User Friendly BPEL and Dynamic Workflow Execution in LEAD Grid

Aleksander Slominski, Satoshi Shirasuna
Indiana University

P21

Research Objectives

- Easy to use graphical workflow representation tailored for BPEL but not requiring users to know XML
- Enable and explore dynamic aspects of LEAD Orchestration WOORDS in BPEL
  - **W**orkflow **O**rchestration for **O**n-Demand, **R**eal-Time, **D**ynamically Adaptive **S**ystems

Workflow Composer: GUI on top, XML beneath

- **Workflow users should not be required to have a detailed knowledge of BPEL or XML**
- Easy to use – we are developing in an iterative way "good enough" subset of BPEL functionality
- Allow monitoring of workflow execution in graphical environment
- *Allow to "replay" history of workflow execution*
- *Allow to visualize state of running instance by connecting to it*

Graph to BPEL XML Translation strategy:

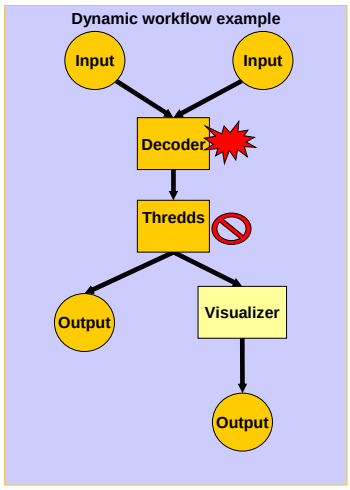
- **Composed workflow is a Web Service described in WSDL (hierarchical compensability)**
  - **Input nodes forms parts of input message**
  - **Output nodes form parts of output message**
- Workflow receives input message into a variable and sends output message based on variable
- Services exchange XML messages represented as variables in BPEL. Interspersed <assign> are automatically generated to pass relevant message parts between services.
- Services are invoked over asynchronous channel (it is OK for service to takes hours to produce response message) using one-way or request-response <invoke>
- Concurrency is enabled by <flow>

Simplified example of generated BPEL XML :

```
<sequence>
  <receive partnerLink="workflowUserPartner"
variable="workflowInput" />
 <assign source-variable  target-variable… />
 <invoke parnerLink="decoder" …/>
  <assign … />
  <invoke parnerLink="thredds" …/>
  <assign dest="workflowOutput"  … />
  <flow> <!– parallel execution -- >
    <invoke parnerLink="viz1" …/>
    <invoke parnerLink="viz2" …/>
  </flow>
 <reply partnerLink="workflowUserPartner"
variable="workflowOutput" />
</sequence>
```

Dynamic Aspects

- Execution can be paused on breakpoint
  - Debugging and tracing
- If service execution fails an alternative service can be used
- Allow introspection and modification of workflow state (parameter values etc.)
- New services can be added and connected into running workflow
- Running workflow can be "cloned" and alternative path of execution explored
  - Allow back-in-time and what-if scenarios
- **Allow extensive monitoring of workflow state and any kind of modifications to it**



Example of Dynamic Modifications

Scenario

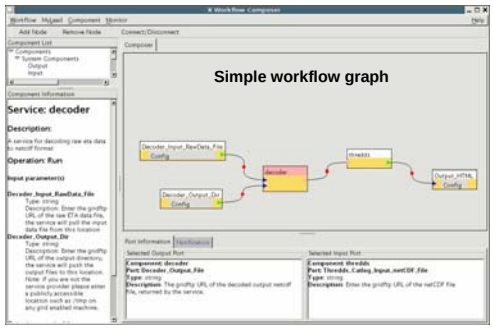
- Execution of a decoder fails
- User is notified
- User requests re-execution of the failed activity
- User selects an alternative decoder service
- Execution continues
- Execution is paused on thredds (breakpoint)
- New activity is added
- Execution continues
- New output is returned
- *Show how to "clone" workflow and services*

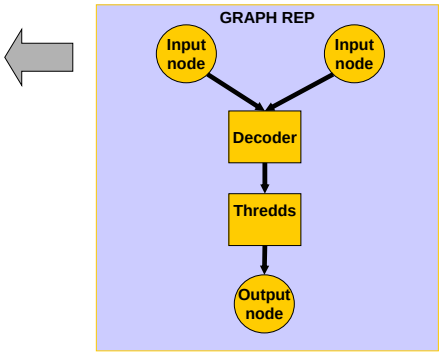
Workflow Reuse

- BPEL Workflow is also a service and has WSDL
- Workflow can use other workflow as services
- *GUI tool should allow to browse workflow template and expand nested sub-workflows*

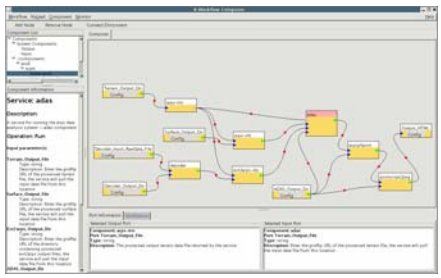
How to compose a workflow:

- Need only Web Service Description (WSDL) to use a web service in a workflow.
- The list of WSDLs are acquired from various places. e.g. web page, local file systems, registry services.
- Drop the WSDL of a web service to the composer. It will show you the connections for each input/output.
- Connect an output of a service to an input of another service as you want.
- The composer generates a BPEL script and WSDL (workflow is a service) based on the graph.





More complex workflow with parallel execution flows:



Future Work and Open Issues

- How to make dynamic workflow capabilities easy to use (GUI)?
- Integrating UNC performance analysis and monitoring tools into workflow engine to support optimal scheduling resources needed during workflow execution
- Supporting long running workflows:
  - In particular: listening for events and starting services

This work is supported primarily by the National Science Foundation under the following cooperative agreements: ATM03-31574, 31578, 31579, 31480, 31586, 31587, 31591, and 31594. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.