Web services:

* interoperability between applications by using Web standards
* specifications:
  + SOAP
    - XML messenging protocol for service interoperability
  + Web Services Description Language (WSDL)
    - common grammar for describing services
  + Universal Description, Discovery, and Integration (UDDI)
    - infrastructure required to publish and discover services in a systematic way

**Business Process Execution Language** (**BPEL** - **Web Services Business Process Execution Language** (**WS-BPEL**)

* standard
* executable language
* describes actions within business processes using web services - orchestration
* exposes the business process as a web service itself
* xml based
* defines control structures such as if, repeat, while and calls to other web services to describe how functionality is obtained

Structure of a business process

<process name="NCName" targetNamespace="anyURI"

   queryLanguage="anyURI"?

   expressionLanguage="anyURI"?

   suppressJoinFailure="yes|no"?

   exitOnStandardFault="yes|no"?

   xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable">

   <extensions>?

      <extension namespace="anyURI" mustUnderstand="yes|no" />+

   </extensions>

   <import namespace="anyURI"?

      location="anyURI"?

      importType="anyURI" />\*

   <partnerLinks>?

      <!-- Note: At least one role must be specified. -->

      <partnerLink name="NCName"

         partnerLinkType="QName"

         myRole="NCName"?

         partnerRole="NCName"?

         initializePartnerRole="yes|no"?>+

      </partnerLink>

   </partnerLinks>

   <messageExchanges>?

      <messageExchange name="NCName" />+

   </messageExchanges>

   <variables>?

      <variable name="BPELVariableName"

         messageType="QName"?

         type="QName"?

         element="QName"?>+

         from-spec?

      </variable>

   </variables>

   <correlationSets>?

      <correlationSet name="NCName" properties="QName-list" />+

   </correlationSets>

   <faultHandlers>?

      <!-- Note: There must be at least one faultHandler -->

      <catch faultName="QName"?

         faultVariable="BPELVariableName"?

         ( faultMessageType="QName" | faultElement="QName" )? >\*

         activity

      </catch>

      <catchAll>?

         activity

      </catchAll>

   </faultHandlers>

   <eventHandlers>?

      <!-- Note: There must be at least one onEvent or onAlarm. -->

      <onEvent partnerLink="NCName"

         portType="QName"?

         operation="NCName"

         ( messageType="QName" | element="QName" )?

         variable="BPELVariableName"?

         messageExchange="NCName"?>\*

         <correlations>?

            <correlation set="NCName" initiate="yes|join|no"? />+

         </correlations>

         <fromParts>?

            <fromPart part="NCName" toVariable="BPELVariableName" />+

         </fromParts>

         <scope ...>...</scope>

      </onEvent>

      <onAlarm>\*

         <!-- Note: There must be at least one expression. -->

         (

         <for expressionLanguage="anyURI"?>duration-expr</for>

         |

         <until expressionLanguage="anyURI"?>deadline-expr</until>

         )?

         <repeatEvery expressionLanguage="anyURI"?>

            duration-expr

         </repeatEvery>?

         <scope ...>...</scope>

      </onAlarm>

   </eventHandlers>

   activity

</process>

Activities:

Receive

<receive partnerLink="NCName"

   portType="QName"?

   operation="NCName"

   variable="BPELVariableName"?

   createInstance="yes|no"?

   messageExchange="NCName"?

   standard-attributes>

   standard-elements

   <correlations>?

      <correlation set="NCName" initiate="yes|join|no"? />+

   </correlations>

   <fromParts>?

      <fromPart part="NCName" toVariable="BPELVariableName" />+

   </fromParts>

</receive>

* allows the business process to wait for a matching message to arrive
* completes when the message arrives
* portType attribute is included for readability, the value of the portType attribute MUST match the portType value implied by the combination of the specified partnerLink and the role implicitly specified by the activity

Reply

<reply partnerLink="NCName"

   portType="QName"?

   operation="NCName"

   variable="BPELVariableName"?

   faultName="QName"?

   messageExchange="NCName"?

   standard-attributes>

   standard-elements

   <correlations>?

      <correlation set="NCName" initiate="yes|join|no"? />+

   </correlations>

   <toParts>?

      <toPart part="NCName" fromVariable="BPELVariableName" />+

   </toParts>

</reply>

* send a message in reply to a message that was received by an inbound message activity (<receive>, <onMessage>, <onEvent>)

Invoke

<invoke partnerLink="NCName"

   portType="QName"?

   operation="NCName"

   inputVariable="BPELVariableName"?

   outputVariable="BPELVariableName"?

   standard-attributes>

   standard-elements

   <correlations>?

      <correlation set="NCName" initiate="yes|join|no"?

         pattern="request|response|request-response"? />+

   </correlations>

   <catch faultName="QName"?

      faultVariable="BPELVariableName"?

      faultMessageType="QName"?

      faultElement="QName"?>\*

      activity

   </catch>

   <catchAll>?

      activity

   </catchAll>

   <compensationHandler>?

      activity

   </compensationHandler>

   <toParts>?

      <toPart part="NCName" fromVariable="BPELVariableName" />+

   </toParts>

   <fromParts>?

      <fromPart part="NCName" toVariable="BPELVariableName" />+

   </fromParts>

</invoke>

* invoke a one-way or request-response operation on a portType offered by a partner
* the invoke activity completes when the response is received

Assign

<assign validate="yes|no"? standard-attributes>

   standard-elements

   (

   <copy keepSrcElementName="yes|no"? ignoreMissingFromData="yes|no"?>

      from-spec

      to-spec

   </copy>

   |

   <extensionAssignOperation>

      assign-element-of-other-namespace

   </extensionAssignOperation>

   )+

</assign>

* update the values of variables with new data

Validate

<validate variables="BPELVariableNames" standard-attributes>

   standard-elements

</validate>

* validate the values of variables against their associated XML and WSDL data definition
* variables - variables being validated

Throw

<throw faultName="QName"

   faultVariable="BPELVariableName"?

   standard-attributes>

   standard-elements

</throw>

* generate a fault from inside the business process

Wait

<wait standard-attributes>

   standard-elements

   (

   <for expressionLanguage="anyURI"?>duration-expr</for>

   |

   <until expressionLanguage="anyURI"?>deadline-expr</until>

   )

</wait>

* wait for a given time period or until a certain point in time has been reached

Empty

<empty standard-attributes>

   standard-elements

</empty>

* no operation

Sequence

<sequence standard-attributes>

   standard-elements

   activity+

</sequence>

* a collection of activities to be performed sequentially in lexical order

If

<if standard-attributes>

   standard-elements

   <condition expressionLanguage="anyURI"?>bool-expr</condition>

   activity

   <elseif>\*

      <condition expressionLanguage="anyURI"?>bool-expr</condition>

      activity

   </elseif>

   <else>?

      activity

   </else>

</if>

* select exactly one activity for execution from a set of choices

While

<while standard-attributes>

   standard-elements

   <condition expressionLanguage="anyURI"?>bool-expr</condition>

   activity

</while>

* define that the child activity is to be repeated as long as the  specified <condition> is true

Repeat Until

<repeatUntil standard-attributes>

   standard-elements

   activity

   <condition expressionLanguage="anyURI"?>bool-expr</condition>

</repeatUntil>

* define that the child activity is to be repeated until the specified <condition> becomes true

For each

<forEach counterName="BPELVariableName" parallel="yes|no"

   standard-attributes>

   standard-elements

   <startCounterValue expressionLanguage="anyURI"?>

      unsigned-integer-expression

   </startCounterValue>

   <finalCounterValue expressionLanguage="anyURI"?>

      unsigned-integer-expression

   </finalCounterValue>

   <completionCondition>?

      <branches expressionLanguage="anyURI"?

         successfulBranchesOnly="yes|no"?>?

         unsigned-integer-expression

      </branches>

   </completionCondition>

   <scope ...>...</scope>

</forEach>

* iterates its child scope activity exactly N+1 times where N equals the <finalCounterValue> minus the <startCounterValue>.
* If parallel="yes" then this is a parallel <forEach> where the N+1 instances of the enclosed <scope> activity SHOULD occur in parallel. In essence an implicit flow is dynamically created with N+1 copies of the <forEach>'s <scope> activity as children.
* A <completionCondition> may be used within the <forEach> to allow the <forEach> activity to complete without executing or finishing all the branches specified.

Pick

<pick createInstance="yes|no"? standard-attributes>

   standard-elements

   <onMessage partnerLink="NCName"

      portType="QName"?

      operation="NCName"

      variable="BPELVariableName"?

      messageExchange="NCName"?>+

      <correlations>?

         <correlation set="NCName" initiate="yes|join|no"? />+

      </correlations>

      <fromParts>?

         <fromPart part="NCName" toVariable="BPELVariableName" />+

      </fromParts>

      activity

   </onMessage>

   <onAlarm>\*

      (

      <for expressionLanguage="anyURI"?>duration-expr</for>

      |

      <until expressionLanguage="anyURI"?>deadline-expr</until>

      )

      activity

   </onAlarm>

</pick>

* wait for one of several possible messages to arrive or for a time-out to occur
* When one of these triggers occurs, the associated child activity is performed.
* When the child activity completes then the <pick> activity completes

Flow

<flow standard-attributes>

   standard-elements

   <links>?

      <link name="NCName" />+

   </links>

   activity+

</flow>

* specify one or more activities to be performed concurrently.
* <links> can be used within a <flow> to define explicit control dependencies between nested child activities

Scope

<scope isolated="yes|no"? exitOnStandardFault="yes|no"?

   standard-attributes>

   standard-elements

   <partnerLinks>?

      ... see above under <process> for syntax ...

   </partnerLinks>

   <messageExchanges>?

      ... see above under <process> for syntax ...

   </messageExchanges>

   <variables>?

      ... see above under <process> for syntax ...

   </variables>

   <correlationSets>?

      ... see above under <process> for syntax ...

   </correlationSets>

   <faultHandlers>?

      ... see above under <process> for syntax ...

   </faultHandlers>

   <compensationHandler>?

      ...

   </compensationHandler>

   <terminationHandler>?

      ...

   </terminationHandler>

   <eventHandlers>?

      ... see above under <process> for syntax ...

   </eventHandlers>

   activity

</scope>

* define a nested activity with its own associated <partnerLinks>, <messageExchanges>, <variables>, <correlationSets>, <faultHandlers>, <compensationHandler>, <terminationHandler>, and <eventHandlers>

Compensate Scope

<compensateScope target="NCName" standard-attributes>

   standard-elements

</compensateScope>

* start compensation on a specified inner scope that has already completed successfully.
* This activity MUST only be used from within a fault handler, another compensation handler, or a termination handler

Compensate

<compensate standard-attributes>

   standard-elements

</compensate>

* start compensation on all inner scopes that have already completed successfully, in default order
* This activity MUST only be used from within a fault handler, another compensation handler, or a termination handler.

Exit

<exit standard-attributes>

   standard-elements

</exit>

* immediately end a business process instance within which the <exit> activity is contained.

Rethrow

<rethrow standard-attributes>

   standard-elements

</rethrow>

* rethrow the fault that was originally caught by the immediately enclosing fault handler.
* The <rethrow> activity MUST only be used within a fault handler (i.e. <catch> and <catchAll> elements).
* This syntactic constraint MUST be statically enforced.

Extension activity

<extensionActivity>

   <*anyElementQName* standard-attributes>

      standard-elements

   </*anyElementQName*>

</extensionActivity>

* extend WS-BPEL by introducing a new activity type. The contents of an <extensionActivity> element MUST be a single element that MUST make available WS-BPEL's standard-attributes and standard-elements.

Standard attributes

name="NCName"? suppressJoinFailure="yes|no"?

         name: No default value (that is, the default is unnamed)

        suppressJoinFailure: When this attribute is not specified for an activity, it inherits its value from its closest enclosing activity or from the process if no enclosing activity specifies this attribute.

The "*standard-elements*" referenced above are:

<targets>?

   <joinCondition expressionLanguage="anyURI"?>?

      bool-expr

   </joinCondition>

   <target linkName="NCName" />+

</targets>

<sources>?

   <source linkName="NCName">+

      <transitionCondition expressionLanguage="anyURI"?>?

         bool-expr

      </transitionCondition>

   </source>

</sources>

In WSDL file:

Partner Link Types

* characterizes the conversational relationship between two services by defining the roles played by each of the services in the conversation and specifying the portType provided by each service to receive messages within the context of the conversation. Each <role> specifies exactly one WSDL portType

<wsdl:definitions name="NCName" targetNamespace="anyURI" ...>

   ...

   <plnk:partnerLinkType name="NCName">

      <plnk:role name="NCName" portType="QName" />

      <plnk:role name="NCName" portType="QName" />?

   </plnk:partnerLinkType>

   ...

</wsdl:definitions>

Partner Links

* services with which the business process interacts

<partnerLinks>

   <partnerLink name="NCName"

      partnerLinkType="QName"

      myRole="NCName"?

      partnerRole="NCName"?

      initializePartnerRole="yes|no"? />+

</partnerLinks>

Example

Test.bpel file

<!-- test BPEL Process [Generated by the Eclipse BPEL Designer] -->

<!-- Date: Wed Oct 31 15:07:35 EET 2012 -->

<bpel:process name=*"test"*

targetNamespace=*"http://eclipse.org/bpel/sample"*

suppressJoinFailure=*"yes"*

xmlns:tns=*"http://eclipse.org/bpel/sample"*

xmlns:bpel=*"http://docs.oasis-open.org/wsbpel/2.0/process/executable"*

>

<!-- Import the client WSDL -->

<bpel:import location=*"testArtifacts.wsdl"* namespace=*"http://eclipse.org/bpel/sample"*

importType=*"http://schemas.xmlsoap.org/wsdl/"* />

<!-- ================================================================= -->

<!-- PARTNERLINKS -->

<!-- List of services participating in this BPEL process -->

<!-- ================================================================= -->

<bpel:partnerLinks>

<!-- The 'client' role represents the requester of this service. -->

<bpel:partnerLink name=*"client"*

partnerLinkType=*"tns:test"*

myRole=*"testProvider"*

/>

</bpel:partnerLinks>

<!-- ================================================================= -->

<!-- VARIABLES -->

<!-- List of messages and XML documents used within this BPEL process -->

<!-- ================================================================= -->

<bpel:variables>

<!-- Reference to the message passed as input during initiation -->

<bpel:variable name=*"input"*

messageType=*"tns:testRequestMessage"*/>

<!--

Reference to the message that will be returned to the requester

-->

<bpel:variable name=*"output"*

messageType=*"tns:testResponseMessage"*/>

</bpel:variables>

<!-- ================================================================= -->

<!-- ORCHESTRATION LOGIC -->

<!-- Set of activities coordinating the flow of messages across the -->

<!-- services integrated within this business process -->

<!-- ================================================================= -->

<bpel:sequence name=*"main"*>

<!-- Receive input from requester.

Note: This maps to operation defined in test.wsdl

-->

<bpel:receive name=*"receiveInput"* partnerLink=*"client"*

portType=*"tns:test"*

operation=*"process"* variable=*"input"*

createInstance=*"yes"*/>

<!-- Generate reply to synchronous request -->

<bpel:assign validate=*"no"* name=*"Assign"*>

<bpel:copy>

<bpel:from><bpel:literal><tns:testResponse xmlns:tns=*"http://eclipse.org/bpel/sample"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*>

<tns:result>tns:result</tns:result>

</tns:testResponse>

</bpel:literal></bpel:from>

<bpel:to variable=*"output"* part=*"payload"*></bpel:to>

</bpel:copy>

<bpel:copy>

<bpel:from part=*"payload"* variable=*"input"*>

<bpel:query queryLanguage=*"urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"*><![CDATA[tns:input]]></bpel:query>

</bpel:from>

<bpel:to part=*"payload"* variable=*"output"*>

<bpel:query queryLanguage=*"urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"*><![CDATA[tns:result]]></bpel:query>

</bpel:to>

</bpel:copy>

</bpel:assign>

<bpel:reply name=*"replyOutput"*

partnerLink=*"client"*

portType=*"tns:test"*

operation=*"process"*

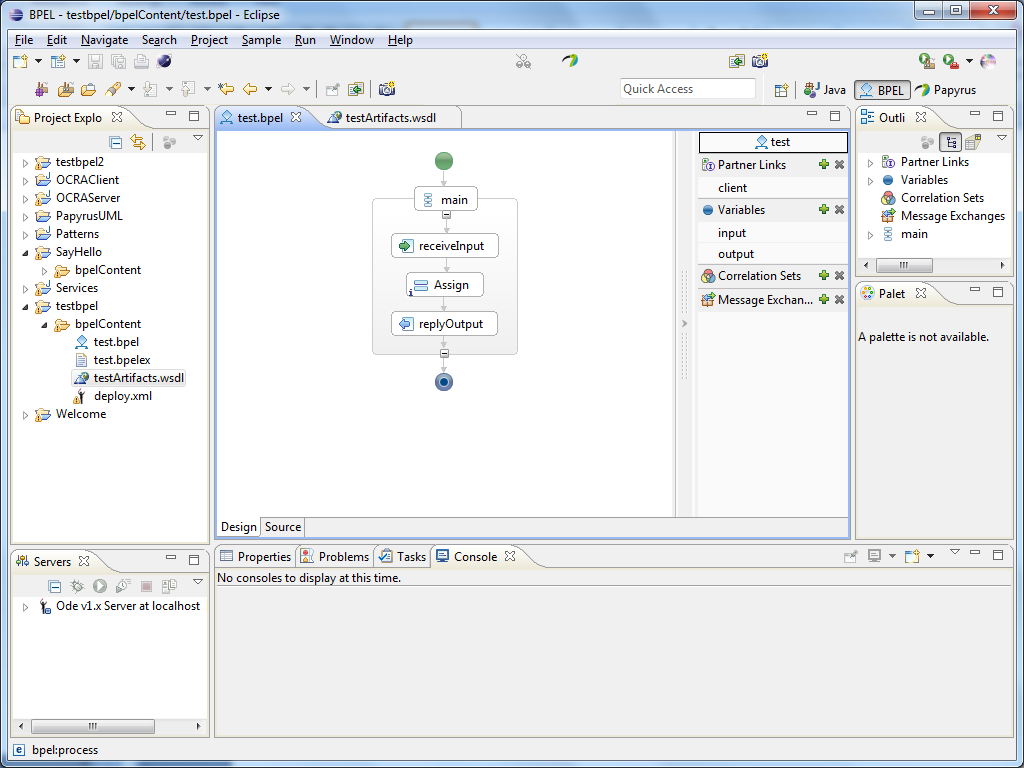
variable=*"output"*

/>

</bpel:sequence>

</bpel:process>

Graphical representation in BPEL Designer



bpelArtifacts.wsdl

<?xml version=*"1.0"*?>

<definitions name=*"test"*

targetNamespace=*"http://eclipse.org/bpel/sample"*

xmlns:tns=*"http://eclipse.org/bpel/sample"*

xmlns:plnk=*"http://docs.oasis-open.org/wsbpel/2.0/plnktype"*

xmlns=*"http://schemas.xmlsoap.org/wsdl/"*

xmlns:soap=*"http://schemas.xmlsoap.org/wsdl/soap/"*

>

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

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=*"unqualified"* elementFormDefault=*"qualified"*

targetNamespace=*"http://eclipse.org/bpel/sample"*

xmlns=*"http://www.w3.org/2001/XMLSchema"*>

<element name=*"testRequest"*>

<complexType>

<sequence>

<element name=*"input"* type=*"string"*/>

</sequence>

</complexType>

</element>

<element name=*"testResponse"*>

<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=*"testRequestMessage"*>

<part name=*"payload"* element=*"tns:testRequest"*/>

</message>

<message name=*"testResponseMessage"*>

<part name=*"payload"* element=*"tns:testResponse"*/>

</message>

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

PORT TYPE DEFINITION - A port type groups a set of operations into

a logical service unit.

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

<!-- portType implemented by the test BPEL process -->

<portType name=*"test"*>

<operation name=*"process"*>

<input message=*"tns:testRequestMessage"* />

<output message=*"tns:testResponseMessage"*/>

</operation>

</portType>

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

PARTNER LINK TYPE DEFINITION

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

<plnk:partnerLinkType name=*"test"*>

<plnk:role name=*"testProvider"* portType=*"tns:test"*/>

</plnk:partnerLinkType>

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

BINDING DEFINITION - Defines the message format and protocol details

for a web service.

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

<binding name=*"testBinding"* type=*"tns:test"*>

<soap:binding style=*"document"*

transport=*"http://schemas.xmlsoap.org/soap/http"* />

<operation name=*"process"*>

<soap:operation

soapAction=*"http://eclipse.org/bpel/sample/process"* />

<input>

<soap:body use=*"literal"* />

</input>

<output>

<soap:body use=*"literal"* />

</output>

</operation>

</binding>

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

SERVICE DEFINITION - A service groups a set of ports into

a service unit.

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

<service name=*"testService"*>

<port name=*"testPort"* binding=*"tns:testBinding"*>

<soap:address location=*"http://localhost:8080/ode/processes/test"* />

</port>

</service>

</definitions>