Pattern-based Evaluation of Oracle-BPEL (v.10.1.2)

N.A. Mulyar

Department of Technology Management, Eindhoven University of Technology
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
{n.mulyar@tm.tue.nl}

Table of contents

Introduction

In the light of the emerging paradigm, known as web-services composition, for enabling application integration within and across organizational boundaries several steps have been made to systematically evaluate the capabilities and limitations of languages and techniques proposed by different vendors and coalitions. In particular, an in-depth analysis of the Business Process Execution Language for Web Services (BPEL4WS) has been made in [1]. BPEL is a language that defines business processes and how these processes interact within/between organizations. The comparison of BPEL to other web services composition languages has been made in [2], [3], [4] and [15]. In all referred sources the evaluation of the web services composition languages has been made from the control-flow perspective, i.e. the degree of support of the 20 workflow patterns (cf. www.workflowpatterns.com) has been analyzed.

To make BPEL operational, it has been implemented in numerous tools. Similar to the evaluation of the BPEL core, the facilities offered by such tools have been a subject for the evaluation. In particular, an analysis of web services workflow patterns in Collaxa has been made in [11]. Based on the Collaxa engine [12] Oracle has implemented BPEL PM Designer that offers facilities for modeling of processes which can be deployed and made operational via the BPEL Console. The purpose of this work is to evaluate up to which degree the workflow patterns are supported by Oracle BPEL. For the purposes of this analysis we used not only the control-flow patterns described in [8], but also workflow data patterns and workflow resource patterns described in [9] and [10] respectively.

While Workflow Control Patterns characterize the range of control flow constructs that might be encountered when modeling and analyzing workflow, Workflow Data Patterns and Workflow Resource Patterns capture the various ways in which data resources are represented and utilized in workflows respectively. Workflow Control Patterns consist of basic control patterns, advanced branching and synchronization patterns, structural patterns, patterns involving multiple instances, state-based patterns and cancellation patterns. Workflow Data Patterns consist of data visibility, data interaction, data transfer and data-based routing patterns. In turn, Workflow Resource Patterns consist of creation patterns, push patterns, pull patterns, detour patterns, auto-start patterns, visibility patterns, and multiple resource patterns. These patterns have served as a reference point for the evaluation of Oracle BPEL PM.

As a knowledge base for the tool we used the tutorials and training material Oracle provided on-line (cf. http://www.oracle.com/technology/products/ias/bpel/index.html).

This document summarizes the results of evaluation of process modeling facilities offered by Oracle-BPEL PM (v.10.1.2) based on the evaluation of the contemporary set of workflow control, data and resource patterns. Since the evaluation of Oracle BPEL PM has been started for the v.2.1.2, and on the half-way of the evaluation the new version of Oracle BPEL PM has appeared, the majority of the results obtained from the evaluation of v.2.1.2 from the control-flow and data perspectives has been reused. The evaluation of the resource perspective is based purely on the v.10.1.2. Section 1 contains the detailed description of the Oracle BPEL PM facilities offered for support of the control-flow patterns. Sections 2 and 3 offer the detailed evaluation from the data and resource perspectives respectively. Finally, this document is completed by the Conclusions section.

1. Evaluation of Oracle BPEL PM from the control-flow perspective

CFP1: Sequence
Description: An activity in a workflow process is enabled after the completion of another activity in the same process.

Oracle BPEL PM supports this pattern directly. Graphically, two activities available at the BPEL Palette that can be connected by an arrow form a sequence. An example of the <sequence> is given in Figure 1. The <receive> activity that gets an input message from the client is followed by three <assign> activities which modify the input message. The result of modification is returned synchronously to the client by means of the <reply> activity.

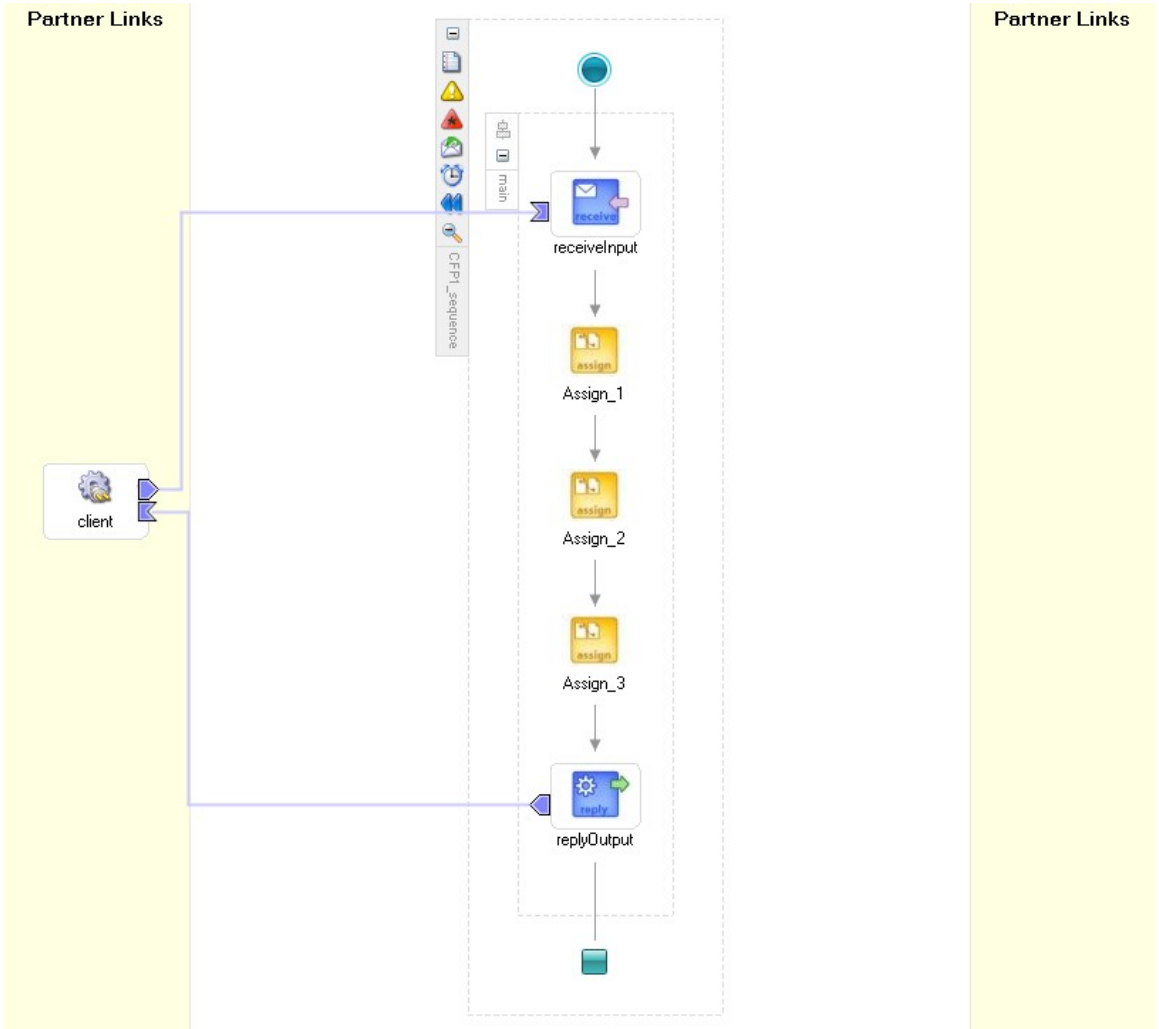


Figure 1 Sequence pattern

The code snippets corresponding to this block diagram are shown below.

```
<process name="CFP1_sequence"
targetNamespace="http://xmlns.oracle.com/CFP1_sequence"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.Xpath20"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:client="http://xmlns.oracle.com/CFP1_sequence"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.ExtFunc"><!--
================================================================= --><!-
- PARTNERLINKS                                                      --
><!-- List of services participating in this BPEL process
--><!--
================================================================= -->
  <partnerLinks><!--
    The 'client' role represents the requester of this service. It is
    used for callback. The location and correlation information
associated
    with the client role are automatically set using WS-Addressing.
    -->
    <partnerLink name="client" partnerLinkType="client:CFP1_sequence"
myRole="CFP1_sequenceProvider"/>
  </partnerLinks><!--
================================================================= --><!-
- VARIABLES                                                         --
><!-- List of messages and XML documents used within this BPEL process
--><!--
================================================================= -->
  <variables><!-- Reference to the message passed as input during
initiation -->
    <variable name="inputVariable"
messageType="client:CFP1_sequenceRequestMessage"/><!--
    Reference to the message that will be returned to the requester
    -->
    <variable name="outputVariable"
messageType="client:CFP1_sequenceResponseMessage"/>
  </variables><!--
================================================================= --><!-
- ORCHESTRATION LOGIC                                               --
><!-- Set of activities coordinating the flow of messages across the
--><!-- services integrated within this business process
--><!--
================================================================= -->
  <sequence name="main"><!-- Receive input from requestor.
    Note: This maps to operation defined in CFP1_sequence.wsdl
    -->
    <receive name="receiveInput" partnerLink="client"
portType="client:CFP1_sequence" operation="process"
variable="inputVariable" createInstance="yes"/><!-- Generate reply to
synchronous request -->
```

```
    <assign name="Assign_1">
      <copy>
        <from
expression="concat(bpws:getVariableData('inputVariable','payload','/clie
nt:CFP1_sequenceProcessRequest/client:input'),'1')"/>
        <to variable="outputVariable" part="payload"
query="/client:CFP1_sequenceProcessResponse/client:result"/>
      </copy>
    </assign>
    <assign name="Assign_2">
      <copy>
        <from
expression="concat(bpws:getVariableData('outputVariable','payload','/cli
ent:CFP1_sequenceProcessResponse/client:result'),'2')"/>
        <to variable="outputVariable" part="payload"
query="/client:CFP1_sequenceProcessResponse/client:result"/>
      </copy>
    </assign>
    <assign name="Assign_3">
      <copy>
        <from
expression="concat(bpws:getVariableData('outputVariable','payload','/cli
ent:CFP1_sequenceProcessResponse/client:result'),'3')"/>
        <to variable="outputVariable" part="payload"
query="/client:CFP1_sequenceProcessResponse/client:result"/>
      </copy>
    </assign>
    <reply name="replyOutput" partnerLink="client"
portType="client:CFP1_sequence" operation="process"
variable="outputVariable"/>
  </sequence>
</process>
```

The content of the wsdl file is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="CFP1_sequence"
             targetNamespace="http://xmlns.oracle.com/CFP1_sequence"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:client="http://xmlns.oracle.com/CFP1_sequence"
             xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-
link/">

      <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      TYPE DEFINITION - List of services participating in this BPEL
process
      The default output of the BPEL designer uses strings as input and
      output to the BPEL Process. But you can define or import any XML
      Schema type and us them as part of the message types.
      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~ -->
      <types>
            <schema attributeFormDefault="qualified"
                  elementFormDefault="qualified"
```

```
      targetNamespace="http://xmlns.oracle.com/CFP1_sequence"
                  xmlns="http://www.w3.org/2001/XMLSchema">
                  <element name="CFP1_sequenceProcessRequest">
                        <complexType>
                              <sequence>
                                    <element name="input"
type="string"/>
                              </sequence>
                        </complexType>
                  </element>
                  <element name="CFP1_sequenceProcessResponse">
                        <complexType>
                              <sequence>
                                    <element name="result"
type="string"/>
                              </sequence>
                        </complexType>
                  </element>
            </schema>
      </types>

      <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      MESSAGE TYPE DEFINITION - Definition of the message types used as
      part of the port type defintions
      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~ -->
      <message name="CFP1_sequenceRequestMessage">
            <part name="payload"
element="client:CFP1_sequenceProcessRequest"/>
      </message>
      <message name="CFP1_sequenceResponseMessage">
            <part name="payload"
element="client:CFP1_sequenceProcessResponse"/>
      </message>

      <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      PORT TYPE DEFINITION - A port type groups a set of operations into
      a logical service unit.
      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~ -->

      <!-- portType implemented by the CFP1_sequence BPEL process -->
      <portType name="CFP1_sequence">
            <operation name="process">
                  <input  message="client:CFP1_sequenceRequestMessage"
/>
                  <output
message="client:CFP1_sequenceResponseMessage"/>
            </operation>
      </portType>

      <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      PARTNER LINK TYPE DEFINITION
```

```
      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~ -->
      <plnk:partnerLinkType name="CFP1_sequence">
            <plnk:role name="CFP1_sequenceProvider">
                  <plnk:portType name="client:CFP1_sequence"/>
            </plnk:role>
      </plnk:partnerLinkType>
</definitions>
```

An audit trail visualizing the execution history of the considered example is shown below:

[2005/08/10 10:30:25] New instance of BPEL process "CFP1_sequence" initiated (# "202").

<process>

<sequence>

receiveInput

[2005/08/10 10:30:25] Received "inputVariable" call from partner "client" More...

Assign_1

[2005/08/10 10:30:25] Updated variable "outputVariable" less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP1_sequenceProcessResponse xmlns="http://xmlns.oracle.com/CFP1_sequence">
<result>b1</result>
      </CFP1_sequenceProcessResponse>
      </part>
      </outputVariable>
```

Assign_2

[2005/08/10 10:30:25] Updated variable "outputVariable" less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP1_sequenceProcessResponse xmlns="http://xmlns.oracle.com/CFP1_sequence">
<result>b12</result>
      </CFP1_sequenceProcessResponse>
      </part>
      </outputVariable>
```

Assign_3

[2005/08/10 10:30:25] Updated variable "outputVariable" less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP1_sequenceProcessResponse xmlns="http://xmlns.oracle.com/CFP1_sequence">
<result>b123</result>
      </CFP1_sequenceProcessResponse>
      </part>
      </outputVariable>
```

replyOutput

[2005/08/10 10:30:25] Reply to partner "client". less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP1_sequenceProcessResponse xmlns="http://xmlns.oracle.com/CFP1_sequence">
```

```
<result>b123</result>
        </CFP1_sequenceProcessResponse>
        </part>
        </outputVariable>
        </sequence>
        [2005/08/10 10:30:25]   BPEL process instance "202" completed
        </process>
```

Another possibility to implement the sequence, which is not the most straightforward solution, is to use links within the <flow> structure. In particular, to ensure that activities *Assign_1, Assign_2, and Assign_3 are* executed in the sequential lexical order, these activities are associated with links *Link12* and *Link23*. As such the activity *Assign_2* serves as a target of the *Link12* and a source of the *Link23.* The process model of the considered example is shown in Figure 2.

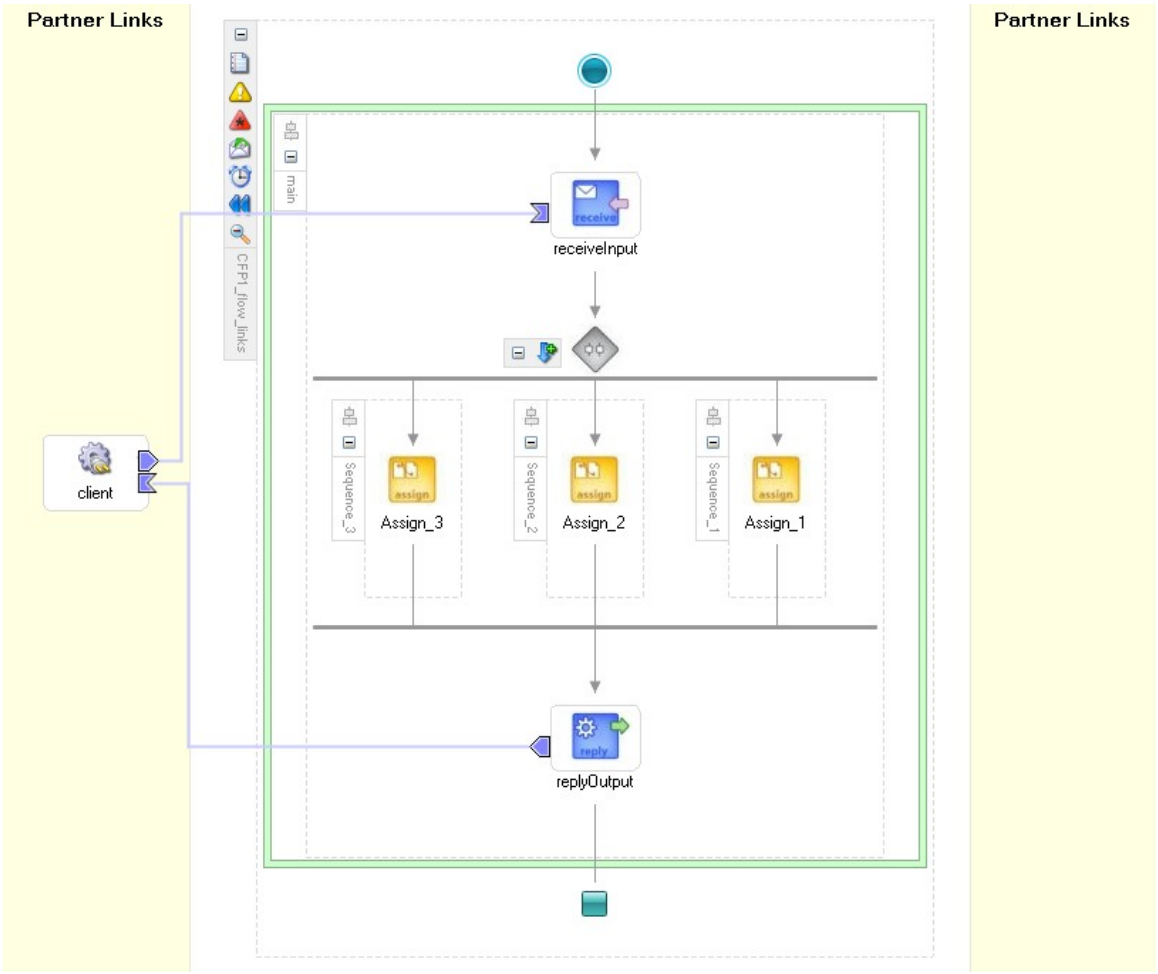


Figure 2 Sequence by means of <flow> and links

The code snippets of the considered example are shown below:

```
<process name="CFP1_flow_links"
targetNamespace="http://xmlns.oracle.com/CFP1_flow_links"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.Xpath20"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:client="http://xmlns.oracle.com/CFP1_flow_links"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.ExtFunc"><!--
================================================================= --><!-
- PARTNERLINKS                                                      --
><!-- List of services participating in this BPEL process
--><!--
================================================================= -->
  <partnerLinks><!--
    The 'client' role represents the requester of this service. It is
    used for callback. The location and correlation information
associated
    with the client role are automatically set using WS-Addressing.
    -->
    <partnerLink name="client" partnerLinkType="client:CFP1_flow_links"
myRole="CFP1_flow_linksProvider"/>
  </partnerLinks><!--
================================================================= --><!-
- VARIABLES                                                         --
><!-- List of messages and XML documents used within this BPEL process
--><!--
================================================================= -->
  <variables><!-- Reference to the message passed as input during
initiation -->
    <variable name="inputVariable"
messageType="client:CFP1_flow_linksRequestMessage"/><!--
    Reference to the message that will be returned to the requester
    -->
    <variable name="outputVariable"
messageType="client:CFP1_flow_linksResponseMessage"/>
  </variables><!--
================================================================= --><!-
- ORCHESTRATION LOGIC                                               --
><!-- Set of activities coordinating the flow of messages across the
--><!-- services integrated within this business process
--><!--
================================================================= -->
  <sequence name="main"><!-- Receive input from requestor.
    Note: This maps to operation defined in CFP1_flow_links.wsdl
    -->
    <receive name="receiveInput" partnerLink="client"
portType="client:CFP1_flow_links" operation="process"
variable="inputVariable" createInstance="yes"/><!-- Generate reply to
synchronous request -->
    <flow name="Flow_1">
      <links>
```

```
        <link name="Link12"/>
        <link name="Link23"/>
      </links>
      <sequence name="Sequence_3">
        <target linkName="Link23"/>
        <assign name="Assign_3">
          <copy>
            <from
expression="concat(bpws:getVariableData('outputVariable','payload','/cli
ent:CFP1_flow_linksProcessResponse/client:result'),'3')"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP1_flow_linksProcessResponse/client:result"/>
          </copy>
        </assign>
      </sequence>
      <sequence name="Sequence_2">
        <target linkName="Link12"/>
        <source linkName="Link23"/>
        <assign name="Assign_2">
          <copy>
            <from
expression="concat(bpws:getVariableData('outputVariable','payload','/cli
ent:CFP1_flow_linksProcessResponse/client:result'),'2')"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP1_flow_linksProcessResponse/client:result"/>
          </copy>
        </assign>
      </sequence>
      <sequence name="Sequence_1">
        <source linkName="Link12"/>
        <assign name="Assign_1">
          <copy>
            <from
expression="concat(bpws:getVariableData('inputVariable','payload','/clie
nt:CFP1_flow_linksProcessRequest/client:input'),'1')"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP1_flow_linksProcessResponse/client:result"/>
          </copy>
        </assign>
      </sequence>
    </flow>
    <reply name="replyOutput" partnerLink="client"
portType="client:CFP1_flow_links" operation="process"
variable="outputVariable"/>
  </sequence>
</process>
```

An audit trail visualizing the execution history of the considered example is shown below:

[2005/08/10 10:45:24] New instance of BPEL process "CFP1_flow_links" initiated (# "203").

<process>

<sequence>

receiveInput

[2005/08/10 10:45:24] Received "inputVariable" call from partner "client" less

<inputVariable>

<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP1_flow_linksProcessRequest xmlns="http://xmlns.oracle.com/CFP1_flow_links">
<input>a</input>
</CFP1_flow_linksProcessRequest>
</part>
</inputVariable>
<flow>
<sequence>

Assign_1

[2005/08/10 10:45:24] Updated variable "outputVariable" less

<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP1_flow_linksProcessResponse xmlns="http://xmlns.oracle.com/CFP1_flow_links">
<result>a1</result>
</CFP1_flow_linksProcessResponse>
</part>
</outputVariable>
</sequence>
<sequence>

Assign_2

[2005/08/10 10:45:24] Updated variable "outputVariable" less

<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP1_flow_linksProcessResponse xmlns="http://xmlns.oracle.com/CFP1_flow_links">
<result>a12</result>
</CFP1_flow_linksProcessResponse>
</part>
</outputVariable>
</sequence>
<sequence>

Assign_3

[2005/08/10 10:45:24] Updated variable "outputVariable" less

<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP1_flow_linksProcessResponse xmlns="http://xmlns.oracle.com/CFP1_flow_links">
<result>a123</result>
</CFP1_flow_linksProcessResponse>
</part>
</outputVariable>
</sequence>
</flow>

replyOutput

[2005/08/10 10:45:24] Reply to partner "client". less

<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP1_flow_linksProcessResponse xmlns="http://xmlns.oracle.com/CFP1_flow_links">
<result>a123</result>

```
</CFP1_flow_linksProcessResponse>
</part>
</outputVariable>
</sequence>
[2005/08/10 10:45:24]   BPEL process instance "203" completed
</process>
```

CFP2 Parallel Split
Description: A point in the process where a single thread of control splits into multiple threads of control which can be executed in parallel, thus allowing activities to be executed simultaneously or in any order.

Oracle BPEL PM supports this pattern directly by means of the <flow> construct. The <flow> construct allows creating multiple branches, which are independent and may execute in any order. Oracle BPEL PM implemented this construct with an assumption that multiple activities will be present in every branch, therefore offering the <sequence> construct as the wrapper for these activities.

An example of implementing the Parallel split pattern is shown in Figure 3. In this example three <assign> activities are executed in parallel. In order to check whether the branches are executed in parallel, each of the <assign> activities is preceded by the <wait> block.

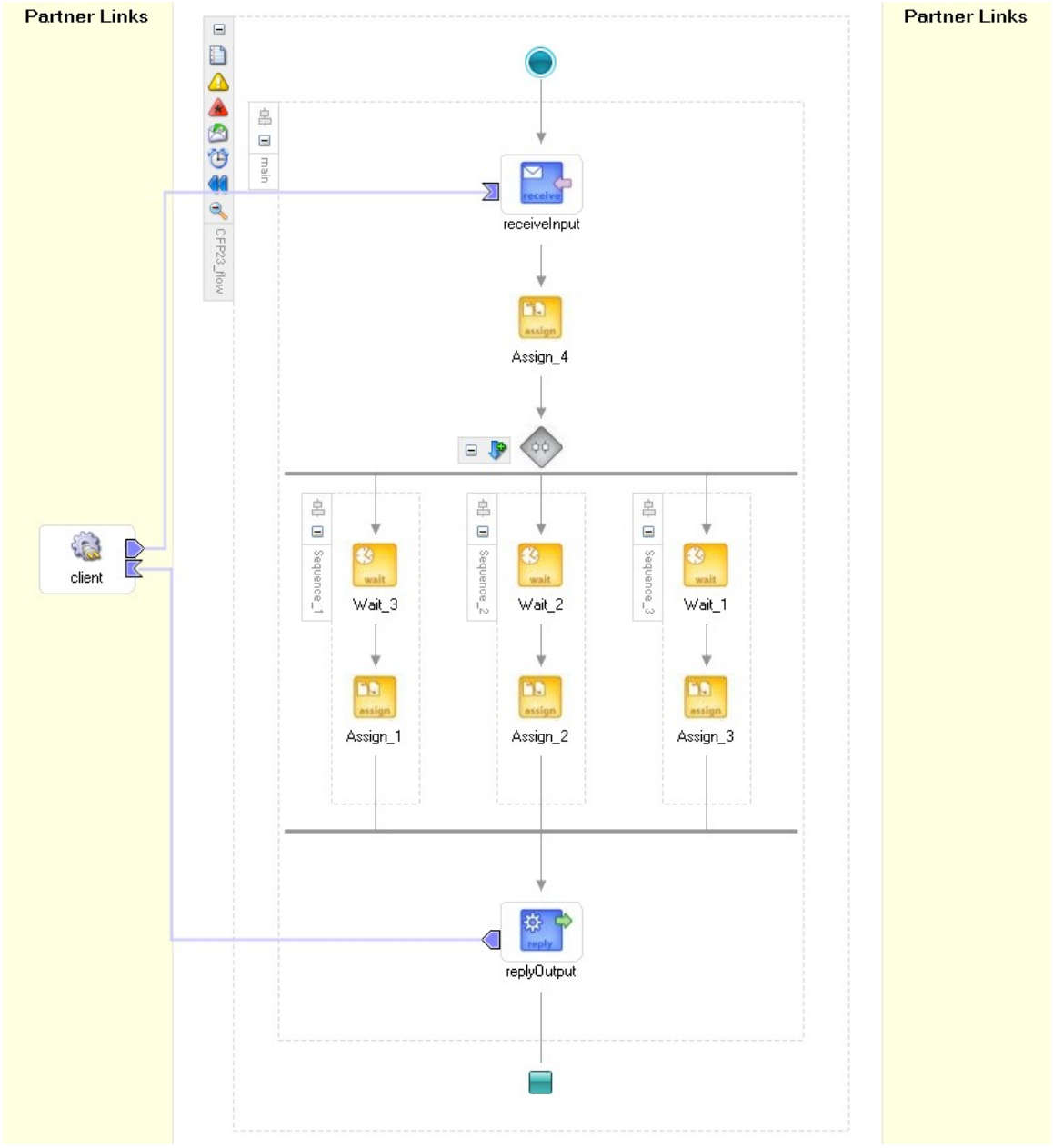


Figure 3 Parallel split by means of <flow>

The code snippets corresponding to the considered example are shown below:

```
<process name="CFP23_flow"
targetNamespace="http://xmlns.oracle.com/CFP23_flow"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.Xpath20"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:client="http://xmlns.oracle.com/CFP23_flow"
```

```
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.ExtFunc">
<!-- ======================================================>
<!-- PARTNERLINKS                                           -
-><!-- List of services participating in this BPEL process
--><!--
================================================================= -->
  <partnerLinks><!--
    The 'client' role represents the requester of this service. It is
    used for callback. The location and correlation information
associated
    with the client role are automatically set using WS-Addressing.
    -->
    <partnerLink name="client" partnerLinkType="client:CFP23_flow"
myRole="CFP23_flowProvider"/>
  </partnerLinks><!--
================================================================= --><!-
- VARIABLES                                                       --
><!-- List of messages and XML documents used within this BPEL process
--><!--
================================================================= -->
  <variables><!-- Reference to the message passed as input during
initiation -->
    <variable name="inputVariable"
messageType="client:CFP23_flowRequestMessage"/><!--
    Reference to the message that will be returned to the requester
    -->
    <variable name="outputVariable"
messageType="client:CFP23_flowResponseMessage"/>
  </variables><!--
================================================================= --><!-
- ORCHESTRATION LOGIC                                             --
><!-- Set of activities coordinating the flow of messages across the
--><!-- services integrated within this business process
--><!--
================================================================= -->
  <sequence name="main"><!-- Receive input from requestor.
    Note: This maps to operation defined in CFP23_flow.wsdl
    -->
    <receive name="receiveInput" partnerLink="client"
portType="client:CFP23_flow" operation="process"
variable="inputVariable" createInstance="yes"/><!-- Generate reply to
synchronous request -->
    <assign name="Assign_4">
      <copy>
        <from variable="inputVariable" part="payload"
query="/client:CFP23_flowProcessRequest/client:input"/>
        <to variable="outputVariable" part="payload"
query="/client:CFP23_flowProcessResponse/client:result"/>
      </copy>
    </assign>
    <flow name="Flow_1">
      <sequence name="Sequence_3">
        <wait name="Wait_1" for="'PT1M'"/>
        <assign name="Assign_3">
          <copy>
```

```
            <from
expression="concat(bpws:getVariableData('outputVariable','payload','/cli
ent:CFP23_flowProcessResponse/client:result'),'3')"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP23_flowProcessResponse/client:result"/>
          </copy>
        </assign>
      </sequence>
      <sequence name="Sequence_2">
        <wait name="Wait_2" for="'PT2M'"/>
        <assign name="Assign_2">
          <copy>
            <from
expression="concat(bpws:getVariableData('outputVariable','payload','/cli
ent:CFP23_flowProcessResponse/client:result'),'2')"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP23_flowProcessResponse/client:result"/>
          </copy>
        </assign>
      </sequence>
      <sequence name="Sequence_1">
        <wait name="Wait_3" for="'PT3M'"/>
        <assign name="Assign_1">
          <copy>
            <from
expression="concat(bpws:getVariableData('outputVariable','payload','/cli
ent:CFP23_flowProcessResponse/client:result'),'1')"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP23_flowProcessResponse/client:result"/>
          </copy>
        </assign>
      </sequence>
    </flow>
    <reply name="replyOutput" partnerLink="client"
portType="client:CFP23_flow" operation="process"
variable="outputVariable"/>
  </sequence>
</process>
```

An audit trail visualizing the execution history of the considered example is shown below:

[2005/08/10 10:57:53] New instance of BPEL process "CFP23_flow" initiated (# "205").

<process>

<sequence>

receiveInput

[2005/08/10 10:57:53] Received "inputVariable" call from partner "client" less

```
<inputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP23_flowProcessRequest xmlns="http://xmlns.oracle.com/CFP23_flow">
<input>b</input>
    </CFP23_flowProcessRequest>
    </part>
    </inputVariable>
```

Assign_4

[2005/08/10 10:57:53] Updated variable "outputVariable" less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP23_flowProcessResponse xmlns="http://xmlns.oracle.com/CFP23_flow">
<result>b</result>
    </CFP23_flowProcessResponse>
    </part>
    </outputVariable>
```

<flow>

<sequence>

Wait_3 - pending

[2005/08/10 10:57:53] Waiting for the expiry time "2005/08/10 11:00:53".

<sequence>

Wait_2 - pending

[2005/08/10 10:57:53] Waiting for the expiry time "2005/08/10 10:59:53".

<sequence>

Wait_1

[2005/08/10 10:57:53] Waiting for the expiry time "2005/08/10 10:58:53".

[2005/08/10 10:58:53] Wait has finished.

Assign_3

[2005/08/10 10:58:53] Updated variable "outputVariable" less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP23_flowProcessResponse xmlns="http://xmlns.oracle.com/CFP23_flow">
<result>b3</result>
    </CFP23_flowProcessResponse>
    </part>
    </outputVariable>
```

</sequence>

CFP3 Synchronization

Description: A point in the process where multiple parallel branches converge into one single thread of control, thus synchronizing multiple threads. It is an assumption of this pattern that after an incoming branch has been completed, it cannot be completed again while the merge is still waiting for other branches to be completed. Also, it is assumed that the threads to be synchronized belong to the same global process instance (i.e., to the same case in workflow terminology).

Oracle BPEL PM supports this pattern directly by means of the <flow> construct. Every branch of the flow construct may be executed only once, and it completes only after all branches have completed. See as a reference an example implementing Parallel Split pattern.

CFP4 Exclusive Choice
Description: A point in the workflow process where, based on a decision or workflow control data, one of several branches is chosen.

Oracle BPEL PM supports this pattern directly by means of the <switch> construct. The <switch> allows making deterministic choice between several branches, i.e. cases, depending on the fulfillment of the case conditions. If none of the specified conditions is fulfilled, the <otherwise> branch is taken (or deemed to be taken). If the conditions for several branches are satisfied, then the first one specified in the lexical order is taken.

An example demonstrating the use of the <switch> construct is given in Figure 4. Each of the three branches is associated with a certain data conditions; the *otherwise* branch handles all conditions not specified for the other two branches. Each of these data conditions is evaluated after a string input has been supplied by the client. In order to check which branch is to be selected when conditions associated with multiple branches are satisfied, the branches with activities *Assign_1* and *Assign_2* are associated with overlapping data conditions. As a result, if multiple cases can be selected, the first branch specified in the lexical order is selected.

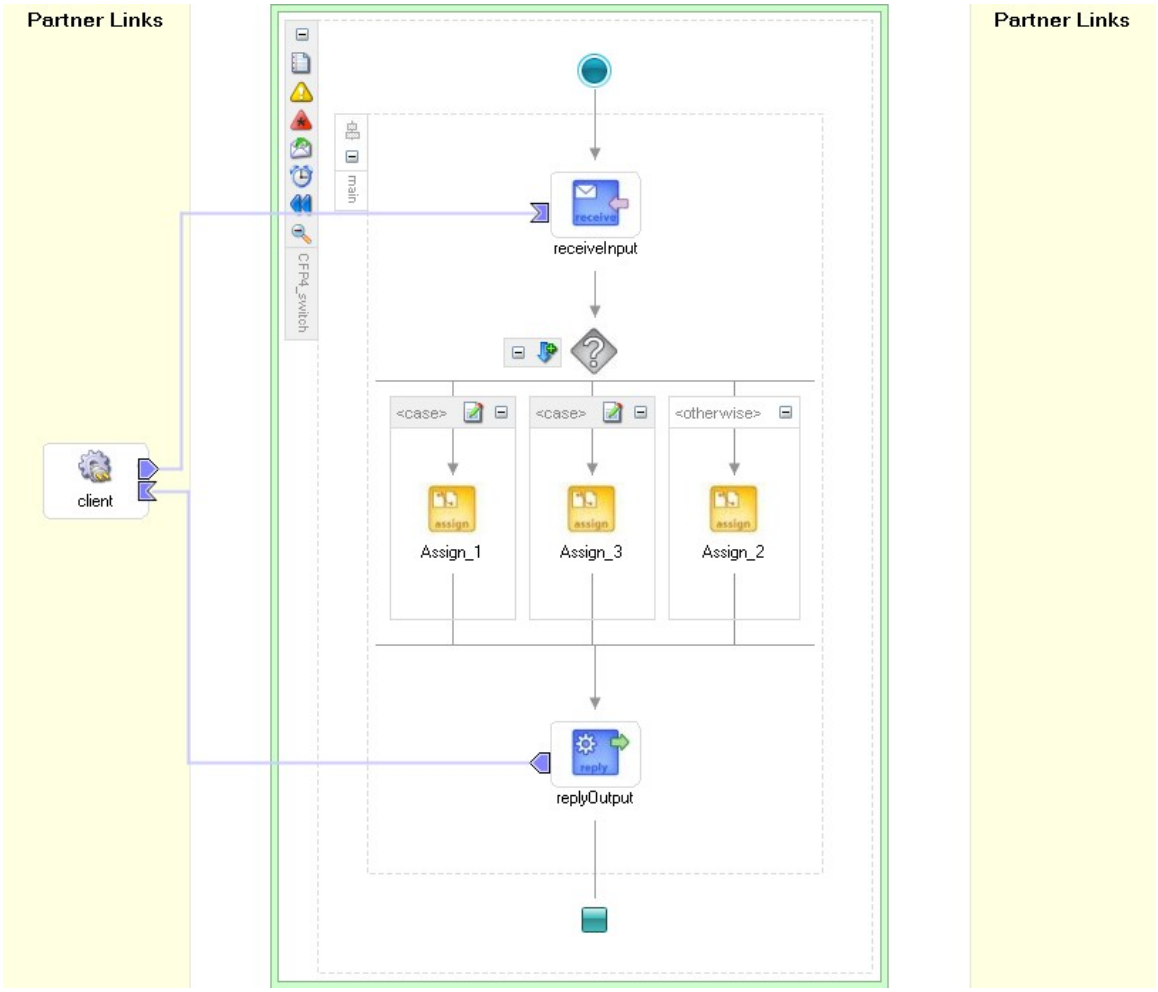


Figure 4 Exclusive choice

The code snippets describing the process model presented in Figure 4 are shown below:

```
<process name="CFP4_switch"
targetNamespace="http://xmlns.oracle.com/CFP4_switch"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.Xpath20"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:client="http://xmlns.oracle.com/CFP4_switch"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.ExtFunc"><!--
================================================================= --><!-
- PARTNERLINKS                                                      --
><!-- List of services participating in this BPEL process
--><!--
================================================================= -->
  <partnerLinks><!--
    The 'client' role represents the requester of this service. It is
    used for callback. The location and correlation information
associated
    with the client role are automatically set using WS-Addressing.
    -->
    <partnerLink name="client" partnerLinkType="client:CFP4_switch"
myRole="CFP4_switchProvider"/>
  </partnerLinks><!--
================================================================= --><!-
- VARIABLES                                                         --
><!-- List of messages and XML documents used within this BPEL process
--><!--
================================================================= -->
  <variables><!-- Reference to the message passed as input during
initiation -->
    <variable name="inputVariable"
messageType="client:CFP4_switchRequestMessage"/><!--
    Reference to the message that will be returned to the requester
    -->
    <variable name="outputVariable"
messageType="client:CFP4_switchResponseMessage"/>
  </variables><!--
================================================================= --><!-
- ORCHESTRATION LOGIC                                               --
><!-- Set of activities coordinating the flow of messages across the
--><!-- services integrated within this business process
--><!--
================================================================= -->
  <sequence name="main"><!-- Receive input from requestor.
    Note: This maps to operation defined in CFP4_switch.wsdl
    -->
    <receive name="receiveInput" partnerLink="client"
portType="client:CFP4_switch" operation="process"
variable="inputVariable" createInstance="yes"/><!-- Generate reply to
synchronous request -->
```

```
    <switch name="Switch_1">
      <case condition="contains('cde',
bpws:getVariableData('inputVariable','payload','/client:CFP4_switchProce
ssRequest/client:input'))">
        <assign name="Assign_3">
          <copy>
            <from expression="'cde'"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP4_switchProcessResponse/client:result"/>
          </copy>
        </assign>
      </case>
      <case condition="contains('abc',
bpws:getVariableData('inputVariable','payload','/client:CFP4_switchProce
ssRequest/client:input'))">
        <assign name="Assign_1">
          <copy>
            <from expression="'abc'"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP4_switchProcessResponse/client:result"/>
          </copy>
        </assign>
      </case>
      <otherwise>
        <assign name="Assign_2">
          <copy>
            <from variable="inputVariable" part="payload"
query="/client:CFP4_switchProcessRequest/client:input"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP4_switchProcessResponse/client:result"/>
          </copy>
        </assign>
      </otherwise>
    </switch>
    <reply name="replyOutput" partnerLink="client"
portType="client:CFP4_switch" operation="process"
variable="outputVariable"/>
  </sequence>
</process>
```

An audit trail visualizing the execution history of the considered example is shown below:

[2005/08/10 11:00:55] New instance of BPEL process "CFP4_switch" initiated (# "206").

<process>

<sequence>

receiveInput

[2005/08/10 11:00:55] Received "inputVariable" call from partner "client" less

<inputVariable>

<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">

<CFP4_switchProcessRequest xmlns="http://xmlns.oracle.com/CFP4_switch">

<input>a</input>

</CFP4_switchProcessRequest>

</part>

</inputVariable>

<switch>

Assign_1

[2005/08/10 11:00:55] Updated variable "outputVariable" less

<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP4_switchProcessResponse xmlns="http://xmlns.oracle.com/CFP4_switch">
<result>abc</result>
</CFP4_switchProcessResponse>
</part>
</outputVariable>

</switch>

replyOutput

[2005/08/10 11:00:55] Reply to partner "client". less

<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP4_switchProcessResponse xmlns="http://xmlns.oracle.com/CFP4_switch">
<result>abc</result>
</CFP4_switchProcessResponse>
</part>
</outputVariable>

</sequence>

[2005/08/10 11:00:55] BPEL process instance "206" completed

</process>

[2005/08/10 11:00:47] New instance of BPEL process "CFP4_switch" initiated (# "207").

<process>

<sequence>

receiveInput

[2005/08/10 11:00:47] Received "inputVariable" call from partner "client" less

<inputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP4_switchProcessRequest xmlns="http://xmlns.oracle.com/CFP4_switch">
<input>c</input>
</CFP4_switchProcessRequest>
</part>
</inputVariable>

<switch>

Assign_3

[2005/08/10 11:00:47] Updated variable "outputVariable" less

<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP4_switchProcessResponse xmlns="http://xmlns.oracle.com/CFP4_switch">
<result>cde</result>
</CFP4_switchProcessResponse>
</part>

</outputVariable>
</switch>

replyOutput

[2005/08/10 11:00:47] Reply to partner "client". less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP4_switchProcessResponse xmlns="http://xmlns.oracle.com/CFP4_switch">
<result>cde</result>
</CFP4_switchProcessResponse>
</part>
</outputVariable>
```

</sequence>

[2005/08/10 11:00:47] BPEL process instance "207" completed

</process>

[2005/08/10 11:11:41] New instance of BPEL process "CFP4_switch" initiated (# "208").

<process>

<sequence>

receiveInput

[2005/08/10 11:11:41] Received "inputVariable" call from partner "client" less

```
<inputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP4_switchProcessRequest xmlns="http://xmlns.oracle.com/CFP4_switch">
<input>f</input>
</CFP4_switchProcessRequest>
</part>
</inputVariable>
```

<switch>

Assign_2

[2005/08/10 11:11:41] Updated variable "outputVariable" less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP4_switchProcessResponse xmlns="http://xmlns.oracle.com/CFP4_switch">
<result>f</result>
</CFP4_switchProcessResponse>
</part>
</outputVariable>
```

</switch>

replyOutput

[2005/08/10 11:11:41] Reply to partner "client". less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP4_switchProcessResponse xmlns="http://xmlns.oracle.com/CFP4_switch">
<result>f</result>
</CFP4_switchProcessResponse>
</part>
```

```
</outputVariable>
</sequence>
[2005/08/10 11:11:41]  BPEL process instance "208" completed
</process>
```

Another possibility to realize the Exclusive Choice pattern is to use links with disjoint conditions within the <flow> construct (see Figure 5).

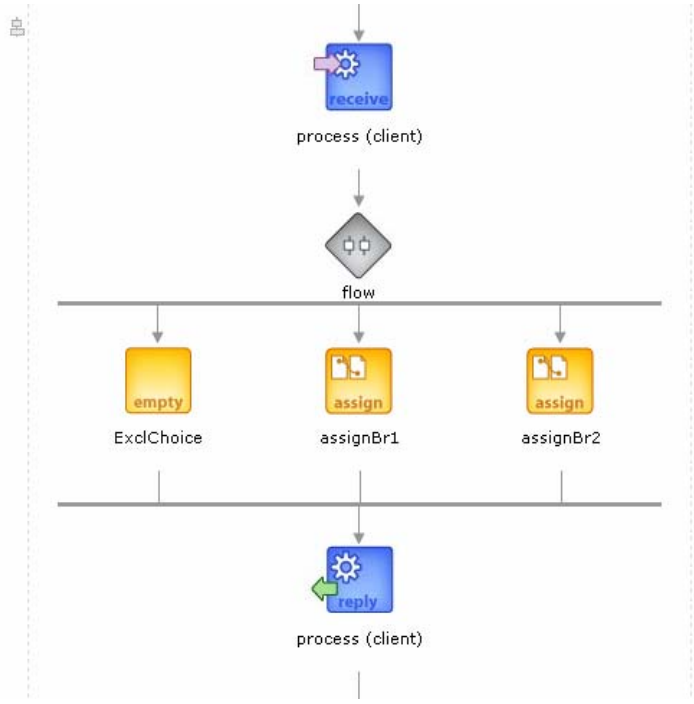


Figure 5 Exclusive choice by means of links and <flow>

The source links associated with activities *assignBr1* and *assignBr2* are supplied with the *transitionCondition* based on which the status of the link is defined. If both links are set to negative, then the corresponding activities are skipped. All available branches are synchronized by the <flow> construct after statuses of all available links have been determined.

The source code corresponding to Figure 5 is shown below:

```
<sequence xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-
process/" name="main">
  <!-- Receive input from requester.
            Note: This maps to operation defined in
TestExclusiveChoice3.wsdl
            -->
  <receive name="receiveInput" partnerLink="client"
portType="tns:TestExclusiveChoice3" operation="process" variable="input"
createInstance="yes"/>
  <!-- Generate reply to synchronous request -->
  <flow xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
name="flow">
```

```
    <links>
      <link name="L1"/>
      <link name="L2"/>
      <!--    <link name="L1s"/>
    <link name="L2s"/> -->
    </links>
    <!--  <sequence name="flow-sequence-1"> -->
    <empty xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-
process/" name="ExclChoice">
      <source linkName="L1"
transitionCondition="bpws:getVariableData(&quot;input&quot;,&quot;payloa
d&quot;,&quot;/tns:TestExclusiveChoice3Request/tns:input&quot;)=
&quot;Car&quot;;"/>
      <source linkName="L2"
transitionCondition="bpws:getVariableData(&quot;input&quot;,&quot;payloa
d&quot;,&quot;/tns:TestExclusiveChoice3Request/tns:input&quot;)=
&quot;Hotel&quot;;"/>
    </empty>
    <!--  </sequence>-->
    <!-- <sequence name="flow-sequence-2"> -->
    <assign name="assignBr1">
      <target linkName="L1"/>
      <!--    <source linkName="L1s"/>-->
      <copy>
        <from expression="&quot;b1&quot;;"></from>
        <to variable="output" part="payload"
query="/tns:TestExclusiveChoice3Response/tns:result2"/>
      </copy>
      <copy>
        <from variable="input" part="payload"
query="/tns:TestExclusiveChoice3Request/tns:input"></from>
        <to variable="output" part="payload"
query="/tns:TestExclusiveChoice3Response/tns:result1"/>
      </copy>
    </assign>
    <!-- </sequence>-->
    <!-- <sequence name="flow-sequence-3">-->
    <assign name="assignBr2">
      <target linkName="L2"/>
      <!--    <source linkName="L2s"/> -->
      <copy>
        <from variable="input" part="payload"
query="/tns:TestExclusiveChoice3Request/tns:input"></from>
        <to variable="output" part="payload"
query="/tns:TestExclusiveChoice3Response/tns:result2"/>
      </copy>
      <copy>
        <from expression="&quot;B2&quot;;"></from>
        <to variable="output" part="payload"
query="/tns:TestExclusiveChoice3Response/tns:result1"/>
      </copy>
    </assign>
    <!--  </sequence> -->
    <!--    <sequence name="flow-sequence-4" joinCondition="L1s OR L2s">
      <target linkName="L1s"/>
      <target linkName="L2s"/>
      <empty name="simpleMerge">
```

```
      </empty>
      <assign name="assign-1">
        <copy>
          <from variable="input" part="payload"
query="/tns:TestExclusiveChoice3Request/tns:input"></from>
          <to variable="output" part="payload"
query="/tns:TestExclusiveChoice3Response/tns:result1"/>
        </copy>
      </assign>
    </sequence> -->
  </flow>
  <reply name="replyOutput" partnerLink="client"
portType="tns:TestExclusiveChoice3" operation="process"
variable="output"/>
</sequence>
```

An audit trail visualizing the execution history of the considered process is shown below:

[2005/07/11 16:43:49] New instance of BPEL process "TestExclusiveChoice3" initiated (# "1220").

<process>

<sequence>

client (process)

[2005/07/11 16:43:49] Received "input" call from partner "client" Less

<input>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<TestExclusiveChoice3Request xmlns="http://acm.org/samples">
<input>Car</input>
</TestExclusiveChoice3Request>
</part>
</input>

<flow>

Empty

[2005/07/11 16:43:49] BPEL "empty" activity is executed.

assignBr2

[2005/07/11 16:43:49] Activity skipped

assignBr1

[2005/07/11 16:43:49] Updated variable "output" Less

<output>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<TestExclusiveChoice3Response xmlns="http://acm.org/samples">
<result1 />
<result2>b1</result2>
</TestExclusiveChoice3Response>
</part>
</output>

[2005/07/11 16:43:49] Updated variable "output" Less

<output>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<TestExclusiveChoice3Response xmlns="http://acm.org/samples">

<result1>Car</result1>
<result2>b1</result2>
</TestExclusiveChoice3Response>
</part>
</output>
</flow>

client

[2005/07/11 16:43:49] Reply to partner "client". More...

</sequence>

CFP5 Simple Merge

Description: A point in the workflow process where two or more alternative branches come together without synchronization. It is an assumption of this pattern that none of the alternative branches is ever executed in parallel (if it is not the case, then see the patterns Multi-Merge and Discriminator).

Oracle BPEL PM supports this pattern directly by means of the <switch> construct or links incorporated in the <flow> activity as it has been described for CFP4. By default the functionality associated with the <switch> construct is a combination of a CFP4 Exclusive Choice and CFP5 Simple Merge. Note that <switch> allows only one branch to be selected, thus none of the alternative branches is ever executed in parallel. As soon as the selected branch finished the execution, the <switch> normally terminates.

Usage of links with disjoint conditions within the <flow> construct permits only one branch to be selected. As soon as the selected branch finished the execution, the <flow> terminates.

CFP6 Multi-Choice

Description: A point in the process, where, based on a decision or control data, a number of branches are chosen and executed as parallel threads.

This pattern can be realized by means of links embedded into the <flow> construct, where activities *Assign-2, Assign-3, Assign-4* are related to each other via links, the source of which is an activity *Assign_1*. The synchronization of multiple branches is done automatically by the <flow> construct, which terminates after the statuses of all links have been determined and no activities that could execute are left. The process model created for this pattern is shown in Figure 6. Based on the string input provided by the client the *transitionCondition*s associated with <assign> activities are evaluated. Thus, if several conditions are satisfied, the correspondent branches become enabled and the <assign> activities execute in parallel. If the client provided an input other than specified in the *transitionCondition*s, all branches are skipped.

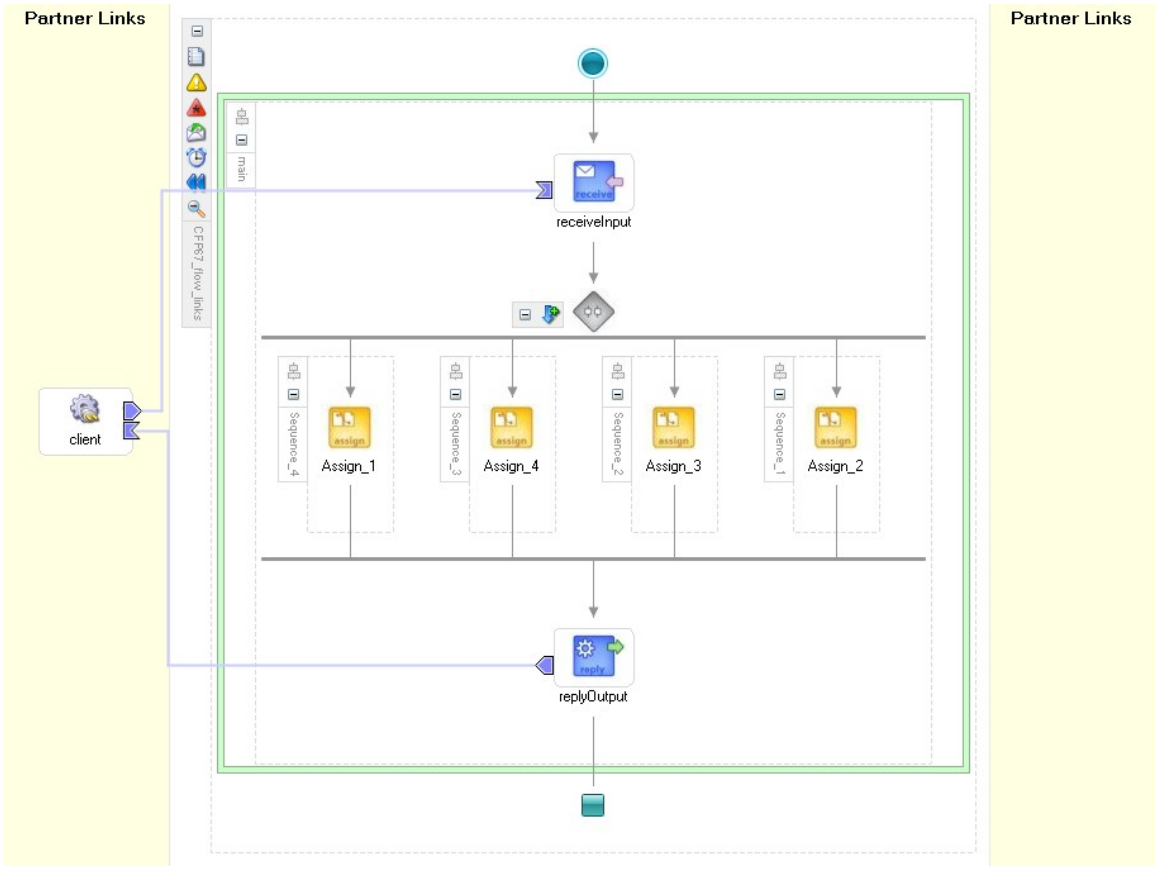


Figure 6 Multi-choice by means of <flow> and links

The source code corresponding to Figure 6 is given below:

```
<process name="CFP67_flow_links" suppressJoinFailure="yes"
targetNamespace="http://xmlns.oracle.com/CFP67_flow_links"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.Xpath20"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:client="http://xmlns.oracle.com/CFP67_flow_links"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.ExtFunc"><!--
================================================================= --><!-
- PARTNERLINKS                                                      --
><!-- List of services participating in this BPEL process
--><!--
================================================================= -->
  <partnerLinks><!--
    The 'client' role represents the requester of this service. It is
    used for callback. The location and correlation information
associated
    with the client role are automatically set using WS-Addressing.
```

```
    -->
    <partnerLink name="client" partnerLinkType="client:CFP67_flow_links"
myRole="CFP67_flow_linksProvider"/>
  </partnerLinks><!--
================================================================= --><!-
- VARIABLES                                                        --
><!-- List of messages and XML documents used within this BPEL process
--><!--
================================================================= -->
  <variables><!-- Reference to the message passed as input during
initiation -->
    <variable name="inputVariable"
messageType="client:CFP67_flow_linksRequestMessage"/><!--
    Reference to the message that will be returned to the requester
    -->
    <variable name="outputVariable"
messageType="client:CFP67_flow_linksResponseMessage"/>
  </variables><!--
================================================================= --><!-
- ORCHESTRATION LOGIC                                              --
><!-- Set of activities coordinating the flow of messages across the
--><!-- services integrated within this business process
--><!--
================================================================= -->
  <sequence name="main"><!-- Receive input from requestor.
    Note: This maps to operation defined in CFP67_flow_links.wsdl
    -->
    <receive name="receiveInput" partnerLink="client"
portType="client:CFP67_flow_links" operation="process"
variable="inputVariable" createInstance="yes"/><!-- Generate reply to
synchronous request -->
    <flow name="Flow_1">
      <links>
        <link name="Link14"/>
        <link name="Link13"/>
        <link name="Link12"/>
      </links>
      <sequence name="Sequence_4">
        <assign name="Assign_1">
          <source linkName="Link12"
transitionCondition="contains('abc',bpws:getVariableData('outputVariable
','payload','/client:CFP67_flow_linksProcessResponse/client:result'))"/>
          <source linkName="Link13"
transitionCondition="contains('bcd',bpws:getVariableData('outputVariable
','payload','/client:CFP67_flow_linksProcessResponse/client:result'))"/>
          <source linkName="Link14"
transitionCondition="contains('cde',bpws:getVariableData('outputVariable
','payload','/client:CFP67_flow_linksProcessResponse/client:result'))"/>
          <copy>
            <from variable="inputVariable" part="payload"
query="/client:CFP67_flow_linksProcessRequest/client:input"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP67_flow_linksProcessResponse/client:result"/>
          </copy>
        </assign>
      </sequence>
      <sequence name="Sequence_3">
```

```
        <target linkName="Link13"/>
        <assign name="Assign_4">
          <copy>
            <from
expression="concat(bpws:getVariableData('outputVariable','payload','/cli
ent:CFP67_flow_linksProcessResponse/client:result'),'4')"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP67_flow_linksProcessResponse/client:result"/>
          </copy>
        </assign>
      </sequence>
      <sequence name="Sequence_2">
        <target linkName="Link12"/>
        <assign name="Assign_3">
          <copy>
            <from
expression="concat(bpws:getVariableData('outputVariable','payload','/cli
ent:CFP67_flow_linksProcessResponse/client:result'),'3')"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP67_flow_linksProcessResponse/client:result"/>
          </copy>
        </assign>
      </sequence>
      <sequence name="Sequence_1">
        <target linkName="Link14"/>
        <assign name="Assign_2">
          <copy>
            <from
expression="concat(bpws:getVariableData('outputVariable','payload','/cli
ent:CFP67_flow_linksProcessResponse/client:result'),'2')"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP67_flow_linksProcessResponse/client:result"/>
          </copy>
        </assign>
      </sequence>
    </flow>
    <reply name="replyOutput" partnerLink="client"
portType="client:CFP67_flow_links" operation="process"
variable="outputVariable"/>
  </sequence>
</process>
```

The audit trail reflecting the execution of the considered process is shown below:

[2005/08/10 11:31:19] New instance of BPEL process "CFP67_flow_links" initiated (# "211").

<process>

<sequence>

receiveInput

[2005/08/10 11:31:19] Received "inputVariable" call from partner "client" less

<inputVariable>

<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">

<CFP67_flow_linksProcessRequest xmlns="http://xmlns.oracle.com/CFP67_flow_links">

<input>a</input>

</CFP67_flow_linksProcessRequest>

</part>
</inputVariable>
<flow>
<sequence>
Assign_1
[2005/08/10 11:31:19] Updated variable "outputVariable" less
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP67_flow_linksProcessResponse xmlns="http://xmlns.oracle.com/CFP67_flow_links">
<result>a</result>
</CFP67_flow_linksProcessResponse>
</part>
</outputVariable>
</sequence>
<sequence>
Skipping
[2005/08/10 11:31:19] Block skipped
</sequence>
<sequence>
Assign_3
[2005/08/10 11:31:19] Updated variable "outputVariable" less
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP67_flow_linksProcessResponse xmlns="http://xmlns.oracle.com/CFP67_flow_links">
<result>a3</result>
</CFP67_flow_linksProcessResponse>
</part>
</outputVariable>
</sequence>
<sequence>
Skipping
[2005/08/10 11:31:19] Block skipped
</sequence>
</flow>
replyOutput
[2005/08/10 11:31:19] Reply to partner "client". less
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP67_flow_linksProcessResponse xmlns="http://xmlns.oracle.com/CFP67_flow_links">
<result>a3</result>
</CFP67_flow_linksProcessResponse>
</part>
</outputVariable>
</sequence>
[2005/08/10 11:31:19] BPEL process instance "211" completed
</process>

CFP7 Synchronizing Merge
Description: A point in the process where multiple paths converge into one single thread. Some of these paths are active (i.e. they are being executed) and some are not. If only one path is active, the activity after the merge is triggered as soon as this path completes. If more than one path is active, synchronization of all active paths needs to take place.

Oracle BPEL PM supports this pattern by means of links as it has been described for the CFP6 Multiple Choice. Links with transition conditions are used to define which branches within the <flow> construct are to be selected. The synchronization of branches is done by the <flow> activity. The <flow> activity will only complete when each of its sub-activities has either completed or has been skipped. The continuation of the process after the synchronizing merge can be placed after the <flow> activity. Alternatively, the continuation of the process can be placed in a separate branch of the <flow>, and the *joinCondition* can be attached to the activity directly following the synchronizing merge.

CFP8 Multi-Merge
Description: A point in a process where two or more branches re-converge without synchronization. If more than one branch gets activated, possibly concurrently, the activity following the merge is started for every action of every incoming branch.

This pattern in Oracle BPEL PM can be implemented by means of an event handler attached to the scope in which multiple branches reside. In every branch enclosed in the <flow> construct, an <invoke> activity should be placed to invoke a synchronous dummy service (denoted by *PartnerLink_1* in Figure 7). The response message produced by this dummy service needs to be processed by an event handler attached to the scope outer of the <flow> construct. As the result, as many branches complete, as many times an activity associated with the event handler will be executed. For some reason, the event handlers attached to the process scope seem to be unable to respond on messages, while the alternative *onMessage* handler within the <pick> construct does this properly. Since the desired behavior hasn't been achieved by means of event handlers, this pattern is considered to be not supported.

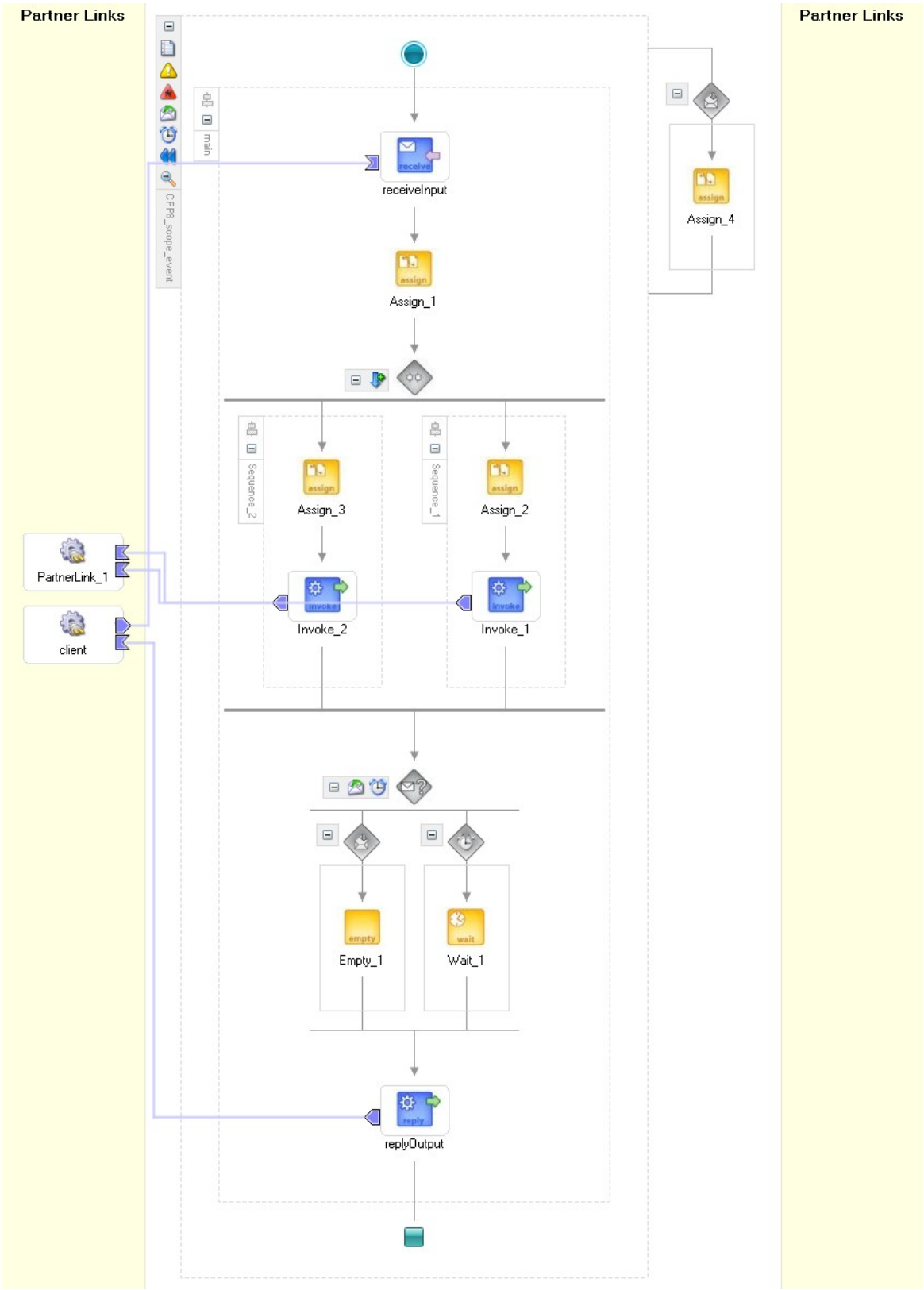


Figure 7 Multi-merge

The code snippets corresponding to Figure 7 are shown below:

```
<process name="CFP8_scope_event"
targetNamespace="http://xmlns.oracle.com/CFP8_scope_event"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.Xpath20"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:ns1="http://xmlns.oracle.com/dummy"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns2="http://xmlns.oracle.com/AsyncDummy"
xmlns:client="http://xmlns.oracle.com/CFP8_scope_event"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.ExtFunc"><!--
================================================================= --><!-
- PARTNERLINKS                                                      --
><!-- List of services participating in this BPEL process
--><!--
================================================================= -->
  <partnerLinks><!--
    The 'client' role represents the requester of this service. It is
    used for callback. The location and correlation information
associated
    with the client role are automatically set using WS-Addressing.
    -->
    <partnerLink name="client" partnerLinkType="client:CFP8_scope_event"
myRole="CFP8_scope_eventProvider"/>
    <partnerLink myRole="AsyncDummyRequester" name="PartnerLink_1"
partnerRole="AsyncDummyProvider" partnerLinkType="ns2:AsyncDummy"/>
  </partnerLinks><!--
================================================================= --><!-
- VARIABLES                                                         --
><!-- List of messages and XML documents used within this BPEL process
--><!--
================================================================= -->
  <variables><!-- Reference to the message passed as input during
initiation --><!--
    Reference to the message that will be returned to the requester
    -->
    <variable name="inputVariable"
messageType="client:CFP8_scope_eventRequestMessage"/>
    <variable name="outputVariable"
messageType="client:CFP8_scope_eventResponseMessage"/>
    <variable name="OnMessage_process_DummyVariable"
messageType="client:CFP8_scope_eventRequestMessage"/>
    <variable name="StringVariable"
messageType="ns2:AsyncDummyRequestMessage"/>
    <variable name="OnMessage_onResult_InputVariable"
messageType="ns2:AsyncDummyResponseMessage"/>
  </variables>
  <eventHandlers>
    <onMessage portType="ns2:AsyncDummyCallback" operation="onResult"
variable="OnMessage_onResult_InputVariable" partnerLink="PartnerLink_1">
```

```
      <assign name="Assign_4">
        <copy>
          <from
expression="concat(bpws:getVariableData('outputVariable','payload','/cli
ent:CFP8_scope_eventProcessResponse/client:result'),'4')"/>
          <to variable="outputVariable" part="payload"
query="/client:CFP8_scope_eventProcessResponse/client:result"/>
        </copy>
      </assign>
    </onMessage>
  </eventHandlers><!--
================================================================= --><!-
- ORCHESTRATION LOGIC                                               --
><!-- Set of activities coordinating the flow of messages across the
--><!-- services integrated within this business process
--><!--
================================================================= -->
  <sequence name="main"><!-- Receive input from requestor.
    Note: This maps to operation defined in CFP8_scope_event.wsdl
    -->
    <receive name="receiveInput" partnerLink="client"
portType="client:CFP8_scope_event" operation="process"
variable="inputVariable" createInstance="yes"/><!-- Generate reply to
synchronous request -->
    <assign name="Assign_1">
      <copy>
        <from variable="inputVariable" part="payload"
query="/client:CFP8_scope_eventProcessRequest/client:input"/>
        <to variable="outputVariable" part="payload"
query="/client:CFP8_scope_eventProcessResponse/client:result"/>
      </copy>
    </assign>
    <flow name="Flow_1">
      <sequence name="Sequence_2">
        <assign name="Assign_3">
          <copy>
            <from
expression="concat(bpws:getVariableData('outputVariable','payload','/cli
ent:CFP8_scope_eventProcessResponse/client:result'),'3')"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP8_scope_eventProcessResponse/client:result"/>
          </copy>
          <copy>
            <from expression="'3'"/>
            <to variable="StringVariable" part="payload"
query="/ns2:AsyncDummyProcessRequest/ns2:input"/>
          </copy>
        </assign>
        <invoke name="Invoke_2" partnerLink="PartnerLink_1"
portType="ns2:AsyncDummy" operation="initiate"
inputVariable="StringVariable"/>
      </sequence>
      <sequence name="Sequence_1">
        <assign name="Assign_2">
          <copy>
```

```
            <from
expression="concat(bpws:getVariableData('outputVariable','payload','/cli
ent:CFP8_scope_eventProcessResponse/client:result'),'2')"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP8_scope_eventProcessResponse/client:result"/>
          </copy>
          <copy>
            <from expression="'2'"/>
            <to variable="StringVariable" part="payload"
query="/ns2:AsyncDummyProcessRequest/ns2:input"/>
          </copy>
        </assign>
        <invoke name="Invoke_1" partnerLink="PartnerLink_1"
portType="ns2:AsyncDummy" operation="initiate"
inputVariable="StringVariable"/>
      </sequence>
    </flow>
    <pick name="Pick_1">
      <onMessage portType="ns2:AsyncDummyCallback" operation="onResult"
variable="OnMessage_onResult_InputVariable" partnerLink="PartnerLink_1">
        <empty name="Empty_1"/>
      </onMessage>
      <onAlarm for="'PT10S'">
        <wait name="Wait_1" for="'PT30S'"/>
      </onAlarm>
    </pick>
    <reply name="replyOutput" partnerLink="client"
portType="client:CFP8_scope_event" operation="process"
variable="outputVariable"/>
  </sequence>
</process>
```

The content of the wsdl file is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="CFP8_scope_event"
             targetNamespace="http://xmlns.oracle.com/CFP8_scope_event"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:client="http://xmlns.oracle.com/CFP8_scope_event"
             xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-
link/">

    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    TYPE DEFINITION - List of services participating in this BPEL
process
    The default output of the BPEL designer uses strings as input and
    output to the BPEL Process. But you can define or import any XML
    Schema type and us them as part of the message types.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <types>
        <schema attributeFormDefault="qualified"
            elementFormDefault="qualified"
            targetNamespace="http://xmlns.oracle.com/CFP8_scope_event"
            xmlns="http://www.w3.org/2001/XMLSchema">
```

```
            <element name="CFP8_scope_eventProcessRequest">
                <complexType>
                    <sequence>
                        <element name="input" type="string"/>
                    </sequence>
                </complexType>
            </element>
            <element name="CFP8_scope_eventProcessResponse">
                <complexType>
                    <sequence>
                        <element name="result" type="string"/>
                    </sequence>
                </complexType>
            </element>
        </schema>
    </types>

    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    MESSAGE TYPE DEFINITION - Definition of the message types used as
    part of the port type defintions

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <message name="CFP8_scope_eventRequestMessage">
        <part name="payload"
element="client:CFP8_scope_eventProcessRequest"/>
    </message>
    <message name="CFP8_scope_eventResponseMessage">
        <part name="payload"
element="client:CFP8_scope_eventProcessResponse"/>
    </message>

    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    PORT TYPE DEFINITION - A port type groups a set of operations into
    a logical service unit.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>

    <!-- portType implemented by the CFP8_scope_event BPEL process -->
    <portType name="CFP8_scope_event">
        <operation name="process">
            <input  message="client:CFP8_scope_eventRequestMessage" />
            <output message="client:CFP8_scope_eventResponseMessage"/>
        </operation>
    </portType>

    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    PARTNER LINK TYPE DEFINITION

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <plnk:partnerLinkType name="CFP8_scope_event">
        <plnk:role name="CFP8_scope_eventProvider">
```

```
            <plnk:portType name="client:CFP8_scope_event"/>
        </plnk:role>
    </plnk:partnerLinkType>
</definitions>
```

An audit trail demonstrating the execution history of the considered process model (based on <pick> construct) is shown below:

[2005/08/10 14:40:27] New instance of BPEL process "CFP8_scope_event" initiated (# "231").

<process>

<sequence>

receiveInput

[2005/08/10 14:40:27] Received "inputVariable" call from partner "client" less

```
<inputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP8_scope_eventProcessRequest xmlns="http://xmlns.oracle.com/CFP8_scope_event">
<input>a</input>
    </CFP8_scope_eventProcessRequest>
    </part>
    </inputVariable>
```

Assign_1

[2005/08/10 14:40:27] Updated variable "outputVariable" less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP8_scope_eventProcessResponse xmlns="http://xmlns.oracle.com/CFP8_scope_event">
<result>a</result>
    </CFP8_scope_eventProcessResponse>
    </part>
    </outputVariable>
```

<flow>

<sequence>

Assign_2

[2005/08/10 14:40:27] Updated variable "outputVariable" less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP8_scope_eventProcessResponse xmlns="http://xmlns.oracle.com/CFP8_scope_event">
<result>a2</result>
    </CFP8_scope_eventProcessResponse>
    </part>
    </outputVariable>
```

[2005/08/10 14:40:27] Updated variable "StringVariable" less

```
<StringVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<AsyncDummyProcessRequest xmlns="http://xmlns.oracle.com/AsyncDummy">
<input>2</input>
    </AsyncDummyProcessRequest>
    </part>
    </StringVariable>
```

Invoke_1

[2005/08/10 14:40:27] Invoked 1-way operation "initiate" on partner "PartnerLink_1".

less

<StringVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<AsyncDummyProcessRequest xmlns="http://xmlns.oracle.com/AsyncDummy">
<input>3</input>
</AsyncDummyProcessRequest>
</part>
</StringVariable>

</sequence>

<sequence>

Assign_3

[2005/08/10 14:40:27] Updated variable "outputVariable" less

<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP8_scope_eventProcessResponse xmlns="http://xmlns.oracle.com/CFP8_scope_event">
<result>a23</result>
</CFP8_scope_eventProcessResponse>
</part>
</outputVariable>

[2005/08/10 14:40:27] Updated variable "StringVariable" less

<StringVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<AsyncDummyProcessRequest xmlns="http://xmlns.oracle.com/AsyncDummy">
<input>3</input>
</AsyncDummyProcessRequest>
</part>
</StringVariable>

Invoke_2

[2005/08/10 14:40:27] Invoked 1-way operation "initiate" on partner "PartnerLink_1".

less

<StringVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<AsyncDummyProcessRequest xmlns="http://xmlns.oracle.com/AsyncDummy">
<input>3</input>
</AsyncDummyProcessRequest>
</part>
</StringVariable>

</sequence>

</flow>

<pick>

onMessage (77)

[2005/08/10 14:40:27] Waiting for message from "PartnerLink_1", operation is "onResult".

[2005/08/10 14:40:29] Received "onResult" callback from partner "PartnerLink_1" less

<OnMessage_onResult_InputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">

<AsyncDummyProcessResponse xmlns="http://xmlns.oracle.com/AsyncDummy">
<result>3</result>
</AsyncDummyProcessResponse>
</part>
</OnMessage_onResult_InputVariable>

onAlarm (80) (cancelled)

[2005/08/10 14:40:27] Alarm started. Alarm will go off at time "2005/08/10 14:40:37".

[2005/08/10 14:40:29] BPEL "onAlarm" cancelled before being triggered.

<onMessage>

Empty

[2005/08/10 14:40:29] BPEL "empty" activity is executed.

</onMessage>

</pick>

replyOutput

[2005/08/10 14:40:29] Reply to partner "client". less

<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP8_scope_eventProcessResponse xmlns="http://xmlns.oracle.com/CFP8_scope_event">
<result>a23</result>
</CFP8_scope_eventProcessResponse>
</part>
</outputVariable>

</sequence>

[2005/08/10 14:40:29] BPEL process instance "231" completed

</process>

An audit trail of the invoked dummy service is shown below:

[2005/08/10 14:40:30] New instance of BPEL process "AsyncDummy" initiated (# "233").

<process>

<sequence>

receiveInput

[2005/08/10 14:40:30] Received "inputVariable" call from partner "client" less

<inputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<AsyncDummyProcessRequest xmlns="http://xmlns.oracle.com/AsyncDummy">
<input>3</input>
</AsyncDummyProcessRequest>
</part>
</inputVariable>

Assign_1

[2005/08/10 14:40:30] Updated variable "outputVariable" less

<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<AsyncDummyProcessResponse xmlns="http://xmlns.oracle.com/AsyncDummy">
<result>3</result>

```
</AsyncDummyProcessResponse>
</part>
</outputVariable>
callbackClient
[2005/08/10 14:40:31]  Invoked 1-way operation "onResult" on partner "client". less
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<AsyncDummyProcessResponse xmlns="http://xmlns.oracle.com/AsyncDummy">
<result>3</result>
</AsyncDummyProcessResponse>
</part>
</outputVariable>
</sequence>
[2005/08/10 14:40:31]  BPEL process instance "233" completed
</process>
```

CFP9 Discriminator

Description: A point in the workflow process that waits for one of the incoming branches to complete before activating the subsequent activity. From that moment on it waits for all remaining branches to complete and 'ignores' them. Once all incoming branches have been triggered, it resets itself so that it can be triggered again (which is important otherwise it could not really be used in the context of a loop).

Oracle BPEL PM does not support this pattern directly since according to the BPEL specification *joinCondition* is evaluated after all branches were triggered, but not after the first branch has completed.

CFP10 Arbitrary Cycles

Description: A point where a portion of the process (including one or more activities and connectors) needs to be visited repeatedly without imposing restrictions on the number, location, and nesting of these points.

Oracle BPEL PM does not support this pattern directly. The only possibility to define loops offered by Oracle BPEL PM is by using the <while> construct (see Figure 8), however it allows only one input and output end-points, and does not permit arbitrary jumps into the body of a loop.

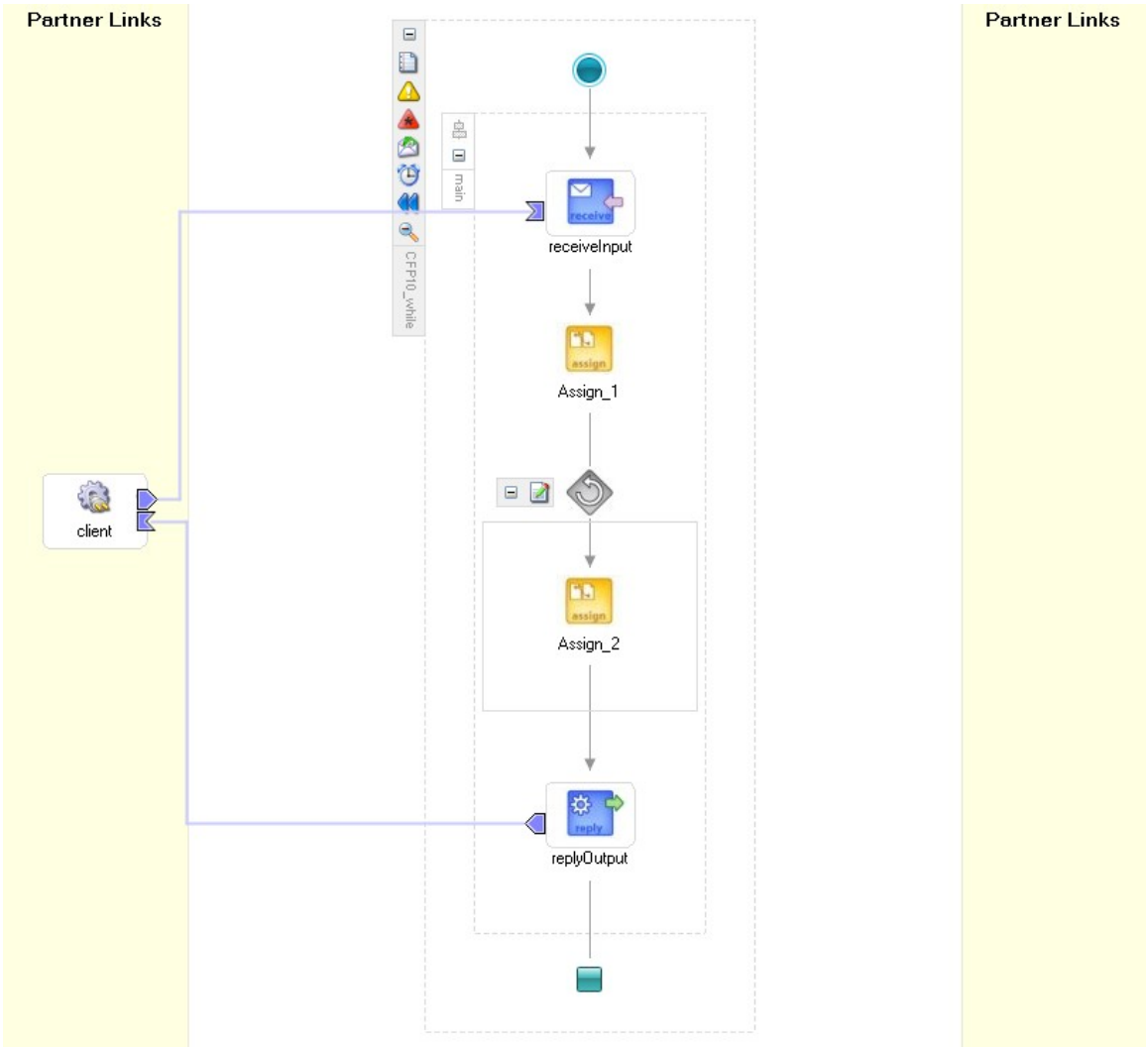


Figure 8 The <while> construct

The code snippets corresponding to Figure 8 are shown below:

```
<process name="CFP10_while"
targetNamespace="http://xmlns.oracle.com/CFP10_while"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.Xpath20" xmlns:ns1="http://www.w3.org/2001/XMLSchema"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:client="http://xmlns.oracle.com/CFP10_while"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.ExtFunc"><!--
================================================================= --><!-
- PARTNERLINKS                                                      --
><!-- List of services participating in this BPEL process
```

```
--><!--
================================================================= -->
  <partnerLinks><!--
    The 'client' role represents the requester of this service. It is
    used for callback. The location and correlation information
associated
    with the client role are automatically set using WS-Addressing.
    -->
    <partnerLink name="client" partnerLinkType="client:CFP10_while"
myRole="CFP10_whileProvider"/>
  </partnerLinks><!--
================================================================= --><!-
- VARIABLES                                                        --
><!-- List of messages and XML documents used within this BPEL process
--><!--
================================================================= -->
  <variables><!-- Reference to the message passed as input during
initiation --><!--
    Reference to the message that will be returned to the requester
    -->
    <variable name="inputVariable"
messageType="client:CFP10_whileRequestMessage"/>
    <variable name="outputVariable"
messageType="client:CFP10_whileResponseMessage"/>
    <variable name="LoopCounter" type="ns1:integer"/>
  </variables><!--
================================================================= --><!-
- ORCHESTRATION LOGIC                                              --
><!-- Set of activities coordinating the flow of messages across the
--><!-- services integrated within this business process
--><!--
================================================================= -->
  <sequence name="main"><!-- Receive input from requestor.
    Note: This maps to operation defined in CFP10_while.wsdl
    -->
    <receive name="receiveInput" partnerLink="client"
portType="client:CFP10_while" operation="process"
variable="inputVariable" createInstance="yes"/><!-- Generate reply to
synchronous request -->
    <assign name="Assign_1">
      <copy>
        <from expression="3"/>
        <to variable="LoopCounter"/>
      </copy>
      <copy>
        <from variable="inputVariable" part="payload"
query="/client:CFP10_whileProcessRequest/client:input"/>
        <to variable="outputVariable" part="payload"
query="/client:CFP10_whileProcessResponse/client:result"/>
      </copy>
    </assign>
    <while name="While_1" condition="bpws:getVariableData('LoopCounter')
&gt; 0">
      <assign name="Assign_2">
        <copy>
          <from
expression="concat(bpws:getVariableData('outputVariable','payload','/cli
```

```
ent:CFP10_whileProcessResponse/client:result'),string(bpws:getVariableDa
ta('LoopCounter')))"/>
          <to variable="outputVariable" part="payload"
query="/client:CFP10_whileProcessResponse/client:result"/>
        </copy>
        <copy>
          <from expression="bpws:getVariableData('LoopCounter') - 1"/>
          <to variable="LoopCounter"/>
        </copy>
      </assign>
    </while>
    <reply name="replyOutput" partnerLink="client"
portType="client:CFP10_while" operation="process"
variable="outputVariable"/>
  </sequence>
</process>
```

The wsdl code is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="CFP10_while"
             targetNamespace="http://xmlns.oracle.com/CFP10_while"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:client="http://xmlns.oracle.com/CFP10_while"
             xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-
link/">

    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    TYPE DEFINITION - List of services participating in this BPEL
process
    The default output of the BPEL designer uses strings as input and
    output to the BPEL Process. But you can define or import any XML
    Schema type and us them as part of the message types.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <types>
        <schema attributeFormDefault="qualified"
            elementFormDefault="qualified"
            targetNamespace="http://xmlns.oracle.com/CFP10_while"
            xmlns="http://www.w3.org/2001/XMLSchema">
            <element name="CFP10_whileProcessRequest">
                <complexType>
                    <sequence>
                        <element name="input" type="string"/>
                    </sequence>
                </complexType>
            </element>
            <element name="CFP10_whileProcessResponse">
                <complexType>
                    <sequence>
                        <element name="result" type="string"/>
                    </sequence>
                </complexType>
            </element>
        </schema>
```

```
    </types>
    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    MESSAGE TYPE DEFINITION - Definition of the message types used as
    part of the port type defintions
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <message name="CFP10_whileRequestMessage">
        <part name="payload"
element="client:CFP10_whileProcessRequest"/>
    </message>
    <message name="CFP10_whileResponseMessage">
        <part name="payload"
element="client:CFP10_whileProcessResponse"/>
    </message>
    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    PORT TYPE DEFINITION - A port type groups a set of operations into
    a logical service unit.
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <!-- portType implemented by the CFP10_while BPEL process -->
    <portType name="CFP10_while">
        <operation name="process">
            <input  message="client:CFP10_whileRequestMessage" />
            <output message="client:CFP10_whileResponseMessage"/>
        </operation>
    </portType>
    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    PARTNER LINK TYPE DEFINITION
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <plnk:partnerLinkType name="CFP10_while">
        <plnk:role name="CFP10_whileProvider">
            <plnk:portType name="client:CFP10_while"/>
        </plnk:role>
    </plnk:partnerLinkType>
</definitions>
```

The audit trail reflecting the execution of the considered process is shown below:

[2005/08/10 13:45:03] New instance of BPEL process "CFP10_while" initiated (# "215").

<process>

<sequence>

receiveInput

[2005/08/10 13:45:03] Received "inputVariable" call from partner "client" less

```
<inputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP10_whileProcessRequest xmlns="http://xmlns.oracle.com/CFP10_while">
<input>a</input>
    </CFP10_whileProcessRequest>
    </part>
    </inputVariable>
```

Assign_1

[2005/08/10 13:45:03] Updated variable "LoopCounter" less

```
<LoopCounter>3</LoopCounter>
```

[2005/08/10 13:45:03] Updated variable "outputVariable" less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP10_whileProcessResponse xmlns="http://xmlns.oracle.com/CFP10_while">
<result>a</result>
</CFP10_whileProcessResponse>
</part>
</outputVariable>
```

<while>

Assign_2

[2005/08/10 13:45:03] Updated variable "outputVariable" less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP10_whileProcessResponse xmlns="http://xmlns.oracle.com/CFP10_while">
<result>a3</result>
</CFP10_whileProcessResponse>
</part>
</outputVariable>
```

[2005/08/10 13:45:03] Updated variable "LoopCounter" less

```
<LoopCounter>2</LoopCounter>
```

Assign_2

[2005/08/10 13:45:03] Updated variable "outputVariable" less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP10_whileProcessResponse xmlns="http://xmlns.oracle.com/CFP10_while">
<result>a32</result>
</CFP10_whileProcessResponse>
</part>
</outputVariable>
```

[2005/08/10 13:45:03] Updated variable "LoopCounter" less

```
<LoopCounter>1</LoopCounter>
```

Assign_2

[2005/08/10 13:45:03] Updated variable "outputVariable" less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP10_whileProcessResponse xmlns="http://xmlns.oracle.com/CFP10_while">
<result>a321</result>
</CFP10_whileProcessResponse>
</part>
</outputVariable>
```

[2005/08/10 13:45:03] Updated variable "LoopCounter" less

```
<LoopCounter>0</LoopCounter>
```

</while>

replyOutput

[2005/08/10 13:45:03] Reply to partner "client". less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP10_whileProcessResponse xmlns="http://xmlns.oracle.com/CFP10_while">
<result>a321</result>
      </CFP10_whileProcessResponse>
      </part>
      </outputVariable>
      </sequence>
      [2005/08/10 13:45:03]   BPEL process instance "215" completed
      </process>
```

CFP11 Implicit Termination
Description: A given sub-process is terminated when there is nothing left to do, i.e., termination does not require an explicit termination activity.

Oracle BPEL PM supports this pattern directly by the <flow> construct, which terminates when no activities within its body can be triggered and executed any more.

CFP12 MI without Synchronization
Description: Within the context of a single case multiple instances of an activity may be created, i.e. there is a facility for spawning of new threads of control, all of them independent of each other. The instances might be created consecutively, but they will be able to run in parallel, which distinguishes this pattern from the pattern for Arbitrary Cycles.

Oracle BPEL PM supports this pattern by placing an <invoke> construct within the body of a <while> loop (see Figure 9).

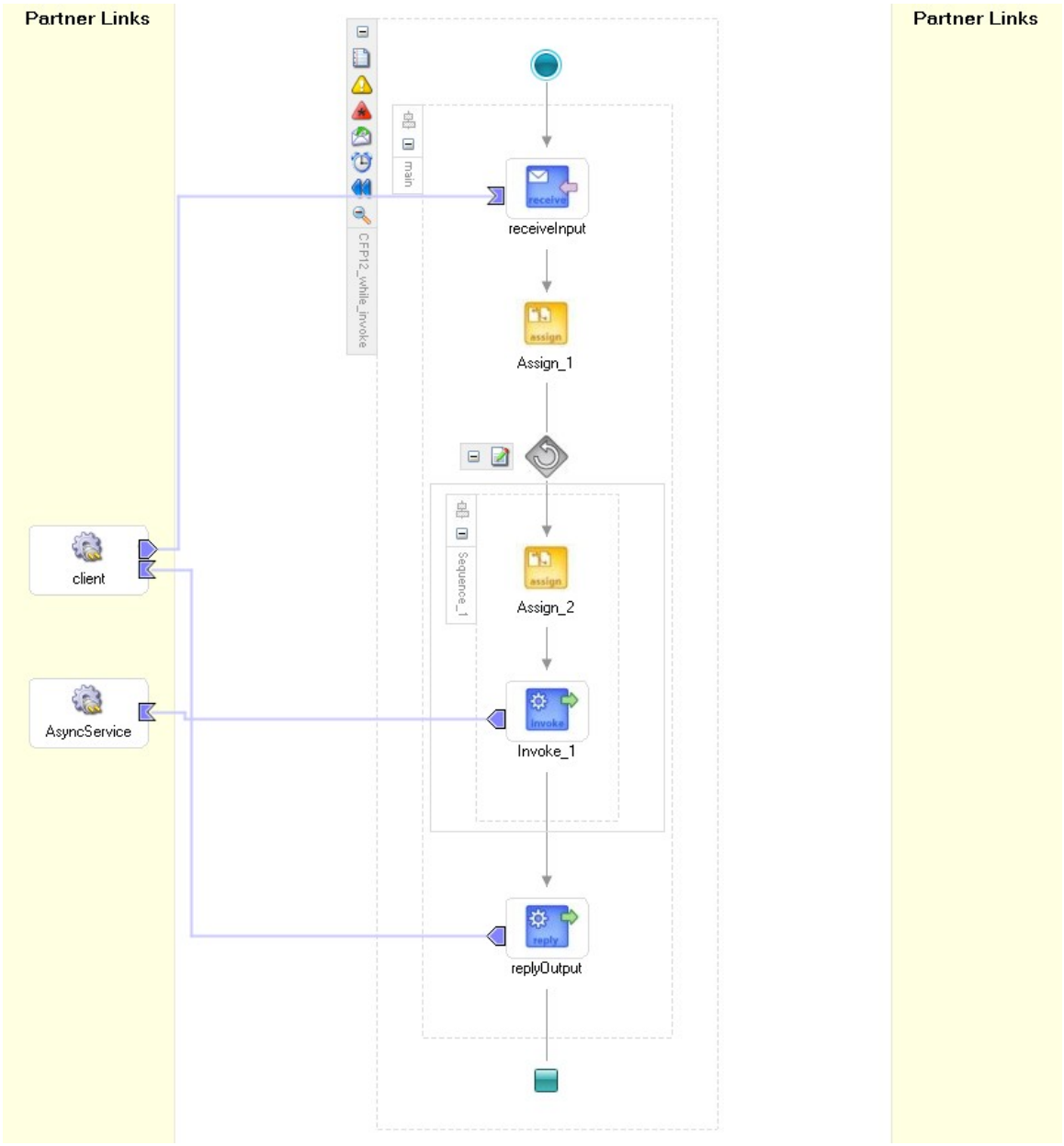


Figure 9 MI instances without synchronization

The code snippets corresponding to Figure 9 are shown below:

```
<process name="CFP12_while_invoke"
targetNamespace="http://xmlns.oracle.com/CFP12_while_invoke"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.Xpath20"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:ns1="http://www.w3.org/2001/XMLSchema"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns2="http://xmlns.oracle.com/AsyncDummy"
xmlns:client="http://xmlns.oracle.com/CFP12_while_invoke"
```

```
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.ExtFunc"><!--
================================================================= --><!-
- PARTNERLINKS                                                      --
><!-- List of services participating in this BPEL process
--><!--
================================================================= -->
  <partnerLinks><!--
    The 'client' role represents the requester of this service. It is
    used for callback. The location and correlation information
associated
    with the client role are automatically set using WS-Addressing.
    -->
    <partnerLink name="client"
partnerLinkType="client:CFP12_while_invoke"
myRole="CFP12_while_invokeProvider"/>
    <partnerLink myRole="AsyncDummyRequester" name="AsyncService"
partnerRole="AsyncDummyProvider" partnerLinkType="ns2:AsyncDummy"/>
  </partnerLinks><!--
================================================================= --><!-
- VARIABLES                                                         --
><!-- List of messages and XML documents used within this BPEL process
--><!--
================================================================= -->
  <variables><!-- Reference to the message passed as input during
initiation --><!--
    Reference to the message that will be returned to the requester
    -->
    <variable name="inputVariable"
messageType="client:CFP12_while_invokeRequestMessage"/>
    <variable name="outputVariable"
messageType="client:CFP12_while_invokeResponseMessage"/>
    <variable name="LoopCounter" type="ns1:integer"/>
    <variable name="StringVariable"
messageType="ns2:AsyncDummyRequestMessage"/>
  </variables><!--
================================================================= --><!-
- ORCHESTRATION LOGIC                                               --
><!-- Set of activities coordinating the flow of messages across the
--><!-- services integrated within this business process
--><!--
================================================================= -->
  <sequence name="main"><!-- Receive input from requestor.
    Note: This maps to operation defined in CFP12_while_invoke.wsdl
    -->
    <receive name="receiveInput" partnerLink="client"
portType="client:CFP12_while_invoke" operation="process"
variable="inputVariable" createInstance="yes"/><!-- Generate reply to
synchronous request -->
    <assign name="Assign_1">
      <copy>
        <from expression="3"/>
        <to variable="LoopCounter"/>
      </copy>
      <copy>
```

```
        <from variable="inputVariable" part="payload"
query="/client:CFP12_while_invokeProcessRequest/client:input"/>
        <to variable="outputVariable" part="payload"
query="/client:CFP12_while_invokeProcessResponse/client:result"/>
      </copy>
      <copy>
        <from variable="inputVariable" part="payload"
query="/client:CFP12_while_invokeProcessRequest/client:input"/>
        <to variable="StringVariable" part="payload"
query="/ns2:AsyncDummyProcessRequest/ns2:input"/>
      </copy>
    </assign>
    <while name="While_1" condition="bpws:getVariableData('LoopCounter')
&gt; 0">
      <sequence name="Sequence_1">
        <assign name="Assign_2">
          <copy>
            <from expression="bpws:getVariableData('LoopCounter') - 1"/>
            <to variable="LoopCounter"/>
          </copy>
          <copy>
            <from
expression="concat(bpws:getVariableData('StringVariable','payload','/ns2
:AsyncDummyProcessRequest/ns2:input'), '.')"/>
            <to variable="StringVariable" part="payload"
query="/ns2:AsyncDummyProcessRequest/ns2:input"/>
          </copy>
        </assign>
        <invoke name="Invoke_1" partnerLink="AsyncService"
portType="ns2:AsyncDummy" operation="initiate"
inputVariable="StringVariable"/>
      </sequence>
    </while>
    <reply name="replyOutput" partnerLink="client"
portType="client:CFP12_while_invoke" operation="process"
variable="outputVariable"/>
  </sequence>
</process>
```

The audit trail reflecting the execution of the considered process is shown below:

[2005/08/10 14:17:12] New instance of BPEL process "CFP12_while_invoke" initiated (# "221").

<process>

<sequence>

receiveInput

[2005/08/10 14:17:12] Received "inputVariable" call from partner "client" less

<inputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP12_while_invokeProcessRequest xmlns="http://xmlns.oracle.com/CFP12_while_invoke">
<input>a</input>
</CFP12_while_invokeProcessRequest>
</part>
</inputVariable>

Assign_1

[2005/08/10 14:17:12] Updated variable "LoopCounter" less

<LoopCounter>3</LoopCounter>

[2005/08/10 14:17:12] Updated variable "outputVariable" less

<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP12_while_invokeProcessResponse xmlns="http://xmlns.oracle.com/CFP12_while_invoke">
<result>a</result>
</CFP12_while_invokeProcessResponse>
</part>
</outputVariable>

[2005/08/10 14:17:12] Updated variable "StringVariable" less

<StringVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<AsyncDummyProcessRequest xmlns="http://xmlns.oracle.com/AsyncDummy">
<input>a</input>
</AsyncDummyProcessRequest>
</part>
</StringVariable>

<while>

<sequence>

Assign_2

[2005/08/10 14:17:12] Updated variable "LoopCounter" less

<LoopCounter>2</LoopCounter>

[2005/08/10 14:17:12] Updated variable "StringVariable" less

<StringVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<AsyncDummyProcessRequest xmlns="http://xmlns.oracle.com/AsyncDummy">
<input>a.</input>
</AsyncDummyProcessRequest>
</part>
</StringVariable>

Invoke_1

[2005/08/10 14:17:13] Invoked 1-way operation "initiate" on partner "AsyncService". less

<StringVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<AsyncDummyProcessRequest xmlns="http://xmlns.oracle.com/AsyncDummy">
<input>a.</input>
</AsyncDummyProcessRequest>
</part>
</StringVariable>

</sequence>

<sequence>

Assign_2

[2005/08/10 14:17:13] Updated variable "LoopCounter" less

<LoopCounter>1</LoopCounter>

[2005/08/10 14:17:13] Updated variable "StringVariable" less

<StringVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<AsyncDummyProcessRequest xmlns="http://xmlns.oracle.com/AsyncDummy">
<input>a..</input>
</AsyncDummyProcessRequest>
</part>
</StringVariable>

Invoke_1

[2005/08/10 14:17:13] Invoked 1-way operation "initiate" on partner "AsyncService". less

<StringVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<AsyncDummyProcessRequest xmlns="http://xmlns.oracle.com/AsyncDummy">
<input>a..</input>
</AsyncDummyProcessRequest>
</part>
</StringVariable>
</sequence>
<sequence>

Assign_2

[2005/08/10 14:17:13] Updated variable "LoopCounter" less

<LoopCounter>0</LoopCounter>

[2005/08/10 14:17:13] Updated variable "StringVariable" less

<StringVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<AsyncDummyProcessRequest xmlns="http://xmlns.oracle.com/AsyncDummy">
<input>a...</input>
</AsyncDummyProcessRequest>
</part>
</StringVariable>

Invoke_1

[2005/08/10 14:17:13] Invoked 1-way operation "initiate" on partner "AsyncService". less

<StringVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<AsyncDummyProcessRequest xmlns="http://xmlns.oracle.com/AsyncDummy">
<input>a...</input>
</AsyncDummyProcessRequest>
</part>
</StringVariable>
</sequence>
</while>

replyOutput

[2005/08/10 14:17:13] Reply to partner "client". less

<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP12_while_invokeProcessResponse xmlns="http://xmlns.oracle.com/CFP12_while_invoke">
<result>a</result>

</CFP12_while_invokeProcessResponse>
</part>
</outputVariable>
</sequence>
[2005/08/10 14:17:13] BPEL process instance "221" completed
</process>

In the <while> loop an asynchronous AsyncDummy service is invoked, the execution history of which is shown below:

[2005/08/10 14:17:15] New instance of BPEL process "AsyncDummy" initiated (# "224").
<process>
<sequence>

receiveInput

[2005/08/10 14:17:15] Received "inputVariable" call from partner "client" less

```
<inputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<AsyncDummyProcessRequest xmlns="http://xmlns.oracle.com/AsyncDummy">
<input>a..</input>
</AsyncDummyProcessRequest>
</part>
</inputVariable>
```

Assign_1

[2005/08/10 14:17:15] Updated variable "outputVariable" less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<AsyncDummyProcessResponse xmlns="http://xmlns.oracle.com/AsyncDummy">
<result>a..</result>
</AsyncDummyProcessResponse>
</part>
</outputVariable>
```

callbackClient

[2005/08/10 14:17:15] Invoked 1-way operation "onResult" on partner "client". less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<AsyncDummyProcessResponse xmlns="http://xmlns.oracle.com/AsyncDummy">
<result>a..</result>
</AsyncDummyProcessResponse>
</part>
</outputVariable>
```

</sequence>
[2005/08/10 14:17:15] BPEL process instance "224" completed
</process>

The wsdl code associated with the considered process is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<definitions name="CFP12_while_invoke"

targetNamespace="http://xmlns.oracle.com/CFP12_while_invoke"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:client="http://xmlns.oracle.com/CFP12_while_invoke"
             xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-
link/">
    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    TYPE DEFINITION - List of services participating in this BPEL
process
    The default output of the BPEL designer uses strings as input and
    output to the BPEL Process. But you can define or import any XML
    Schema type and us them as part of the message types.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <types>
        <schema attributeFormDefault="qualified"
            elementFormDefault="qualified"
            targetNamespace="http://xmlns.oracle.com/CFP12_while_invoke"
            xmlns="http://www.w3.org/2001/XMLSchema">
            <element name="CFP12_while_invokeProcessRequest">
                <complexType>
                    <sequence>
                        <element name="input" type="string"/>
                    </sequence>
                </complexType>
            </element>
            <element name="CFP12_while_invokeProcessResponse">
                <complexType>
                    <sequence>
                        <element name="result" type="string"/>
                    </sequence>
                </complexType>
            </element>
        </schema>
    </types>
    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    MESSAGE TYPE DEFINITION - Definition of the message types used as
    part of the port type defintions
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <message name="CFP12_while_invokeRequestMessage">
        <part name="payload"
element="client:CFP12_while_invokeProcessRequest"/>
    </message>
    <message name="CFP12_while_invokeResponseMessage">
        <part name="payload"
element="client:CFP12_while_invokeProcessResponse"/>
    </message>
    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    PORT TYPE DEFINITION - A port type groups a set of operations into
    a logical service unit.
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <!-- portType implemented by the CFP12_while_invoke BPEL process -->
    <portType name="CFP12_while_invoke">
        <operation name="process">
            <input  message="client:CFP12_while_invokeRequestMessage" />
            <output message="client:CFP12_while_invokeResponseMessage"/>
        </operation>
    </portType>
    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    PARTNER LINK TYPE DEFINITION
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <plnk:partnerLinkType name="CFP12_while_invoke">
        <plnk:role name="CFP12_while_invokeProvider">
            <plnk:portType name="client:CFP12_while_invoke"/>
        </plnk:role>
    </plnk:partnerLinkType>
</definitions>
```

The invoked AsyncDummy process must have an attribute *createInstance* set to "yes".

CFP13-CFP15 MI with Synchronization
Description: A point in a workflow where a number of instances of a given activity is initiated and these instances are later synchronized, before proceeding with the rest of the process.

In **CFP13** the number of instances to be started/ synchronized is known at the design time.

Oracle BPEL PM supports this pattern by placing replicas of an activity on the separate branches of the <flow> activity. There should be so many branches as many instances are required. For the details of the <flow> construct see the pattern Parallel Split.

An alternative implementation is to use the <flowN> construct offered by Oracle BPEL PM for creating the specified number of instances of the activities placed in the branch of <flowN> (see Figure 10).

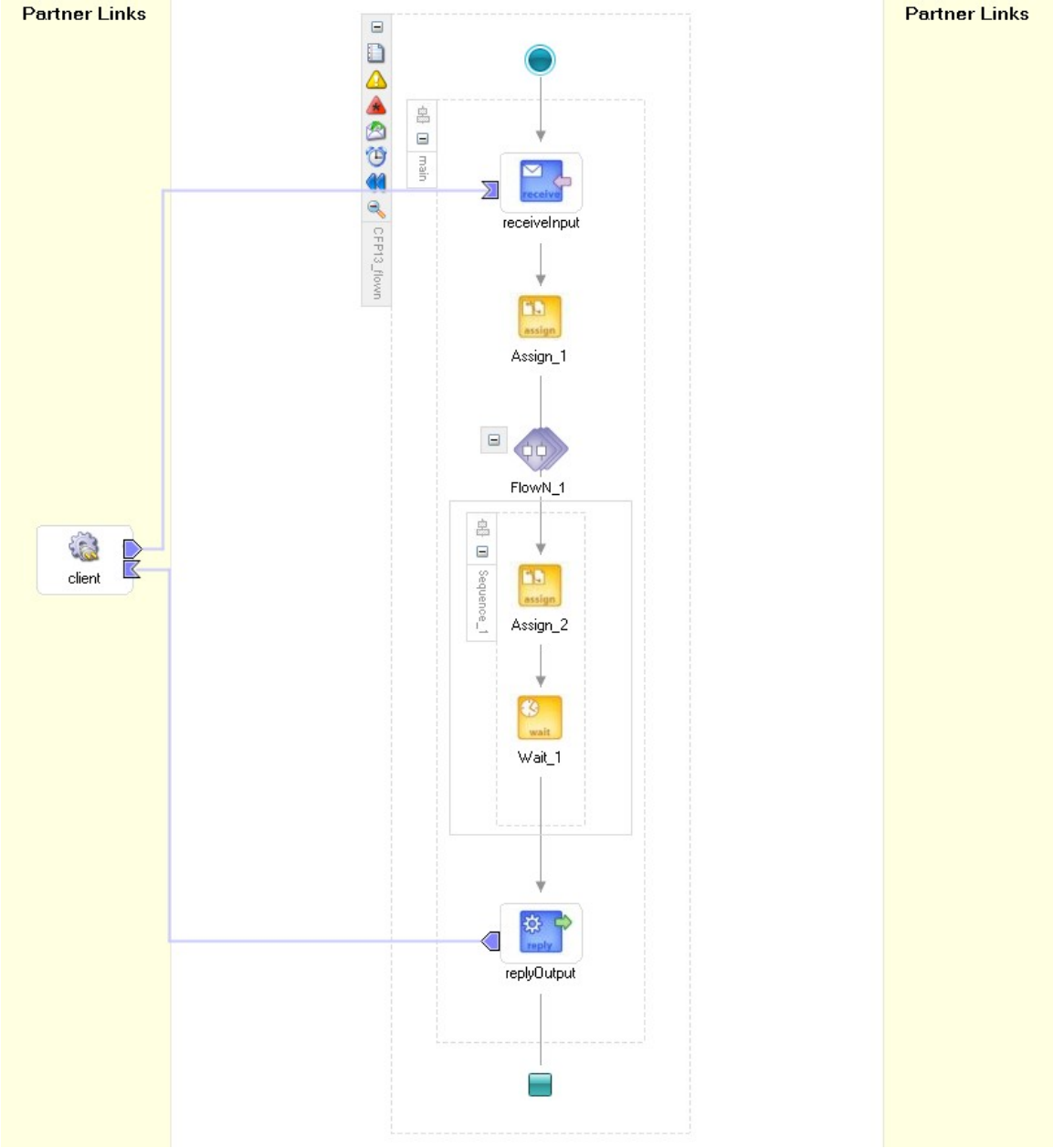


Figure 10 MI task with a-priori design-time knowledge

The code snippets corresponding to Figure 10 are shown below:

```
<process name="CFP13_flown"
targetNamespace="http://xmlns.oracle.com/CFP13_flown"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.Xpath20"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:client="http://xmlns.oracle.com/CFP13_flown"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
```

```
xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.ExtFunc"><!--
================================================================== --><!-
- PARTNERLINKS                                                      --
><!-- List of services participating in this BPEL process
--><!--
================================================================== -->
  <partnerLinks><!--
    The 'client' role represents the requester of this service. It is
    used for callback. The location and correlation information
associated
    with the client role are automatically set using WS-Addressing.
    -->
    <partnerLink name="client" partnerLinkType="client:CFP13_flown"
myRole="CFP13_flownProvider"/>
  </partnerLinks><!--
================================================================== --><!-
- VARIABLES                                                         --
><!-- List of messages and XML documents used within this BPEL process
--><!--
================================================================== -->
  <variables><!-- Reference to the message passed as input during
initiation -->
    <variable name="inputVariable"
messageType="client:CFP13_flownRequestMessage"/><!--
    Reference to the message that will be returned to the requester
    -->
    <variable name="outputVariable"
messageType="client:CFP13_flownResponseMessage"/>
    <variable name="FlowN_1_Variable" type="xsd:int"/>
  </variables><!--
================================================================== --><!-
- ORCHESTRATION LOGIC                                               --
><!-- Set of activities coordinating the flow of messages across the
--><!-- services integrated within this business process
--><!--
================================================================== -->
  <sequence name="main"><!-- Receive input from requestor.
    Note: This maps to operation defined in CFP13_flown.wsdl
    -->
    <receive name="receiveInput" partnerLink="client"
portType="client:CFP13_flown" operation="process"
variable="inputVariable" createInstance="yes"/><!-- Generate reply to
synchronous request -->
    <assign name="Assign_1">
      <copy>
        <from variable="inputVariable" part="payload"
query="/client:CFP13_flownProcessRequest/client:input"/>
        <to variable="outputVariable" part="payload"
query="/client:CFP13_flownProcessResponse/client:result"/>
      </copy>
    </assign>
    <bpelx:flowN name="FlowN_1" N="3" indexVariable="FlowN_1_Variable">
      <sequence name="Sequence_1">
        <assign name="Assign_2">
          <copy>
```

```
            <from
expression="concat(bpws:getVariableData('outputVariable','payload','/cli
ent:CFP13_flownProcessResponse/client:result'), '.')"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP13_flownProcessResponse/client:result"/>
          </copy>
        </assign>
        <wait name="Wait_1" for="'PT10S'"/>
      </sequence>
    </bpelx:flowN>
    <reply name="replyOutput" partnerLink="client"
portType="client:CFP13_flown" operation="process"
variable="outputVariable"/>
  </sequence>
</process>
```

The audit trail reflecting the execution of the considered process is shown below:

[2005/08/10 14:57:15] New instance of BPEL process "CFP13_flown" initiated (# "234").

<process>

<sequence>

receiveInput

[2005/08/10 14:57:16] Received "inputVariable" call from partner "client" less

<inputVariable>

<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">

<CFP13_flownProcessRequest xmlns="http://xmlns.oracle.com/CFP13_flown">

<input>a</input>

</CFP13_flownProcessRequest>

</part>

</inputVariable>

Assign_1

[2005/08/10 14:57:16] Updated variable "outputVariable" less

<outputVariable>

<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">

<CFP13_flownProcessResponse xmlns="http://xmlns.oracle.com/CFP13_flown">

<result>a</result>

</CFP13_flownProcessResponse>

</part>

</outputVariable>

<flowN>

<FlowN_1_Variable=0>

<sequence>

Assign_2

[2005/08/10 14:57:16] Updated variable "outputVariable" less

<outputVariable>

<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">

<CFP13_flownProcessResponse xmlns="http://xmlns.oracle.com/CFP13_flown">

<result>a...</result>

</CFP13_flownProcessResponse>

</part>

</outputVariable>

Wait_1

[2005/08/10 14:57:16] Waiting for the expiry time "2005/08/10 14:57:26".

[2005/08/10 14:57:28] Wait has finished.

</sequence>

</FlowN_1_Variable=0>

<FlowN_1_Variable=1>

<sequence>

Assign_2

[2005/08/10 14:57:16] Updated variable "outputVariable" less

<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP13_flownProcessResponse xmlns="http://xmlns.oracle.com/CFP13_flown">
<result>a..</result>
</CFP13_flownProcessResponse>
</part>
</outputVariable>

Wait_1

[2005/08/10 14:57:16] Waiting for the expiry time "2005/08/10 14:57:26".

[2005/08/10 14:57:27] Wait has finished.

</sequence>

</FlowN_1_Variable=1>

<FlowN_1_Variable=2>

<sequence>

Assign_2

[2005/08/10 14:57:16] Updated variable "outputVariable" less

<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP13_flownProcessResponse xmlns="http://xmlns.oracle.com/CFP13_flown">
<result>a.</result>
</CFP13_flownProcessResponse>
</part>
</outputVariable>

Wait_1

[2005/08/10 14:57:16] Waiting for the expiry time "2005/08/10 14:57:26".

[2005/08/10 14:57:26] Wait has finished.

</sequence>

</FlowN_1_Variable=2>

</flowN>

replyOutput

[2005/08/10 14:57:28] Reply to partner "client". less

<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP13_flownProcessResponse xmlns="http://xmlns.oracle.com/CFP13_flown">
<result>a...</result>
</CFP13_flownProcessResponse>

```
</part>
</outputVariable>
</sequence>
[2005/08/10 14:57:28]   BPEL process instance "234" completed
</process>
```

In **CFP14** the number of instances is known at some stage during run time, but before the initiation of the instances has started.

Standard BPEL does not support this pattern, however Oracle BPEL PM has implemented the <flowN> construct which supports it. The <flowN> on the moment of initiation of the process instance requests the number of task instances that should be created, and creates the specified number of instances at run-time. An example of creating multiple instances of an activity *Assign_2* with run-time a-priori knowledge is shown in Figure 11.

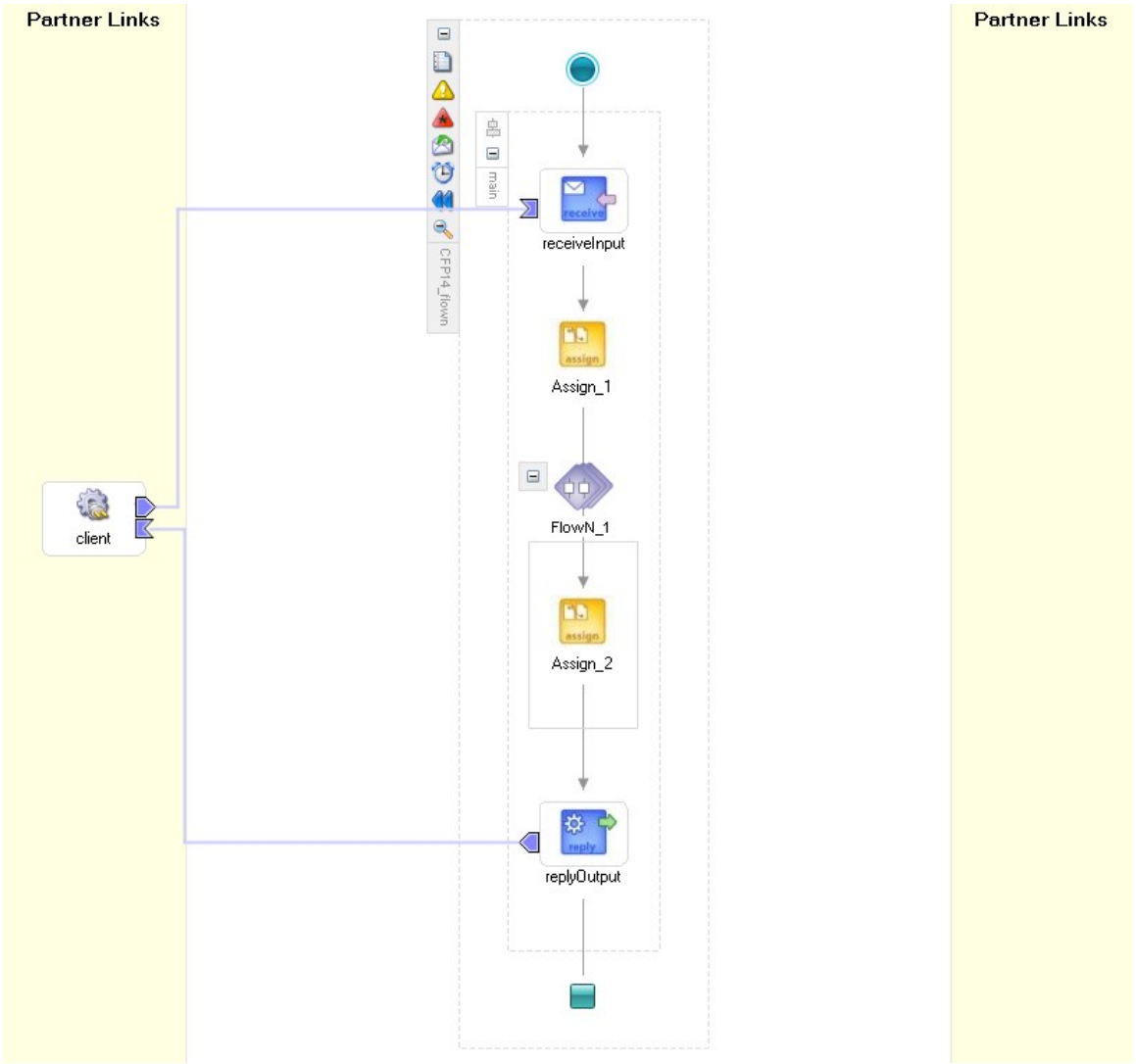


Figure 11 MI with synchronization a-priori run-time

The code snippets corresponding to the considered process model are shown below:

```
<process name="CFP14_flown"
targetNamespace="http://xmlns.oracle.com/CFP14_flown"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.Xpath20" xmlns:ns1="http://www.w3.org/2001/XMLSchema"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:client="http://xmlns.oracle.com/CFP14_flown"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.ExtFunc"><!--
================================================================= --><!-
- PARTNERLINKS                                                     --
><!-- List of services participating in this BPEL process
--><!--
================================================================= -->
  <partnerLinks><!--
    The 'client' role represents the requester of this service. It is
    used for callback. The location and correlation information
associated
    with the client role are automatically set using WS-Addressing.
    -->
    <partnerLink name="client" partnerLinkType="client:CFP14_flown"
myRole="CFP14_flownProvider"/>
  </partnerLinks><!--
================================================================= --><!-
- VARIABLES                                                        --
><!-- List of messages and XML documents used within this BPEL process
--><!--
================================================================= -->
  <variables><!-- Reference to the message passed as input during
initiation --><!--
    Reference to the message that will be returned to the requester
    -->
    <variable name="inputVariable"
messageType="client:CFP14_flownRequestMessage"/>
    <variable name="outputVariable"
messageType="client:CFP14_flownResponseMessage"/>
    <variable name="FlowN_1_Variable" type="xsd:int"/>
    <variable name="Counter" type="ns1:integer"/>
  </variables><!--
================================================================= --><!-
- ORCHESTRATION LOGIC                                              --
><!-- Set of activities coordinating the flow of messages across the
--><!-- services integrated within this business process
--><!--
================================================================= -->
  <sequence name="main"><!-- Receive input from requestor.
    Note: This maps to operation defined in CFP14_flown.wsdl
    -->
```

```
    <receive name="receiveInput" partnerLink="client"
portType="client:CFP14_flown" operation="process"
variable="inputVariable" createInstance="yes"/><!-- Generate reply to
synchronous request -->
    <assign name="Assign_1">
      <copy>
        <from variable="inputVariable" part="payload"
query="/client:CFP14_flownProcessRequest/client:input"/>
        <to variable="outputVariable" part="payload"
query="/client:CFP14_flownProcessResponse/client:result"/>
      </copy>
      <copy>
        <from
expression="number(bpws:getVariableData('inputVariable','payload','/clie
nt:CFP14_flownProcessRequest/client:input'))"/>
        <to variable="Counter"/>
      </copy>
    </assign>
    <bpelx:flowN name="FlowN_1" N="bpws:getVariableData('Counter')"
indexVariable="FlowN_1_Variable">
      <assign name="Assign_2">
        <copy>
          <from
expression="concat(bpws:getVariableData('outputVariable','payload','/cli
ent:CFP14_flownProcessResponse/client:result'), '.')"/>
          <to variable="outputVariable" part="payload"
query="/client:CFP14_flownProcessResponse/client:result"/>
        </copy>
      </assign>
    </bpelx:flowN>
    <reply name="replyOutput" partnerLink="client"
portType="client:CFP14_flown" operation="process"
variable="outputVariable"/>
  </sequence>
</process>
```

An audit trail visualizing the execution history of the considered example is shown below:

[2005/08/10 15:00:38] New instance of BPEL process "CFP14_flown" initiated (# "235").

<process>

<sequence>

receiveInput

[2005/08/10 15:00:38] Received "inputVariable" call from partner "client" less

<inputVariable>

<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">

<CFP14_flownProcessRequest xmlns="http://xmlns.oracle.com/CFP14_flown">

<input>3</input>

</CFP14_flownProcessRequest>

</part>

</inputVariable>

Assign_1

[2005/08/10 15:00:38] Updated variable "outputVariable" less

<outputVariable>

<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP14_flownProcessResponse xmlns="http://xmlns.oracle.com/CFP14_flown">
<result>3</result>
</CFP14_flownProcessResponse>
</part>
</outputVariable>
[2005/08/10 15:00:38] Updated variable "Counter" less
<Counter>3</Counter>
<flowN>
<FlowN_1_Variable=0>

Assign_2

[2005/08/10 15:00:38] Updated variable "outputVariable" less
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP14_flownProcessResponse xmlns="http://xmlns.oracle.com/CFP14_flown">
<result>3.</result>
</CFP14_flownProcessResponse>
</part>
</outputVariable>
</FlowN_1_Variable=0>
<FlowN_1_Variable=1>

Assign_2

[2005/08/10 15:00:38] Updated variable "outputVariable" less
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP14_flownProcessResponse xmlns="http://xmlns.oracle.com/CFP14_flown">
<result>3..</result>
</CFP14_flownProcessResponse>
</part>
</outputVariable>
</FlowN_1_Variable=1>
<FlowN_1_Variable=2>

Assign_2

[2005/08/10 15:00:38] Updated variable "outputVariable" less
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP14_flownProcessResponse xmlns="http://xmlns.oracle.com/CFP14_flown">
<result>3...</result>
</CFP14_flownProcessResponse>
</part>
</outputVariable>
</FlowN_1_Variable=2>
</flowN>

replyOutput

[2005/08/10 15:00:38] Reply to partner "client". less
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">

<CFP14_flownProcessResponse xmlns="http://xmlns.oracle.com/CFP14_flown">
<result>3...</result>
</CFP14_flownProcessResponse>
</part>
</outputVariable>
</sequence>
[2005/08/10 15:00:38] BPEL process instance "235" completed
</process>

[2005/08/10 15:10:00] New instance of BPEL process "CFP14_flown" initiated (# "236").
<process>
<sequence>
receiveInput
[2005/08/10 15:10:00] Received "inputVariable" call from partner "client" less
<inputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP14_flownProcessRequest xmlns="http://xmlns.oracle.com/CFP14_flown">
<input>0</input>
</CFP14_flownProcessRequest>
</part>
</inputVariable>
Assign_1
[2005/08/10 15:10:00] Updated variable "outputVariable" less
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP14_flownProcessResponse xmlns="http://xmlns.oracle.com/CFP14_flown">
<result>0</result>
</CFP14_flownProcessResponse>
</part>
</outputVariable>
[2005/08/10 15:10:00] Updated variable "Counter" less
<Counter>0</Counter>
<flowN>

The following inputs caused unhandled exceptions to be thrown:

Input = -3:
Your test request generated the following exception/fault:
`java.lang.NegativeArraySizeException`

`Input = 1000:`
Your test request generated the following exception/fault:
```
java.rmi.RemoteException: No Exception - originate
from:java.lang.Exception: No Exception - originate from:; nested
exception is:
        java.lang.Exception: No Exception - originate from:
```

In **CFP15** the number of instances to be created is not known in advance: new instances are created on demand, until no more instances are required.

Oracle BPEL PM offers no direct support for this pattern, however the workaround can be found by means of placing the <pick> construct within the <while> loop. The <pick> should be configured to react on three types of messages: the first message Start corresponds to the creation of an activity; the second message Finish corresponds to the completion of the started activity; the third message NoMore notifies that no more instances of the activity will be created any more. Thus the logic of this pattern is based on the logical expression evaluating the value of a counter i, which is increased and decreased if a new task instance is created and finished respectively, and the status of the Boolean variable *moreInstances* indicating whether new instances are still to be created. The process model incorporating this pattern is shown in Figure 12.

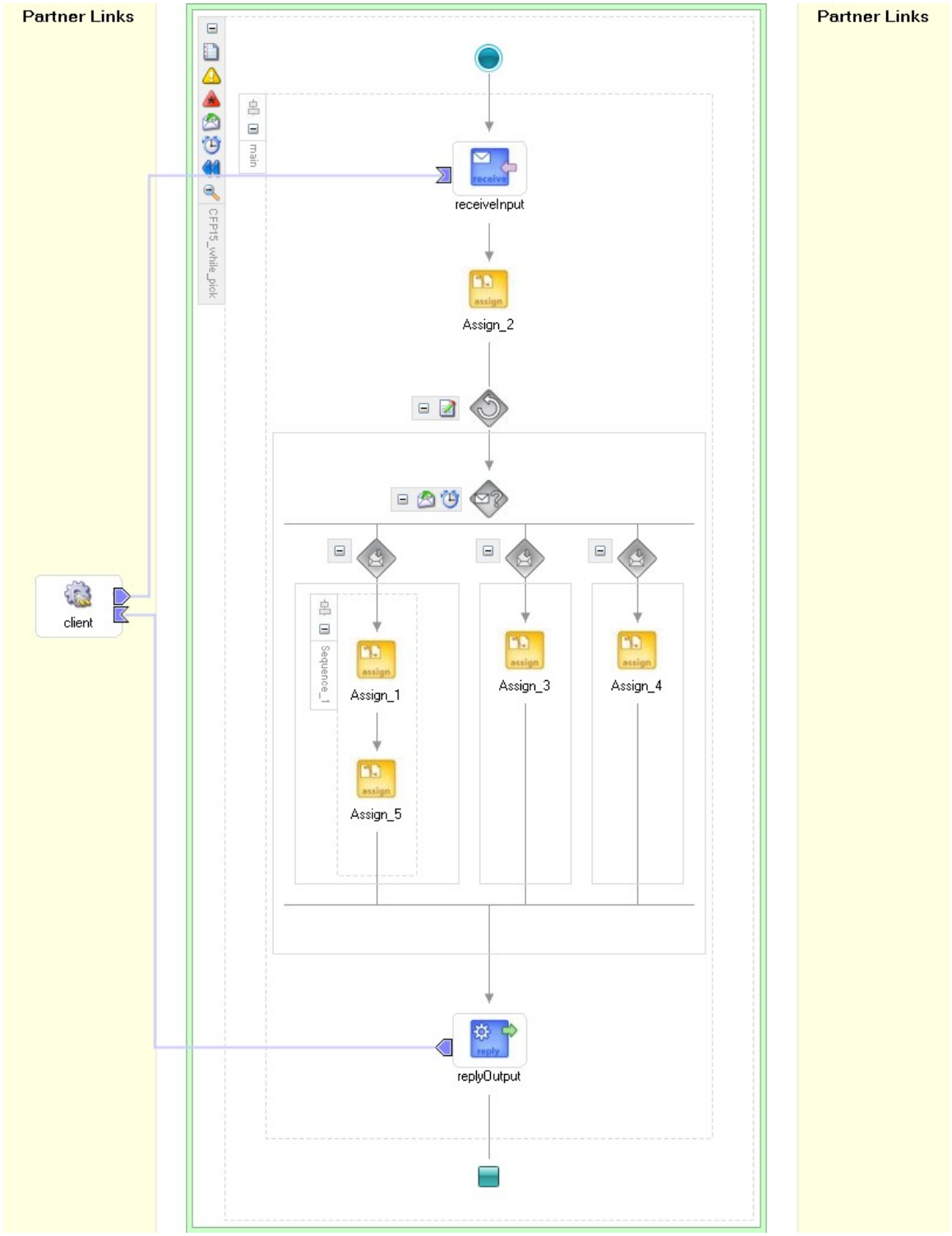


Figure 12 MI with synchronization (run-time)

The source code corresponding to the considered process model is shown below:

```
<process name="CFP15_while_pick"
targetNamespace="http://xmlns.oracle.com/CFP15_while_pick"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
```

```
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.Xpath20"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:ns1="http://www.w3.org/2001/XMLSchema"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:client="http://xmlns.oracle.com/CFP15_while_pick"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.ExtFunc"><!--
================================================================= --><!-
- PARTNERLINKS                                                      --
><!-- List of services participating in this BPEL process
--><!--
================================================================= -->
  <partnerLinks><!--
    The 'client' role represents the requester of this service. It is
    used for callback. The location and correlation information
associated
    with the client role are automatically set using WS-Addressing.
    -->
    <partnerLink name="client" partnerLinkType="client:CFP15_while_pick"
myRole="CFP15_while_pickProvider"/>
  </partnerLinks><!--
================================================================= --><!-
- VARIABLES                                                         --
><!-- List of messages and XML documents used within this BPEL process
--><!--
================================================================= -->
  <variables><!-- Reference to the message passed as input during
initiation --><!--
    Reference to the message that will be returned to the requester
    -->
    <variable name="inputVariable"
messageType="client:CFP15_while_pickRequestMessage"/>
    <variable name="outputVariable"
messageType="client:CFP15_while_pickResponseMessage"/>
    <variable name="OnMessage_start_InputVariable"
messageType="client:CFP15_while_pickStartMessage"/>
    <variable name="Counter" type="ns1:integer"/>
    <variable name="MoreInstances" type="ns1:integer"/>
    <variable name="OnMessage_finish_InputVariable"
messageType="client:CFP15_while_pickFinishMessage"/>
    <variable name="OnMessage_noMore_InputVariable"
messageType="client:CFP15_while_pickNoMoreMessage"/>
  </variables><!--
================================================================= --><!-
- ORCHESTRATION LOGIC                                               --
><!-- Set of activities coordinating the flow of messages across the
--><!-- services integrated within this business process
--><!--
================================================================= -->
  <sequence name="main"><!-- Receive input from requestor.
    Note: This maps to operation defined in CFP15_while_pick.wsdl
    -->
```

```
    <receive name="receiveInput" partnerLink="client"
portType="client:CFP15_while_pick" operation="process"
variable="inputVariable" createInstance="yes"/><!-- Generate reply to
synchronous request -->
    <assign name="Assign_2">
      <copy>
        <from variable="inputVariable" part="payload"
query="/client:CFP15_while_pickProcessRequest/client:input"/>
        <to variable="outputVariable" part="payload"
query="/client:CFP15_while_pickProcessResponse/client:result"/>
      </copy>
      <copy>
        <from expression="0"/>
        <to variable="Counter"/>
      </copy>
      <copy>
        <from expression="1"/>
        <to variable="MoreInstances"/>
      </copy>
    </assign>
    <while name="While_1"
condition="bpws:getVariableData('MoreInstances') +
bpws:getVariableData('Counter') &gt; 0">
      <pick name="Pick_1">
        <onMessage portType="client:CFP15_while_pick" operation="start"
variable="OnMessage_start_InputVariable" partnerLink="client">
          <sequence name="Sequence_1">
            <assign name="Assign_1">
              <copy>
                <from expression="bpws:getVariableData('Counter') + 1"/>
                <to variable="Counter"/>
              </copy>
            </assign>
            <assign name="Assign_5">
              <copy>
                <from
expression="concat(bpws:getVariableData('outputVariable','payload','/cli
ent:CFP15_while_pickProcessResponse/client:result'), '.')"/>
                <to variable="outputVariable" part="payload"
query="/client:CFP15_while_pickProcessResponse/client:result"/>
              </copy>
            </assign>
          </sequence>
        </onMessage>
        <onMessage portType="client:CFP15_while_pick" operation="finish"
variable="OnMessage_finish_InputVariable" partnerLink="client">
          <assign name="Assign_3">
            <copy>
              <from expression="bpws:getVariableData('Counter') - 1"/>
              <to variable="Counter"/>
            </copy>
          </assign>
        </onMessage>
        <onMessage portType="client:CFP15_while_pick" operation="noMore"
variable="OnMessage_noMore_InputVariable" partnerLink="client">
          <assign name="Assign_4">
            <copy>
```

```
              <from expression="0"/>
              <to variable="MoreInstances"/>
            </copy>
          </assign>
        </onMessage>
      </pick>
    </while>
    <reply name="replyOutput" partnerLink="client"
portType="client:CFP15_while_pick" operation="process"
variable="outputVariable"/>
  </sequence>
</process>
```

The semantics of “OR” when it is used in the condition of the <while> loop differs from expected if a single identifier used in a Boolean Expression. As such, an expression containing the variable *MoreInstances* of the Boolean type in the condition of the <while> is not evaluated properly. According to the syntax of XPath a single identifier cannot be taken as a Boolean expression. Therefore, the variable of an integer type or an expression containing a relational operator should be used.

To make this example operational several types, messages, operations and partner links were defined in the *wsdl* file as shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="CFP15_while_pick"
             targetNamespace="http://xmlns.oracle.com/CFP15_while_pick"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:client="http://xmlns.oracle.com/CFP15_while_pick"
             xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-
link/">

    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    TYPE DEFINITION - List of services participating in this BPEL
process
    The default output of the BPEL designer uses strings as input and
    output to the BPEL Process. But you can define or import any XML
    Schema type and us them as part of the message types.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <types>
        <schema attributeFormDefault="qualified"
            elementFormDefault="qualified"
            targetNamespace="http://xmlns.oracle.com/CFP15_while_pick"
            xmlns="http://www.w3.org/2001/XMLSchema">
            <element name="CFP15_while_pickProcessRequest">
                <complexType>
                    <sequence>
                        <element name="input" type="string"/>
                    </sequence>
                </complexType>
            </element>
            <element name="CFP15_while_pickProcessStart">
                <complexType>
                    <sequence>
```

```
                        <element name="input" type="string"/>
                    </sequence>
                </complexType>
            </element>
            <element name="CFP15_while_pickProcessFinish">
                <complexType>
                    <sequence>
                        <element name="input" type="string"/>
                    </sequence>
                </complexType>
            </element>
            <element name="CFP15_while_pickProcessNoMore">
                <complexType>
                    <sequence>
                        <element name="input" type="string"/>
                    </sequence>
                </complexType>
            </element>
            <element name="CFP15_while_pickProcessResponse">
                <complexType>
                    <sequence>
                        <element name="result" type="string"/>
                    </sequence>
                </complexType>
            </element>
        </schema>
    </types>

    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    MESSAGE TYPE DEFINITION - Definition of the message types used as
    part of the port type defintions

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <message name="CFP15_while_pickRequestMessage">
        <part name="payload"
element="client:CFP15_while_pickProcessRequest"/>
    </message>
    <message name="CFP15_while_pickStartMessage">
        <part name="payload"
element="client:CFP15_while_pickProcessStart"/>
    </message>
    <message name="CFP15_while_pickFinishMessage">
        <part name="payload"
element="client:CFP15_while_pickProcessFinish"/>
    </message>
    <message name="CFP15_while_pickNoMoreMessage">
        <part name="payload"
element="client:CFP15_while_pickProcessNoMore"/>
    </message>
    <message name="CFP15_while_pickResponseMessage">
        <part name="payload"
element="client:CFP15_while_pickProcessResponse"/>
    </message>
```

```
    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    PORT TYPE DEFINITION - A port type groups a set of operations into
    a logical service unit.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>

    <!-- portType implemented by the CFP15_while_pick BPEL process -->
    <portType name="CFP15_while_pick">
        <operation name="process">
            <input  message="client:CFP15_while_pickRequestMessage" />
            <output message="client:CFP15_while_pickResponseMessage"/>
        </operation>
        <operation name="start">
            <input  message="client:CFP15_while_pickStartMessage" />
        </operation>
        <operation name="finish">
            <input  message="client:CFP15_while_pickFinishMessage" />
        </operation>
        <operation name="noMore">
            <input  message="client:CFP15_while_pickNoMoreMessage" />
        </operation>
    </portType>

    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    PARTNER LINK TYPE DEFINITION

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <plnk:partnerLinkType name="CFP15_while_pick">
        <plnk:role name="CFP15_while_pickProvider">
            <plnk:portType name="client:CFP15_while_pick"/>
        </plnk:role>
        <plnk:role name="CFP15_while_pickSender">
            <plnk:portType name="client:CFP15_while_pick"/>
        </plnk:role>
    </plnk:partnerLinkType>
</definitions>
```

An audit trail corresponding visualizing the execution history of the considered process model is shown below:

[2005/08/10 15:58:00] New instance of BPEL process "CFP15_while_pick" initiated (# "241").

⊟<process>

⊟<sequence>

receiveInput

[2005/08/10 15:58:00] Received "inputVariable" call from partner "client" More...

Assign_2

[2005/08/10 15:58:00] Updated variable "outputVariable" less

<outputVariable>

<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">

<CFP15_while_pickProcessResponse xmlns="http://xmlns.oracle.com/CFP15_while_pick">

<result>a</result>
</CFP15_while_pickProcessResponse>
</part>
</outputVariable>
[2005/08/10 15:58:00] Updated variable "Counter" More...
[2005/08/10 15:58:00] Updated variable "MoreInstances" More...
<while>
<pick>
onMessage (73) (cancelled)
[2005/08/10 15:58:00] Waiting for message from "client", operation is "noMore".
[2005/08/10 15:59:31] Receive activity has been cancelled.
onMessage (65) (cancelled)
[2005/08/10 15:58:00] Waiting for message from "client", operation is "finish".
[2005/08/10 15:59:31] Receive activity has been cancelled.
onMessage (49)
[2005/08/10 15:58:00] Waiting for message from "client", operation is "start".
[2005/08/10 15:59:31] Received "start" callback from partner "client" More...
<onMessage>
<sequence>
Assign_1
[2005/08/10 15:59:31] Updated variable "Counter" More...
Assign_5
[2005/08/10 15:59:31] Updated variable "outputVariable" less
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP15_while_pickProcessResponse xmlns="http://xmlns.oracle.com/CFP15_while_pick">
<result>a.</result>
</CFP15_while_pickProcessResponse>
</part>
</outputVariable>
</sequence>
</onMessage>
</pick>
<pick>
onMessage (73)
[2005/08/10 15:59:31] Waiting for message from "client", operation is "noMore".
[2005/08/10 15:59:58] Received "noMore" callback from partner "client" More...
onMessage (65) (cancelled)
[2005/08/10 15:59:31] Waiting for message from "client", operation is "finish".
[2005/08/10 15:59:58] Receive activity has been cancelled.
onMessage (49) (cancelled)
[2005/08/10 15:59:31] Waiting for message from "client", operation is "start".
[2005/08/10 15:59:58] Receive activity has been cancelled.
<onMessage>
Assign_4

[2005/08/10 15:59:58] Updated variable "MoreInstances" More...

</onMessage>

</pick>

<pick>

onMessage (73) (cancelled)

[2005/08/10 15:59:58] Waiting for message from "client", operation is "noMore".

[2005/08/10 16:00:27] Receive activity has been cancelled.

onMessage (65)

[2005/08/10 15:59:59] Waiting for message from "client", operation is "finish".

[2005/08/10 16:00:27] Received "finish" callback from partner "client" More...

onMessage (49) (cancelled)

[2005/08/10 15:59:59] Waiting for message from "client", operation is "start".

[2005/08/10 16:00:27] Receive activity has been cancelled.

<onMessage>

Assign_3

[2005/08/10 16:00:27] Updated variable "Counter" More...

</onMessage>

</pick>

</while>

replyOutput (faulted)

[2005/08/10 16:00:27] "NullPointerException" has been thrown. less

```
java.lang.NullPointerException
```

</sequence>

[2005/08/10 16:00:27] "NullPointerException" has not been caught by a catch block.

[2005/08/10 16:00:27] BPEL process instance "241" cancelled

</process>

[2005/08/10 16:03:14] New instance of BPEL process "CFP15_while_pick" initiated (# "242").
<process>
<sequence>
receiveInput
[2005/08/10 16:03:14] Received "inputVariable" call from partner "client" More...
Assign_2
[2005/08/10 16:03:14] Updated variable "outputVariable" More...
[2005/08/10 16:03:14] Updated variable "Counter" More...
[2005/08/10 16:03:14] Updated variable "MoreInstances" More...
<while>
<pick>
onMessage (73)
[2005/08/10 16:03:15] Waiting for message from "client", operation is "noMore".
[2005/08/10 16:04:27] Received "noMore" callback from partner "client" More...
onMessage (65) (cancelled)
[2005/08/10 16:03:15] Waiting for message from "client", operation is "finish".
[2005/08/10 16:04:27] Receive activity has been cancelled.
onMessage (49) (cancelled)
[2005/08/10 16:03:15] Waiting for message from "client", operation is "start".
[2005/08/10 16:04:27] Receive activity has been cancelled.
<onMessage>
Assign_4
[2005/08/10 16:04:27] Updated variable "MoreInstances" More...
</onMessage>
</pick>
</while>
replyOutput (faulted)
[2005/08/10 16:04:27] "NullPointerException" has been thrown. More...
</sequence>
[2005/08/10 16:04:27] "NullPointerException" has not been caught by a catch block.
[2005/08/10 16:04:27] BPEL process instance "242" cancelled
</process>

CFP16 Deferred Choice
Description: A point in a process where one among several alternative branches is chosen based on information which is not necessarily available when this point is reached. This differs from the normal exclusive choice, in that the choice is not made immediately when the point is reached, but instead several alternatives are offered, and the choice between them is delayed until the occurrence of some event.

Oracle BPEL PM supports this pattern directly by the <pick> construct, which allows only one of several possible activities or (a set of activities) to be executed based on the type of the message received. Alternatively, <pick> allows the time trigger to be specified for the timeouts discarding all other alternative tasks. Figure 13 shows an example of a

process model with the deferred choice. Activities *Assign_2* or *Assign_3* are enabled if the client provided a message "left" or "right" respectively. If none of these messages had been received before the time alarm set for the third branch has expired, an activity *Assign_4* is executed.

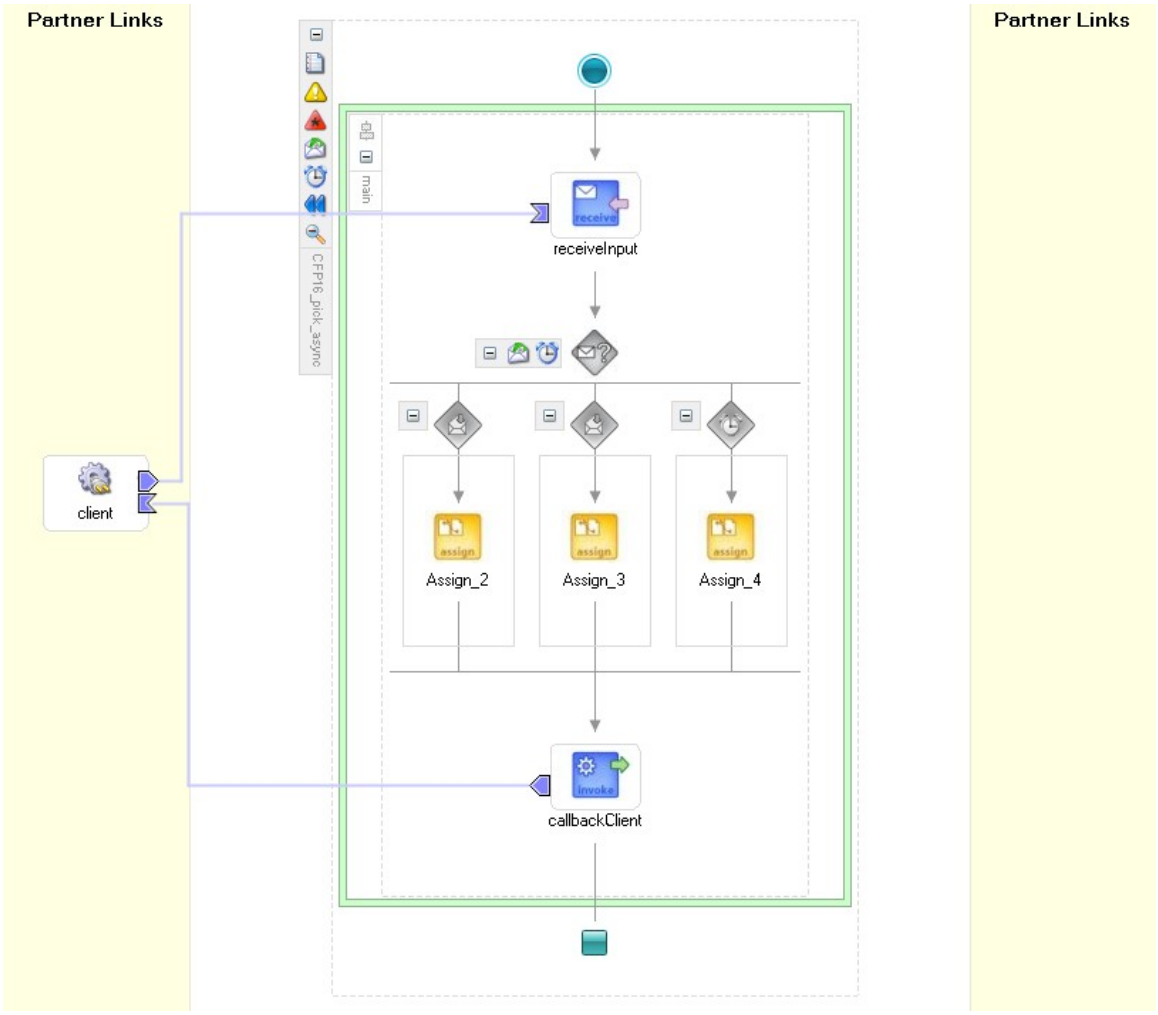


Figure 13 Deferred choice

The source code corresponding to the considered process model is shown below:

```
<process name="CFP16_pick_async"
targetNamespace="http://xmlns.oracle.com/CFP16_pick_async"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.Xpath20"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:client="http://xmlns.oracle.com/CFP16_pick_async"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.ExtFunc"><!--
```

```
================================================================== --><!-
- PARTNERLINKS                                                      --
><!-- List of services participating in this BPEL process
--><!--
================================================================== -->
  <partnerLinks><!--
    The 'client' role represents the requester of this service. It is
    used for callback. The location and correlation information
associated
    with the client role are automatically set using WS-Addressing.
    -->
    <partnerLink name="client" partnerLinkType="client:CFP16_pick_async"
myRole="CFP16_pick_asyncProvider"
partnerRole="CFP16_pick_asyncRequester"/>
  </partnerLinks><!--
================================================================== --><!-
- VARIABLES                                                         --
><!-- List of messages and XML documents used within this BPEL process
--><!--
================================================================== -->
  <variables><!-- Reference to the message passed as input during
initiation -->
    <variable name="inputVariable"
messageType="client:CFP16_pick_asyncRequestMessage"/><!-- Reference to
the message that will be sent back to the
    requester during callback
    -->
    <variable name="outputVariable"
messageType="client:CFP16_pick_asyncResponseMessage"/>
    <variable name="OnMessage_left_InputVariable"
messageType="client:CFP16_pick_asyncLeftMessage"/>
    <variable name="OnMessage_right_InputVariable"
messageType="client:CFP16_pick_asyncRightMessage"/>
  </variables><!--
================================================================== --><!-
- ORCHESTRATION LOGIC                                               --
><!-- Set of activities coordinating the flow of messages across the
--><!-- services integrated within this business process
--><!--
================================================================== -->
  <sequence name="main"><!-- Receive input from requestor.
    Note: This maps to operation defined in CFP16_pick_async.wsdl
    -->
    <receive name="receiveInput" partnerLink="client"
portType="client:CFP16_pick_async" operation="initiate"
variable="inputVariable" createInstance="yes"/><!-- Asynchronous
callback to the requester.
     Note: the callback location and correlation id is transparently
handled
     using WS-addressing.
     -->
    <pick name="Pick_1">
      <onMessage portType="client:CFP16_pick_async" operation="left"
variable="OnMessage_left_InputVariable" partnerLink="client">
        <assign name="Assign_2">
          <copy>
```

```
            <from
expression="concat(bpws:getVariableData('inputVariable','payload','/clie
nt:CFP16_pick_asyncProcessRequest/client:input'), 'Left')"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP16_pick_asyncProcessResponse/client:result"/>
          </copy>
        </assign>
      </onMessage>
      <onMessage portType="client:CFP16_pick_async" operation="right"
variable="OnMessage_right_InputVariable" partnerLink="client">
        <assign name="Assign_3">
          <copy>
            <from
expression="concat(bpws:getVariableData('inputVariable','payload','/clie
nt:CFP16_pick_asyncProcessRequest/client:input'), 'Right')"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP16_pick_asyncProcessResponse/client:result"/>
          </copy>
        </assign>
      </onMessage>
      <onAlarm for="'PT2M'">
        <assign name="Assign_4">
          <copy>
            <from
expression="concat(bpws:getVariableData('inputVariable','payload','/clie
nt:CFP16_pick_asyncProcessRequest/client:input'), 'Alarm')"/>
            <to variable="outputVariable" part="payload"
query="/client:CFP16_pick_asyncProcessResponse/client:result"/>
          </copy>
        </assign>
      </onAlarm>
    </pick>
    <invoke name="callbackClient" partnerLink="client"
portType="client:CFP16_pick_asyncCallback" operation="onResult"
inputVariable="outputVariable"/>
  </sequence>
</process>
```

The declarations of data types, messages, operations, and port types specified in the WP16.wsdl file are shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="CFP16_pick_async"
             targetNamespace="http://xmlns.oracle.com/CFP16_pick_async"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:client="http://xmlns.oracle.com/CFP16_pick_async"
             xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-
link/">

    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    TYPE DEFINITION - List of services participating in this BPEL
process
    The default output of the BPEL designer uses strings as input and
    output to the BPEL Process. But you can define or import any XML
    Schema type and us them as part of the message types.
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <types>
        <schema attributeFormDefault="qualified"
            elementFormDefault="qualified"
            targetNamespace="http://xmlns.oracle.com/CFP16_pick_async"
            xmlns="http://www.w3.org/2001/XMLSchema">
            <element name="CFP16_pick_asyncProcessRequest">
                <complexType>
                    <sequence>
                        <element name="input" type="string"/>
                    </sequence>
                </complexType>
            </element>
            <element name="CFP16_pick_asyncProcessLeft">
                <complexType>
                    <sequence>
                        <element name="input" type="string"/>
                    </sequence>
                </complexType>
            </element>
            <element name="CFP16_pick_asyncProcessRight">
                <complexType>
                    <sequence>
                        <element name="input" type="string"/>
                    </sequence>
                </complexType>
            </element>
            <element name="CFP16_pick_asyncProcessResponse">
                <complexType>
                    <sequence>
                        <element name="result" type="string"/>
                    </sequence>
                </complexType>
            </element>
        </schema>
    </types>

    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    MESSAGE TYPE DEFINITION - Definition of the message types used as
    part of the port type defintions
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <message name="CFP16_pick_asyncRequestMessage">
        <part name="payload"
element="client:CFP16_pick_asyncProcessRequest"/>
    </message>

    <message name="CFP16_pick_asyncLeftMessage">
        <part name="payload"
element="client:CFP16_pick_asyncProcessLeft"/>
    </message>

    <message name="CFP16_pick_asyncRightMessage">
```

```
        <part name="payload"
element="client:CFP16_pick_asyncProcessRight"/>
    </message>

    <message name="CFP16_pick_asyncResponseMessage">
        <part name="payload"
element="client:CFP16_pick_asyncProcessResponse"/>
    </message>

    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    PORT TYPE DEFINITION - A port type groups a set of operations into
    a logical service unit.
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <!-- portType implemented by the CFP16_pick_async BPEL process -->
    <portType name="CFP16_pick_async">
        <operation name="initiate">
            <input message="client:CFP16_pick_asyncRequestMessage"/>
        </operation>
        <operation name="left">
            <input message="client:CFP16_pick_asyncLeftMessage"/>
        </operation>
        <operation name="right">
            <input message="client:CFP16_pick_asyncRightMessage"/>
        </operation>
    </portType>

    <!-- portType implemented by the requester of CFP16_pick_async BPEL
process
    for asynchronous callback purposes
    -->
    <portType name="CFP16_pick_asyncCallback">
        <operation name="onResult">
            <input message="client:CFP16_pick_asyncResponseMessage"/>
        </operation>
    </portType>

    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    PARTNER LINK TYPE DEFINITION
    the CFP16_pick_async partnerLinkType binds the provider and
    requester portType into an asynchronous conversation.
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <plnk:partnerLinkType name="CFP16_pick_async">
        <plnk:role name="CFP16_pick_asyncProvider">
            <plnk:portType name="client:CFP16_pick_async"/>
        </plnk:role>
        <plnk:role name="CFP16_pick_asyncSender">
            <plnk:portType name="client:CFP16_pick_async"/>
        </plnk:role>
        <plnk:role name="CFP16_pick_asyncRequester">
            <plnk:portType name="client:CFP16_pick_asyncCallback"/>
        </plnk:role>
    </plnk:partnerLinkType>
</definitions>
```

An audit trail visualizing the execution history of the considered process is shown below. This audit trail shows how the <pick> construct reacts on the message "right" received from the client.

[2005/08/10 16:29:01] New instance of BPEL process "CFP16_pick_async" initiated (# "245").
<process>
<sequence>
receiveInput
[2005/08/10 16:29:01] Received "inputVariable" call from partner "client" less
<inputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP16_pick_asyncProcessRequest xmlns="http://xmlns.oracle.com/CFP16_pick_async">
<input>a</input>
</CFP16_pick_asyncProcessRequest>
</part>
</inputVariable>
<pick>
onMessage (42)
[2005/08/10 16:29:01] Waiting for message from "client", operation is "right".
[2005/08/10 16:29:22] Received "right" callback from partner "client" less
<OnMessage_right_InputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP16_pick_asyncProcessRight xmlns="http://xmlns.oracle.com/CFP16_pick_async">
<input>1</input>
</CFP16_pick_asyncProcessRight>
</part>
</OnMessage_right_InputVariable>
onMessage (34) (cancelled)
[2005/08/10 16:29:01] Waiting for message from "client", operation is "left".
[2005/08/10 16:29:22] Receive activity has been cancelled.
onAlarm (50) (cancelled)
[2005/08/10 16:29:01] Alarm started. Alarm will go off at time "2005/08/10 16:31:01".
[2005/08/10 16:29:22] BPEL "onAlarm" cancelled before being triggered.
<onMessage>
Assign_3
[2005/08/10 16:29:22] Updated variable "outputVariable" less
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP16_pick_asyncProcessResponse xmlns="http://xmlns.oracle.com/CFP16_pick_async">
<result>aRight</result>
</CFP16_pick_asyncProcessResponse>
</part>
</outputVariable>
</onMessage>
</pick>
callbackClient

[2005/08/10 16:29:22] Skipped callback "onResult" on partner "client". less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP16_pick_asyncProcessResponse xmlns="http://xmlns.oracle.com/CFP16_pick_async">
<result>aRight</result>
    </CFP16_pick_asyncProcessResponse>
    </part>
    </outputVariable>
    </sequence>
```

[2005/08/10 16:29:22] BPEL process instance "245" completed

```
    </process>
```

This audit trail shows how the <pick> construct reacts on the message "left" received from the client.

[2005/08/10 16:30:00] New instance of BPEL process "CFP16_pick_async" initiated (# "246").

```
    <process>
    <sequence>
```

receiveInput

[2005/08/10 16:30:00] Received "inputVariable" call from partner "client" less

```
<inputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP16_pick_asyncProcessRequest xmlns="http://xmlns.oracle.com/CFP16_pick_async">
<input>b</input>
    </CFP16_pick_asyncProcessRequest>
    </part>
    </inputVariable>
    <pick>
```

onMessage (42) (cancelled)

[2005/08/10 16:30:00] Waiting for message from "client", operation is "right".

[2005/08/10 16:30:19] Receive activity has been cancelled.

onMessage (34)

[2005/08/10 16:30:00] Waiting for message from "client", operation is "left".

[2005/08/10 16:30:19] Received "left" callback from partner "client" less

```
<OnMessage_left_InputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP16_pick_asyncProcessLeft xmlns="http://xmlns.oracle.com/CFP16_pick_async">
<input>2</input>
    </CFP16_pick_asyncProcessLeft>
    </part>
    </OnMessage_left_InputVariable>
```

onAlarm (50) (cancelled)

[2005/08/10 16:30:00] Alarm started. Alarm will go off at time "2005/08/10 16:32:08".

[2005/08/10 16:30:20] BPEL "onAlarm" cancelled before being triggered.

```
    <onMessage>
```

Assign_2

[2005/08/10 16:30:20] Updated variable "outputVariable" less

```
<outputVariable>
```

<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP16_pick_asyncProcessResponse xmlns="http://xmlns.oracle.com/CFP16_pick_async">
<result>bLeft</result>
</CFP16_pick_asyncProcessResponse>
</part>
</outputVariable>
</onMessage>
</pick>

callbackClient

[2005/08/10 16:30:20] Skipped callback "onResult" on partner "client". less

<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP16_pick_asyncProcessResponse xmlns="http://xmlns.oracle.com/CFP16_pick_async">
<result>bLeft</result>
</CFP16_pick_asyncProcessResponse>
</part>
</outputVariable>
</sequence>
[2005/08/10 16:30:20] BPEL process instance "246" completed
</process>

This audit trail shows how the <pick> construct behaves if no messages have been provided by the client and the time alarm has expired.

[2005/08/10 16:31:29] New instance of BPEL process "CFP16_pick_async" initiated (# "247").
<process>
<sequence>

receiveInput

[2005/08/10 16:31:29] Received "inputVariable" call from partner "client" less

<inputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP16_pick_asyncProcessRequest xmlns="http://xmlns.oracle.com/CFP16_pick_async">
<input>c</input>
</CFP16_pick_asyncProcessRequest>
</part>
</inputVariable>
<pick>

onMessage (42) (cancelled)

[2005/08/10 16:31:29] Waiting for message from "client", operation is "right".
[2005/08/10 16:33:29] Receive activity has been cancelled.

onMessage (34) (cancelled)

[2005/08/10 16:31:29] Waiting for message from "client", operation is "left".
[2005/08/10 16:33:29] Receive activity has been cancelled.

onAlarm (50)

[2005/08/10 16:31:29] Alarm started. Alarm will go off at time "2005/08/10 16:33:29".
[2005/08/10 16:33:29] BPEL "onAlarm" triggered.
<onAlarm>

Assign_4

[2005/08/10 16:33:29] Updated variable "outputVariable" less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP16_pick_asyncProcessResponse xmlns="http://xmlns.oracle.com/CFP16_pick_async">
<result>cAlarm</result>
    </CFP16_pick_asyncProcessResponse>
    </part>
    </outputVariable>
    </onAlarm>
    </pick>
```

callbackClient

[2005/08/10 16:33:29] Skipped callback "onResult" on partner "client". less

```
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP16_pick_asyncProcessResponse xmlns="http://xmlns.oracle.com/CFP16_pick_async">
<result>cAlarm</result>
    </CFP16_pick_asyncProcessResponse>
    </part>
    </outputVariable>
    </sequence>
```

[2005/08/10 16:33:29] BPEL process instance "247" completed

```
</process>
```

CFP17 Interleaved Parallel Routing

Description: A set of activities is executed in an arbitrary order. Each activity in the set is executed exactly once. The order between the activities is decided at run-time: it is not until one activity is completed that the decision on what to do next is taken. In any case, no two activities in the set can be active at the same time.

Although BPEL specification supports this pattern by means of the serializable scopes, Oracle BPEL PM doesn't seem to support it. Having two scopes placed on the independent branches of the <flow> (see Figure 14) and setting in both scopes variableAccessSerializable="yes", an access to the input variable should be exclusive. However, as it is visualized in the audit trail, after one branch modified the value of the input variable, the other branch is propagated the initial value of the input variable, but not the newly assigned one, which is against the BPEL specification.

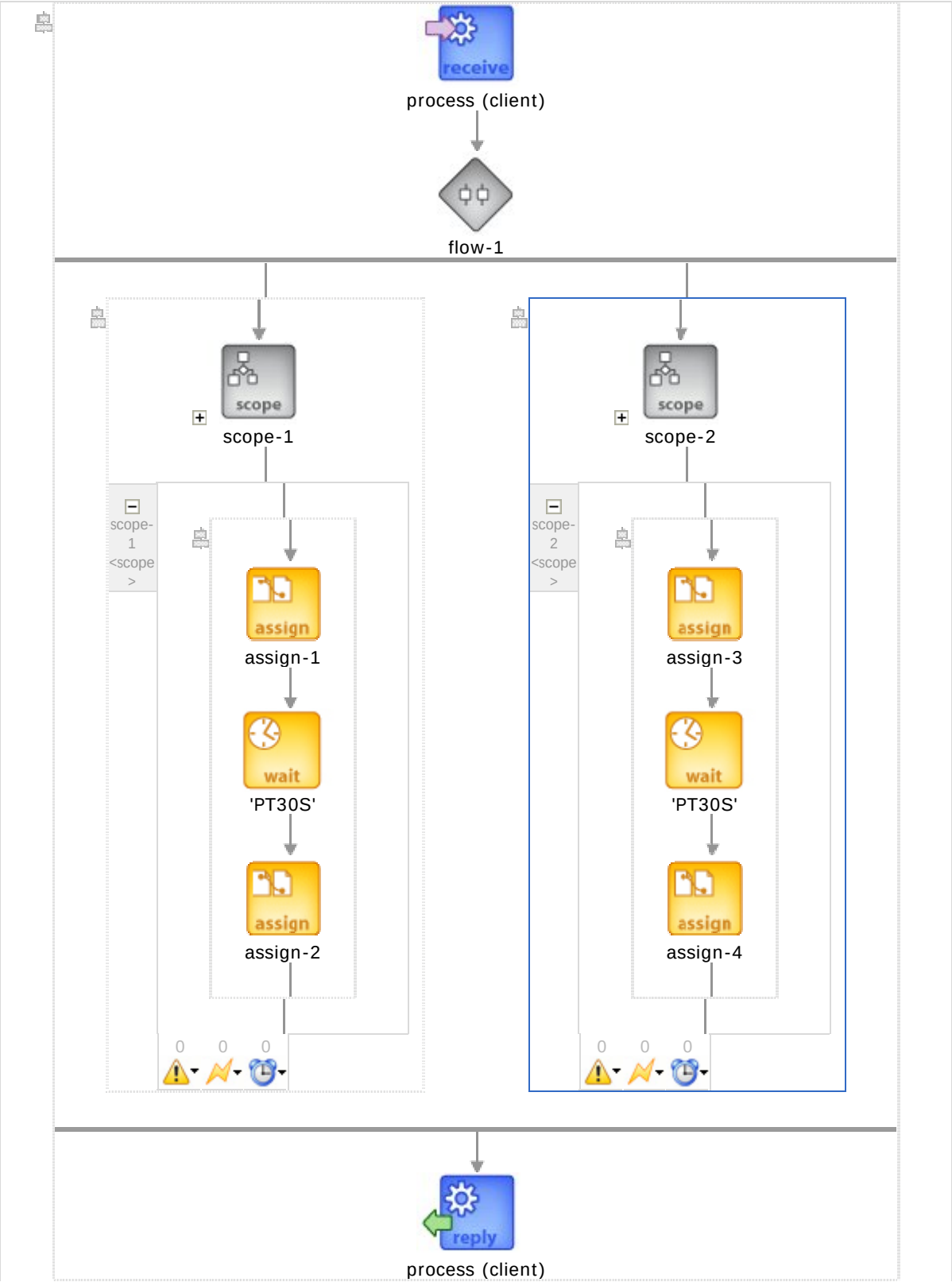


Figure 14 Interleaved parallel routing

The source code corresponding to the Figure 14 is shown below:

```
<sequence xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-
process/" name="main">
  <!-- Receive input from requester.
             Note: This maps to operation defined in WP17.wsdl
             -->
  <receive name="receiveInput" partnerLink="client" portType="tns:WP17"
operation="process" variable="input" createInstance="yes"/>
  <!-- Generate reply to synchronous request -->
  <flow name="flow-1">
    <sequence name="flow-sequence-1"><scope name="scope-1"
variableAccessSerializable="yes"><sequence name="sequence-1"><assign
name="assign-1">
            <copy>
              <from expression="&quot;Start left&quot;"></from>
              <to variable="input" part="payload"
query="/tns:WP17Request"/>
            </copy>
          </assign>
          <wait for="'PT30S'" name="wait-1"/>
          <assign name="assign-2">
            <copy>
              <from expression="&quot;End left&quot;"></from>
              <to variable="input" part="payload"
query="/tns:WP17Request"/>
            </copy>
          </assign>
        </sequence>
      </scope>
    </sequence>
    <sequence name="flow-sequence-2"><scope name="scope-2"
variableAccessSerializable="yes"><sequence><assign name="assign-3">
            <copy>
              <from expression="&quot;Start right&quot;"></from>
              <to variable="input" part="payload"
query="/tns:WP17Request"/>
            </copy>
          </assign>
          <wait for="'PT30S'" name="wait-2"/>
          <assign name="assign-4">
            <copy>
              <from expression="&quot;End right&quot;"></from>
              <to variable="input" part="payload"
query="/tns:WP17Request"/>
            </copy>
          </assign>
        </sequence>
      </scope>
    </sequence>
  </flow>
  <reply name="replyOutput" partnerLink="client" portType="tns:WP17"
operation="process" variable="output"/>
</sequence>
```

An audit trail showing the execution history of the considered process is given below:

[2005/07/13 17:31:28] New instance of BPEL process "WP17" initiated (# "1304").
<process>
<sequence>
client (process)
[2005/07/13 17:31:28] Received "input" call from partner "client" More...
<flow>
<sequence>
<scope name="scope-1">
<sequence>
assign-3
[2005/07/13 17:31:28] Updated variable "input" Less
<input>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<WP17Request xmlns="http://acm.org/samples">
<input>start</input>
Start right
</WP17Request>
</part>
</input>
2005.07.13 05:31
[2005/07/13 17:31:28] Waiting for the expiry time "7/13/05 5:31 PM".
[2005/07/13 17:31:58] Wait has finished.
assign-4
[2005/07/13 17:31:58] Updated variable "input" Less
<input>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<WP17Request xmlns="http://acm.org/samples">
<input>start</input>
End right
</WP17Request>
</part>
</input>
</sequence>
<scope>
</sequence>
<sequence>
<scope name="scope-1">
<sequence>
assign-1
[2005/07/13 17:31:28] Updated variable "input" Less
<input>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<WP17Request xmlns="http://acm.org/samples">
<input>start</input>

Start left
</WP17Request>
</part>
</input>
2005.07.13 05:31
[2005/07/13 17:31:28] Waiting for the expiry time "7/13/05 5:31 PM".
[2005/07/13 17:31:58] Wait has finished.
assign-2
[2005/07/13 17:31:58] Updated variable "input" Less
<input>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<WP17Request xmlns="http://acm.org/samples">
<input>start</input>
End left
</WP17Request>
</part>
</input>
</sequence>
<scope>
</sequence>
</flow>
client
[2005/07/13 17:31:58] Reply to partner "client". Less
<output>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<WP17Response xmlns="http://acm.org/samples" />
</part>
</output>
</sequence>
[2005/07/13 17:31:58] BPEL process instance "1304" completed
</process>

CFP18 Milestone
Description: A given activity E can only be enabled if a certain milestone has been reached which has not yet expired. A milestone is defined as a point in the process where a given activity A has finished and an activity B following it has not yet started.

Oracle BPEL PM does not offer a direct support for this pattern in terms of the single activity, but the workaround can be found by means of the <pick> construct placed in the <while> loop, which can be executed only once (see Figure 15). When executing this process model, the client specifies whether the activity E or an activity B is to be executed. If the message *onE* has been received, then the activity E, i.e. *actE,* followed by the activity B, i.e. *<reply>*, is executed. Otherwise, if the message *onB* has been received, the activity B (*<reply>*) will be executed and the activity E won't be able to execute any more. Each of the branches in the <pick> construct contains the assignment activity, which is needed for updating the loop counter.

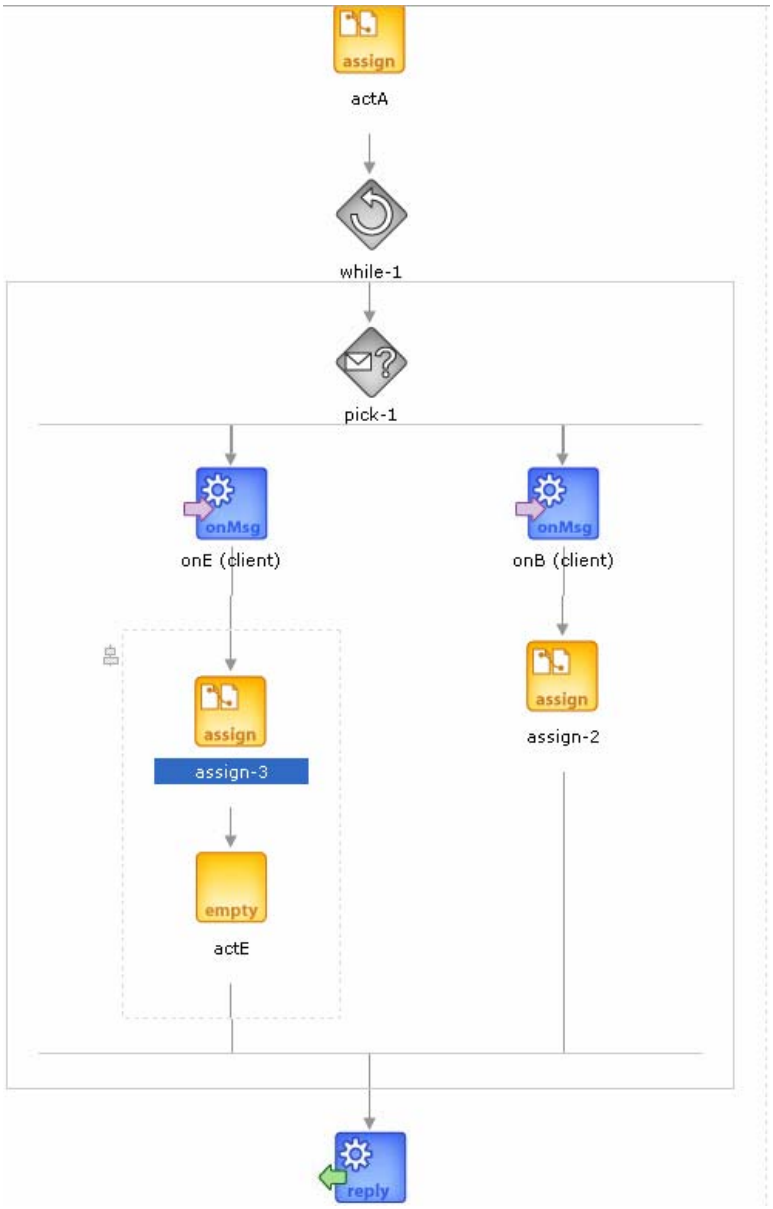


Figure 15 Milestone

The source code corresponding to the process visualized in Figure 15 is given below:

```
<sequence xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-
process/" name="main">
  <!-- Receive input from requester.
            Note: This maps to operation defined in Milestone.wsdl
            -->
  <receive name="receiveInput" partnerLink="client"
portType="tns:Milestone" operation="process" variable="input"
createInstance="yes"/>
  <!-- Generate reply to synchronous request -->
  <assign name="actA">
    <copy>
      <from expression="false()"></from>
      <to variable="B_chosen"/>
```

```
    </copy>
  </assign>
  <while name="while-1"
condition="not(bpws:getVariableData('B_chosen'))"><pick name="pick-1">
      <onMessage partnerLink="client" portType="tns:Milestone"
operation="onE" variable="E_var">
        <sequence><assign name="assign-3">
            <copy>
              <from expression="true()"></from>
              <to variable="B_chosen"/>
            </copy>
          </assign>
          <empty name="actE"/>
        </sequence>
      </onMessage>
      <onMessage partnerLink="client" portType="tns:Milestone"
operation="onB" variable="B_var">
        <assign name="assign-2">
          <copy>
            <from expression="true()"></from>
            <to variable="B_chosen"/>
          </copy>
        </assign>
      </onMessage>
    </pick>
  </while>
  <reply name="replyOutput" partnerLink="client"
portType="tns:Milestone" operation="process" variable="output"/>
</sequence
```

The declaration of data types, messages, port types and partner links are shown below:

```
<?xml version="1.0"?>
<definitions name="Milestone"
        targetNamespace="http://acm.org/samples"
        xmlns:tns="http://acm.org/samples"
        xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
        xmlns="http://schemas.xmlsoap.org/wsdl/"
        ><!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     TYPE DEFINITION - List of types participating in this BPEL process
     The BPEL Designer will generate default request and response types
     but you can define or import any XML Schema type and use them as
part
     of the message types.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <types>
        <schema attributeFormDefault="qualified"
elementFormDefault="qualified"
                targetNamespace="http://acm.org/samples"
                xmlns="http://www.w3.org/2001/XMLSchema">

            <element name="MilestoneRequest">
                <complexType>
                    <sequence>
```

```
                        <element name="input" type="string"/>
                    </sequence>
                </complexType>
            </element>

            <element name="MilestoneResponse">
                <complexType>
                    <sequence>
                        <element name="result" type="string"/>
                    </sequence>
                </complexType>
            </element>

            <element name="MilestoneSend">
                <complexType>
                    <sequence>
                        <element name="result" type="string"/>
                    </sequence>
                </complexType>
            </element>

        </schema>
    </types>
<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     MESSAGE TYPE DEFINITION - Definition of the message types used as
     part of the port type definitions
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <message name="MilestoneRequestMessage">
        <part name="payload" element="tns:MilestoneRequest"/>
    </message>
    <message name="MilestoneResponseMessage">
        <part name="payload" element="tns:MilestoneResponse"/>
    </message>
   <message name="MilestoneEMessage">
        <part name="payload" element="tns:MilestoneSend"/>
    </message>
       <message name="MilestoneBMessage">
        <part name="payload" element="tns:MilestoneSend"/>
    </message>
<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     PORT TYPE DEFINITION - A port type groups a set of operations into
     a logical service unit.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <!-- portType implemented by the Milestone BPEL process -->
    <portType name="Milestone">
        <operation name="process">
            <input  message="tns:MilestoneRequestMessage" />
            <output message="tns:MilestoneResponseMessage"/>
        </operation>
        <operation name="onE">
            <input message="tns:MilestoneEMessage"/>
        </operation>
        <operation name="onB">
```

```
            <input message="tns:MilestoneBMessage"/>
        </operation>
    </portType>

<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     PARTNER LINK TYPE DEFINITION
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <plnk:partnerLinkType name="Milestone">
        <plnk:role name="MilestoneProvider">
            <plnk:portType name="tns:Milestone"/>
        </plnk:role>
       <plnk:role name="MilestoneSender">
            <plnk:portType name="tns:MilestoneSend"/>
        </plnk:role>

    </plnk:partnerLinkType>

</definitions>
```

An audit trail visualizing the execution history of the considered process is shown below:

[2005/07/11 14:12:24] New instance of BPEL process "Milestone" initiated (# "1209").
<process>
<sequence>
client (process)
[2005/07/11 14:12:24] Received "input" call from partner "client" More...
actA
[2005/07/11 14:12:24] Updated variable "B_chosen" More...
<while>
<pick>
client (onB) (cancelled)
[2005/07/11 14:12:24] Waiting for message from "client", operation is "onB".
[2005/07/11 14:12:31] Receive activity has been cancelled.
client (onE)
[2005/07/11 14:12:24] Waiting for message from "client", operation is "onE".
[2005/07/11 14:12:30] Received "onE" callback from partner "client" Less
<E_var>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<MilestoneSend xmlns="http://acm.org/samples">
<result>10</result>
</MilestoneSend>
</part>
</E_var>
<onMessage>
<sequence>
assign-3
[2005/07/11 14:12:31] Updated variable "B_chosen" More...

Empty

[2005/07/11 14:12:31] BPEL "empty" activity is executed.

</sequence>

</onMessage>

</pick>

</while>

client

[2005/07/11 14:12:31] Reply to partner "client". More...

</sequence>

[2005/07/11 14:12:31] BPEL process instance "1209" completed

</process>

Not that the offered solution is very specific and does not allow other types of the milestone to be supported.

Another possibility to implement the Milestone is by means of <while>, <wait> and <flow>, as it is shown in Figure 16. An activity placed in the <while> may execute multiple times after an activity *Assign_2* has been executed but before an activity *Assign_3* has been executed. The status of the activity *Assign_3* is modeled by means of a Boolean variable *Milestone*, which is set to true after this activity has been executed. Thus, while *Milestone=false* the activities places in the <while> loop may be executed.

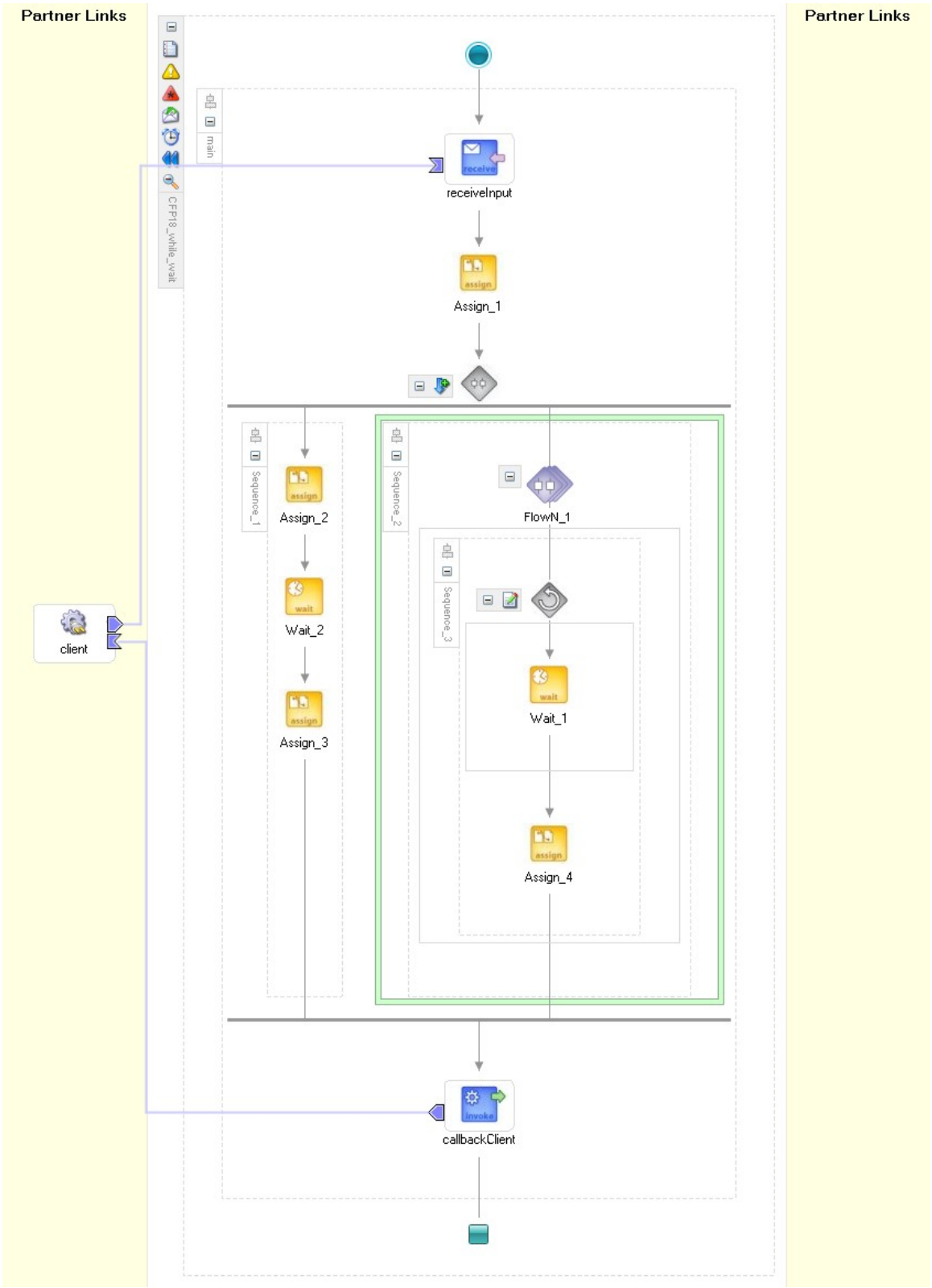


Figure 16 Milestone- the alternative implementation

```
<process name="CFP18_while_wait"
targetNamespace="http://xmlns.oracle.com/CFP18_while_wait"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.Xpath20" xmlns:ns1="http://www.w3.org/2001/XMLSchema"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:client="http://xmlns.oracle.com/CFP18_while_wait"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.ExtFunc"><!--
================================================================= --><!-
- PARTNERLINKS                                                      --
><!-- List of services participating in this BPEL process
--><!--
================================================================= -->
  <partnerLinks><!--
    The 'client' role represents the requester of this service. It is
    used for callback. The location and correlation information
associated
    with the client role are automatically set using WS-Addressing.
    -->
    <partnerLink name="client" partnerLinkType="client:CFP18_while_wait"
myRole="CFP18_while_waitProvider"
partnerRole="CFP18_while_waitRequester"/>
  </partnerLinks><!--
================================================================= --><!-
- VARIABLES                                                         --
><!-- List of messages and XML documents used within this BPEL process
--><!--
================================================================= -->
  <variables><!-- Reference to the message passed as input during
initiation --><!-- Reference to the message that will be sent back to
the
    requester during callback
    -->
    <variable name="inputVariable"
messageType="client:CFP18_while_waitRequestMessage"/>
    <variable name="outputVariable"
messageType="client:CFP18_while_waitResponseMessage"/>
    <variable name="Milestone" type="ns1:boolean"/>
    <variable name="FlowN_1_Variable" type="xsd:int"/>
  </variables><!--
================================================================= --><!-
- ORCHESTRATION LOGIC                                               --
><!-- Set of activities coordinating the flow of messages across the
--><!-- services integrated within this business process
--><!--
================================================================= -->
  <sequence name="main"><!-- Receive input from requestor.
    Note: This maps to operation defined in CFP18_while_wait.wsdl
    -->
    <receive name="receiveInput" partnerLink="client"
portType="client:CFP18_while_wait" operation="initiate"
```

```
variable="inputVariable" createInstance="yes"/><!-- Asynchronous
callback to the requester.
     Note: the callback location and correlation id is transparently
handled
     using WS-addressing.
     -->
    <assign name="Assign_1">
      <copy>
        <from expression="false()"/>
        <to variable="Milestone"/>
      </copy>
      <copy>
        <from variable="inputVariable" part="payload"
query="/client:CFP18_while_waitProcessRequest/client:input"/>
        <to variable="outputVariable" part="payload"
query="/client:CFP18_while_waitProcessResponse/client:result"/>
      </copy>
    </assign>
    <flow name="Flow_1">
      <sequence name="Sequence_2">
        <bpelx:flowN name="FlowN_1" N="3"
indexVariable="FlowN_1_Variable">
          <sequence name="Sequence_3">
            <while name="While_1"
condition="bpws:getVariableData('Milestone') = false()">
              <wait name="Wait_1" for="'PT1S'"/>
            </while>
            <assign name="Assign_4">
              <copy>
                <from
expression="concat(bpws:getVariableData('outputVariable','payload','/cli
ent:CFP18_while_waitProcessResponse/client:result'), '.')"/>
                <to variable="outputVariable" part="payload"
query="/client:CFP18_while_waitProcessResponse/client:result"/>
              </copy>
            </assign>
          </sequence>
        </bpelx:flowN>
      </sequence>
      <sequence name="Sequence_1">
        <assign name="Assign_2">
          <copy>
            <from expression="true()"/>
            <to variable="Milestone"/>
          </copy>
        </assign>
        <wait name="Wait_2" for="'PT10S'"/>
        <assign name="Assign_3">
          <copy>
            <from expression="false()"/>
            <to variable="Milestone"/>
          </copy>
        </assign>
      </sequence>
    </flow>
```

```
    <invoke name="callbackClient" partnerLink="client"
portType="client:CFP18_while_waitCallback" operation="onResult"
inputVariable="outputVariable"/>
  </sequence>
</process>
```

The wsdl code associated with the considered process model is given below:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="CFP18_while_wait"
             targetNamespace="http://xmlns.oracle.com/CFP18_while_wait"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:client="http://xmlns.oracle.com/CFP18_while_wait"
             xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-
link/">
    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    TYPE DEFINITION - List of services participating in this BPEL
process
    The default output of the BPEL designer uses strings as input and
    output to the BPEL Process. But you can define or import any XML
    Schema type and us them as part of the message types.
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <types>
        <schema attributeFormDefault="qualified"
            elementFormDefault="qualified"
            targetNamespace="http://xmlns.oracle.com/CFP18_while_wait"
            xmlns="http://www.w3.org/2001/XMLSchema">
            <element name="CFP18_while_waitProcessRequest">
                <complexType>
                    <sequence>
                        <element name="input" type="string"/>
                    </sequence>
                </complexType>
            </element>
            <element name="CFP18_while_waitProcessResponse">
                <complexType>
                    <sequence>
                        <element name="result" type="string"/>
                    </sequence>
                </complexType>
            </element>
        </schema>
    </types>

    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    MESSAGE TYPE DEFINITION - Definition of the message types used as
    part of the port type defintions
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <message name="CFP18_while_waitRequestMessage">
        <part name="payload"
element="client:CFP18_while_waitProcessRequest"/>
    </message>
```

```
    <message name="CFP18_while_waitResponseMessage">
        <part name="payload"
element="client:CFP18_while_waitProcessResponse"/>
    </message>
    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    PORT TYPE DEFINITION - A port type groups a set of operations into
    a logical service unit.
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <!-- portType implemented by the CFP18_while_wait BPEL process -->
    <portType name="CFP18_while_wait">
        <operation name="initiate">
            <input message="client:CFP18_while_waitRequestMessage"/>
        </operation>
    </portType>

    <!-- portType implemented by the requester of CFP18_while_wait BPEL
process
    for asynchronous callback purposes
    -->
    <portType name="CFP18_while_waitCallback">
        <operation name="onResult">
            <input message="client:CFP18_while_waitResponseMessage"/>
        </operation>
    </portType>
    <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    PARTNER LINK TYPE DEFINITION
    the CFP18_while_wait partnerLinkType binds the provider and
    requester portType into an asynchronous conversation.
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <plnk:partnerLinkType name="CFP18_while_wait">
        <plnk:role name="CFP18_while_waitProvider">
            <plnk:portType name="client:CFP18_while_wait"/>
        </plnk:role>
        <plnk:role name="CFP18_while_waitRequester">
            <plnk:portType name="client:CFP18_while_waitCallback"/>
        </plnk:role>
    </plnk:partnerLinkType>
</definitions>
```

An audit trail demonstrating the execution history of the considered process is shown below:

[2005/08/10 16:49:50] New instance of BPEL process "CFP18_while_wait" initiated (# "249").

<process>

<sequence>

receiveInput

[2005/08/10 16:49:50] Received "inputVariable" call from partner "client" less

<inputVariable>

<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">

<CFP18_while_waitProcessRequest xmlns="http://xmlns.oracle.com/CFP18_while_wait">
<input>a</input>
</CFP18_while_waitProcessRequest>
</part>
</inputVariable>

Assign_1

[2005/08/10 16:49:50] Updated variable "Milestone" less
<Milestone>false</Milestone>
[2005/08/10 16:49:50] Updated variable "outputVariable" less
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP18_while_waitProcessResponse xmlns="http://xmlns.oracle.com/CFP18_while_wait">
<result>a</result>
</CFP18_while_waitProcessResponse>
</part>
</outputVariable>
<flow>
<sequence>

Assign_2

[2005/08/10 16:49:50] Updated variable "Milestone" less
<Milestone>true</Milestone>

Wait_2

[2005/08/10 16:49:50] Waiting for the expiry time "2005/08/10 16:50:00".
[2005/08/10 16:50:00] Wait has finished.

Assign_3

[2005/08/10 16:50:00] Updated variable "Milestone" less
<Milestone>false</Milestone>
</sequence>
<sequence>
<flowN>
<FlowN_1_Variable=0>
<sequence>
<while>
</while>

Assign_4

[2005/08/10 16:49:50] Updated variable "outputVariable" less
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP18_while_waitProcessResponse xmlns="http://xmlns.oracle.com/CFP18_while_wait">
<result>a..</result>
</CFP18_while_waitProcessResponse>
</part>
</outputVariable>
</sequence>
</FlowN_1_Variable=0>
<FlowN_1_Variable=1>

<sequence>
<while>
</while>
Assign_4
[2005/08/10 16:49:50] Updated variable "outputVariable" less
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP18_while_waitProcessResponse xmlns="http://xmlns.oracle.com/CFP18_while_wait">
<result>a...</result>
</CFP18_while_waitProcessResponse>
</part>
</outputVariable>
</sequence>
</FlowN_1_Variable=1>
<FlowN_1_Variable=2>
<sequence>
<while>
</while>
Assign_4
[2005/08/10 16:49:50] Updated variable "outputVariable" less
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP18_while_waitProcessResponse xmlns="http://xmlns.oracle.com/CFP18_while_wait">
<result>a.</result>
</CFP18_while_waitProcessResponse>
</part>
</outputVariable>
</sequence>
</FlowN_1_Variable=2>
</flowN>
</sequence>
</flow>
callbackClient
[2005/08/10 16:50:00] Skipped callback "onResult" on partner "client". less
<outputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
<CFP18_while_waitProcessResponse xmlns="http://xmlns.oracle.com/CFP18_while_wait">
<result>a...</result>
</CFP18_while_waitProcessResponse>
</part>
</outputVariable>
</sequence>
[2005/08/10 16:50:00] BPEL process instance "249" completed
</process>

CFP19 Cancel Activity
Description: A cancel activity terminates a running instance of an activity.

An Oracle BPEL PM provides support for this pattern by means of using a scope as wrapper for an activity that should be canceled (see Figure 17), defining a fault that must be thrown when the client cancels the activity (see Figure 19) and the fault handler to catch this fault message (see Figure 18).

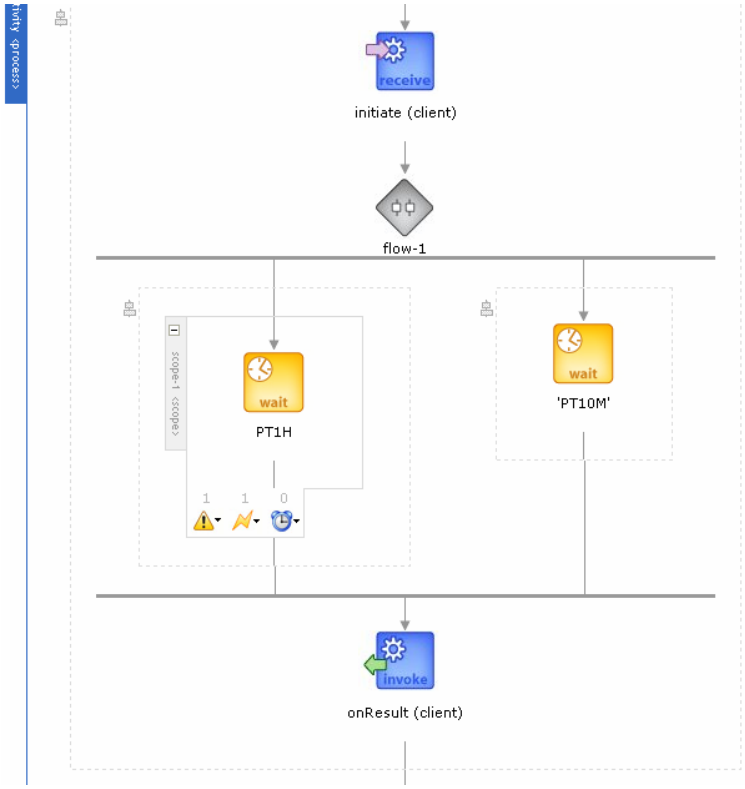


Figure 17 Cancel activity

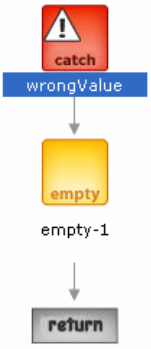


Figure 18 Catch fault

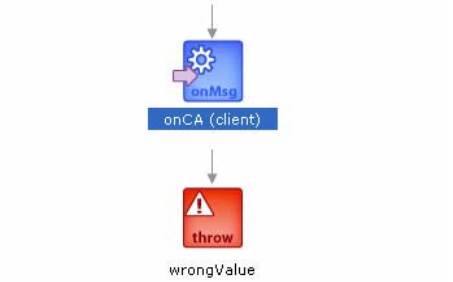


Figure 19 Throw fault

The process model of the canceling a *wait* activity *PT1H* is shown in Figure 17, while the corresponding source code is shown below:

```
<!-- TestCancelActivity BPEL Process [Generated by the Oracle BPEL
Designer]  -->
<process name="TestCancelActivity"
targetNamespace="http://acm.org/samples" suppressJoinFailure="yes"
xmlns:tns="http://acm.org/samples"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:ora="http://schemas.oracle.com/xpath/extension">
      <!--
================================================================= -->
      <!-- PARTNERLINKS
-->
      <!-- List of services participating in this BPEL process
-->
      <!--
================================================================= -->
      <partnerLinks>
            <!--
        The 'client' role represents the requester of this service. It
is
        used for callback. The location and correlation information
associated
        with the client role are automatically set using WS-Addressing.
        -->
            <partnerLink name="client"
partnerLinkType="tns:TestCancelActivity"
myRole="TestCancelActivityProvider"
partnerRole="TestCancelActivityRequester"/>
      </partnerLinks>
      <!--
================================================================= -->
      <!-- VARIABLES
-->
      <!-- List of messages and XML documents used within this BPEL
process  -->
      <!--
================================================================= -->
      <variables>
```

```
            <!-- Reference to the message passed as input during
initiation -->
            <variable name="input"
messageType="tns:TestCancelActivityRequestMessage"/>
            <!-- Reference to the message that will be sent back to the
             requester during callback
             -->
            <variable name="output"
messageType="tns:TestCancelActivityResponseMessage"/>
            <variable name="ca"
messageType="tns:TestCancelActivityCAMessage"/>
      </variables>
      <!--
================================================================= -->
      <!-- ORCHESTRATION LOGIC
-->
      <!-- Set of activities coordinating the flow of messages across
the    -->
      <!-- services integrated within this business process
-->
      <!--
================================================================= -->
      <sequence name="main">
            <!-- Receive input from requestor.
             Note: This maps to operation defined in
TestCancelActivity.wsdl
             -->
            <receive name="receiveInput" partnerLink="client"
portType="tns:TestCancelActivity" operation="initiate" variable="input"
createInstance="yes"/>
            <!-- Asynchronous callback to the requester.
             Note: the callback location and correlation id is
transparently handled
             using WS-addressing.
             -->
            <flow name="flow-1">
                  <sequence name="flow-sequence-1"><scope name="scope-
1"><faultHandlers>
                                    <catch faultName="wrongValue"><empty
name="empty-1"/>
                                    </catch>
                              </faultHandlers>
                              <eventHandlers>
                                    <onMessage partnerLink="client"
portType="tns:TestCancelActivity" operation="onCA" variable="ca">
                                          <throw name="throw-1"
faultName="wrongValue"/>
                                    </onMessage>
                              </eventHandlers>
                              <wait for="PT1H" name="wait-2"/>
                        </scope>
                  </sequence>
                  <sequence name="flow-sequence-2"><wait for="'PT10M'"
name="wait-1"/>
                  </sequence>
            </flow>
```

```
            <invoke name="callbackClient" partnerLink="client"
portType="tns:TestCancelActivityCallback" operation="onResult"
inputVariable="output"/>
      </sequence>
</process>
```

Declarations of the data types, messages, partner links and port types are shown below:

```
<?xml version="1.0"?>
<definitions name="TestCancelActivity"
        targetNamespace="http://acm.org/samples"
        xmlns:tns="http://acm.org/samples"
        xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
        xmlns="http://schemas.xmlsoap.org/wsdl/"
        >

<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     TYPE DEFINITION - List of services participating in this BPEL
process
     The default output of the BPEL designer uses strings as input and
     output to the BPEL Process. But you can define or import any XML
     Schema type and us them as part of the message types.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <types>
        <schema attributeFormDefault="qualified"
                elementFormDefault="qualified"
                targetNamespace="http://acm.org/samples"
                xmlns="http://www.w3.org/2001/XMLSchema"
                >

            <element name="TestCancelActivityRequest">
                <complexType>
                    <sequence>
                        <element name="input" type="string" />
                    </sequence>
                </complexType>
            </element>

            <element name="TestCancelActivityResponse">
                <complexType>
                    <sequence>
                        <element name="result" type="string"/>
                    </sequence>
                </complexType>
            </element>

            <element name="TestCancelActivityCA">
                <complexType>
                    <sequence>
                        <element name="ca" type="string"/>
                    </sequence>
                </complexType>
            </element>
```

```
        </schema>
    </types>

<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     MESSAGE TYPE DEFINITION - Definition of the message types used as
     part of the port type defintions

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <message name="TestCancelActivityRequestMessage">
        <part name="payload" element="tns:TestCancelActivityRequest"/>
    </message>

    <message name="TestCancelActivityResponseMessage">
        <part name="payload" element="tns:TestCancelActivityResponse"/>
    </message>

    <message name="TestCancelActivityCAMessage">
        <part name="payload" element="tns:TestCancelActivityCA"/>
    </message>


<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     PORT TYPE DEFINITION - A port type groups a set of operations into
     a logical service unit.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <!-- portType implemented by the TestCancelActivity BPEL process -->
    <portType name="TestCancelActivity">
        <operation name="initiate">
            <input message="tns:TestCancelActivityRequestMessage"/>
        </operation>
    </portType>

    <!-- portType implemented by the requester of TestCancelActivity
BPEL process
         for asynchronous callback purposes
         -->
    <portType name="TestCancelActivityCallback">
        <operation name="onResult">
            <input message="tns:TestCancelActivityResponseMessage"/>
        </operation>
    </portType>

    <portType name="TestCancelActivity">
        <operation name="onCA">
            <input message="tns:TestCancelActivityCAMessage"/>
        </operation>
    </portType>


<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
     PARTNER LINK TYPE DEFINITION
         the TestCancelActivity partnerLinkType binds the provider and
         requester portType into an asynchronous conversation.
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <plnk:partnerLinkType name="TestCancelActivity">
        <plnk:role name="TestCancelActivityProvider">
            <plnk:portType name="tns:TestCancelActivity"/>
        </plnk:role>
        <plnk:role name="TestCancelActivityRequester">
            <plnk:portType name="tns:TestCancelActivityCallback"/>
        </plnk:role>
        <plnk:role name="TestCancelActivityCAer">
            <plnk:portType name="tns:TestCancelActivityCAPT"/>
        </plnk:role>
    </plnk:partnerLinkType>
</definitions>
```

An audit trail visualizing the execution history of the considered process is shown below:

[2005/07/15 11:04:44] New instance of BPEL process "TestCancelActivity" initiated (# "1503").

<process>

<sequence>

client (initiate)

[2005/07/15 11:04:44] Received "input" call from partner "client" More...

<flow>

<sequence>

2005.07.15 11:14 - pending

[2005/07/15 11:04:44] Waiting for the expiry time "7/15/05 11:14 AM".

<sequence>

<scope name="scope-1">

<eventHandlers>

client (onCA) (cancelled)

[2005/07/15 11:04:44] Waiting for message from "client", operation is "onCA".

[2005/07/15 11:04:58] Received "onCA" callback from partner "client" More...

[2005/07/15 11:04:58] Receive activity has been cancelled.

<onMessage>

Throw (faulted)

[2005/07/15 11:04:58] "{http://acm.org/samples}wrongValue" has been thrown. Less

```
<wrongValue xmlns="http://acm.org/samples" />
```

</onMessage>

</eventHandlers>

2005.07.15 12:04 (cancelled)

[2005/07/15 11:04:44] Waiting for the expiry time "7/15/05 12:04 PM".

<catchFault>

Empty

```
[2005/07/15 11:04:58]   BPEL "empty" activity is executed.
</catchFault>
<scope>
</sequence>
```

CFP20 Cancel Case
Description: A cancel case activity leads to removal of the whole workflow instance

Oracle BPEL PM supports this pattern directly by means of the <terminate> construct. The execution of the <terminate> activity leads to canceling of the whole process instance.

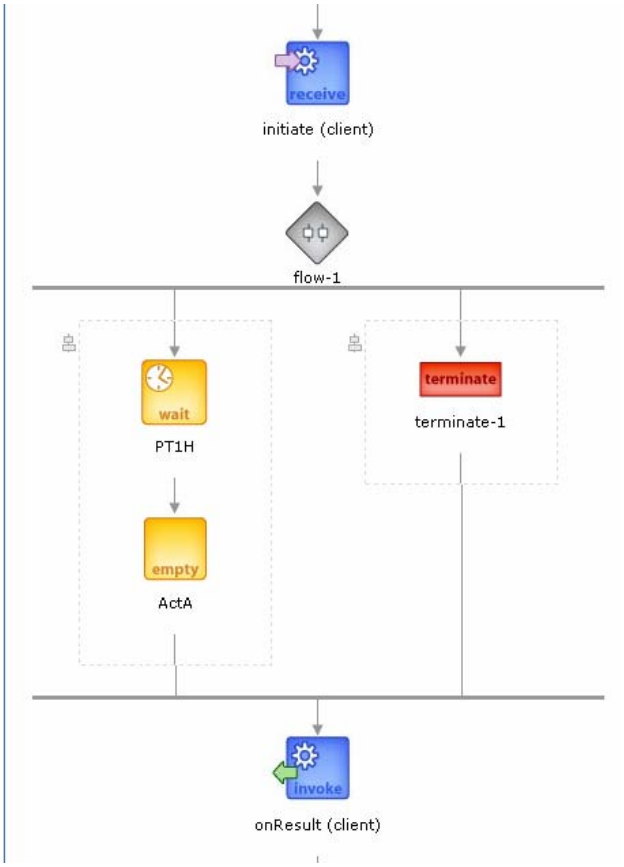


Figure 20 Cancel case

The source code corresponding to Figure 20 is shown below:

```
<sequence xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-
process/" name="main">
      <!-- Receive input from requestor.
             Note: This maps to operation defined in WP20.wsdl
             -->
      <receive name="receiveInput" partnerLink="client"
portType="tns:WP20" operation="initiate" variable="input"
createInstance="yes"/>
      <!-- Asynchronous callback to the requester.
```

```
            Note: the callback location and correlation id is
transparently handled
            using WS-addressing.
            -->
      <flow name="flow-1">
            <sequence name="flow-sequence-1"><wait for="PT1H"
name="wait-1"/>
                  <empty name="ActA"/>
            </sequence>
            <sequence name="flow-sequence-2"><terminate name="terminate-
1"/>
            </sequence>
      </flow>
      <invoke name="callbackClient" partnerLink="client"
portType="tns:WP20Callback" operation="onResult"
inputVariable="output"/>
</sequence>
```

An audit trail visualizing the execution history of the considered process is shown below:

[2005/07/13 17:47:19] New instance of BPEL process "WP20" initiated (# "1305").

<process>

<sequence>

client (initiate)

[2005/07/13 17:47:19] Received "input" call from partner "client" More...

<flow>

<sequence>

Terminate

[2005/07/13 17:47:19] Instance terminated.

<sequence>

2005.07.13 06:47 (aborted)

[2005/07/13 17:47:19] Waiting for the expiry time "7/13/05 6:47 PM".

[2005/07/13 17:47:19] Activity "2005.07.13 06:47" aborted

[2005/07/13 17:47:19] BPEL process instance "1305" aborted

2. Evaluation of Oracle BPEL PM from the data perspective

Data visibility

DP 1 (Task Data)
Description: Data elements can be defined by tasks which are accessible only within the context of individual execution instances of that task.

Oracle BPEL PM offers no direct support for this pattern. However, it can be accomplished through the <scope> construct. The <scope> is a collection of activities that can have its own local variables. The number of nested activities into a scope defines the visibility of data, i.e. if a scope contains only one task, then a variable defined within this scope will be visible only to this task but not to the higher levels.

Figure 21, Figure 22 and Figure 23 show how to declare variables local to a task via the properties of the <scope>.

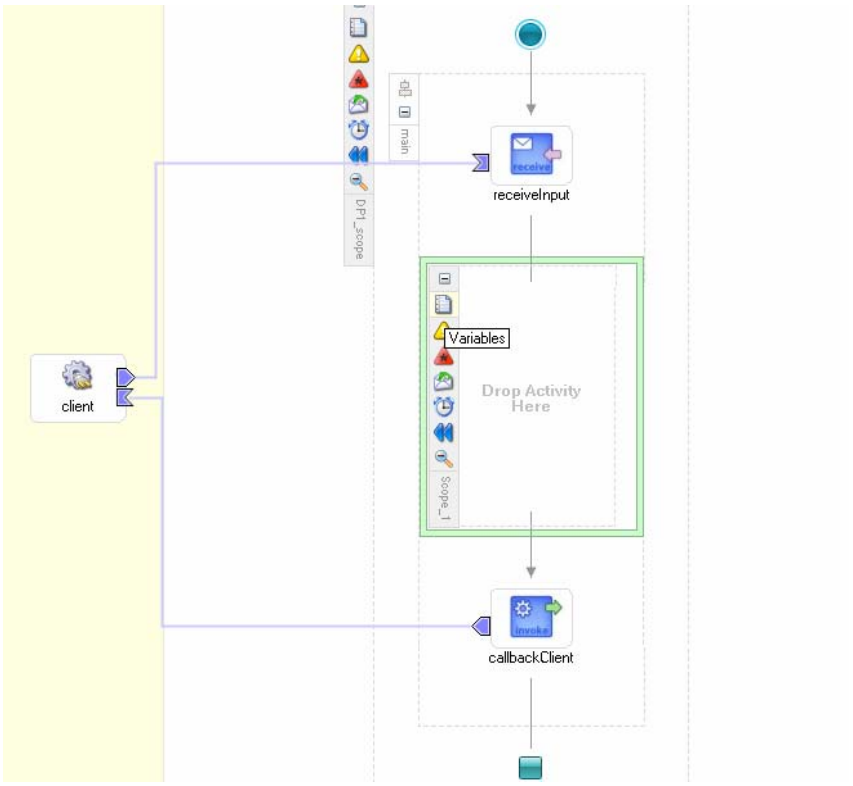


Figure 21 The <scope> construct

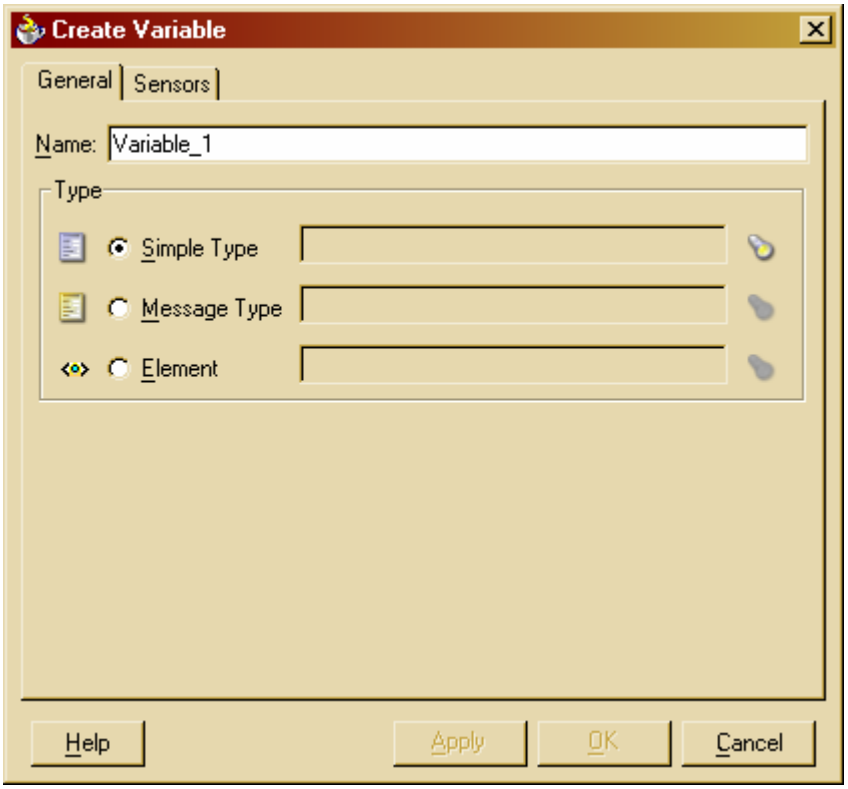


Figure 22 <scope>: create variables

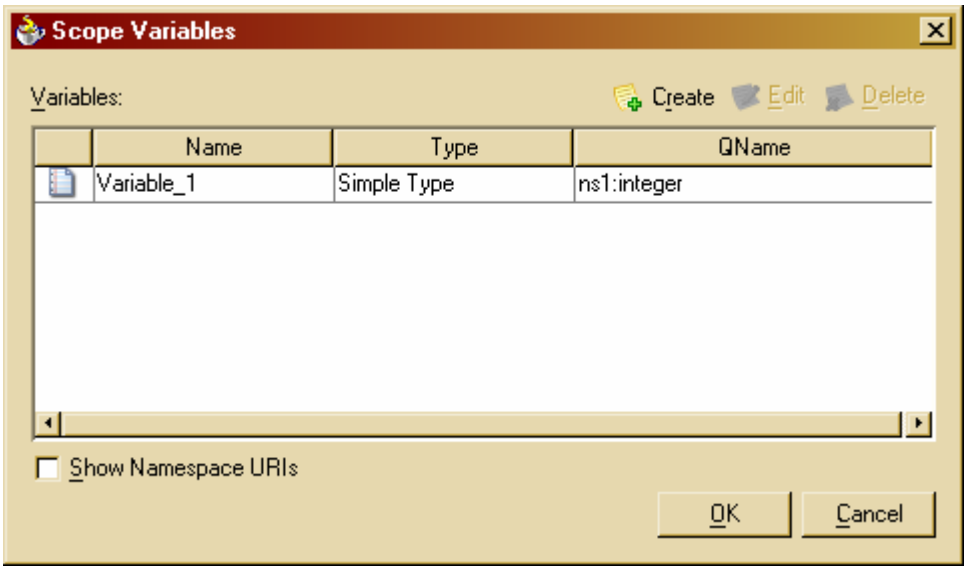


Figure 23 Scope variables

The source code associated with a process definition contains the section for declaration of variables, as it is shown below. Note that on the process scope level, any variable is treated as global, i.e. it is visible to all process components.

```
<variables>
<!-- Reference to the message passed as input during initiation -->
<variable name="input" messageType="tns:TestDataRequestMessage"/>
<!--
 Reference to the message that will be returned to the requester-->
<variable name="output" messageType="tns:TestDataResponseMessage"/>
<variable name="testlocalvar" type="xsd:date"/>
<variable name="testglobalvar" type="xsd:dateTime"/>
</variables>
```

To define a variable local to a scope, the declaration of the data variable should be performed in the boundaries of the <scope>.

```
Definition of scope variable:
<scope name="scope-1">
<variables>
      <variable name="testscopevar" type="xsd:boolean"/>
</variables>
      <assign name="assign-1">
            <copy>
                  <from expression="True;"></from>
                  <to variable="testscopevar"/>
            </copy>
      </assign>
</scope>
```

DP 2 (Block Data)
Description: Block tasks (i.e. tasks which can be described in terms of a corresponding sub-workflow) are able to define data elements which are accessible by each of the components of the corresponding sub-workflow.

Oracle BPEL PM: does not have a notion of a sub-workflow, it does not allow unfolding a block task to another process model, but is oriented towards web services. Therefore this pattern is not supported.

DP 3 (Scope Data)
Description: Data elements can be defined which are accessible by a subset of the tasks in a case.

Oracle BPEL PM supports this pattern directly. The process designer offers the <scope> construct which encompasses several tasks in the hierarchical manner. The data variables declared within the <scope> are visible and shared between all scope components.

DP 4 (Multiple Instance Data)
Description: Tasks which are able to execute multiple times within a single workflow case can define data elements which are specific to an individual execution instance.

Oracle BPEL PM supports this pattern partially. MI task without synchronization, when the number of instances is known at the design time, may have data variables associated with every separate task instance. Every task instance the parameter *createInstance* of which is set to *yes* has a distinct se of data elements. Every activity instantiated in this

way operates in its own address space. MI with synchronization, the number of instances of which is known at the design time, also may have data variables local to every instance. However this imposes the constraint that every task instance must be wrapped into the <scope>. MI task with synchronization, the number of instances of which is known at the run-time, is realized in Oracle BPEL PM by means of the <flowN> construct. On the one hand, the semantics of <flowN> is not specified and this construct is Oracle-specific. On the other hand the result of testing showed that all instances of the task that were created run-time share all data variables, thus are not instance-specific.

DP 5 (Case Data)
Description: Data elements are supported which are specific to a process instance or a workflow case. They can be accessed by all components of the workflow during the execution of the case.

Oracle BPEL PM supports this pattern by means of declaring data variables in the uppermost (process) scope. The declaration can be done directly in the source code of the process bpel source, as it is shown in Figure 24, or with the help of the graphical interface in the designer editor as it is shown in Figure 25.

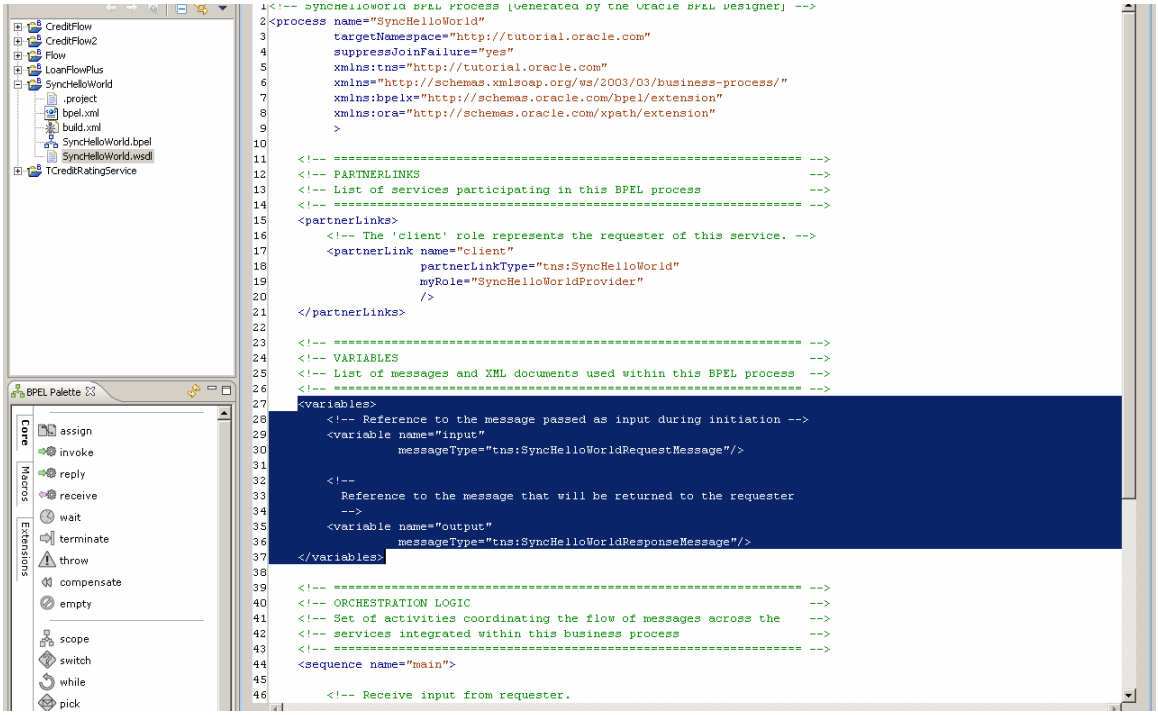


Figure 24 Defining global variables directly in the bpel source

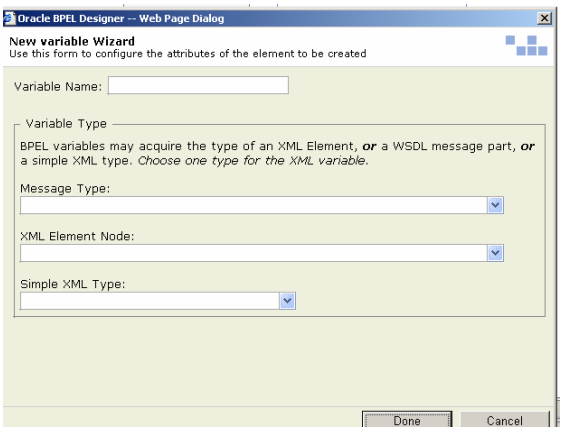


Figure 25 Data declaration wizard

DP 6 (Folder Data)
Description: Data elements can be defined which are accessible by multiple cases on a selective basis.

Oracle BPEL PM offers no support for this pattern.

DP 7 (Workflow Data)
Description: Data elements are supported which are accessible to all components in each and every case of the workflow and are within the control of the workflow system.

Oracle BPEL PM supports this pattern by means of preference properties that are defined in the deployment descriptor. Preferences are name-value properties that are accessed at run time by BPEL process. It allows changing the value of preferences without having to redeploy the BPEL process.

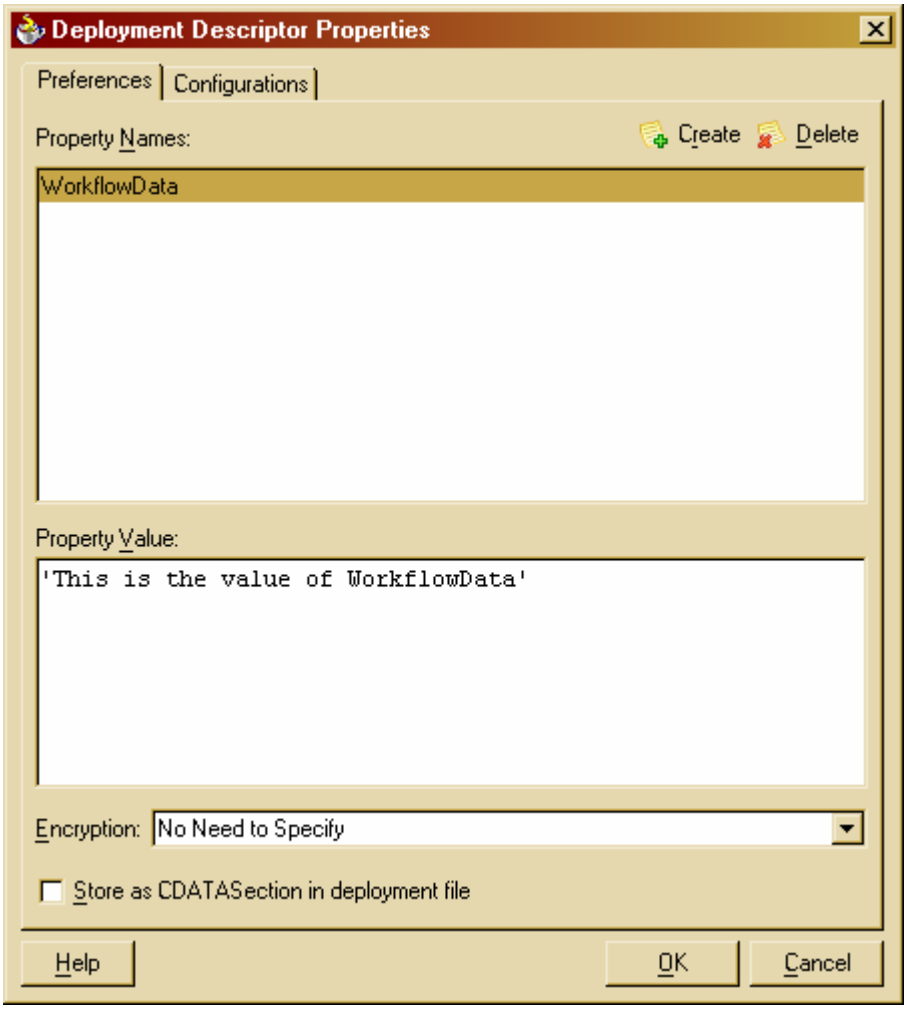
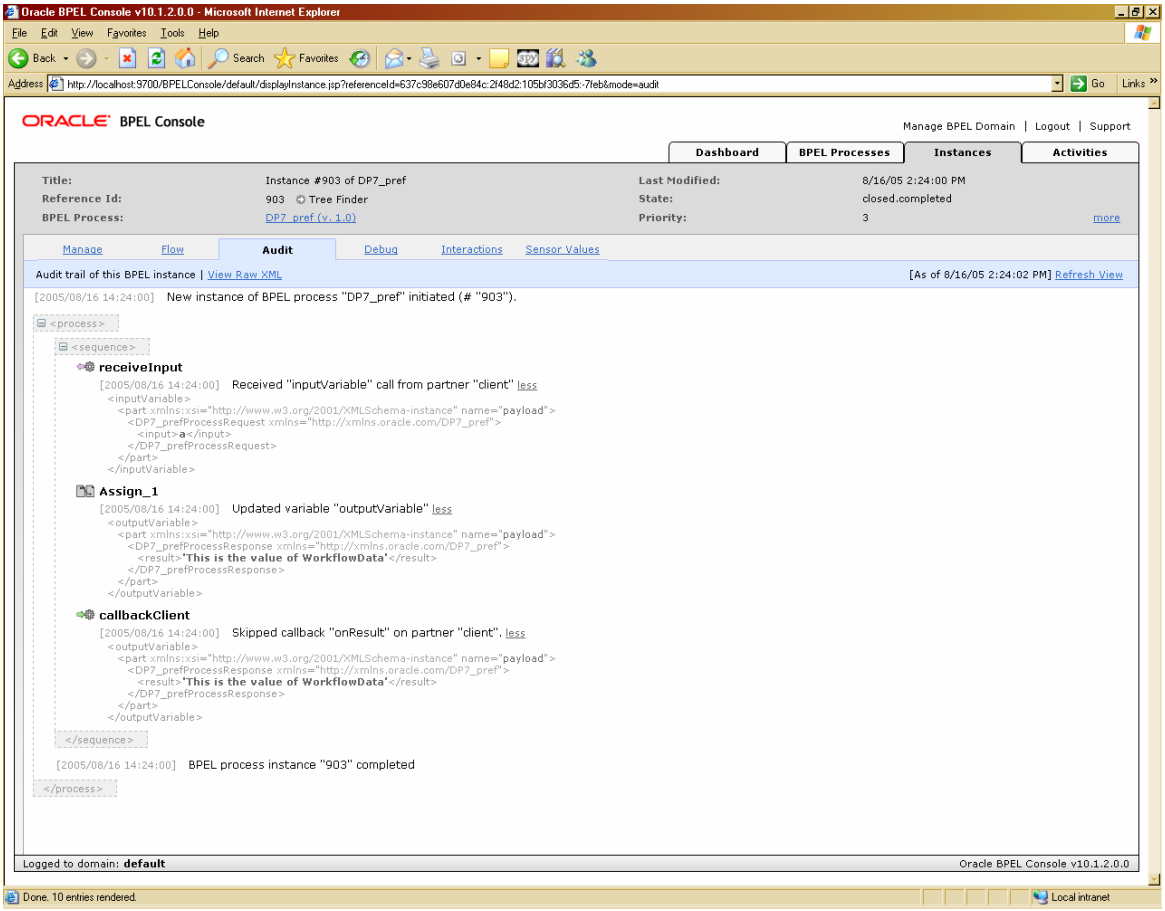


Figure 26 Deployment descriptor properties

An access to the preference property set in Figure 26 and its value are shown in the audit trail below.



DP 8 (Environment Data)
Description: Data elements which exist in the external operating environment are able to be accessed by components of the workflow during execution.

Oracle BPEL PM allows accessing data provided by external operating environment by means of synchronous invocation of an external service. In particular, by means of <invoke> an external service can be invoked, and the data wrapped into messages is sent by the invoked service back to the requester. The initiator of the request receives the message through <receive>.

BPEL code snippet illustrating the use of <invoke> is shown below:

```
<variable name="inputToCr"
messageType="nsxml0:CreditRatingServiceRequestMessage"/>
<variable name="outputFromCr"
messageType="nsxml0:CreditRatingServiceResponseMessage"/>
...
<invoke name="invoke-1" operation="process" inputVariable="inputToCr"
outputVariable="outputFromCr" partnerLink="CreditRatingService"
portType="nsxml0:CreditRatingService"/>
```

Note that the data types, message types and port types need to be defined in the wsdl format:

```
<?xml version="1.0"?>
<definitions name="TestMIwithoutSync"
        targetNamespace="http://acm.org/samples"
        xmlns:tns="http://acm.org/samples"
        xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
        xmlns="http://schemas.xmlsoap.org/wsdl/"
        >
<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     TYPE DEFINITION - List of types participating in this BPEL process
     The BPEL Designer will generate default request and response types
     but you can define or import any XML Schema type and use them as
part
     of the message types.
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <types>
            <schema attributeFormDefault="qualified"
            elementFormDefault="qualified"
                targetNamespace="http://acm.org/samples"
                xmlns="http://www.w3.org/2001/XMLSchema">

            <element name="TestMIwithoutSyncRequest">
                <complexType>
                    <sequence>
                        <element name="input" type="string"/>
                    </sequence>
                </complexType>
            </element>

            <element name="TestMIwithoutSyncResponse">
                <complexType>
                    <sequence>
                        <element name="result" type="integer"/>
                    </sequence>
                </complexType>
            </element>
        </schema>
    </types>

<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     MESSAGE TYPE DEFINITION - Definition of the message types used as
     part of the port type defintions
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <message name="TestMIwithoutSyncRequestMessage">
        <part name="payload" element="tns:TestMIwithoutSyncRequest"/>
    </message>
    <message name="TestMIwithoutSyncResponseMessage">
        <part name="payload" element="tns:TestMIwithoutSyncResponse"/>
    </message>

<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     PORT TYPE DEFINITION - A port type groups a set of operations into
```

```
     a logical service unit.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <!-- portType implemented by the TestMIwithoutSync BPEL process -->
    <portType name="TestMIwithoutSync">
        <operation name="process">
            <input  message="tns:TestMIwithoutSyncRequestMessage" />
            <output message="tns:TestMIwithoutSyncResponseMessage"/>
        </operation>
    </portType>
<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     PARTNER LINK TYPE DEFINITION
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <plnk:partnerLinkType name="TestMIwithoutSync">
        <plnk:role name="TestMIwithoutSyncProvider">
            <plnk:portType name="tns:TestMIwithoutSync"/>
        </plnk:role>
    </plnk:partnerLinkType>

</definitions>
```

Internal data interaction

DP 9 (Data Interaction- Task to Task)
Description: The ability to communicate data elements between one task instance and another within the same case.

Oracle BPEL PM adopts “no data passing” strategy, i.e. the flow of control is passed from one task to another, but the control-flow channel is not integrated with a data channel, nor there is a private data channel connecting tasks. Data within the same case are not passed between tasks but are made available via access to globally shared data.
An access to the data variables is accomplished via the <assign> activity. The <assign> activity allows the definition of Copy Rules based on the XPATH Query in order to assign a value of a variable, a part or a query (see Figure 27).

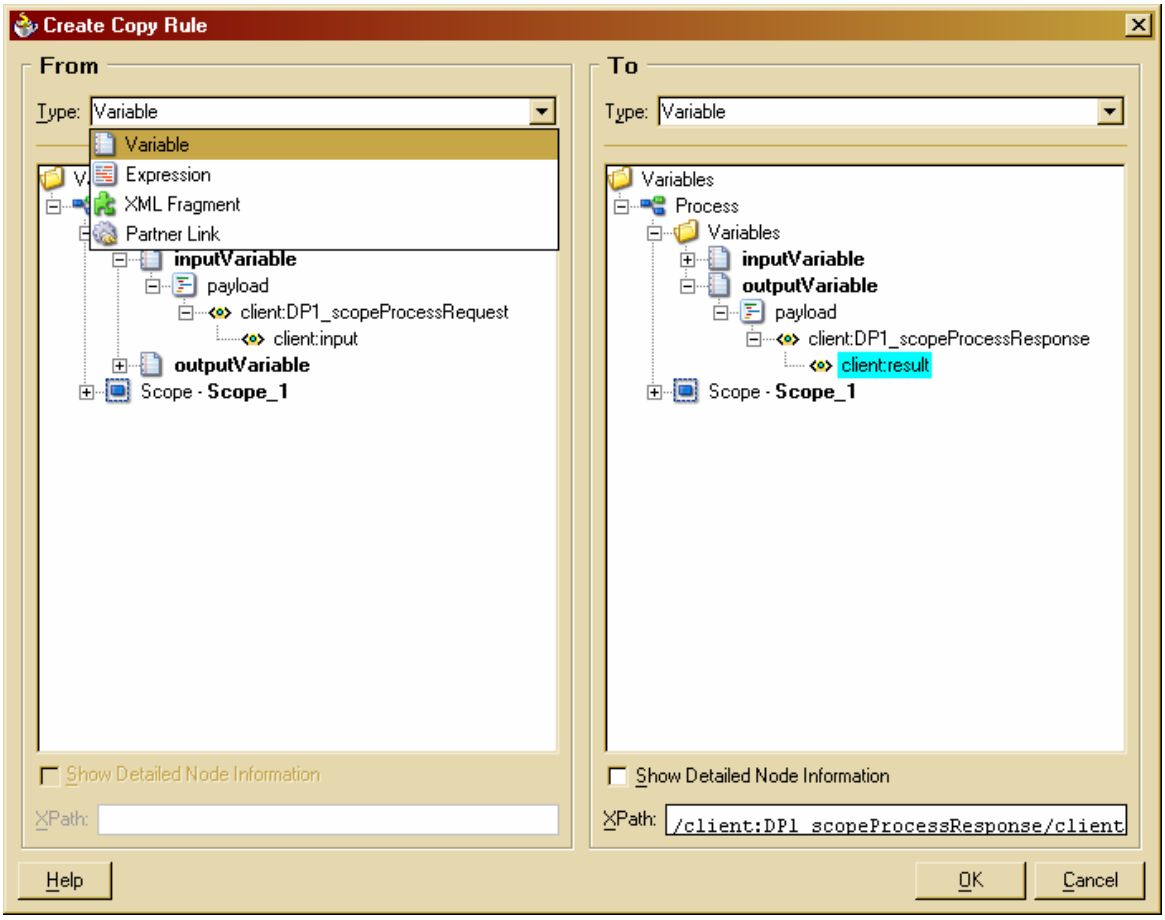


Figure 27 Assign activity

DP 10 (Data Interaction - Block Task to Sub-Workflow Decomposition)
Description: The ability to pass data elements from a block task instance to the corresponding sub-workflow that defines its implementation.

Not applicable, since the concept of the Block task, which can be unfolded to a sub-workflow, is not supported.

DP 11 (Data Interaction- Sub-Workflow Decomposition to Block Task)
Description: The ability to pass data elements from the underlying sub-workflow back to the corresponding block task instance.

Not applicable, since the concept of the Block task, which can be unfolded to a sub-workflow, is not supported.

DP 12 (Data Interaction- to Multiple Instance Task)
Description: The ability to pass data elements from a preceding task instance to a subsequent task which is able to support multiple execution instances. This may involve passing the data elements to all instances of the multiple instance task or distributing them on a selective basis.

Since not all variants of MI task are supported by Oracle BPEL PM, this patterns is respectively not supported directly. It is possible to pass data to all instances or to a specific instance of the MI task without synchronization. The same is valid for multiple instances task with synchronization created at the design-time. In addition, arrays of data can be used for passing data to specific task instances.

DP 13 (Data Interaction- from Multiple Instance Task)
Description: The ability to pass data elements from a task which supports multiple execution instances to a subsequent task.

Since multiple instances are not supported directly, this pattern is also not directly supported. However for the existing workarounds to implement MI task, the specified number of instances must complete before data can be passed to the subsequent task by means of the <assign> construct. In order to keep track of data associated with task instances created run-time (with or without a-priori knowledge) the array structure can be used. Any of the array elements can be passed to the subsequent task.

DP 14 (Data Interaction- Case to Case)
Description: The passing of data elements from one case of a workflow during its execution to another case that is executing concurrently.

Oracle BPEL PM does not support this pattern.

External data passing

DP 15 (Data Interaction- Task to Environment - Push-Oriented)
Description: The ability of a task to initiate the passing of data elements to a resource or service in the operating environment.

Oracle BPEL PM supports this pattern directly by via the <invoke> construct, which allows passing data by means of the referencing to the variable name in the *inputVariable* attribute on an asynchronous basis.

```
<invoke name="callbackClient" partnerLink="client"
portType="tns:TestMultiChoiceCallback" operation="onResult"
inputVariable="output"/>
```

where input variable *output* is defined as:

```
<variable name="output"
messageType="tns:TestMultiChoiceResponseMessage"/>
```

The corresponding message type, operation, and port type are defined in the wsdl as follows:

```
<?xml version="1.0"?>
<definitions name="TestMultiChoice"
        targetNamespace="http://acm.org/samples"
        xmlns:tns="http://acm.org/samples"
        xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
        xmlns="http://schemas.xmlsoap.org/wsdl/"
```

```
        >

<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     TYPE DEFINITION - List of services participating in this BPEL
process
     The default output of the BPEL designer uses strings as input and
     output to the BPEL Process. But you can define or import any XML
     Schema type and us them as part of the message types.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <types>
        <schema attributeFormDefault="qualified"
                elementFormDefault="qualified"
                targetNamespace="http://acm.org/samples"
                xmlns="http://www.w3.org/2001/XMLSchema"
                >

            <element name="TestMultiChoiceRequest">
                <complexType>
                    <sequence>
                        <element name="input" type="string" />
                    </sequence>
                </complexType>
            </element>

            <element name="TestMultiChoiceResponse">
                <complexType>
                    <sequence>
                        <element name="result1" type="string"/>
                                  <element name="result2"
type="string"/>
                                              <element name="result3"
type="string"/>
                    </sequence>
                </complexType>
            </element>

        </schema>
    </types>

<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     MESSAGE TYPE DEFINITION - Definition of the message types used as
     part of the port type defintions
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ --
>
    <message name="TestMultiChoiceRequestMessage">
        <part name="payload" element="tns:TestMultiChoiceRequest"/>
    </message>

    <message name="TestMultiChoiceResponseMessage">
        <part name="payload" element="tns:TestMultiChoiceResponse"/>
    </message>
<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     PORT TYPE DEFINITION - A port type groups a set of operations into
```

```
    a logical service unit.
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <!-- portType implemented by the TestMultiChoice BPEL process -->
    <portType name="TestMultiChoice">
        <operation name="initiate">
            <input message="tns:TestMultiChoiceRequestMessage"/>
        </operation>
    </portType>

    <!-- portType implemented by the requester of TestMultiChoice BPEL
process
         for asynchronous callback purposes
         -->
    <portType name="TestMultiChoiceCallback">
        <operation name="onResult">
            <input message="tns:TestMultiChoiceResponseMessage"/>
        </operation>
    </portType>
<!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     PARTNER LINK TYPE DEFINITION
         the TestMultiChoice partnerLinkType binds the provider and
         requester portType into an asynchronous conversation.
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <plnk:partnerLinkType name="TestMultiChoice">
        <plnk:role name="TestMultiChoiceProvider">
            <plnk:portType name="tns:TestMultiChoice"/>
        </plnk:role>
        <plnk:role name="TestMultiChoiceRequester">
            <plnk:portType name="tns:TestMultiChoiceCallback"/>
        </plnk:role>
    </plnk:partnerLinkType>
</definitions>
```

DP 16 (Data Interaction- Environment to Task - Pull-Oriented)
Description: The ability of a workflow task to request data elements from resources or services in the operational environment.

Oracle BPEL PM supports this pattern directly by means of <invoke> and <receive> synchronously. The <invoke> activity invokes an external service, and waits for its reply via <receive>. The invoked service must have <reply> implemented in order to send the requested data back.

For instance, the code below shows the invocation of an external service *CreditRatingService.* The *crInput* variable provides an input data needed for the service invocation, the result provided by the service is associated with the output variable respectively.

```
<invoke name="invoke-1" partnerLink="partnerLink1"
portType="nsxml0:CreditRatingService" operation="process"
inputVariable="crInput" outputVariable="crOutput"/>
```

Where an external service has been declared by means of the partner link:

```
<partnerLink name="CreditRatingService"
partnerLinkType="nsxml0:CreditRatingService"
partnerRole="CreditRatingServiceProvider"/>
```

DP 17 (Data Interaction- Environment to Task- Push-Oriented)
Description: The ability for a workflow task to receive and utilize data elements passed to it from services and resources in the operating environment on an unscheduled basis.

Oracle BPEL PM supports this pattern directly by means of <receive> and event handlers (*onMessage* within the <pick> construct) as it is shown below in the code snippets.

```
<receive name="receiveInput" partnerLink="client"
portType="tns:CreditFlow" operation="initiate" variable="input"
createInstance="yes"/>

<onMessage partnerLink="client" portType="tns:WP" operation="onStart"
variable="start">
      <sequence>
            ...
      </sequence>
</onMessage>
```

DP 18 (Data Interaction- Task to Environment- Pull-Oriented)
Description: The ability of a workflow task to receive and respond to requests for data elements from services and resources in the operational environment.

Oracle BPEL PM supports this pattern directly by means of <receive> and <reply> constructs.

DP 19 (Data Interaction- Case to Environment- Push-Oriented)
Description: The ability of a workflow case to initiate the passing of data elements to a resource or service in the operational environment.

Oracle BPEL PM offers no support for this pattern.

DP 20 (Data Interaction- Environment to Case- Pull-Oriented)
Description: The ability of a workflow case to request data from services or resources in the operational environment.

Oracle BPEL PM offers no support for this pattern.

DP 21 (Data Interaction- Environment to Case- Push-Oriented
Description: The ability of a workflow case to accept data elements from services or resources in the operating environment.

Oracle BPEL PM offers no support for this pattern.

DP 22 (Data Interaction- Case to Environment- Pull-Oriented)
Description: The ability of a workflow case to respond to requests for data elements from a service or resource in the operating environment.

Oracle BPEL PM offers no support for this pattern.

DP 23 (Data Interaction- Workflow to Environment- Push-Oriented)
Description: The ability of a workflow engine to pass data elements to resources or services in the operational environment.

Oracle BPEL PM offers no support for this pattern.

DP 24 (Data Interaction- Environment to Workflow- Pull-Oriented)
Description: The ability of a workflow to request workflow-level data elements from external applications.

Oracle BPEL PM offers no support for this pattern.

DP 25 (Data Interaction- Environment to Workflow- Push-Oriented)
Description: The ability of services or resources in the operating environment to pass workflow-level data to a workflow process.

Oracle BPEL PM offers no support for this pattern.

DP 26 (Data Interaction- Workflow to Environment- Pull-Oriented)
Description: The ability of a workflow engine to handle requests for workflow-level data from external applications.

Oracle BPEL PM offers no support for this pattern.

Data Transfer Mechanisms

DP 27 (Data Transfer by Value- Incoming)
Description: The ability of a workflow component to receive incoming data elements by value relieving it from the need to have shared names or common address space with the component(s) from which it receives them.

Oracle BPEL PM supports this pattern directly by means of the <assign> activity.

DP 28 (Data Transfer by Value- Outgoing)
Description: The ability of a workflow component to pass data elements to subsequent components as values relieving it from the need to have shared names or common address space with the component(s) to which it is passing them.

Oracle BPEL PM supports this pattern directly by means of the <assign> activity.

DP 29 (Data Transfer- Copy In/Copy Out)
Description: The ability of a workflow component to copy the values of a set of data elements into its address space at the commencement of execution and to copy their final values back at completion.

This pattern assumes that the entire data needs to migrate from one task to another. Oracle BPEL PM offers no support for this pattern, since it allows only copying data from one task to another by means of an <assign> activity, which is not applicable for data migration.

DP 30 (Data Transfer by Reference- Unlocked)
Description: The ability to communicate data elements between workflow components by utilizing a reference to the location of the data element in some mutually accessible location. No concurrency restrictions apply to the shared data element.

Oracle BPEL PM supports this pattern directly. No concurrency restrictions apply to the shared data.

DP 31 (Data Transfer by Reference- With Lock)
Description: The ability to communicate data elements between workflow components by passing a reference to the location of the data element in some mutually accessible location. Concurrency restrictions are implied with the receiving component receiving the privilege of read-only or dedicated access to the data element.

Although Oracle BPEL PM can indirectly support this pattern by means of serializable scopes, the tested version of Oracle BPEL PM does not seem to work according to the semantics of the serializable scopes.

DP 32 (Data Transformation- Input)
Description: The ability to apply a transformation function to a data element prior to it being passed to a workflow component.

Although Oracle BPEL PM allows the usage of the transformation functions within the <assign> activity, this pattern understands the transformations as inline functions which must occur at the time the task initiated. Therefore, this pattern is not supported.

DP 33 (Data Transformation- Output)
Description: The ability to apply a transformation function to a data element immediately prior to it being passed out of a workflow component.

Oracle BPEL PM does not support this pattern since it is not possible to specify transformation function inline with the task, which would be applied on the task completion.

Data-based Routing

DP 34 (Task Precondition- Data Existence)
Description: Data-based preconditions can be specified for tasks based on the presence of data elements at the time of execution.

Oracle BPEL PM does not support for this pattern.

DP 35 (Task Precondition- Data Value)
Description: Data-based preconditions can be specified for tasks based on the value of specific parameters at the time of execution.

Oracle BPEL PM supports this pattern by means of links and conditions evaluating the status of the links, i.e. *joinCondition*. For instance, assuming that sources of links L1 and L2 have been defined, an empty activity will be executed after the status of the links has been evaluated and *joinCondition* satisfied.

```
<empty name="empty" joinCondition="L1 OR L2">
      <target linkName="L1"/>
      <target linkName="L2"/>
```

DP 36 (Task Post-condition- Data Existence)
Description: Data-based post-conditions can be specified for tasks based on the existence of specific parameters at the time of execution.

Oracle BPEL PM offers no support for this pattern.

DP 37 (Task Postcondition- Data Value)
Description: Data-based postconditions can be specified for tasks based on the value of specific parameters at the time of execution.

Oracle BPEL PM offers no support for this pattern.

DP 38 (Event-based Task Trigger)
Description: The ability for an external event to initiate a task.

Oracle BPEL PM supports this pattern directly by means of <receive> and event handlers *onMessage*. In addition, there is also support for the time events via *onAlarm.*

DP 39 (Data-based Task Trigger)
Description: The ability to trigger a specific task when an expression based on workflow data elements evaluates to true.

Oracle BPEL PM offers not direct support for this pattern.

DP 40 (Data-based Routing)
Description: The ability to alter the control flow within a workflow case as a consequence of the value of data-based expressions.

Oracle BPEL PM offers direct support for this pattern via links and the <switch> construct. Every link can have a data-based transition condition associated with it, and thus be selected if the given transition condition has been satisfied. The <switch> construct allows creating multiple branches and their association with the data conditions. The first branch specified in the lexical order, for which the data conditions holds, is to be selected. An example of the <switch> construct with two predefined and one default conditions is shown below:

```
<switch name="switch-1">
<case
condition="bpws:getVariableData(&quot;input&quot;,&quot;payload&quot;,&q
uot;/tns:TestExclusiveChoiceRequest/tns:input&quot;)&lt;5;">
      <sequence>
          <assign name="actA">
            <copy>
                  <from variable="input" part="payload"
                  query="/tns:TestExclusiveChoiceRequest/tns:input">
                  </from>
                  <to variable="output" part="payload"
                  query="/tns:TestExclusiveChoiceResponse/tns:result"/>
            </copy>
          </assign>
      </sequence>
</case>
<otherwise>
      <sequence>
            <assign name="actB">
              <copy>
                  <from variable="input" part="payload"
                  query="/tns:TestExclusiveChoiceRequest/tns:input">
                  </from>
                  <to variable="output" part="payload"
                  query="/tns:TestExclusiveChoiceResponse/tns:result"/>
              </copy>
            </assign>
      </sequence>
</otherwise>
</switch>
```

3. Evaluation of Oracle BPEL PM from the resource perspective

The evaluation of Oracle BPEL PM from the resource perspective is based on the features of the Workflow Service offered by Oracle BPEL PM to model user interactions. The user task can be added in the process model by means of the *user task macro* block, which allows the given task to be configured and the functionality of which depends on the set of services, namely Task Management Service, Task Action Handler, Idenitity Service, Worklist Service, Task Routing Service and Notification Service. The main purpose of the user task service is to enable the integration of people and manual tasks into bpel flows.

A user task is a service that assists with modeling processes that require user input in order to complete. It is accessible externally through a Java API upon which custom interfaces may be developed. Typically in a process that requires user input, an upstream process element creates a User Task and associates a payload of data with it; subsequently suspending downstream process nodes. That task and its associated data may then be accessed through an external application's user interface. Once the user has completed the required external process, the User Task API is called to assign a status of complete (or any other appropriate value). This in turn activates the remaining downstream processes. User tasks are special instances of activities where the performer of the activity is manual rather than automated. Tasks may be assigned to particular assignees and may have expiration times associated with them [13].

A task may be routed through multiple users via a sequential flow, a parallel flow, or an adhoc flow. Oracle BPEL PM integrated these routing variants into a set of "workflow patterns" (see Figure 28) consisting of

- Simple workflow (single-user task) – used for a business process that required a user's action. Based on the user's action or inaction, the business process modeler defines what the business process has to do.
- Simple workflow with escalation (extension of a single-user task) – used to escalate the task, if the user does not respond within the allotted time.
- Simple workflow with renewal (extension of a single-user task) – used to extend the expiration period id the user doe not respond within the allotted time.
- Sequential workflow – used to route tasks to multiple users in sequence.
- Sequential workflow with escalation (extension of a sequential workflow) – used to escalate the task if a user does not take an action within the allotted time.
- Parallel workflow – used when multiple users, working in parallel, must take action, such as in a hiring situation when multiple users vote to hire or reject an applicant. The voting percentage is needed for specifying the outcome.
- Parallel workflow with final reviewer (extension of parallel workflow) – used when a task is first sent to multiple users in parallel and then sent to a final reviewer for a decision.
- Adhoc (or dynamic) workflow – used to assign a task to one user first, who then decides where the task goes next. The task is routed until one of the assignees completes it.
- FYI task – used when a task is sent to the user, but the business process does not wait for the user response.
- User task 2.0. Macro – supports user tasks from Oracle BPEL PM release 2.0.

- Task continuation – used to build complex workflow patterns, i.e. it allows one workflow to be continued with another workflow.

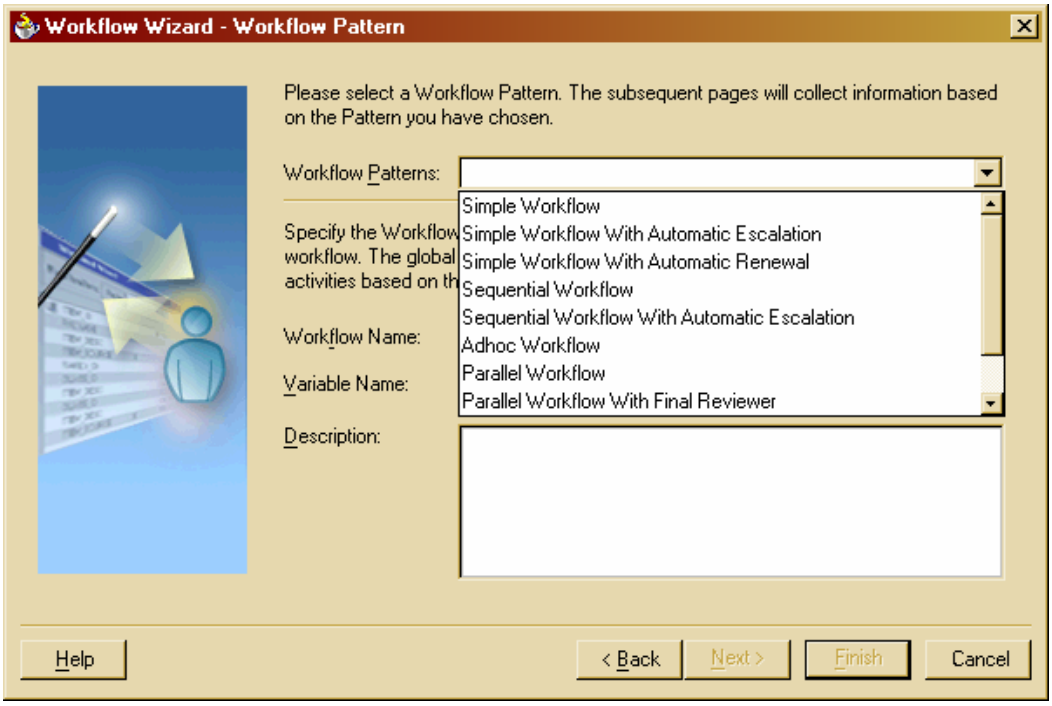


Figure 28 Wokflow patterns

Each of the workflow patterns can be easily configured with help of the workflow wizard. Figures below show an example of configuring the Simple workflow pattern.

Figure 29 Workflow wizard - Step 1

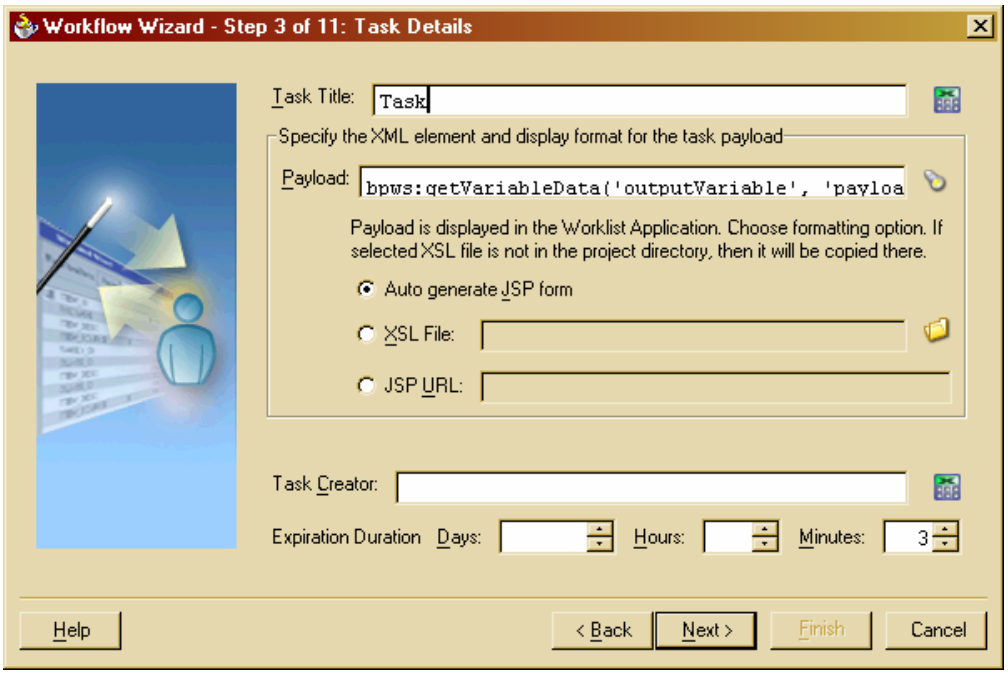


Figure 30 Workflow wizard - Step 3

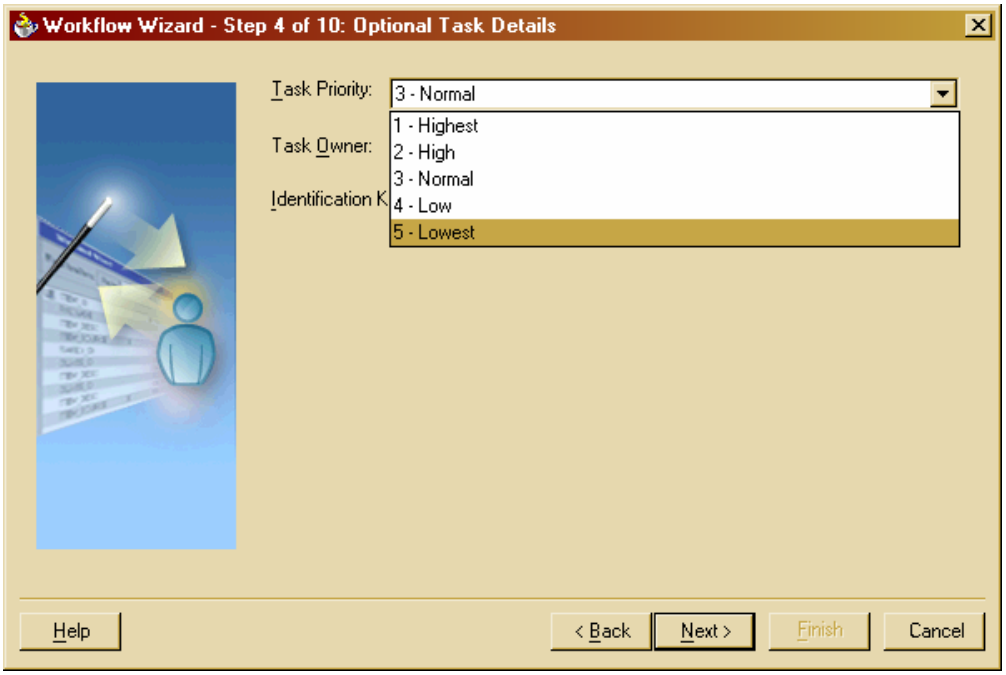


Figure 31 Workflow wizard - Step 4

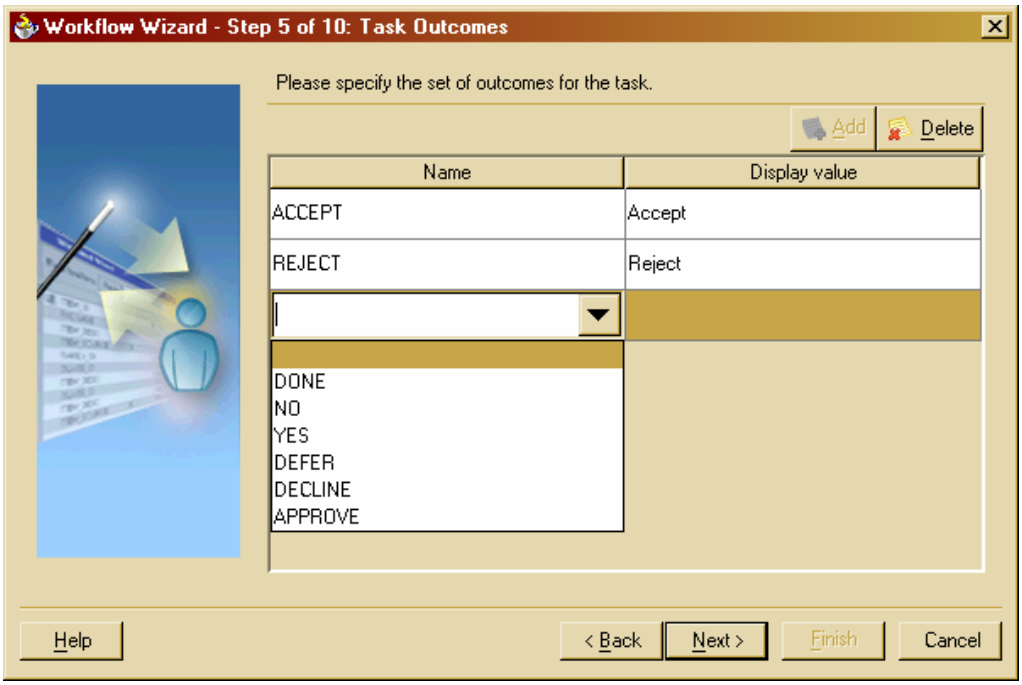


Figure 32 Workflow wizard - Step 5

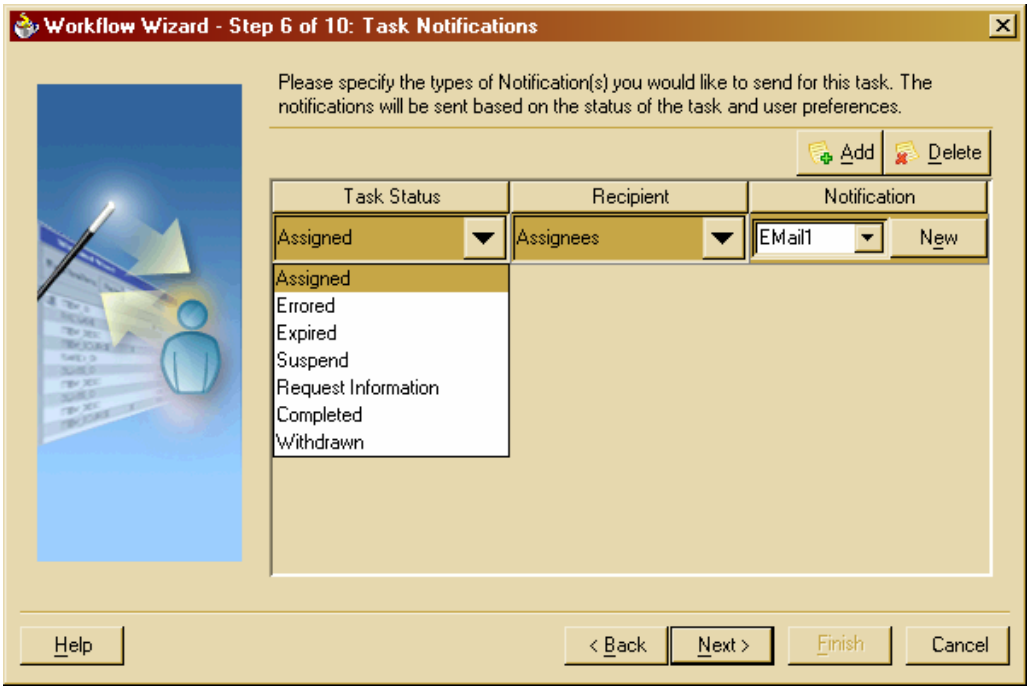


Figure 33 Workflow wizard - Step 6

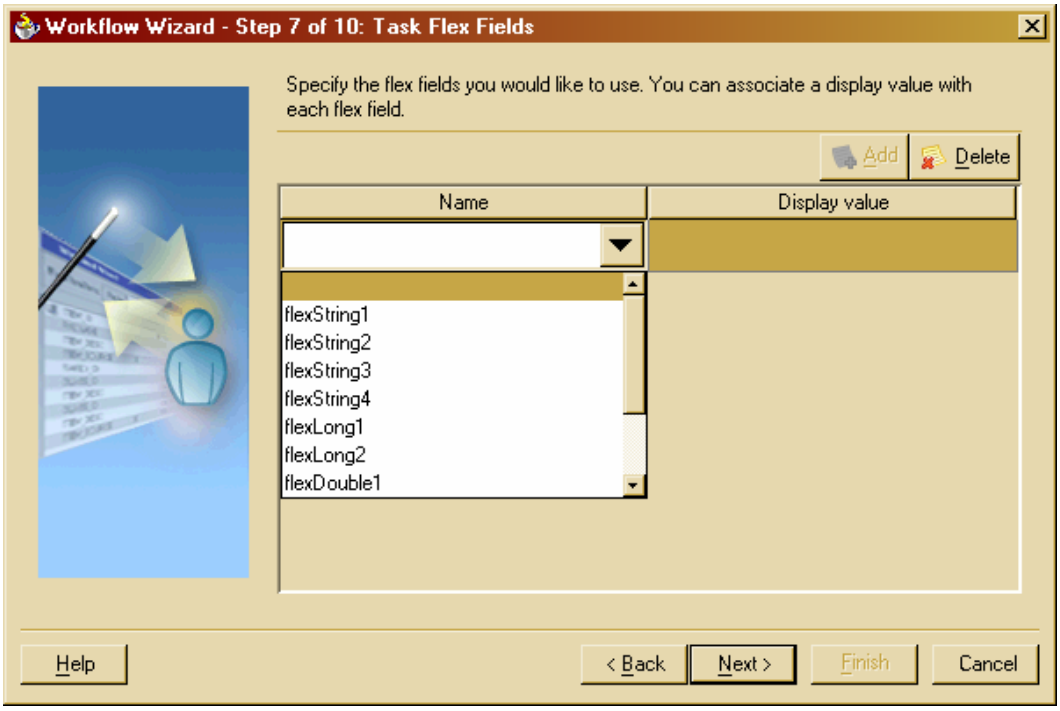


Figure 34 Workflow wizard - Step 7

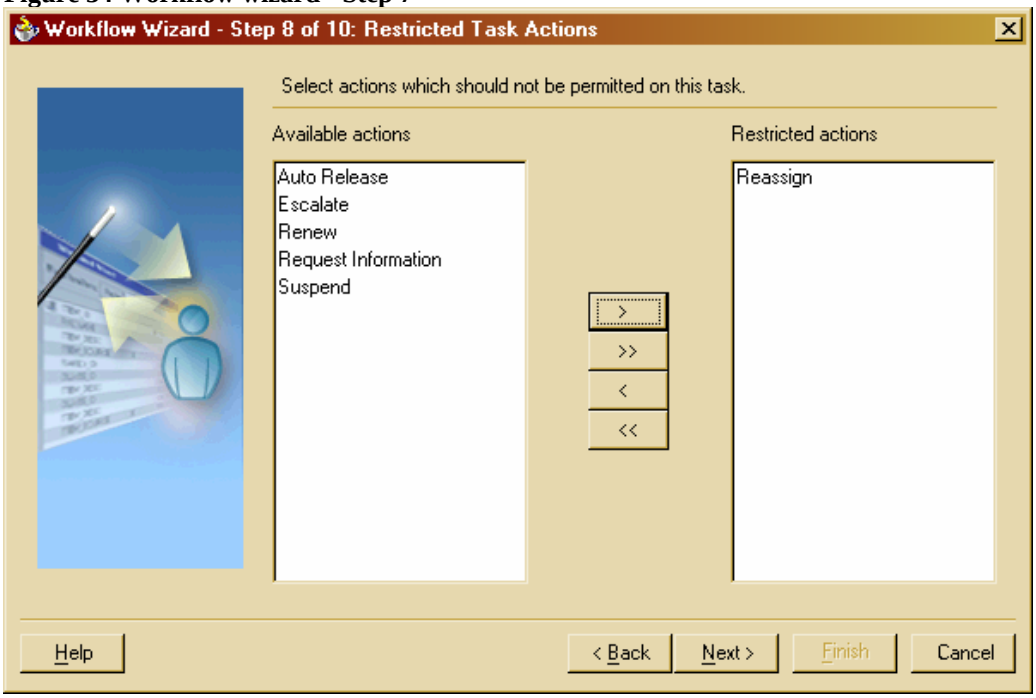


Figure 35 Workflow wizard- Step 8

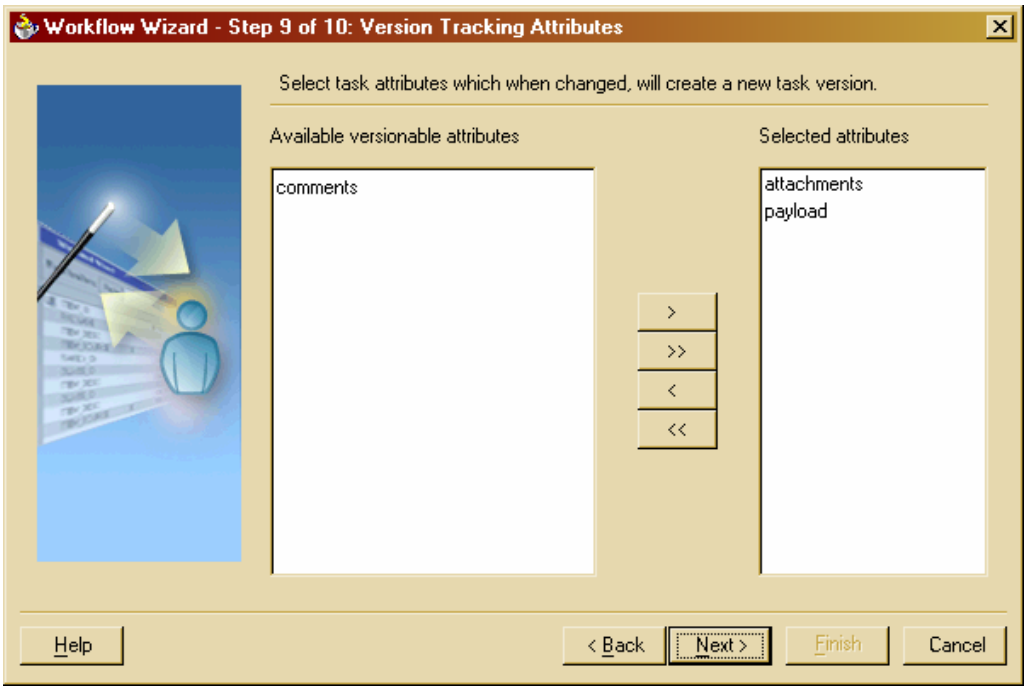


Figure 36 Workflow wizard - Step 9

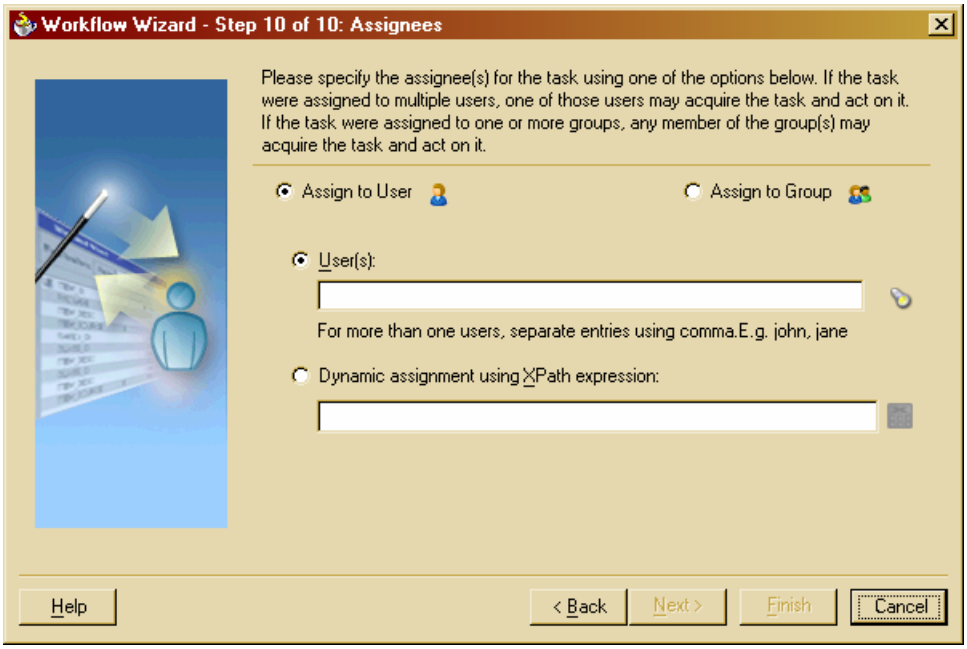


Figure 37 Workflow wizard - Step 10

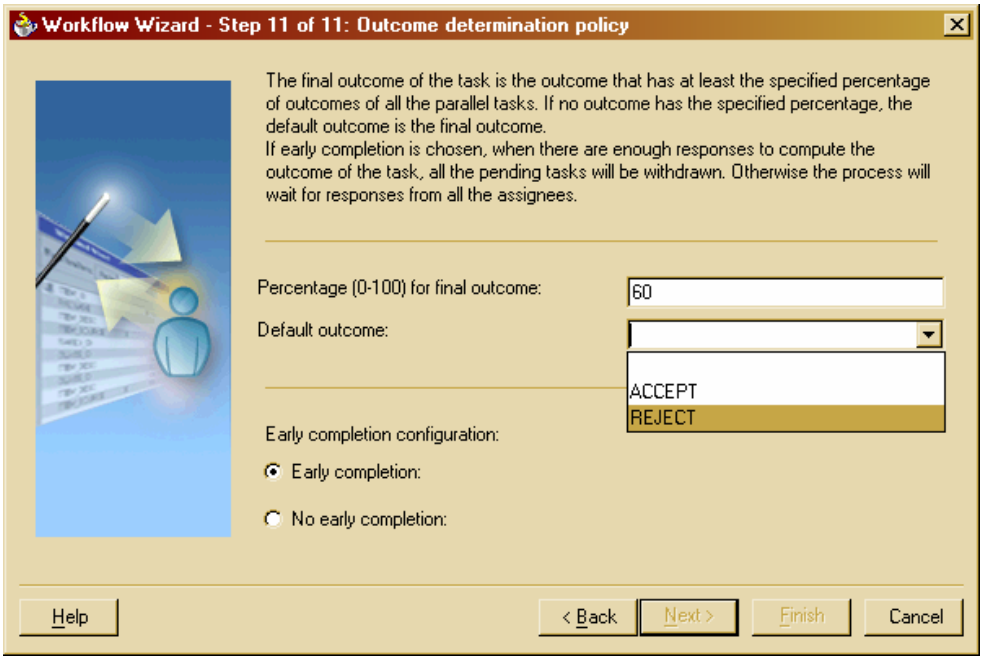


Figure 38 Workflow wizard - Step 11

Each task has the following set of the attributes:

- title: the name of the activity;
- creationDate: the date of the task creation;
- creator: an identifier of the process initiated the task;
- modifyDate: the last modification date of the task;
- modifier: an id of a user or a role performing the updating or completion of the task;
- assignee: id of a user, role, or group responsible for completing the task;
- status: the *active* or *completed* status of the task;
- expired: Boolean status indicating whether the task expired or not;
- expirationDate: an optional field indicating the time of the task expiration;
- duration: an optional field indicating the duration after which the task should expire;
- priority: an optional field indicating the priority of the task;
- customKey: an optional field specifying an alternative to the task identifier key that can be used for referring to the task;
- conclusion: an optional field indicating how the task has completed, i.e. whether it was *Approved, Refused* or *Canceled.*
- attachment: an optional field containing any type of information that might be required for the task execution.

Users access tasks assigned to them via the Worklist Application, which displays tasks assigned to the user, the group the user belongs to, and allows including additional information such as history, comment, etc. Any user may adjust the format of a work list in order to display the work items allocated to this user, to the group the user belongs to, or other users. By selecting a work item allocated to a user, the user may commence execution on it. An example of work list is shown in Figure 39.

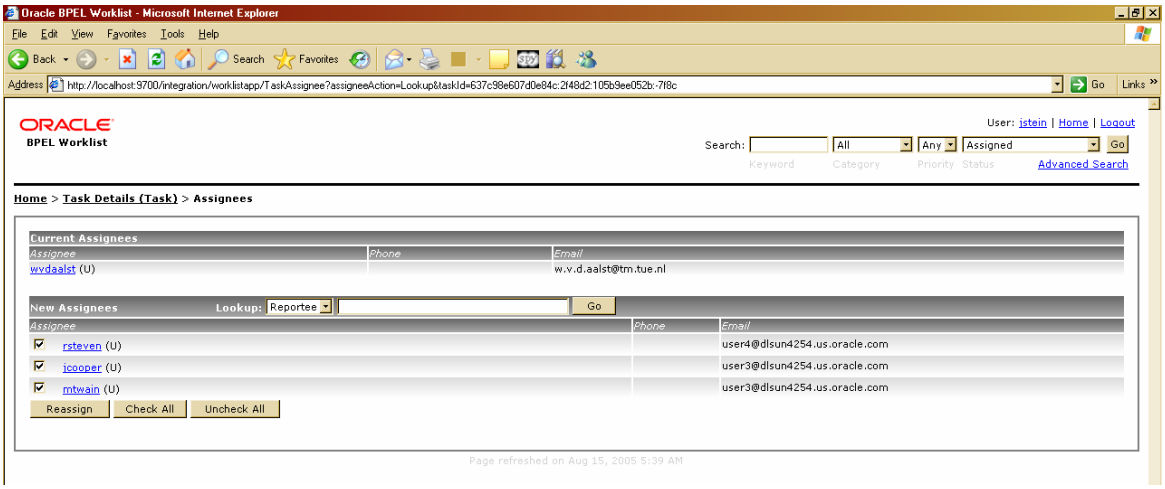


Figure 39 The worklist application

The evaluation of Oracle BPEL PM from the resource perspective is done by means of the resource patterns [10]. The pattern can be directly or partially supported or not supported at all. The criteria for the evaluation are taken from [10].

RP1 (Direct Allocation)
Description: The ability to specify at design time the identity of the resource that will execute a task.

Oracle BPEL PM supports this pattern directly. The resources can be specified statically or dynamically. The organization structure, together with resources, their identities, and other characteristics can be specified via JAZN admintool. When adding a task to the process model, the workflow wizard allows specifying a single user, a set of the users, or a group as it shown in Figure 41. Each of the task parameters specified by means of the wizard can be modified later in the designer view (SimpleUserActivity1 in Figure 40) or in the corresponding BPEL code.

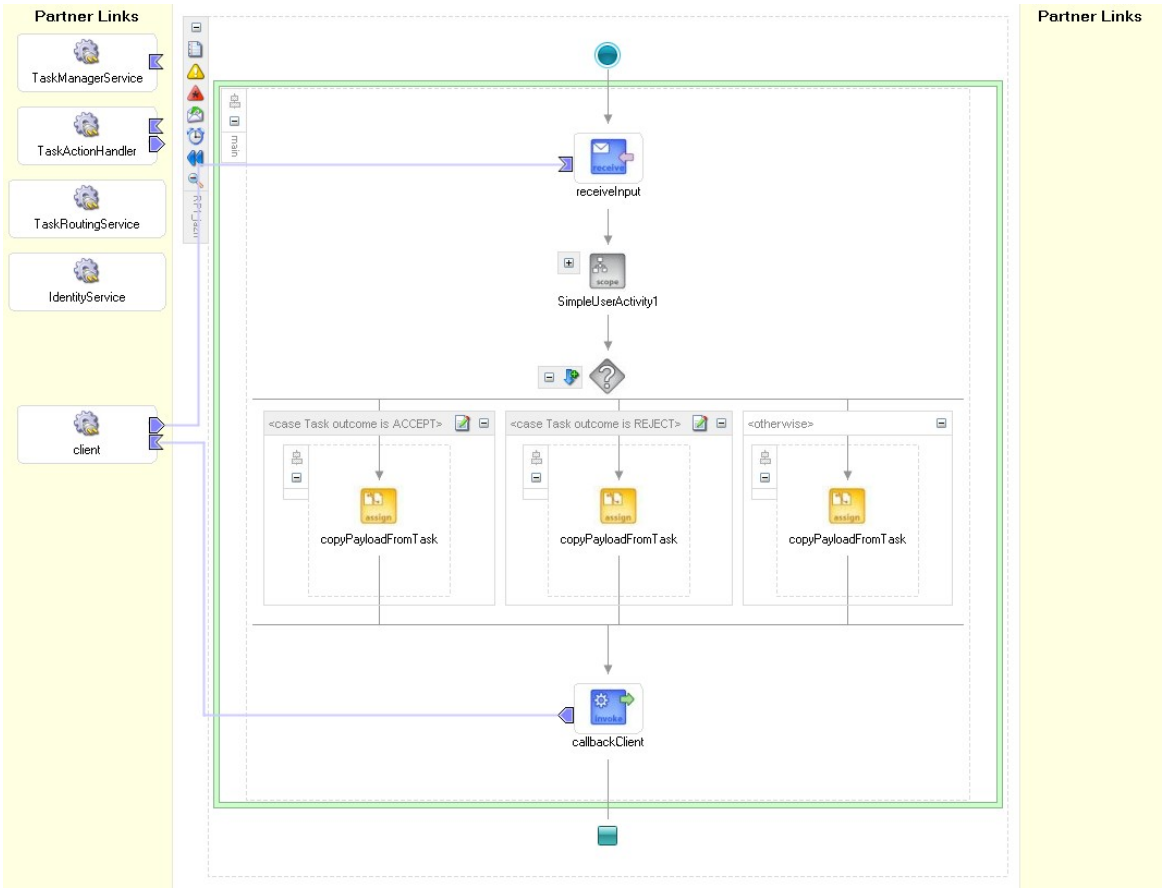


Figure 40 A user task (Simple Workflow pattern)

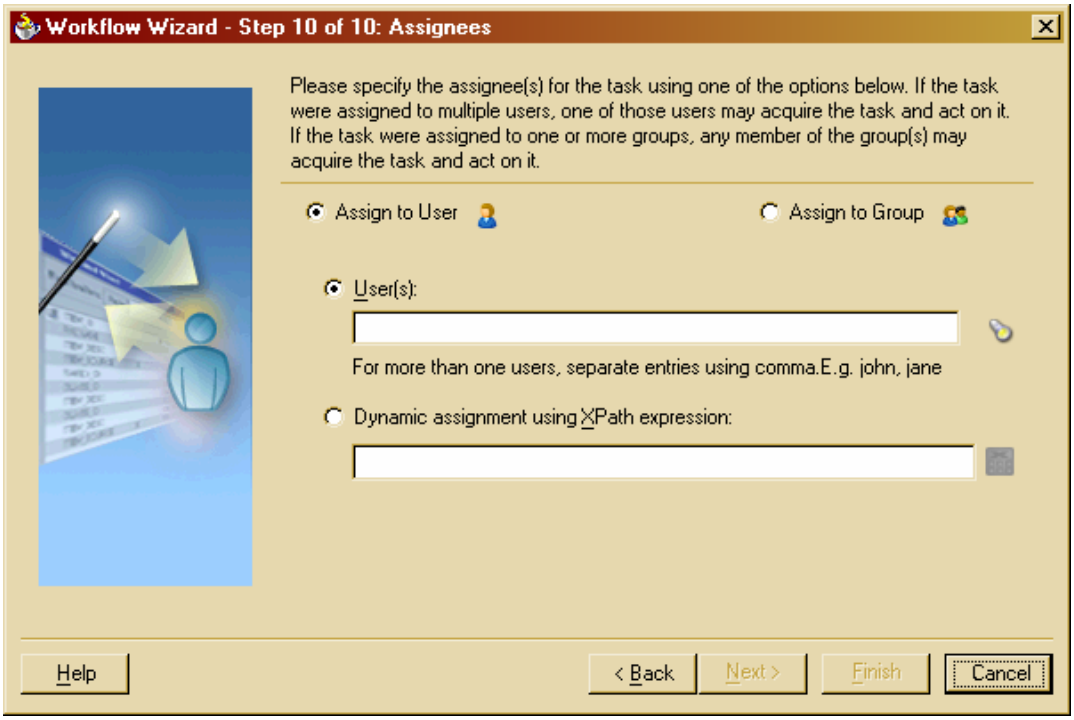


Figure 41 Task assignees

An identity look-up dialog by role and user is shown in Figure 42 and respectively.

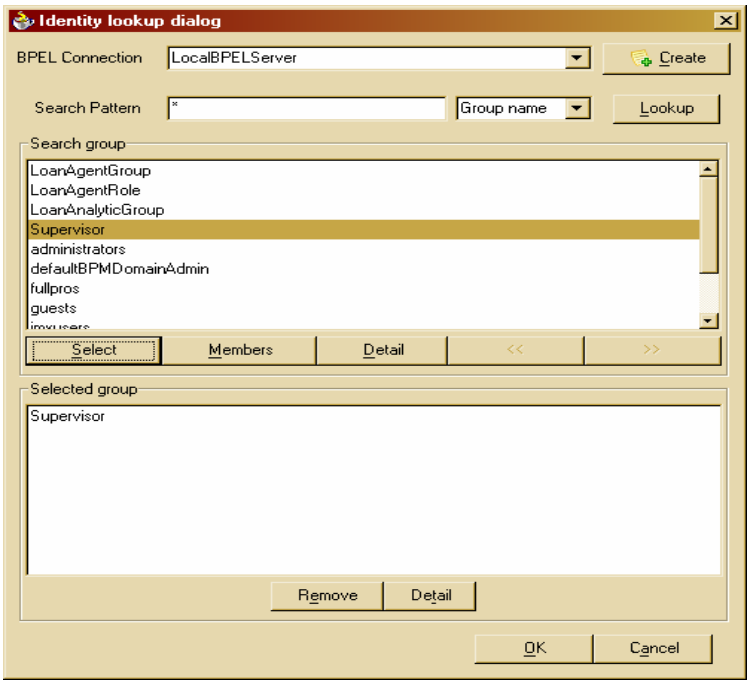


Figure 42 The identity lookup dialog (role)

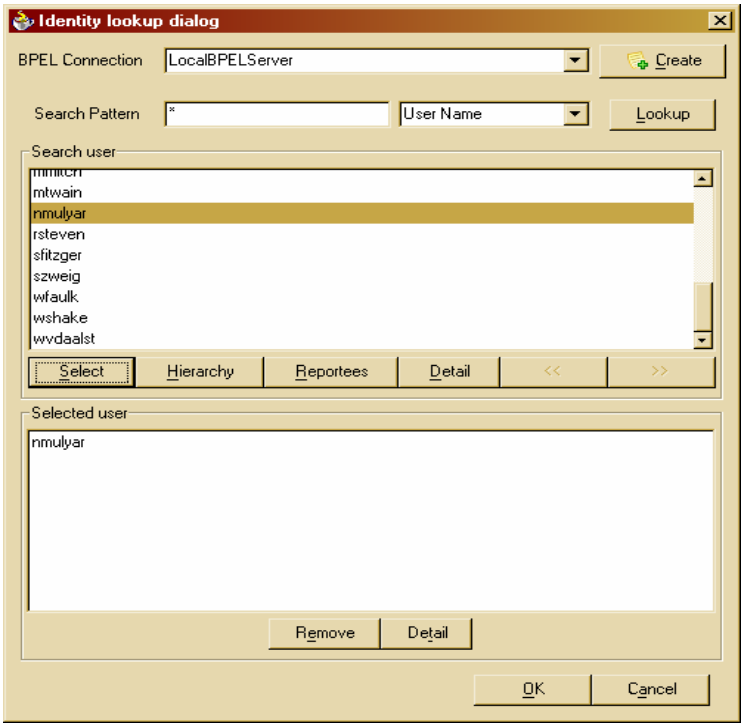


Figure 43 The identity lookup dialog (user)

The code snippets corresponding to the process model in Figure 40 are shown below:

```
<process name="RP1_jazn"
targetNamespace="http://xmlns.oracle.com/RP1_jazn"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.Xpath20"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:client="http://xmlns.oracle.com/RP1_jazn"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:identityservice="http://xmlns.oracle.com/pcbpel/identityservice/lo
cal"
xmlns:taskmngr="http://xmlns.oracle.com/pcbpel/taskservice/taskmanager"
xmlns:taskroutingservice="http://xmlns.oracle.com/pcbpel/taskservice/tas
kroutingservice" xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:task="http://xmlns.oracle.com/pcbpel/taskservice/task"
xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.ExtFunc"
xmlns:taskactionhandler="http://xmlns.oracle.com/pcbpel/taskservice/task
ActionHandler"><!--
================================================================= --><!-
- PARTNERLINKS                                                      --
><!-- List of services participating in this BPEL process
--><!--
================================================================= -->
  <partnerLinks><!--
    The 'client' role represents the requester of this service. It is
    used for callback. The location and correlation information
associated
    with the client role are automatically set using WS-Addressing.
    -->
    <partnerLink name="client" partnerLinkType="client:RP1_jazn"
myRole="RP1_jaznProvider" partnerRole="RP1_jaznRequester"/>
    <partnerLink myRole="TaskManagerCallbackListener"
name="TaskManagerService" partnerRole="TaskManager"
partnerLinkType="taskmngr:TaskManager"/>
    <partnerLink name="TaskRoutingService"
partnerRole="TaskRoutingService"
partnerLinkType="taskroutingservice:TaskRoutingService"/>
    <partnerLink myRole="HandleTaskActionRequester"
name="TaskActionHandler" partnerRole="HandleTaskActionProvider"
partnerLinkType="taskactionhandler:TaskActionHandler"/>
    <partnerLink name="IdentityService"
partnerRole="IdentityServiceProvider"
partnerLinkType="identityservice:IdentityService"/>
  </partnerLinks><!--
================================================================= --><!-
- VARIABLES                                                         --
><!-- List of messages and XML documents used within this BPEL process
--><!--
================================================================= -->
  <variables><!-- Reference to the message passed as input during
initiation -->
    <variable name="inputVariable"
messageType="client:RP1_jaznRequestMessage"/><!-- Reference to the
message that will be sent back to the
    requester during callback
```

```
    -->
    <variable name="outputVariable"
messageType="client:RP1_jaznResponseMessage"/>
    <variable name="SimpleUserActivityVar1" element="task:task"/>
  </variables><!--
================================================================= --><!-
- ORCHESTRATION LOGIC                                               --
><!-- Set of activities coordinating the flow of messages across the
--><!-- services integrated within this business process
--><!--
================================================================= -->
  <sequence name="main"><!-- Receive input from requestor.
    Note: This maps to operation defined in RP1_jazn.wsdl
    -->
    <receive name="receiveInput" partnerLink="client"
portType="client:RP1_jazn" operation="initiate" variable="inputVariable"
createInstance="yes"/><!-- Asynchronous callback to the requester.
     Note: the callback location and correlation id is transparently
handled
     using WS-addressing.
     -->
    <scope name="SimpleUserActivity1" variableAccessSerializable="no"
xmlns:taskactionhandler="http://xmlns.oracle.com/pcbpel/taskservice/task
ActionHandler" xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:bpel="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:taskmngr="http://xmlns.oracle.com/pcbpel/taskservice/taskmanager"
xmlns:task="http://xmlns.oracle.com/pcbpel/taskservice/task"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:wf="http://schemas.oracle.com/bpel/extension/workflow"
wf:key="SimpleUserActivityVar1;taskConfigSimpleUserActivity1.xml;SimpleU
serActivity;Task4Eric;bpws:getVariableData('outputVariable', 'payload',
'/client:RP1_jaznProcessResponse/client:result');;;;;">
      <variables>
        <variable name="oraBPMTaskMessage"
messageType="taskmngr:taskMessage"/>
        <variable name="oraBPMTaskErroredFaultMessage"
messageType="taskmngr:taskErroredMessage"/>
        <variable name="oraBPMTemporaryVariable" type="xsd:string"/>
      </variables>
      <correlationSets>
        <correlationSet name="oraBPMTaskIdCor"
properties="taskmngr:taskId"/>
      </correlationSets>
      <sequence>
        <assign name="setUserDefinedAttributes">
          <copy>
            <from expression="&quot;Task4Eric&quot;"/>
            <to variable="SimpleUserActivityVar1"
query="/task:task/task:title"/>
          </copy>
          <copy>
            <from expression="bpws:getVariableData('outputVariable',
'payload', '/client:RP1_jaznProcessResponse/client:result')"/>
```

```
            <to variable="SimpleUserActivityVar1"
query="/task:task/task:payload"/>
          </copy>
          <copy>
            <from expression="string('hverbeek')"/>
            <to variable="SimpleUserActivityVar1"
query="/task:task/task:assigneeUsers[1]"/>
          </copy>
          <copy>
            <from expression="concat(ora:getProcessURL(),
string('/taskConfigSimpleUserActivity1.xml'))"/>
            <to variable="SimpleUserActivityVar1"
query="/task:task/task:taskType"/>
          </copy>
        </assign>
        <assign name="setSystemDefinedAttributes">
          <copy>
            <from expression="ora:getInstanceId()"/>
            <to variable="SimpleUserActivityVar1"
query="/task:task/task:instanceId"/>
          </copy>
          <copy>
            <from expression="ora:getProcessId()"/>
            <to variable="SimpleUserActivityVar1"
query="/task:task/task:processName"/>
          </copy>
          <copy>
            <from expression="ora:getProcessId()"/>
            <to variable="SimpleUserActivityVar1"
query="/task:task/task:processId"/>
          </copy>
          <copy>
            <from expression="ora:getProcessVersion()"/>
            <to variable="SimpleUserActivityVar1"
query="/task:task/task:processVersion"/>
          </copy>
          <copy>
            <from expression="ora:getDomainId()"/>
            <to variable="SimpleUserActivityVar1"
query="/task:task/task:domainId"/>
          </copy>
          <copy>
            <from expression="ora:getProcessOwnerId()"/>
            <to variable="SimpleUserActivityVar1"
query="/task:task/task:processOwner"/>
          </copy>
          <copy>
            <from expression="string('SINGLE_APPROVAL')"/>
            <to variable="SimpleUserActivityVar1"
query="/task:task/task:pattern"/>
          </copy>
          <copy>
            <from expression="false()"/>
            <to variable="SimpleUserActivityVar1"
query="/task:task/task:hasSubTasks"/>
          </copy>
          <copy>
```

```
            <from variable="SimpleUserActivityVar1"/>
            <to variable="oraBPMTaskMessage" part="payload"/>
          </copy>
        </assign>
        <scope name="initiateTask">
          <faultHandlers>
            <catch faultName="taskmngr:taskErroredFault"
faultVariable="oraBPMTaskErroredFaultMessage">
              <assign name="readErroredTask">
                <copy>
                  <from variable="oraBPMTaskErroredFaultMessage"
part="payload"/>
                  <to variable="oraBPMTaskMessage" part="payload"/>
                </copy>
              </assign>
            </catch>
          </faultHandlers>
          <sequence>
            <invoke name="initiateTask" partnerLink="TaskManagerService"
portType="taskmngr:TaskManager" operation="initiateTask"
inputVariable="oraBPMTaskMessage" outputVariable="oraBPMTaskMessage"/>
          </sequence>
        </scope>
        <sequence>
          <invoke name="initiateTaskActionHandler"
partnerLink="TaskActionHandler"
portType="taskactionhandler:TaskActionHandler" operation="initiate"
inputVariable="oraBPMTaskMessage">
            <correlations>
              <correlation set="oraBPMTaskIdCor" initiate="yes"
pattern="out"/>
            </correlations>
          </invoke>
          <receive name="receiveUpdatedTask"
partnerLink="TaskActionHandler"
portType="taskactionhandler:TaskActionHandlerCallback"
operation="onTaskCompleted" variable="oraBPMTaskMessage"
createInstance="no">
            <correlations>
              <correlation set="oraBPMTaskIdCor" initiate="no"/>
            </correlations>
          </receive>
        </sequence>
        <assign name="readUpdatedTask">
          <copy>
            <from variable="oraBPMTaskMessage" part="payload"/>
            <to variable="SimpleUserActivityVar1"/>
          </copy>
        </assign>
      </sequence>
    </scope>
    <switch name="taskSwitch"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:tt="http://xmlns.oracle.com/pcbpel/taskservice/tasktype"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```
      <case condition="bpws:getVariableData('SimpleUserActivityVar1',
'/task:task/task:state') = 'COMPLETED' and
bpws:getVariableData('SimpleUserActivityVar1',
'/task:task/task:conclusion') = 'ACCEPT'">
        <bpelx:annotation>
          <bpelx:pattern>Task outcome is ACCEPT
          </bpelx:pattern>
        </bpelx:annotation>
        <sequence>
          <assign name="copyPayloadFromTask">
            <copy>
              <from variable="SimpleUserActivityVar1"
query="/task:task/task:payload"/>
              <to variable="outputVariable" part="payload"
query="/client:RP1_jaznProcessResponse/client:result"/>
            </copy>
          </assign>
        </sequence>
      </case>
      <case condition="bpws:getVariableData('SimpleUserActivityVar1',
'/task:task/task:state') = 'COMPLETED' and
bpws:getVariableData('SimpleUserActivityVar1',
'/task:task/task:conclusion') = 'REJECT'">
        <bpelx:annotation>
          <bpelx:pattern>Task outcome is REJECT
          </bpelx:pattern>
        </bpelx:annotation>
        <sequence>
          <assign name="copyPayloadFromTask">
            <copy>
              <from variable="SimpleUserActivityVar1"
query="/task:task/task:payload"/>
              <to variable="outputVariable" part="payload"
query="/client:RP1_jaznProcessResponse/client:result"/>
            </copy>
          </assign>
        </sequence>
      </case>
      <otherwise>
        <bpelx:annotation>
          <bpelx:pattern>Task is EXPIRED, WITHDRAWN or ERRORED
          </bpelx:pattern>
        </bpelx:annotation>
        <sequence>
          <assign name="copyPayloadFromTask">
            <copy>
              <from variable="SimpleUserActivityVar1"
query="/task:task/task:payload"/>
              <to variable="outputVariable" part="payload"
query="/client:RP1_jaznProcessResponse/client:result"/>
            </copy>
          </assign>
        </sequence>
      </otherwise>
    </switch>
```

```
    <invoke name="callbackClient" partnerLink="client"
portType="client:RP1_jaznCallback" operation="onResult"
inputVariable="outputVariable"/>
  </sequence>
</process>
```

RP2 (Role-Based Allocation)
Description: The ability to specify at the design time that a task can only be executed by resources which correspond to a given role.

Oracle BPEL PM supports this pattern directly. The organizational structure, including the roles assigned to the users, can be specified in the *jazn-data.xml* and *user-properties.xml*, the content of which is vizible for the designer creating a process model. Specifying one of the existing roles as a task assignee (Figure 41) would make a work item visible in the worklists of users with the corresponding role. Note that Oracle BPEL PM makes no distinction between groups and roles.

RP3 (Deferred Allocation)
Description: The ability to defer specifying the identity of the resource that will execute a task until runtime.

Oracle BPEL PM supports this pattern directly. It allows specifying a task assignee by means of an XPath expression which is evaluated at run-time.

RP4 (Authorization)
Description: The ability to specify the range of resources that are authorized to execute a task.

This pattern is not supported by Oracle BPEL. It is possible to assign a work item to a specified role, and this work item can be reassigned to any other user/role. However, it is not possible to (re-)assign a task based on the condition, i.e. to a user having a certain authority.

RP5 (Separation of Duties)
Description: The ability to specify that two tasks must be allocated to different resources in a given workflow case.

Oracle BPEL PM does not allow specifying the separation of duties in terms of relationships between tasks, nor it allows the separation of duties based on security mechanisms. Thus this pattern is not supported by Oracle BPEL PM.

RP6 (Case Handling)
Description: The ability to allocate the work items within a given workflow case to the same resource.

Oracle BPEL PM supports this pattern directly. The feature of dynamic assignment using an XPath expression allows specifying that a next task must be assigned to the resource who executed the previous (first) task. In particular, the function *ora:getPreviousTaskApprover()* can be used for this purposes.

RP7 (Retain Familiar)

Description: Where several resources are available to undertake a work item, the ability to allocate a work item within a given workflow case to the same resource that undertook a preceding work item.

Oracle BPEL PM supports this pattern by means of the *ora:getPreviousTaskApprover()* function during the dynamic assignment an assignee to a task.

RP8 (Capability-based Allocation)
Description: The ability to offer or allocate instances of a task to resources based on specific capabilities that they possess.

Oracle BPEL PM supports this pattern directly. It allows defining user properties and store them in *user-propoerties.xml* file, which become accessible via the function *ora:getUserProperty().* This function can be used in the condition associated with the dynamic assignment feature.

RP9 (History-based Allocation)
Description: The ability to offer or allocate work items to resources on the basis of their previous execution history.

Oracle BPEL PM does not offer the direct support for this pattern, but it allows implementing this feature and accessing it via the properties of the dynamic assignment.

RP10 (Organizational Allocation)
Description: The ability to offer or allocate instances of a task to resources based their position within the organization and their relationship with other resources.

Oracle BPEL PM offers an indirect support for this pattern. The organizational structure is stored in the xml format in the *jazn-data.xml* file, and it can be modified and extended. The relationships between roles are specified via the role-hierarchy tree. The roles defined become accessible via the look-up wizard.

RP11 (Automatic Execution)
Description: The ability for an instance of a task to execute without needing to utilize the services of a resource.

Oracle BPEL PM allows specifying tasks which involving the user, but also the tasks which are to be performed automatically. Any of the basic or structured activities offered by Oracle BPEL PM in the BPEL palette are executed automatically. Thus this pattern is directly supported.

RP12 (Distribution by Offer – Single Resource)
Description: The ability to offer a work item to a selected individual resource.

Oracle BPEL PM supports this pattern by offering work item to members of a group. A group containing one user allows this user to "acquire" the offered work item.

RP13 (Distribution by Offer – Multiple Resources)
Description: The ability to offer a work item to a group of selected resources.

Oracle BPEL PM supports this pattern directly by specifying the name of a group as an assignee of the task. As a result, the task will be offered to all members of a group, and any of the members may acquire it. After the work item has been acquired, no other users may acquire this work item any more.

RP14 (Distribution by Allocation – Single Resource)
Description: The ability to directly allocate a work item to a specific resource for execution.

Oracle BPEL PM supports this pattern directly. Assigning a user with a given identity statically or dynamically, automatically allocates the work item to this user. This differs from the pattern RP12 where the work item is offered to the user.

RP15 (Random Allocation)
Description: The ability to offer or allocate work items to suitable resources on a random basis.

Oracle BPEL PM offers no direct support for this pattern. However, the feature of assigning a work item dynamically can be used to support this pattern. For example, a service can be implemented to retrieve resource on the random basis.

RP16 (Round Robin Allocation)
Description: The ability to allocate a work item to available resources on a cyclic basis.

Oracle BPEL PM offers no direct support for this pattern. However, the feature of assigning a work item dynamically can be used to support this pattern. For example, a service can be implemented to retrieve resource based on the round-robin algorithm.

RP17 (Shortest Queue)
Description: The ability to allocate a work item to the resource that has the least number of work items allocated to it.

Oracle BPEL PM offers no direct support for this pattern. However, the feature of assigning a work item dynamically can be used to support this pattern. For example, a service can be implemented to retrieve resource based on the shortest queue algorithm.

RP18 (Early Distribution)
Description: The ability to advertise and potentially allocate work items to resources ahead of the moment at which the work item is actually enabled for execution.

Oracle BPEL PM offers no support for this pattern.

RP19 (Distribution on Enablement)
Description: The ability to advertise and allocate work items to resources at the moment they are enabled for execution.

Oracle BPEL PM supports this pattern directly: as soon as a work item becomes available, it appears in the work list of the assigned resource.

RP20 (Late Distribution)
Description: The ability to advertise and allocate work items to resources after the work item has been enabled.

Oracle BPEL PM does not support this pattern since by any work item requires to have an assignee. Since the work item has to be allocated or offered to a user/role from the moment of creation, the late distribution is not possible.

RP21 (Resource-Initiated Allocation)
Description: The ability for a resource to commit to undertake a work item without needing to commence working on it immediately.

Oracle BPEL PM does not support this pattern directly. If a work item is assigned to a set of users or a group, one of the users in the list can "acquire" the task. However, this corresponds to the commence working on it immediately.

RP22 (Resource-Initiated Execution – Allocated Work Item)
Description: The ability for a resource to commence work on a work item that is allocated to it.

In Oracle BPEL PM resources are able to commence the execution of a work time available at their own worklists at a time of their own choosing, but before the task has expired. Therefore this pattern is supported directly.

RP23 (Resource-Initiated Execution – Offered Work Item)
Description: The ability for a resource to select a work item offered to it and commence work on it immediately.

Oracle BPEL PM supports this pattern directly. When a user is offered a work item, and the user "acquires" it, he commences work on it immediately.

RP24 (System-Determined Work Queue Content)
Description: The ability of the workflow engine to order the content and sequence in which work items are presented to a resource for execution.

Oracle BPEL PM does not impose a default ordering of the work items in the resources work queue, thus offering no support for this pattern.

RP25 (Resource-Determined Work Queue Content)
Description: The ability for resources to specify the format and content of work items listed in the work queue for execution.

Oracle BPEL PM supports this pattern directly. The user is able to format the work items listed in his worklist based on the task id, priority and other parameters. In addition, Oracle ships a JSP based sampe worklist application that can be customized to list specific content on the worklist application.

RP26 (Selection Autonomy)
Description: The ability for resources to select a work item for execution based on its characteristics and their own preferences.

Oracle BPEL PM supports this pattern directly. The user can select and act on any of the task displayed in his work list.

RP27 (Delegation)
Description: The ability for a resource to allocate a work item previously allocated to it to another resource.

Oracle BPEL PM supports this pattern directly by means of *"reassign"* action. As such, a manager can delegate a task to reportees. Similarly, the process owner or a user with BPMWorkflowReassign privileges can delegate a specific task to any other person in the organization. Figure 44 - Figure 47 illustrate how a work item can be reassigned via the work list application.

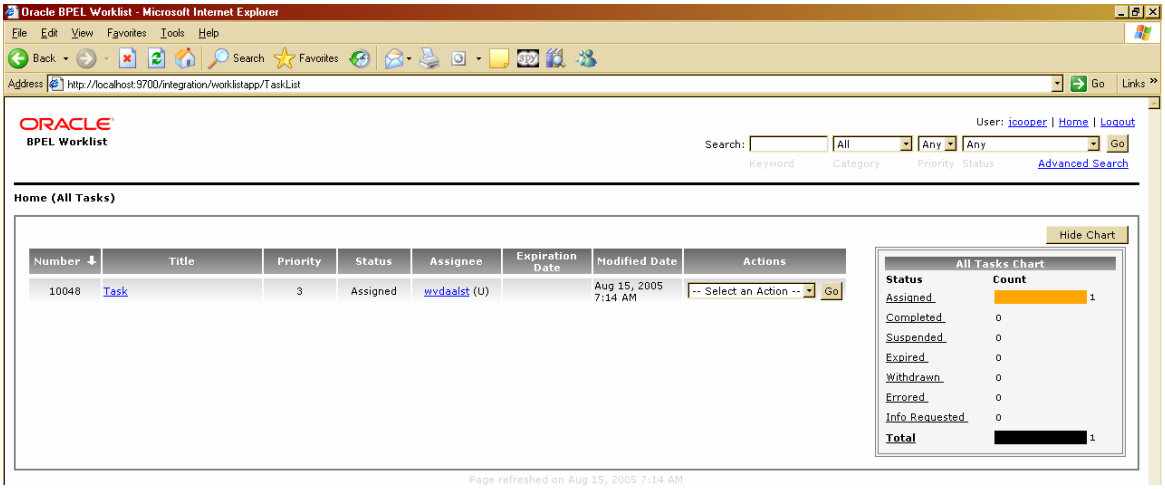


Figure 44 Reassign-1

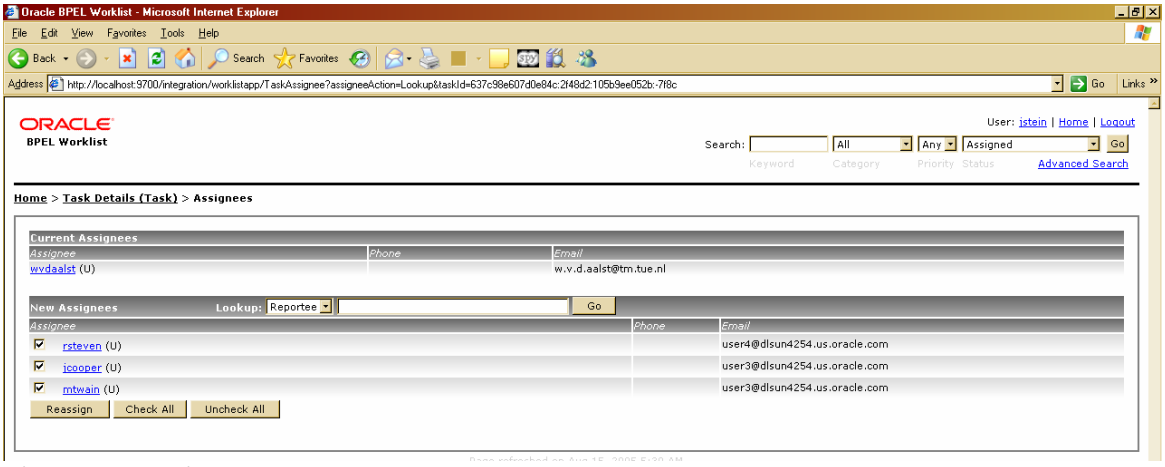


Figure 45 Reassign-2

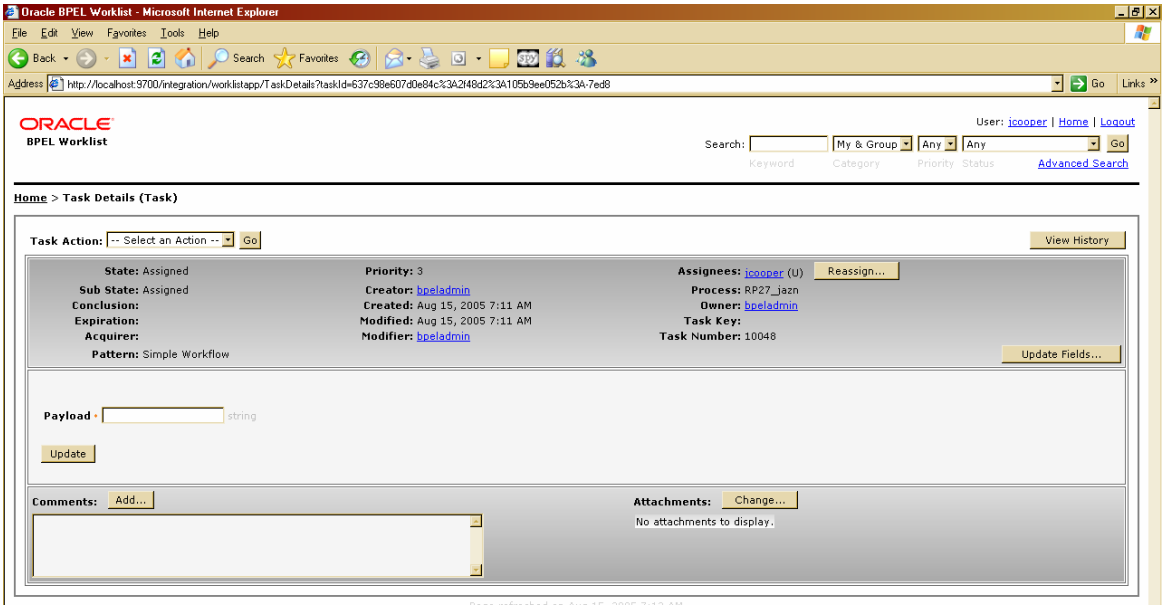


Figure 46 Reassign-3

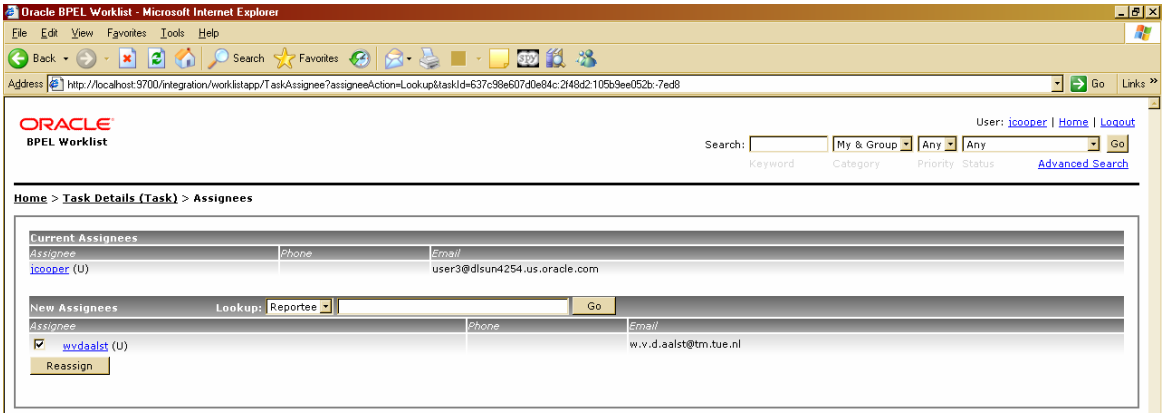


Figure 47 Reassign-4

RP28 (Escalation)

Description: The ability of the workflow system to offer or allocate a work item to a resource or group of resources other than those it has previously been offered or allocated to in an attempt to expedite the completion of the work item.

Oracle BPEL PM supports this pattern directly by allowing escalation of a task to the manager for further action. The escalation continues until a certain user, a certain level (number of escalations to a 'manager'), or a certain title is reached. The escalation feature works correctly if a task has been assigned to a specific user, however if a task has been assigned to a role or a group, Oracle BPEL PM does not seem to know an upper level where a task should be escalated to. Figure 48 shows the setting of the escalation in the workflow wizard.

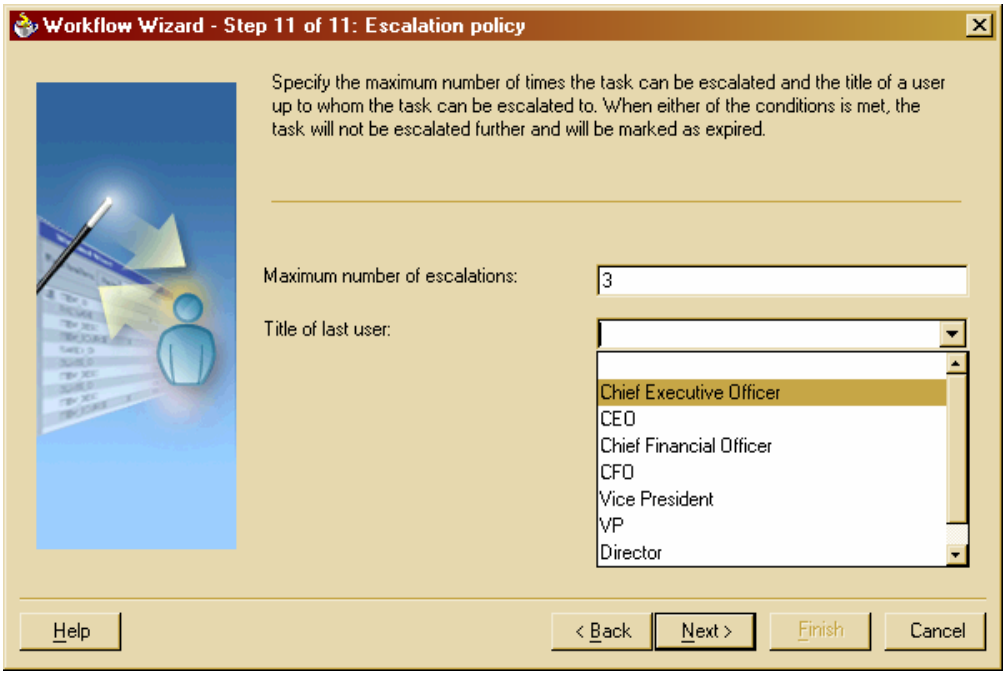


Figure 48 Escalation settings

RP29 (Deallocation)

Description: The ability of a resource (or group of resources) to relinquish a work item which is allocated to it and make it available for allocation to another resource or group of resources.

Oracle BPEL PM supports this pattern directly. If a work item has been assigned to a set of users of a group, one of the users in the list can "acquire" the task. At anytime before the task expires or before a user has updated the task, the user can "release" the task to the set of users/group the task was originally assigned to. Figure 49 - Figure 51demonstrate how a work item can be released via the work list application.

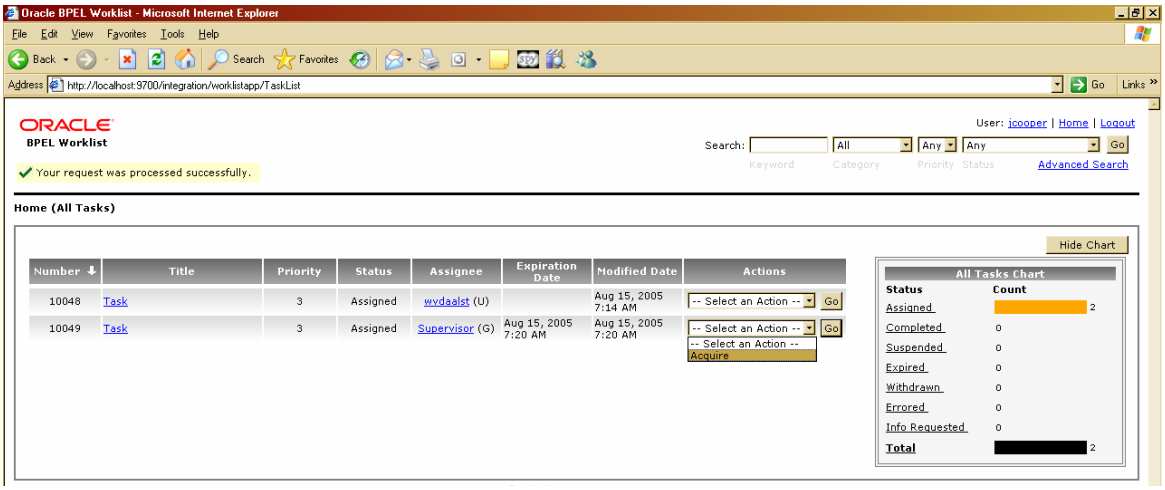


Figure 49 Release-1

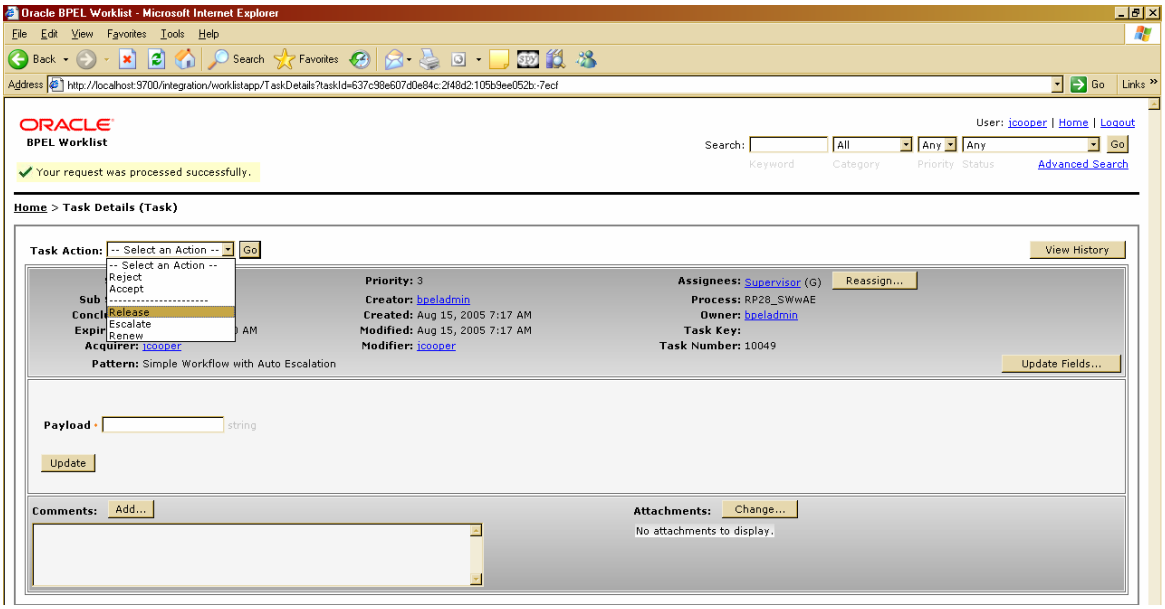


Figure 50 Release-2

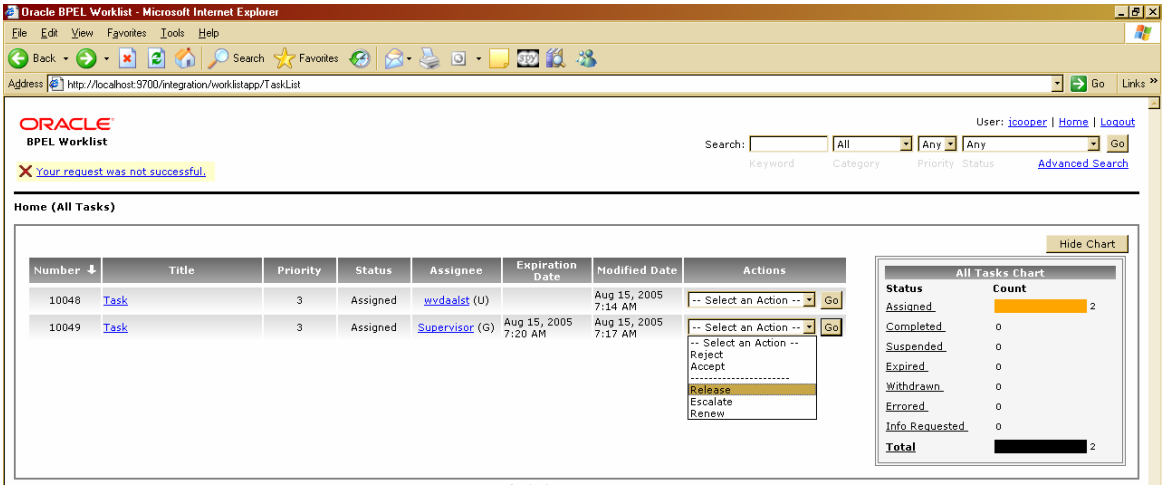


Figure 51 Release-3

RP30 (Stateful Reallocation)

Description: The ability of a resource to allocate a work item to another resource without loss of state data.

Any work item can be "reassigned" to a new set of users/group. If the user has updated a task, after the reassignment the data provided by this user is visible to the new assignee, i.e. the state data is not lost.

RP31 (Stateless Reallocation)

Description: The ability for a resource to reallocate a work item currently being executed to another resource without retention of state.

Oracle BPEL PM does not allow task rollback, thus offering no support for this pattern.

RP32 (Suspension/Resumption)
Description: The ability for a resource to suspend and resume execution of a work item.

Oracle BPEL PM supports this pattern directly. "suspend" and "resume" are available actions in the worklist application.

RP33 (Skip)
Description: The ability for a resource to skip a work item allocated to it and mark the work item as complete.

Oracle BPEL PM supports this pattern directly. As such any work item can be "withdrawn" by the task creator or the administrator. However, there is also possibility to model a user-action "skip", which marks a work item as completed and passes the flow of control to the subsequent task. Figure 52- Figure 54 visualize the "skip" and "withdraw" actions.

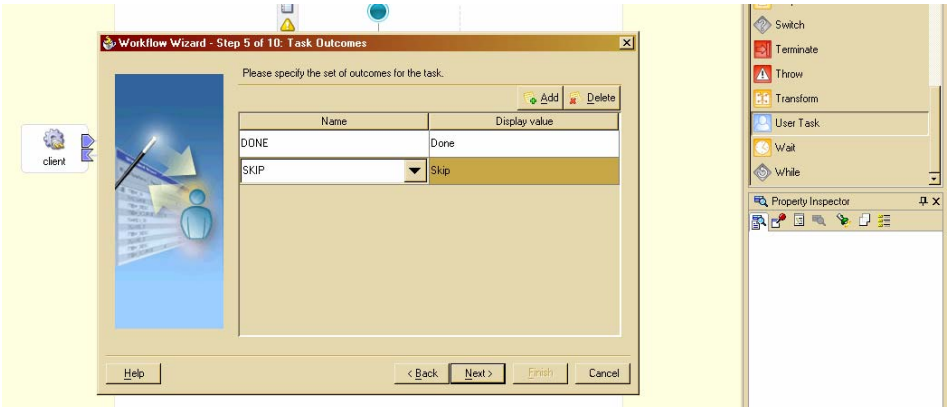


Figure 52 Skip in workflow wizard

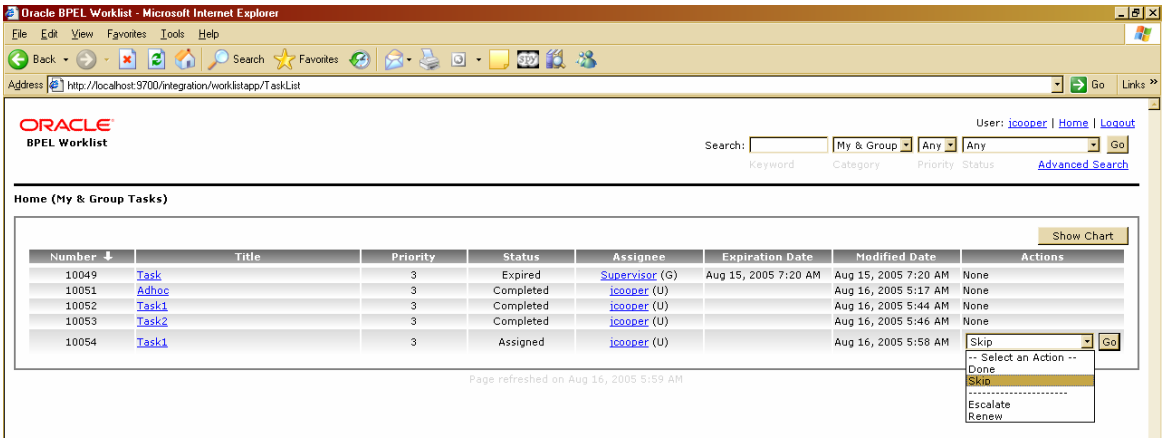


Figure 53 Skip in the work list application

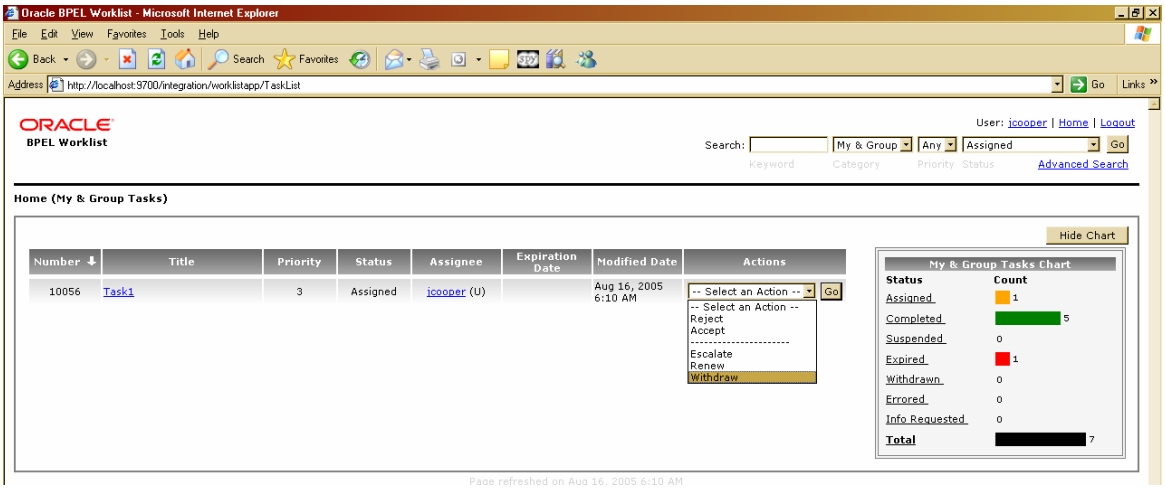


Figure 54 Withdraw

RP34 (Redo)
Description: The ability for a resource to redo a work item that has previously been completed in a case.

Oracle BPEL PM offers no support for this pattern.

RP35 (Pre-Do)
Description: The ability for a resource to execute a work item ahead of the time that it has been offered or allocated to resources working on a given case.

Oracle BPEL PM offers no support for this pattern.

RP36 (Commencement on Creation)
Description: The ability for a resource to commence execution on a work item as soon as it is created.

Oracle BPEL PM offers no support for this pattern since a resource needs to "accept" or "acquire" a work item from the worklist in order to start the execution.

RP37 (Commencement on Allocation)
Description: The ability to commence execution on a work item as soon as it is allocated to a resource.

Oracle BPEL PM offers no support for this pattern since a resource needs to "accept" or "acquire" a work item from the worklist in order to start the execution.

RP38 (Piled Execution)
Description: The ability of the workflow system to initiate the next instance of a workflow task (perhaps in a different case) once the previous one has completed.

Oracle BPEL PM offers no support for this pattern.

RP39 (Chained Execution)

Description: The ability of the workflow engine to automatically start the next work item in a case once the previous one has completed.

Although Oracle BPEL PM offers the "continue task" pattern which allows one workflow to be continued with another workflow, the transition between the workflows is not automatic and requires a work item to be selected from the worklist. Therefore, Oracle BPEL PM offers no support for this pattern.

RP40 (Configurable Unallocated Work Item Visibility)
Description: The ability to configure the visibility of unallocated work items by workflow participants.

Oracle BPEL PM offers no support for this pattern, since any user can see all unallocated work items and there is no option to limit the visibility of unallocated items.

RP41 (Configurable Allocated Work Item Visibility)
Description: The ability to configure the visibility of allocated work items by work-flow participants.

Oracle BPEL PM offers no support for this pattern, since any user can see all allocated work items and there is no option to limit the visibility of allocated items.

RP42 (Simultaneous Execution)
Description: The ability for a resource to execute more than one work item simultaneously.

Oracle BPEL PM supports this pattern partially by allowing a resource to work with multiple browsers related to a single worklist, and thus enabling and executing several work items simultaneously.

RP43 (Additional Resources)
Description: The ability for a given resource to request additional resources to assist in the execution of a work item that they are currently undertaking.

Oracle BPEL PM supports this pattern directly. It offers an "adhoc" pattern which allows assigning the task to any other user run-time and "request for more information" from other users and have them submit information for tasks. Figure 55 - Figure 59 visualize the rerouting of a task and the request for more information.

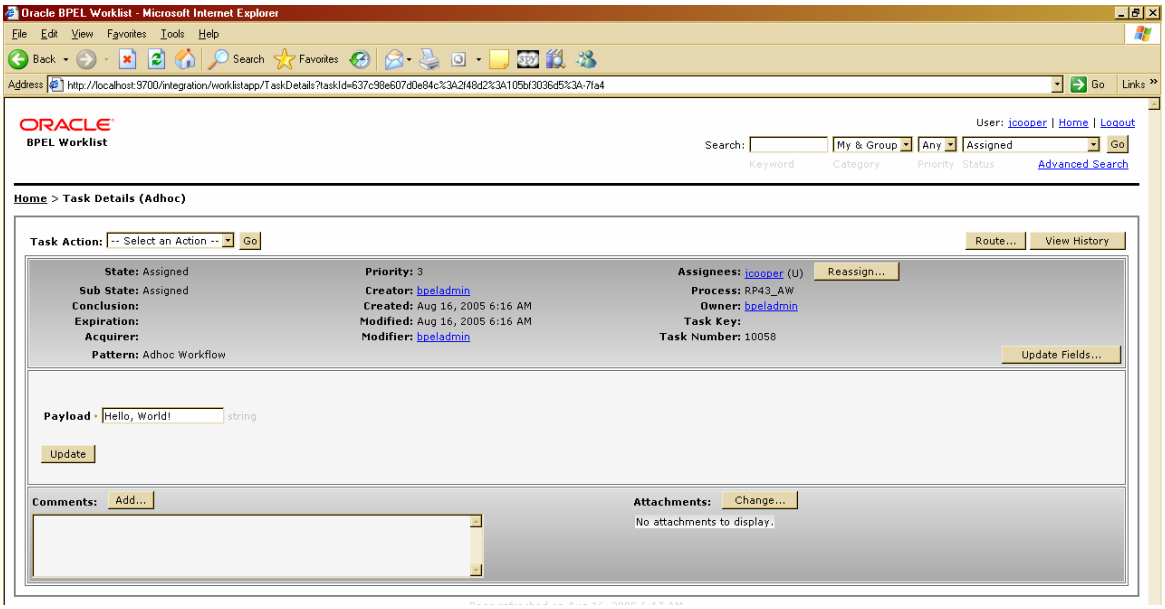


Figure 55 Reroute

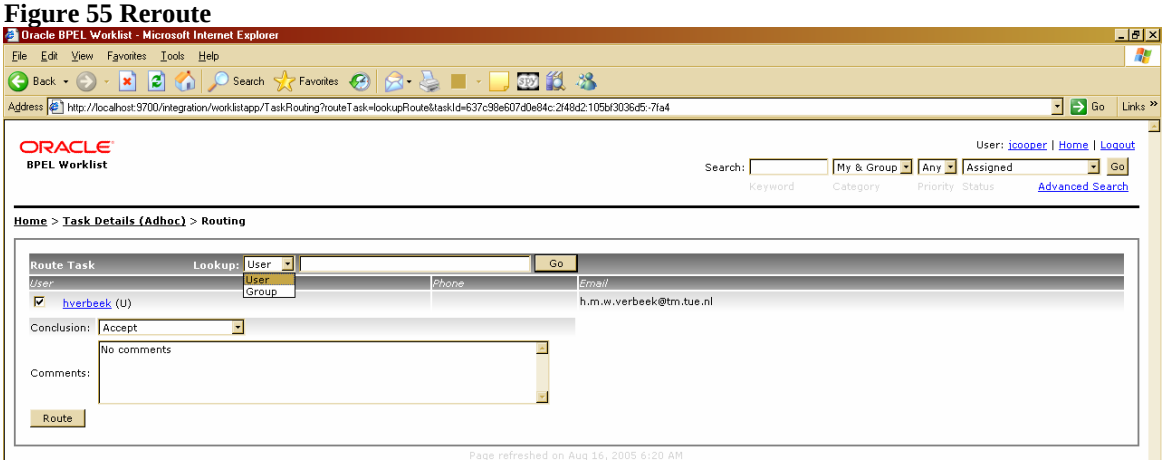


Figure 56 Reroute -2

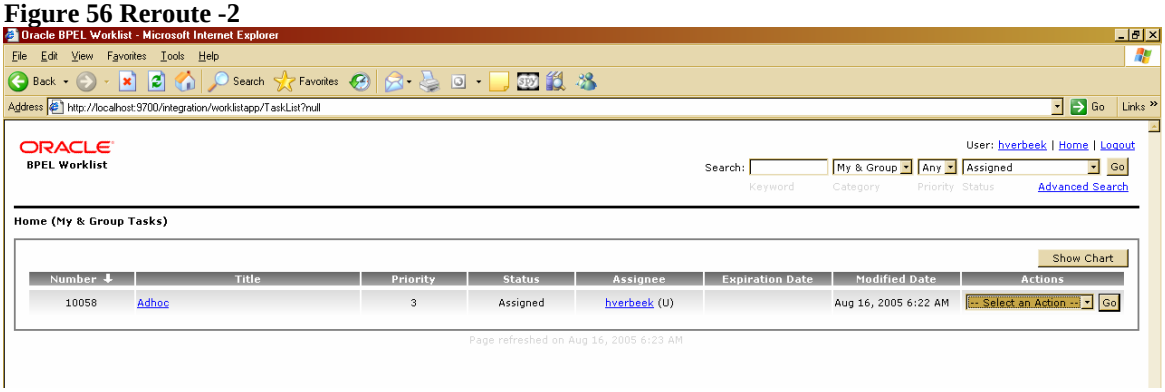


Figure 57 Reroute-3

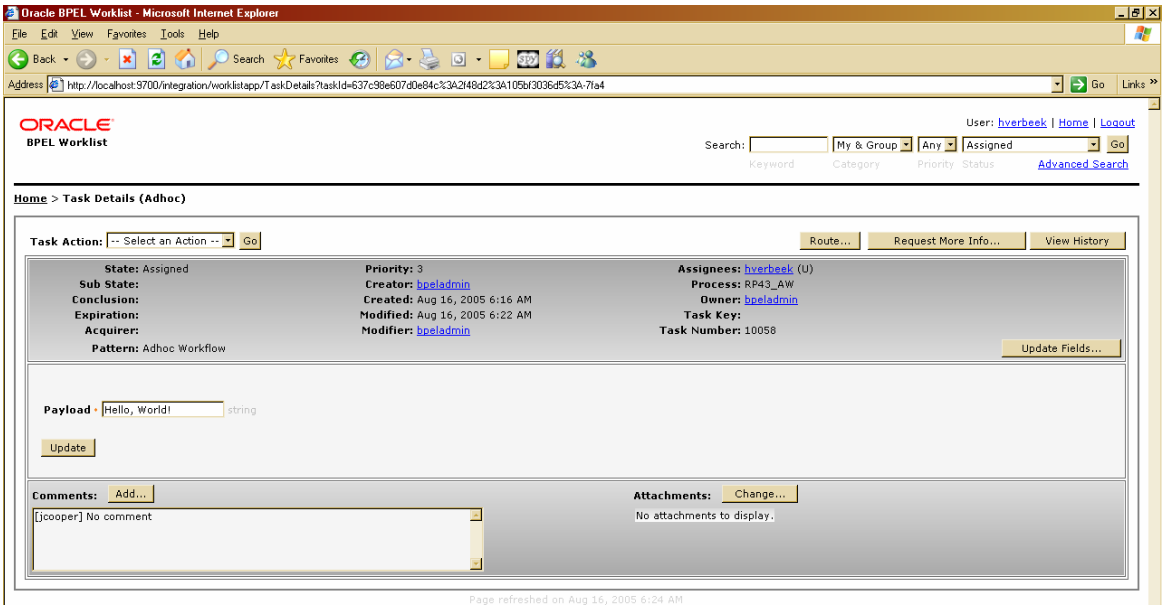


Figure 58 Request for more information 1

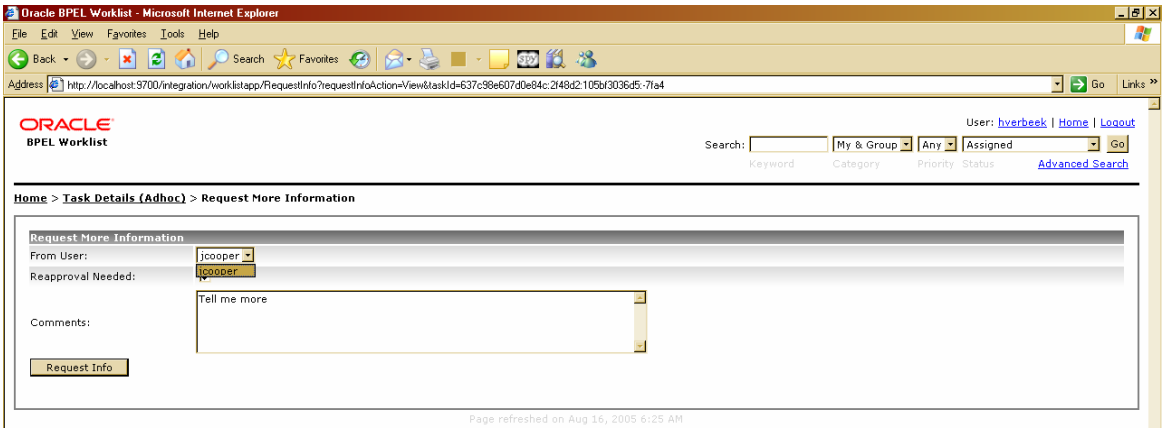


Figure 59 Request for more information -2

Conclusions

The evaluation of Oracle BPEL PM from the control-flow, data and resource perspectives is summarized in Table 1, Table 2 and Table 3 respectively. The evaluation shows that Oracle BPEL PM supports the majority of control patterns, data patterns and resource patterns.

Table 1 Support of Control-flow patterns in Oracle BPEL PM

	Pattern Name	Supported (+: directly, +/-: workaround; -: not supported)	Remarks
CFP1	Sequence	+	by <sequence> or links within <flow>
CFP2	Parallel Split	+	by <flow>
CFP3	Synchronization	+	by <flow>
CFP4	Exclusive Choice	+	by <switch> or links within <flow>
CFP5	Simple Merge	+	by <switch> or links within <flow>
CFP6	Multi-Choice	+	by links within <flow>
CFP7	Synchronizing Merge	+	by links within <flow>
CFP8	Multi-Merge	-	not supported
CFP9	Discriminator	-	not supported
CFP10	Arbitrary Cycles	-	supports only structured loop <while>
CFP11	Implicit Termination	+	by default
CFP12	MI without Synchronization	+	by <invoke> within <while> loop
CFP13	MI with a priori known design time knowledge	+	replication of activities within <flow>
CFP14	MI with a priori known runtime knowledge	+	<flowN>
CFP15	MI with no a priori runtime knowledge	+/-	by <pick> in the <while> loop
CFP16	Deferred Choice	+	by <pick> construct
CFP17	Interleaved Parallel Routing	-	supported by BPEL spec, but not by investigated version of Oracle BPEL PM
CFP18	Milestone	+/-	by <pick> in the <while> loop
CFP19	Cancel Activity	+/-	by messaging and fault handlers
CFP20	Cancel Case	+	by <terminate>

Table 2 Support for Data Patterns in Oracle BPEL PM

Nr	Pattern name	Support: direct (+); partial (+/-); no support (-)	Remarks
	Data visibility		
DP1	Task Data	+/-	A task must be wrapped into a scope
DP2	Block Data	-	Not supported
DP3	Scope Data	+	Directly supported by <scope>
DP4	Multiple Instance Data	+/-	Partial support dependents on the type of the MI task
DP5	Case Data	+	Bound to outermost scope in the process definition
DP6	Folder Data	-	Not supported
DP7	Workflow Data	+	Supported via deployment descriptor properties
DP8	Environment Data	+	Synchronous message interaction <invoke>, <receive>
	Internal data interaction		
DP9	Data interaction – Task to Task	+	No data passing; data elements shared between tasks via access to globally shared data
DP10	Data interaction – Block Task to Sub-Workflow Decomposition	-	Not supported
DP11	Data interaction – Data interaction to Multiple Instance Task	-	Not supported
DP12	Data interaction – to Multiple Instance Task	+/-	Not supported by all variants of MI tasks
DP13	Data interaction – from Multiple Instance Task	+/-	For non-directly supported MI task, a certain number of instances should complete before the data is passed to the next task
DP14	Data Interaction – Case to Case	-	Not supported
	External data passing		
DP15	Data interaction – Task to Environment – Push-oriented	+	Directly supported by means *inputVariable* of <invoke>
DP16	Data interaction- Environment to Task – Pull –Oriented	+	Direct support via <invoke> and <receive>
DP17	Data interaction – Environment to Task – Push-Oriented	+	Direct support via <receive> and event handlers
DP18	Data interaction – Task to Environment – Pull-oriented	+	Directly supported through <receive> and <reply>
DP19	Data interaction – Case to environment – Push-oriented	-	Not supported
DP20	Data interaction – Environment to Case – Pull-oriented	-	Not supported

DP21	Data interaction – Environment to Case- Push-oriented	-	Not supported
DP22	Data interaction – Case to Environment – Pull-oriented	-	Not supported
DP23	Data interaction- Workflow to Environment – Push-oriented	-	Not supported
DP24	Data interaction- Environment to Workflow – Pull-oriented	-	Not supported
DP25	Data interaction – Environment to Workflow – Push-Oriented	-	Not supported
DP26	Data interaction – Workflow to Environment – Pull-oriented	-	Not supported
	Data transfers mechanisms		
DP27	Data passing by Value - Incoming	+	Directly supported by the attributes of <assign> wizard
DP28	Data passing by Value- Outgoing	+	Directly supported by the attributes of <assign> wizard
DP29	Data passing – Copy In/Copy Out	+	Directly supported by means of two <assign> constructs
DP30	Data passing by Reference - Unlocked	+	No concurrency restrictions for accessing global data
DP31	Data passing by Reference- Locked	-	The serializable scope feature does not work according to its semantics
DP32	Data transformation - Input	-	Directly supported by the attributes of <assign> wizard
DP33	Data transformation - Output	-	Directly supported by the attributes of <assign> wizard
	Data-based routing		
DP34	Task precondition – Data existence	-	Not supported
DP35	Task Precondition – Data value	+	Directly supported via joinCondition evaluation the status of links
DP36	Task Postcondition – Data existence	-	Not supported
DP37	Task Postcondition – Data value	-	Not supported
DP38	Event-based Task Trigger	+	Directly supported via <receive> and event handlers
DP39	Data-based Task Trigger	-	Not supported
DP40	Data-based Routing	+	Directly supported via links and <switch>

Table 3 Support of Resource Patterns in Oracle BPEL PM

Nr	Pattern name	Support (+:direct; +/-: partial; -: no support)
RP1	Direct allocation	+
RP2	Role-based Allocation	+
RP3	Deferred Allocation	+
RP4	Authorization	-
RP5	Separation of Duties	-
RP6	Case Handling	+
RP7	Retain Familiar	+
RP8	Capability-based Allocation	+
RP9	History-based Allocation	+/-
RP10	Organizational Allocation	+/-
RP11	Automatic Execution	+
RP12	Distribution by Offer- Single Resource	+
RP13	Distribution by Offer- Multiple Resource	+
RP14	Distribution by Allocation- Single Resource	+
RP15	Random Allocation	+/-
RP16	Round Robin Allocation	+/-
RP17	Shortest Queue	+/-
RP18	Early Distribution	-
RP19	Distribution by Enablement	+
RP20	Late Distribution	-
RP21	Resource-Initiated Allocation	-
RP22	Resource-Initiated Execution – Allocated Work Item	+
RP23	Resource-Initiated Execution – Offered Work Item	+
RP24	System-Determined Work List Management	-
RP25	Resource-Determined Work List Management	+
RP26	Selection Autonomy	+
RP27	Delegation	+
RP28	Escalation	+
RP29	Deallocation	+
RP30	Stateful Reallocation	+
RP31	Stateless Reallocation	-
RP32	Suspension/ Resumption	+
RP33	Skip	+
RP34	Redo	-
RP35	Pre-do	-
RP36	Commencement on Creation	-
RP37	Commencement on Allocation	-
RP38	Piled Execution	-
RP39	Chained Execution	-
RP40	Configurable Unallocated Work Item Visibility	-
RP41	Configurable Allocated Work Item Visibility	-
RP42	Simultaneous Execution	+
RP43	Additional Resources	+

Standardness and Completeness of Oracle BPEL PM

In addition to the availability of BPEL implementation, Oracle offers a set of additional tool-specific features that allow more patterns to be supported than the original BPEL4WS specification does. Among them the <flowN> construct allowing the creation of multiple instances of a task at the run-time. Unfortunately, the semantics of this construct is not clear. From the data perspective, Oracle allows performing basic transformation on data and supporting data patterns 31 and 32. Originally, these patterns are not supported by BPEL [1].

There are some inconsistencies in the implementation which were revealed during the evaluation of Oracle BPEL PM:

- The specification of the seriazable scopes (variableAccessSerializable="yes") does not ensure the exclusive access to the shared data.

Related work

In [11] Martin Vasko and Schahram Dustdar offer the results of evaluation of web services workflow patterns in Collaxa. The majority of their results coincide with the ones reported in this document. However, several remarks can be made:

- The authors claim the direct support for the workflow pattern MI with synchronization (all three variants, i.e. with a-priori design-time knowledge, a-priori run-time knowledge, and no a-priori run-time knowledge), while offering the solution only for the MI with synchronization with a-priori design-time knowledge.
- The canceling patterns (Cancel Activity and Cancel Case) are claimed to be directly supported, while only the solution for the case cancellation is provided.

In [1] Petia Wohed, Wil van der Aalst, Marlon Dumas, and Artur ter Hofstede offer the results of the evaluation of the BPEL4WS based on the workflow control and communication patterns. Relating to the evaluation from the control-flow perspective, the majority of the results of their evaluation coincide with the results reported in this document. However, several remarks can be made:

- In [1] the pattern MI with a-priori run-time knowledge is not supported by BPEL, while Oracle BPEL PM supports it directly by the Oracle specific <flowN> construct;
- In [1] the pattern Interleaved Parallel Routing is partially supported, while the suggested solution implemented in Oracle BPEL PM does not seem to work.

In contrast to the evaluations documented in [1] and [11], which focused only on the control-flow perspective, the scope of this work has been extended also by the data and resource perspectives. The work reported in this document is part of the Workflow Pattern Initiative (cf. www.workflowpatterns.com). In the context of this initiative the workflow control, data and resource patterns have been developed that concentrate on the different perspectives of Process-Aware Information Systems [14].

Acknowledgements

The evaluation of Oracle BPEL PM has been performed with involvement of several parties. I would like to thank Eric Verbeek for the direct involvement in the evaluation; Wil van der Aalst, Marlon Dumas, Nick Russels, and Schahram Dustdar for the constructive feedbacks they provided. Finally, I would like to thank Oracle-representatives David Shaffer, Ravi Rangaswami and Bhagat Nainani for the provided information, their support and collaboration in the evaluation of Oracle BPEL PM.

References

[1] P. Wohed, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Pattern-Based Analysis of BPEL4WS. QUT Technical report, FIT-TR-2002-04, Queensland University of Technology, Brisbane, 2002.

[2] M. Bernauer, G. Kramler, G. Kappel, W. Retschitzegger: Specification of Interorganizational Workflows - A Comparison of Approaches. Proceeding of the 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI), Orlando, USA, 2003.

[3] P. Wohed, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Analysis of Web Services Composition Languages: The Case of BPEL4WS. In I.Y. Song, S.W. Liddle, T.W. Ling, and P. Scheuermann, editors, 22nd International Conference on Conceptual Modeling (ER 2003), volume 2813 of Lecture Notes in Computer Science, pages 200-215. Springer-Verlag, Berlin, 2003.

[4] W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Web Service Composition Languages: Old Wine in New Bottles? In G. Chroust and C. Hofer, editors, Proceeding of the 29th EUROMICRO Conference: New Waves in System Architecture, pages 298-305. IEEE Computer Society, Los Alamitos, CA, 2003.

[5] J. Mendling, M. zur Muehlen, A. Price: Standards for Workflow Definition and Execution. In: M. Dumas, A. ter Hofstede, W.M.P. van der Aalst: Process Aware Information Systems, Wiley Publishing, to appear 2005.

[6] J. Mendling, J. Ziemann: EPK-Visualisierung von BPEL4WS Prozessdefinitionen. Accepted for the 7th Workshop Software-Reengineering (WSR 2005), Bad Honnef, Germany, May 2005.

[7] J. Mendling, M. Strembeck, G. Neumann: Extending BPEL4WS for Multiple Instantiation. In: P. Dadam, M. Reichert (eds.): INFORMATIK 2004, Band 2, Proceedings of the 34th Annual Meeting of German Informatics Society (GI), Workshop "Geschäftsprozessorientierte Architekturen" (GPA 2004), Ulm, Germany. Vol. 51 of Lecture Notes in Informatics (LNI), pages 524-529, September 2004.

[8] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. Distributed and Parallel Databases, 14(1):5-51, 2003.

[9] N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Data Patterns. Queensland University of Technology, Brisbane, December, 2004. http://www.bpm.fit.qut.edu.au/projects/babel/docs/DataPatternsRevised.pdf

[10] N. Russell, A.H.M. ter Hofstede, D. Edmond, W.M.P. van der Aalst. Workflow Resource Patterns, 2004, BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven.

[11] Vasko, M., Dustdar, S. An Analysis of Web services Workflow Patterns in Collaxa. European Conference on Web services (ECOWS) 2004, 27 - 30 September 2004, Erfurt, Germany, Springer LNCS.

[12] Oracle BPEL Process Manager provides SOA and Integration Support. Cover pages hosted by OASIS. http://xml.coverpages.org/ni2004-06-30-a.html

[13] Oracle BPEL Process Manager (10.1.2) developers guide, appeared in June 29, 2005 http://download-uk.oracle.com/otndocs/products/bpel/bpeldev.pdf

[14] M. Dumas, W.M.P. van der Aalst, A.H.M. ter Hofstede. Process-Aware Information Systems, 2005, Wiley.

[15] R. Shapiro. A comparison of XPDL, BPML and BPEL4WS (Version 1.4), 2002, http://xml.coverpages.org/Shapiro-XPDL.pdf