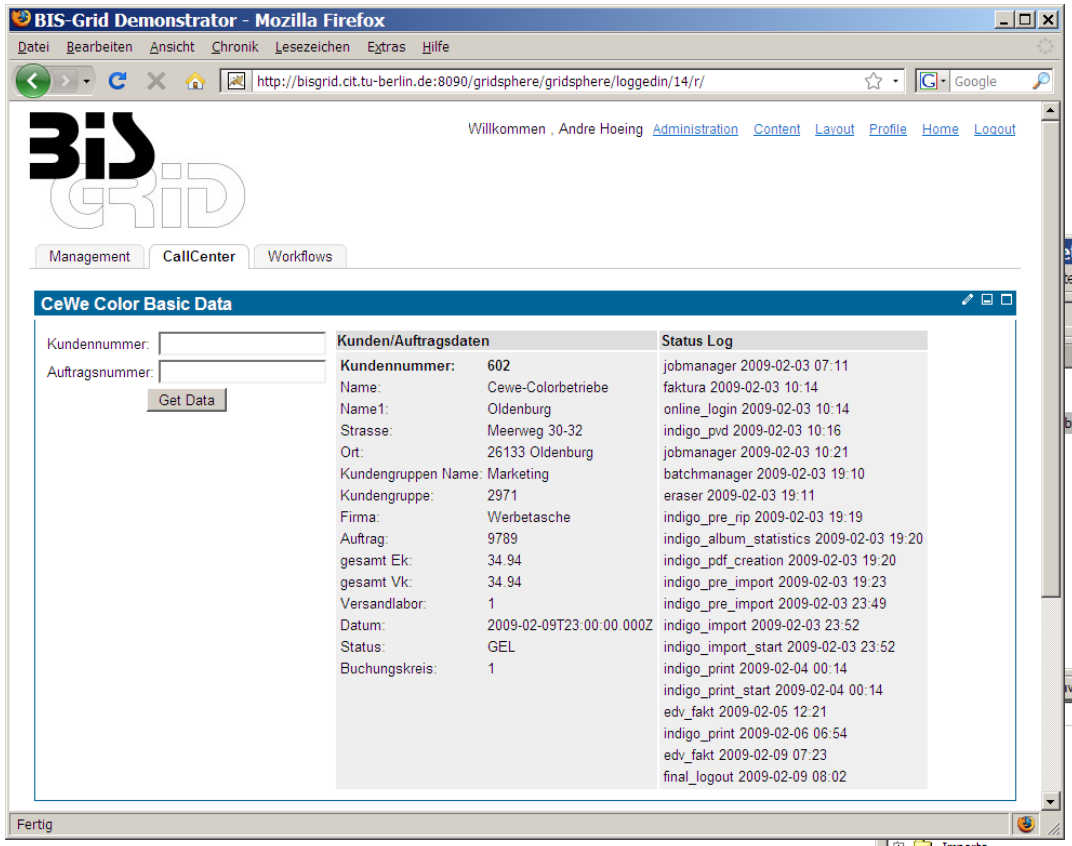
Globus Toolkit 4 interoperability

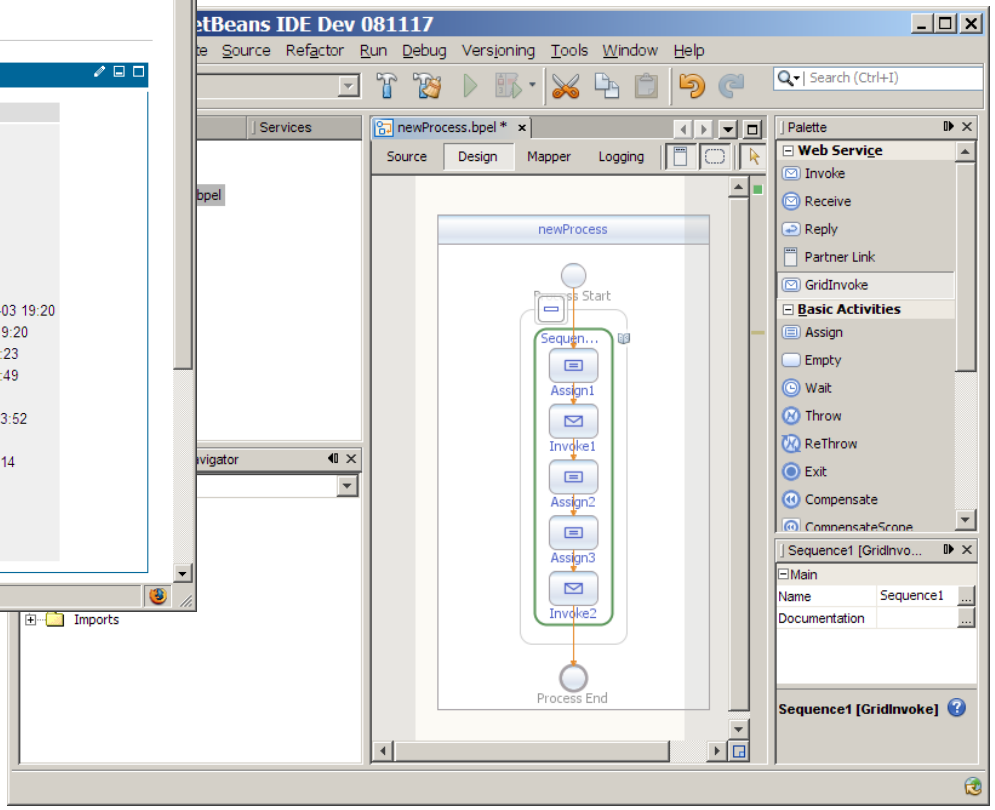
- We already developed a WS-BPEL pattern to invoke GT4 Grid services
- Open: support for GT4 Grid Security Infrastructure (GSI)
  - Evaluation in an appropriate GT4 service orchestration scenario planned (flood simulation)

Performance tests

- Application Scenario Workflows, Job Submission Workflow as basic wf-activity, bottleneck identification & load balancing

Future Work – User Clients





- End user client prototype currently based on GridSphere Portlets, business application scenarios shall integrate with existing tooling
- Extensions for Netbeans workflow designer (especially for deployment, but also for WS-BPEL pattern support)

Contact Information

Stefan Gudenkauf, Guido Scherp

Technology Cluster EAI

OFFIS

R&D Division Energy

Escherweg 2 - 26121 Oldenburg - Germany

26121 Oldenburg

Phone: +49 441 9722 178/122

E-Mail: [stefan.gudenkauf, guido.scherp]@offis.de,

URL: http://www.offis.de

More Infos:

http://www.bisgrid.de
(„Dokumente")

http://bis-grid.sourceforge.net/

André Höing

Technische Universitaet Berlin

Faculty of Electrical Engineering and Computer Science

Department of Telecommunication Systems

Complex and Distributed IT Systems

Einsteinufer 17

10587 Berlin

Phone: +49 30 314 78946

Fax: +49 30 314 21114

e-mail: andre.hoeing@tu-berlin.de

WWW: http://www.cit.tu-berlin.de/