BPEL4WS

José M. Vidal

Tue Apr 13 15:02:52 EDT 2004

We describe workflows and the workflow language BPEL4WS. We study the possibility of using workflows in multiagent enactment.

- Andrews et. al. Business Process Execution Language for Web Services Version 1.1[1]. 2003.
- Unknown tag=José M. Vidal, Paul Buhler, and Christian Stahl. Multiagent Systems with Workflows.[2] *IEEE Internet Computing,* January/February; 8(1):76–82, 2004.

1 Workflows

- A representation of how work flows from person to person. Some examples include:
  - Fulfillment of a purchase order.
  - Handling an request for admissions at a University.
  - Handling an insurance claim.
  - Diagnosis and treatment of a patient.

1.1 Why Workflows?

- Workflows are pervasive in the business world.
  - IBM WebSphere MQ Workflow[3].
  - Lotus Workflow[4].
  - Microsoft BizTalk Server[5].
  - SAP[6] has been selling workflow systems for a long time (ERP). SAP NetWeaver[7] integrates them with SOAP (aka Web Services).
- Many of these tools allow "workflow experts" to draw these just as one would draw flowcharts.
- Replace the people with web services (which might, in turn, call a person) and you have a fully automated workflow.
- Workflows tell us how to put together individual web services into a coherent whole.
- With SOAP standardized, my SAP can talk to your BizTalk!

Note:
Most large companies have "workflow experts" whose job is to develop the workflows that the business implements. For example, an insurance agency might have experts that determine how a claim is to be handled. These workflows are under constant revision since the business' requests and needs are constantly changing. The tools the companies above provide usually provide a GUI that the expert can use to draw workflow, similar to how one might draw a flowchart. The **enactment** engine of the tools take these workflows as input and execute them.

2 BPEL4WS

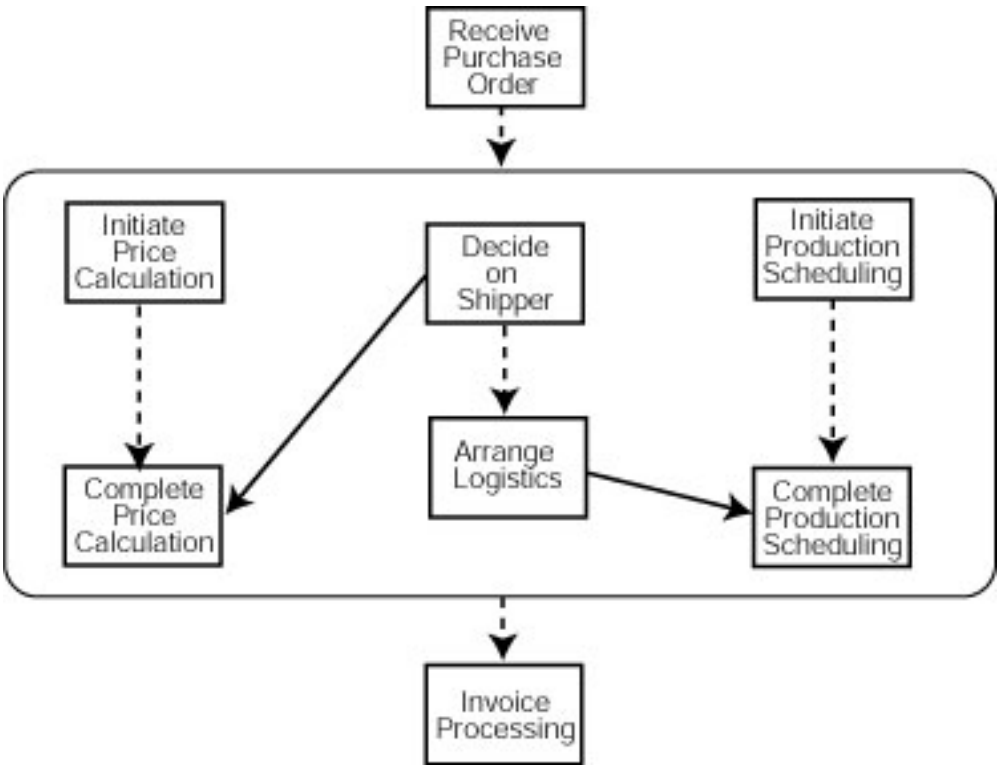
- XML-based standard jointly proposed by Microsoft, IBM, SAP, and others.
- Meant to replace their proprietary formats for storing workflow descriptions.
- Assumes atomic web services are described using WSDL.

Note:
BPEL4WS was created jointly by Microsoft and IBM, along with others. Both companies started with workflow tools that wrote the workflows to files using their own propietary format. One of the goals for BPEL4WS was for it to be able to express most of the constructs that could be implemented with both tools, so that one could save a workflow with one one vendor's tool and read it back with another vendor's tool (in the same way that many browsers can read the same HTML files).

3 BPEL4WS Language Structure

- A BPEL4WS document is divided into several parts:
  - <**partners**> contains a list of the web services invoked as part of the workflow.
  - <**variables**> used in the workflow.
  - <**correlationSets**> specifies precedence constraints between invocations.
  - <**faultHandlers**> are the exception handling routines.
  - <**compensationHandler**> handles compensation.
  - <**eventHandlers**> for asynchronous events.

3.1 PO Example



Purchase order workflow.

```
<process name="purchaseOrderProcess"
        targetNamespace="http://acme.com/ws-bp/purchase"
        xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
        xmlns:lns="http://manufacturing.org/wsdl/purchase">

  <partnerLinks>
    <partnerLink name="purchasing"
            partnerLinkType="lns:purchasingLT"
            myRole="purchaseService"/>
    <partnerLink name="invoicing"
            partnerLinkType="lns:invoicingLT"
            myRole="invoiceRequester"
            partnerRole="invoiceService"/>
    <partnerLink name="shipping"
            partnerLinkType="lns:shippingLT"
            myRole="shippingRequester"
            partnerRole="shippingService"/>
    <partnerLink name="scheduling"
            partnerLinkType="lns:schedulingLT"
            partnerRole="schedulingService"/>
  </partnerLinks>

  <variables>
    <variable name="PO" messageType="lns:POMessage"/>
    <variable name="Invoice"
              messageType="lns:InvMessage"/>
    <variable name="POFault"
              messageType="lns:orderFaultType"/>
    <variable name="shippingRequest"
              messageType="lns:shippingRequestMessage"/>
    <variable name="shippingInfo"
              messageType="lns:shippingInfoMessage"/>
    <variable name="shippingSchedule"
              messageType="lns:scheduleMessage"/>
  </variables>

  <faultHandlers>
    <catch faultName="lns:cannotCompleteOrder"
          faultVariable="POFault">
      <reply   partnerLink="purchasing"
               portType="lns:purchaseOrderPT"
               operation="sendPurchaseOrder"
               variable="POFault"
               faultName="cannotCompleteOrder"/>
    </catch>
  </faultHandlers>

  <sequence>

    <receive partnerLink="purchasing"
            portType="lns:purchaseOrderPT"
            operation="sendPurchaseOrder"
            variable="PO">
```

```
</receive>

<flow>

  <links>
    <link name="ship-to-invoice"/>
    <link name="ship-to-scheduling"/>
  </links>

  <sequence>
    <assign>
      <copy>
        <from variable="PO" part="customerInfo"/>
        <to variable="shippingRequest"
            part="customerInfo"/>
      </copy>
    </assign>

    <invoke  partnerLink="shipping"
             portType="lns:shippingPT"
             operation="requestShipping"
             inputVariable="shippingRequest"
             outputVariable="shippingInfo">
      <source linkName="ship-to-invoice"/>
    </invoke>

    <receive partnerLink="shipping"
             portType="lns:shippingCallbackPT"
             operation="sendSchedule"
             variable="shippingSchedule">
      <source linkName="ship-to-scheduling"/>
    </receive>

  </sequence>

  <sequence>

    <invoke  partnerLink="invoicing"
             portType="lns:computePricePT"
             operation="initiatePriceCalculation"
             inputVariable="PO">
    </invoke>
    <invoke  partnerLink="invoicing"
             portType="lns:computePricePT"
             operation="sendShippingPrice"
             inputVariable="shippingInfo">
      <target linkName="ship-to-invoice"/>
    </invoke>

    <receive partnerLink="invoicing"
             portType="lns:invoiceCallbackPT"
             operation="sendInvoice"
             variable="Invoice"/>
```

```
    </sequence>

    <sequence>
      <invoke  partnerLink="scheduling"
               portType="lns:schedulingPT"
               operation="requestProductionScheduling"
               inputVariable="PO">
      </invoke>
      <invoke  partnerLink="scheduling"
               portType="lns:schedulingPT"
               operation="sendShippingSchedule"
               inputVariable="shippingSchedule">
        <target linkName="ship-to-scheduling"/>
      </invoke>
    </sequence>
  </flow>

  <reply partnerLink="purchasing"
         portType="lns:purchaseOrderPT"
         operation="sendPurchaseOrder"
         variable="Invoice"/>
 </sequence>

</process>
```

3.2 General Structure

```
<process name="ncname" targetNamespace="uri"
        queryLanguage="anyURI"?
        expressionLanguage="anyURI"?
        suppressJoinFailure="yes—no"?
        enableInstanceCompensation="yes—no"?
        abstractProcess="yes—no"?
        xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/">

 <partnerLinks>?
  <!-- Note: At least one role must be specified. -->
  <partnerLink name="ncname" partnerLinkType="qname"
          myRole="ncname"? partnerRole="ncname"?>+
  </partnerLink>
 </partnerLinks>

 <partners>?
   <partner name="ncname">+
    <partnerLink name="ncname"/>+
 </partner>
 </partners>

 <variables>?
  <variable name="ncname" messageType="qname"?
            type="qname"? element="qname"?/>+
 </variables>
```

```
  <correlationSets>?
    <correlationSet name="ncname" properties="qname-list"/>+
  </correlationSets>

  <faultHandlers>?
    <!-- Note: There must be at least one fault handler or default. -->
    <catch faultName="qname"? faultVariable="ncname"?>*
      activity
    </catch>
    <catchAll>?
      activity
    </catchAll>
  </faultHandlers>

  <compensationHandler>?
    activity
  </compensationHandler>

  <eventHandlers>?
    <!-- Note: There must be at least one onMessage or onAlarm handler. -->
    <onMessage partnerLink="ncname" portType="qname"
               operation="ncname" variable="ncname"?>
      <correlations>?
        <correlation set="ncname" initiate="yes—no"?>+
      <correlations>
      activity
    </onMessage>
    <onAlarm for="duration-expr"? until="deadline-expr"?>*
      activity
    </onAlarm>
  </eventHandlers>

  activity
</process>
```

- Where **activity** can be any one of:
  - <receive>
  - <reply>
  - <invoke>
  - <assign>
  - <throw>
  - <terminate>
  - <wait>
  - <empty>
  - <sequence>
  - <switch>
  - <while>
  - <pick>
  - <flow>
  - <scope>
  - <compensate>

3.3 Receive

- The <receive> construct allows the business process to do a blocking wait for a matching message to arrive.

```
<receive partnerLink="ncname" portType="qname" operation="ncname"
        variable="ncname" createInstance="yes—no"
        standard-attributes>
 standard-elements
 <correlations>?
  <correlation set="ncname" initiate="yes—no"?>+
 </correlations>
</receive>
```

  - We are publishing a web service!

3.4 Reply

- The <reply> construct allows the business process to send a message in reply to a message that was received through a <receive>.
- The combination of a <receive> and a <reply> forms a request-response operation on the WSDL portType for the process.

```
<reply partnerLink="ncname" portType="qname" operation="ncname"
      variable="ncname"? faultName="qname"?
      standard-attributes>
 standard-elements
 <correlations>?
   <correlation set="ncname" initiate="yes—no"?>+
 </correlations>
</reply>
```

3.5 Invoke

- The <invoke> construct allows the business process to invoke a one-way or request-response operation on a portType offered by a partner.

```
<invoke partnerLink="ncname" portType="qname" operation="ncname"
       inputVariable="ncname"? outputVariable="ncname"?
       standard-attributes>
 standard-elements
 <correlations>?
   <correlation set="ncname" initiate="yes—no"?
             pattern="in—out—out-in"/>+
 </correlations>
 <catch faultName="qname" faultVariable="ncname"?>*
  activity
 </catch>
 <catchAll>?
  activity
 </catchAll>
 <compensationHandler>?
```

```
   activity
  </compensationHandler>
</invoke>

<!--- for example -->
<invoke partnerLink="Seller" portType="SP:Purchasing"
        operation="SyncPurchase"
        inputVariable="sendPO"
        outputVariable="getResponse">
  <compensationHandler>
     <invoke partnerLink="Seller" portType="SP:Purchasing"
             operation="CancelPurchase"
             inputVariable="getResponse"
             outputVariable="getConfirmation">
  </compensationHandler>
</invoke>
```

- We are invoking a web service!

3.6 Assign

- The <assign> construct can be used to update the values of variables with new data. An <assign> construct can contain any number of elementary assignments. The syntax of the assignment activity is:

```
<assign standard-attributes>
  standard-elements
  <copy>+
    from-spec
    to-spec
  </copy>
</assign>
```

3.7 Throw

- The <throw> construct generates a fault from inside the business process.

```
<throw faultName="qname" faultVariable="ncname"? standard-attributes>
  standard-elements
</throw>
```

3.8 Wait

- The <wait> construct allows you to wait for a given time period or until a certain time has passed. Exactly one of the expiration criteria must be specified.

```
<wait (for="duration-expr" — until="deadline-expr") standard-attributes>
  standard-elements
</wait>
```

3.9 Empty

- The <empty> construct allows you to insert a "no-op" instruction into a business process. This is useful for synchronization of concurrent activities.

```
<empty standard-attributes>
  standard-elements
</empty>
```

3.10 Sequence

- The <sequence> construct allows you to define a collection of activities to be performed sequentially in lexical order.

```
<sequence standard-attributes>
  standard-elementsactivity+
</sequence>
```

3.11 Switch

- The <switch> construct allows you to select exactly one branch of activity from a set of choices.

```
<switch standard-attributes>
  standard-elements
  <case condition="bool-expr">+
    activity
  </case>
  <otherwise>?
    activity
  </otherwise>
</switch>

<!-- for example -->

<switch xmlns:inventory="http://supply-chain.org/inventory"
        xmlns:FLT="http://example.com/faults">
  <case condition= "bpws:getVariableProperty(stockResult,level) > 100">
    <flow>
      <!-- perform fulfillment work -->
    </flow>
  </case>
  <case condition="bpws:getVariableProperty(stockResult,level) >= 0">
    <throw faultName="FLT:OutOfStock"
           variable="RestockEstimate"/>
  </case>
  <otherwise>
    <throw faultName="FLT:ItemDiscontinued"/>
  </otherwise>
</switch>
```

3.12 While

- The <while> construct allows you to indicate that an activity is to be repeated until a certain success criteria has been met.

```
<while condition="bool-expr" standard-attributes>
  standard-elementsactivity
</while>

<!-- for example -->

<variable name="orderDetails" type="xsd:integer"/>
   ...
<while condition=
          "bpws:getVariableData(orderDetails) > 100">
  <scope>
   ...
  </scope>
</while>
```

3.13 Pick

- The <pick> construct allows you to block and wait for a suitable message to arrive or for a time-out alarm to go off.

```
<pick createInstance="yes—no"? standard-attributes>
 standard-elements
 <onMessage partnerLink="ncname" portType="qname"
            operation="ncname" variable="ncname"?>+
  <correlations>?
    <correlation set="ncname" initiate="yes—no"?>+
  </correlations>
  activity
 </onMessage>
 <onAlarm (for="duration-expr" — until="deadline-expr")>*
  activity
 </onAlarm>
</pick>

<!-- for example -->
<pick>
 <onMessage partnerLink="buyer"
              portType="orderEntry"
              operation="inputLineItem"
              variable="lineItem">
     <!-- activity to add line item to order -->
 </onMessage>
 <onMessage partnerLink="buyer"
              portType="orderEntry"
              operation="orderComplete"
              variable="completionDetail">
     <!-- activity to perform order completion -->
  </onMessage>
```

```
  <!-- set an alarm to go after 3 days and 10 hours -->
  <onAlarm for="'P3DT10H'">
      <!-- handle timeout for order completion  -->
  </onAlarm>
</pick>
```

3.14 Flow

- The <flow> construct allows you to specify one or more activities to be performed concurrently.

```
<flow standard-attributes>
 standard-elements
 <links>?
  <link name="ncname">+
 </links>
 activity+
</flow>

<!-- for example -->
<flow>
 <links>
  <link name="XtoY"/>
  <link name="CtoD"/>
 </links>
 <sequence name="X">
   <source linkName="XtoY"/>
   <invoke name="A" .../>
   <invoke name="B" .../>
 </sequence>
 <sequence name"Y">
   <target linkName="XtoY"/>
   <receive name="C" ...>
    <source linkName="CtoD"/>
   </receive>
   <invoke name="E" .../>
 </sequence>
 <invoke partnerLink="D" ...>
  <target linkName="CtoD"/>
 </invoke>
</flow>
```

3.15 Scope

- The <scope> construct allows you to define a nested activity with its own associated variables, fault handlers, and compensation handler.

```
<scope variableAccessSerializable="yes—no" standard-attributes>
   standard-elements
   <variables>?

   </variables>
   <correlationSets>?
```

</**correlationSets**>
<**faultHandlers**>?

</**faultHandlers**>
<**compensationHandler**>?

</**compensationHandler**>
<**eventHandlers**>?
...
</**eventHandlers**>
activity
</**scope**>

3.16 Compensate

- The <compensate> construct is used to invoke compensation on an inner scope that has already completed normally.

<**compensate scope=**"ncname"**? standard-attributes**>
standard-elements
</**compensate**>

4 Process Instances

- A BPEL4WS workflow is like a class.
- An *instance* is created when an activity (receive) occurs which has `createInstance="yes"`.
- Every workflow must have at least one such start activity.
- If more than one, they must be correlated using a `correlationset`.
- The instance dies when either
  1. It reaches the end of the workflow.
  2. It raises a fault.
  3. It is terminated by <`terminate`>
  4. A compensationHandler is called.

5 PartnerLinkType

- The `partner`s are the other businesses (web services) the workflow accesses.
- Each partner plays a role:

<**partnerLinkType name=**"BuyerSellerLink"
xmlns="http://schemas.xmlsoap.org/ws/2003/05/partner-link/">
<**role name=**"Buyer">
<**portType name=**"buy:BuyerPortType"/>
</**role**>

```
    <role name="Seller">
      <portType name="sell:SellerPortType"/>
    </role>
  </partnerLinkType>
```

- Each role is linked to a WSDL PortType.

6 Properties

- You can define **properties** which are just nicknames for particular xsd types.

```
<definitions name="properties"
     targetNamespace="http://example.com/properties.wsdl"
     xmlns:tns="http://example.com/properties.wsdl"
     xmlns:txtyp="http://example.com/taxTypes.xsd"
     xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
     xmlns="http://schemas.xmlsoap.org/wsdl/">

   <!-- define a correlation property -->
   <bpws:property name="taxpayerNumber"
                          type="txtyp:SSN"/>
   ...
</wsdl:definitions>
```

7 Variables

- You can define any new variables and assign them parts of messages, or use them to invoke other services.

```
<complexType name="tAddress">
 <sequence>
  <element name="number" type="xsd:int"/>
  <element name="street" type="xsd:string"/>
  <element name="city" type="xsd:string"/>
  <element name="phone">
   <complexType>
    <sequence>
     <element name="areacode" type="xsd:int"/>
     <element name="exchange" type="xsd:int"/>
     <element name="number" type="xsd:int"/>
    </sequence>
   </complexType>
  </element>
 </sequence>
</complexType>

<element name = "address" type = "tAddress"/>

<!-- this message in the the .wsdl file -->
<message name="person" xmlns:x="http://tempuri.org/bpws/example">
 <part name="full-name" type="xsd:string"/>
```

```
  <part name="address" element="x:address"/>
</message>
<!-- -->

<variable name="c1" messageType="x:person"/>
<variable name="c2" messageType="x:person"/>
<variable name="c3" element="x:address"/>

<assign>
  <copy>
    <from variable="c1"/>
    <to variable="c2"/>
  </copy>
  <copy>
    <from variable="c1" part = "address"/>
    <to variable="c3"/>
  </copy>
</assign>
```

8 CorrelationSets

- Are used to correlate `invoce`, `receive`, and `reply` activities.
- The correlation sets must first be declared:

```
<correlationSets
      xmlns:cor="http://example.com/supplyCorrelation.wsdl">
   <!-- Order numbers are particular to a customer,
        this set is carried in application data -->
  <correlationSet name="PurchaseOrder"
          properties="cor:customerID cor:orderNumber"/>
   <!-- Invoice numbers are particular to a vendor,
        this set is carried in application data -->
  <correlationSet name="Invoice"
          properties="cor:vendorID cor:invoiceNumber"/>
</correlationSets>
```

so they can then be used

```
<receive partnerLink="Buyer" portType="SP:PurchasingPT"
        operation="AsyncPurchase"
        variable="PO">
  <correlations>
    <correlation set="PurchaseOrder" initiate="yes">
  </correlations>
</receive>

<invoke partnerLink="Buyer" portType="SP:BuyerPT"
      operation="AsyncPurchaseResponse" inputVariable="POResponse">
  <correlations>
    <correlation set="PurchaseOrder" initiate="no" pattern="out">
    <correlation set="Invoice" initiate="yes" pattern="out">
```

```
    </correlations>
  </invoke>
```

- The `initiate="yes"` means that the correlation is being started (new instance).
- The `pattern` tells us if the correlation applies to the outbound (request) message, the inbound (response) message, or both.

9 Links

- `link`s have a `name`.
- Activities then have `source`, and `target` which name particular links.
- Picture each activity as a node, the links are directed links from the source activity to the target activity.
- The links signify temporal precedence.

10 Loan Approval Example

```
<process name="loanApprovalProcess"
        targetNamespace="http://acme.com/loanprocessing"
        xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
        xmlns:lns="http://loans.org/wsdl/loan-approval"
        suppressJoinFailure="yes">

  <partnerLinks>
    <partnerLink name="customer"
            partnerLinkType="lns:loanPartnerLinkType"
            myRole="loanService"/>
    <partnerLink name="approver"
            partnerLinkType="lns:loanApprovalLinkType"
            partnerRole="approver"/>
    <partnerLink name="assessor"
            partnerLinkType="lns:riskAssessmentLinkType"
            partnerRole="assessor"/>
  </partnerLinks>

  <variables>
    <variable name="request"
             messageType="lns:creditInformationMessage"/>
    <variable name="risk"
             messageType="lns:riskAssessmentMessage"/>
    <variable name="approval"
             messageType="lns:approvalMessage"/>
    <variable name="error"
             messageType="lns:errorMessage"/>
  </variables>

  <faultHandlers>
    <catch faultName="lns:loanProcessFault"
          faultVariable="error">
```

```
    <reply   partnerLink="customer"
             portType="lns:loanServicePT"
             operation="request"
             variable="error"
             faultName="unableToHandleRequest"/>
  </catch>
</faultHandlers>


<flow>

  <links>
    <link name="receive-to-assess"/>
    <link name="receive-to-approval"/>
    <link name="approval-to-reply"/>
    <link name="assess-to-setMessage"/>
    <link name="setMessage-to-reply"/>
    <link name="assess-to-approval"/>
  </links>

  <receive partnerLink="customer"
           portType="lns:loanServicePT"
           operation="request"
           variable="request" createInstance="yes">
    <source linkName="receive-to-assess"
        transitionCondition=
          "bpws:getVariableData('request','amount')< 10000"/>
    <source linkName="receive-to-approval"
        transitionCondition=
          "bpws:getVariableData('request','amount')>=10000"/>
  </receive>

  <invoke  partnerLink="assessor"
           portType="lns:riskAssessmentPT"
           operation="check"
           inputVariable="request"
           outputVariable="risk">
    <target linkName="receive-to-assess"/>
    <source linkName="assess-to-setMessage"
        transitionCondition=
          "bpws:getVariableData('risk','level')='low'"/>
    <source linkName="assess-to-approval"
        transitionCondition=
          "bpws:getVariableData('risk','level')!='low'"/>
  </invoke>

  <assign>
    <target linkName="assess-to-setMessage"/>
    <source linkName="setMessage-to-reply"/>
    <copy>
      <from expression="'yes'"/>
      <to variable="approval" part="accept"/>
    </copy>
  </assign>
```

```
    <invoke  partnerLink="approver"
             portType="lns:loanApprovalPT"
             operation="approve"
             inputVariable="request"
             outputVariable="approval">
      <target linkName="receive-to-approval"/>
      <target linkName="assess-to-approval"/>
      <source linkName="approval-to-reply" />
    </invoke>

    <reply   partnerLink="customer"
             portType="lns:loanServicePT"
             operation="request"
             variable="approval">
      <target linkName="setMessage-to-reply"/>
      <target linkName="approval-to-reply"/>
    </reply>
  </flow>

</process>
```

11 Adaptive Multiagent Enactment of Workflows

- Current workflow enactment tools are centralized.
- BPEL4WS is centralized.
- BPEL4WS does not allow for any deviation or replacements. Web services are not autonomous.
- We are studying ways to do adaptive multiagent enactment of workflows.
  - Enact one workflow as a multiagent system, using Petri Nets. Buhler's thesis provides the first proof of concept.
  - Add autonomous agents.
  - Instantiate thousands of partial workflows under uncertainty. If many workflows will fail, how does the agent choose which one to participate in?
  - Mechanism design issues: what incentives should we provide agents in order to implement a successful business workflow?

Notes

[1] http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/
[2] http://jmvidal.cse.sc.edu/papers/vidal04a.pdf
[3] http://www-306.ibm.com/software/integration/wmqwf/
[4] http://www.lotus.com/products/domworkflow.nsf
[5] http://www.microsoft.com/biztalk/
[6] http://www.sap.com/
[7] http://www.sap.com/solutions/netweaver/index.asp