UML Modelling of Automated Business Processes with a Mapping to BPEL4WS

Tracy Gardner
IBM UK Laboratories, Hursley Park

First European Workshop on Web Services and Object Orientation, ECOOP 2003
UML to BPEL4WS

- UML is a widely used, standard modelling language for software design with a visual notation.

- BPEL4WS is a language for specifying business processes which can be executed on a BPEL4WS runtime.

- **Goal:** Support automated mapping from (a profile of) UML to BPEL4WS

A profile is a customization of UML for modelling in a particular context.
Target Audience

- The profile subsets and extends UML 1.4 to allow the design of UML models that can be mapped to BPEL4WS

- The profile is aimed at:
  UML-literate IT architects/developers who need to design systems that will be deployed to a BPEL4WS runtime
BPEL4WS

- The Business Process Execution Language for Web Services (BPEL4WS) provides an XML notation and semantics for specifying business process behavior based on Web Services.

- A BPEL4WS process is defined in terms of its interactions with partners. A partner may provide services to the process, require services from the process, or participate in a two-way interaction with the process.
A BPEL ‘program’

```xml
<process name="purchaseOrderProcess" ...>
  <containers>
    <container name="PO" messageType="lns:POMessage"/>
    <container name="Invoice" messageType="lns:InvMessage"/>
    ...
  </containers>

  <partners>
    <partner name="customer" serviceLinkType="lns:purchaseLT" myRole="purchaseService"/>
    ...
  </partners>

  <sequence>
    <receive partner="customer" portType="lns:purchaseOrderPT" operation="sendPurchaseOrder" container="PO">
    <reply partner="customer" portType="lns:purchasePT" operation="sendPurchaseOrder" container="Invoice"/>
    ...
  </sequence>

</process>
```

state
interaction points
behaviour
Receive a message from the customer

Purchase Order Process
Purchase Order Process – Detailed Behaviour

customer

«receive»
receivePO

<Entry/sendPurchaseOrder(PO)>

invoiceProvider

«invoke»
initiatePriceCalculation

<Entry/shippingRequest/customerInfo := PO/customerInfo>

«invoke»
sendShippingPrice

shippingProvider

«receive»
receiveSchedule

<Entry/shippingInfo := requestShipping(shippingRequest)>

«invoke»
requestShipping

shippingInfo

«receive»
receiveShippingPrice

冰淇

«assign»
initialiseShippingRequest

<Entry/shippingInfo := requestShipping(shippingRequest)>

schedulingProvider

«invoke»
requestScheduling

<Entry/requestScheduling(PO)>

invoiceProvider

«invoke»
initiatePriceCalculation

<schedulingProvider>

«receive»
receiveShippingSchedule

InvoiceProvider

«reply»
returnInvoice

<Entry/sendPurchaseOrder() := invoice>
Basic Interactions

- Action syntax based on XPATH which BPEL uses as its default expression language

- **receive**
  ```
  receivePO
  Entry/sendPurchaseOrder(PO)
  ```

- **reply**
  ```
  returnInvoice
  Entry/sendPurchaseOrder() := invoice
  ```

- **invoke**
  ```
  requestShipping
  Entry/shippingInfo := requestShipping(shippingRequest)
  ```

- **invoke**
  ```
  requestScheduling
  Entry/requestProductionScheduling(PO)
  ```

- Synchronous requests have a response

A receive with a corresponding reply handles a synchronous request.
# BPEL Mapping Overview

<table>
<thead>
<tr>
<th>BPEL Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;&lt;process&gt;&gt;</code> class</td>
<td>BPEL process definition</td>
</tr>
<tr>
<td>Activity graph on a <code>&lt;&lt;process&gt;&gt;</code> class</td>
<td>BPEL activity hierarchy</td>
</tr>
<tr>
<td><code>&lt;&lt;port&gt;&gt;</code> associations</td>
<td>BPEL partner declarations</td>
</tr>
<tr>
<td><code>&lt;&lt;process&gt;&gt;</code> class attributes</td>
<td>BPEL containers</td>
</tr>
<tr>
<td>Hierarchical structure and control flow</td>
<td>BPEL sequence and flow activities</td>
</tr>
<tr>
<td>Decision nodes</td>
<td>BPEL switch activities and transition</td>
</tr>
<tr>
<td><code>&lt;&lt;receive&gt;&gt;</code>, <code>&lt;&lt;reply&gt;&gt;</code>, <code>&lt;&lt;invoke&gt;&gt;</code></td>
<td>BPEL receive, reply, invoke activities</td>
</tr>
<tr>
<td>activities</td>
<td></td>
</tr>
<tr>
<td><code>&lt;&lt;protocol&gt;&gt;</code> package with <code>&lt;&lt;role&gt;&gt;</code> classes</td>
<td>BPEL service links types and roles</td>
</tr>
</tbody>
</table>
UML to BPEL alphaWorks Demo

- Part of the Emerging Technologies Toolkit
- Available now
- Supports an end-to-end scenario: Loan Approval
- Version supporting full forward generation from user-defined models coming in July 2003
Conclusions

- The experiences of modelling with UML can be applied to the development of systems that will be deployed using emerging web services standards

- It is possible to specify a profile of UML with sufficient detail that it can be translated automatically to a language such as BPEL4WS

- The approach provides an integration mechanism for multiple standards and specifications which need to be used to build a complex solution
  - This is especially relevant in business integration scenarios
Resources and Contact Details

- UML to BPEL on IBM alphaWorks:
  http://www.alphaworks.ibm.com/tech/ettk
  - Profile definition
  - Example models
  - End-to-end demo

- BPEL4WS Specification:

- BPWS4J on IBM alphaWorks:
  http://www.alphaworks.ibm.com/tech/bpws4j

- Contact: Tracy Gardner <tgardner@uk.ibm.com>