MTIS

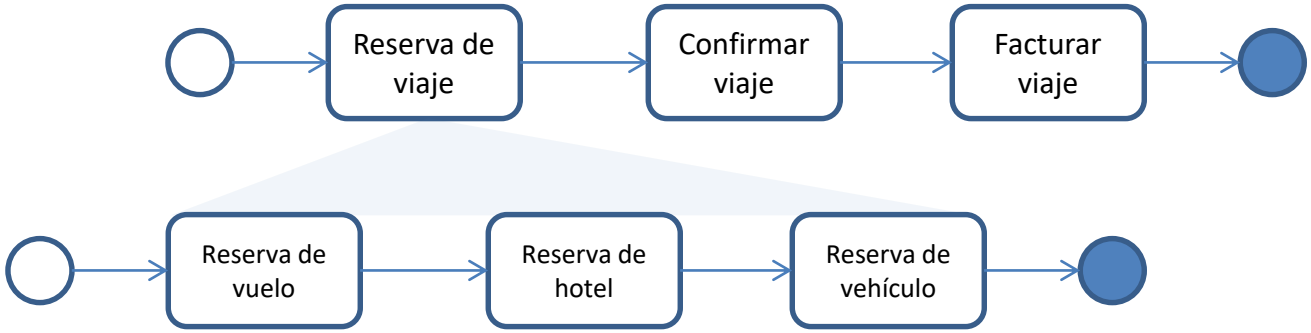
Introducción a la composición de servicios

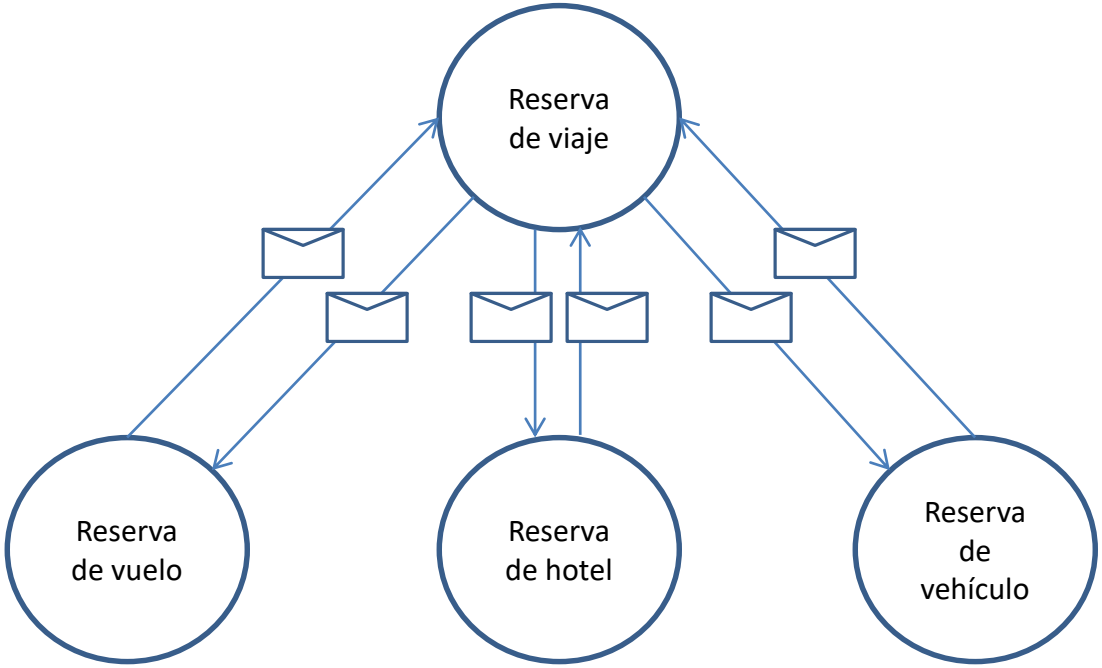
Profesor: Alejandro Sirvent Llamas
Curso: 2019-2020

- Composición de Servicios
- Orquestación de Servicios
- BPEL

- Introducción
  - Agregación coordinada de servicios
  - Pilar fundamental de SOA
  - Acercamiento mundo de negocios e IT
    - Servicios compuestos = Procesos de negocio
    - Servicios atómicos = Actividades
  - Los servicios pueden ser usados o invocados para generar nuevos servicios
    - Encapsulan lógica necesaria para el nuevo servicio

Ejemplo de composición





- Beneficios de la composición
  - Agilidad
    - Alineación procesos de negocio y IT
    - Generación de nuevas aplicaciones (Servicios)
  - Reusabilidad de servicios existentes
  - Productividad
  - ROI

- Retos de la composición
  - Automatización de la composición
  - Composición dinámica e inteligente en base a una base de conocimiento
    - Ontologías (formulación de un exhaustivo y riguroso análisis)
    - Web Semántica

- Tipos
  - Orquestación
    - Coordinación centralizada del flujo de servicios
      - Workflows
    - Centrada en la ejecución de procesos
    - El coordinador centraliza la lógica de interacción
  - Coreografía
    - Coordinación distribuida
      - Protocolos
    - Centrada en la secuencia de mensajes entre las partes implicadas
    - Colaboración entre las partes
    - Ausencia de coordinador central que gestione el funcionamiento del proceso

- Business Process Execution Language
  - Propuesta inicial de IBM y MS
    - Unión de esfuerzos
    - Posteriormente
      - SAP, BEA Systems, Siebel Systems
    - WSFL + XLANG → BPEL4WS → WS-BPEL 2.0 (BPEL)
  - Trasladado a OASIS para su estandarización
    - Estándar de facto
  - Lenguaje de ejecución de procesos de negocio expuestos como Servicios Web
    - Orquestación de servicios
  - Basado en XML

- Favorece la automatización de aplicaciones bajo un enfoque SOA
- Integración de aplicaciones
  - Reutilización de sistemas heredados
    - Funcionalidades como servicios de envoltorio
    - Adaptabilidad a los requerimientos de negocio
- Define el orden exacto en el cual los Servicios Web serán invocados
- Trabaja con WSDL y espacios de nombres

- Tipos de procesos
  - Abstractos
    - Definir conversaciones y protocolos para el uso de un Servicio Web o cómo colaborar entre varios Servicios Web
  - Ejecutables
    - Flujo donde cada entidad implicada es un Servicio Web
    - Se ofrece a su vez como Servicio Web

- Estructura
  - Dos tipos de archivos
    - Archivo BPEL: XML que define el proceso
      - Flujos (secuencial o paralelo)
      - Condicionales
      - Bucles
      - Variables
      - Asignación y copia de valores
      - Excepciones
      - Enlaces
      - Eventos
    - Archivo WSDL: especifica la interfaz del Servicio Web
      - El proceso se expone como un nuevo Servicio Web para que pueda ser invocado

Estructura general BPEL

Proceso: Define el nombre del proceso y el espacio de nombres

Socios: Servicios y procesos con los cuales va a interactuar el proceso

Variables globales: Variables que se pueden usar en el proceso completo

Cuerpo del proceso: Flujo del proceso compuesto por una actividad formada por subactividades

- Socios (partners)
  - Representan a los Servicios Web invocados desde el proceso actual
  - A su vez pueden ser procesos
- Estructuras
  - PartnerLinkTypes (tipo de socio)
    - Enlace genérico para una categoría de Servicios Web
    - Similar al concepto de clase en OO
  - PartnerLinks (enlace de socio):
    - Define el Servicio Web que será invocado
    - Instancia de una clase en OO

- PartnerLinkType
  - Define la relación entre dos servicios para establecer una conversación
  - Define el **rol** que juega cada uno de los servicios en la conversación
  - Indica los *PortTypes* para establecer dicha conversación
    - Cada *rol* especifica un *portType* incluido en una WSDL

El PartnerLinkType es definido en una WSDL externa al proceso BPEL

```
<plnk:partnerLinkType name="BuyerSellerLink">
  <plnk:role name="Buyer" portType="buy:BuyerPortType" />
  <plnk:role name="Seller" portType="sell:SellerPortType" />
</plnk:partnerLinkType>
```

- PartnerLinkType
  - Define la relación entre dos servicios para establecer una conversación
  - Define el **rol** que juega cada uno de los servicios en la conversación
  - Indica los *PortTypes* para establecer dicha conversación
    - Cada *rol* especifica un *portType* incluido en una WSDL

El PortType es definido en una WSDL externa

```
<portType name="PurchaseOrderPortType">
        <operation name="PurchaseOrder">
          <input message="searchdef:bookSearchMessage"/>
          <output message="searchdef:bookSearchReturn"/>
        </operation>
</portType>
```

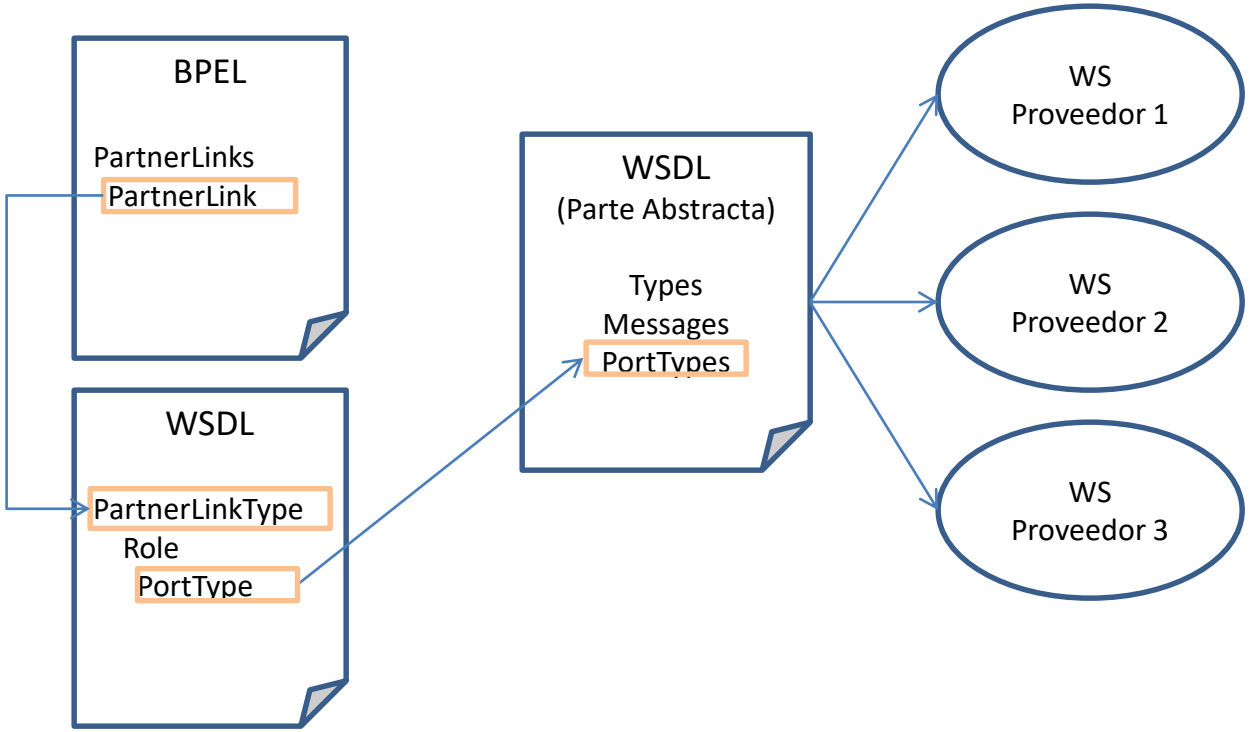
BPEL Elementos

- PartnerLink
  - Servicios con los que el proceso interactúa
  - Modelados en el documento BPEL
  - Caracterizado por el *PartnerLinkType*
    - Por ejemplo, un proceso de compras podría contactar con diferentes proveedores pero compartir el *PartnerLinkType*
  - Definición
    - Nombre único
    - Tipo de partner, alguno de los *PartnerLinkTypes* definidos
    - El rol del proceso de negocio mismo
    - El rol del partner que se invocará

Cuando un PartnerLinkType tiene un rol sólo definiremos uno de los dos

```
<partnerLinks>
  <partnerLink name="PurchaseOrder"
    partnerLinkType="lns:BuyerSellerLink"
    myRole="Buyer"
    partnerRole="Seller">
</partnerLinks>
```

Introducción a la composición de servicios



BPEL Elementos

- Variables
  - Almacenar datos utilizados durante el proceso
  - Los tipos de las variables se definen
    - En el documento WSDL asociado al proceso
    - En un documento WSDL externo
    - Pueden ser definidos como
      - Mensajes completos WSDL
      - Tipos XML básicos o compuestos
      - Elementos XML

- Variables
  - Ejemplos
    - <variable name="orderRequest" messageType="orderdef:orderRequestMessage"/>
    - <variable name="orderResponse" messageType="orderdef:orderResponseMessage"/>
    - <variable name="numResults" type="xsd:int"/>
    - <variable name="orderResult" element="orderdef:orderResult"/>

- Un flujo BPEL se compone de una serie de pasos denominados *Actividades* (activities)
  - Tipos
    - Primitivas
      - *<invoke>*: invocar a un Servicio Web o BPEL
      - *<receive>*: Esperar que el cliente invoque el proceso BPEL
      - *<reply>*: Generar una respuesta para una petición síncrona
      - *<assign>*: Manipular valores de variables
      - *<throw>*: Lanzar excepciones
      - *<wait>*: Esperar un tiempo determinado
      - *<exit>*: Finaliza la instancia del proceso
      - *<empty>*: No hace nada. Usada para sincronizar

- Tipos
  - *<pick>*: esperar que suceda un evento de entre un conjunto
  - *<scope>*: define el ámbito de las variables (global o local)
  - *<compensate>*: compensación en transacciones
  - *<validate>*: Permite validar el esquema de variables XML
  - *<extensionActivity>*: Permite añadir extensiones propietarias
  - Estructuradas
    - *<sequence>*: invocación secuencial de actividades
    - *<if>*: indica condición para realizar alguna actividad
    - *<flow>*: invocación de actividades en paralelo
    - *<switch>*: elección de casos
    - *<while> <repeatuntil> <foreach>*: bucle iterativo

- Invoke
  - Indica el enlace del socio
  - Indica el tipo de puerto
  - Indica la operación que se invocará
  - La variable que contiene el mensaje de la petición
  - La variable que contiene el mensaje resultante

```
<invoke name="invokeBookSearch" partnerLink=“bookSearcher"
portType="apns:bookSearchPT"
operation=“bookSearch"
inputVariable=“searchRequest"
outputVariable=“searchResponse">
</invoke>
```

- Receive
  - Indica el enlace del socio
  - Indica el tipo de puerto
  - Indica la operación que se invocará
  - La variable que almacenará el mensaje de la petición

```
<receive name="receivePetition" partnerLink="bookSearcher"
portType="apns:bookSearchPT"
operation="bookSearch" variable="searchRequest"
createInstance="yes">
```

  - *CreateInstance* indica si se ha de lanzar un nueva instancia con la recepción de la petición

- Reply
  - Indica el enlace del socio
  - Indica el tipo de puerto
  - Indica la operación que se invocará
  - La variable que contiene el mensaje de respuesta

```
<reply name="replyPetition" partnerLink=“bookSearcher"
portType="apns:bookSearchPT"
operation=“bookSearch" variable=“searchResponse">
</reply>
```

- Assign
  - Manipulación de datos
  - Basado en Xpath y funciones predefinidas
    - Puede soportar otros lenguajes (Xquery)
  - Permite transformaciones con XSLT
    - Función *doXslTransform*
  - Xpath
    - Lenguaje para manipulación de datos XML
      - Acceso a partes de un documento XML mediante ruta
      - http://www.w3.org/TR/xpath
      - http://www.w3schools.com/xpath/default.asp
    - Funciones establecidas

- Ejemplo Xpath

```
<searchresponse>
   <numberResults> 10 </numberReults>
<searchresults>
  <searchresult>
    <title> Veinte años despues </title>
    <author> Alejandro Dumas </author>
    <price currency="euro"> 15 </price>
  </searchresult>
  <searchresult>
    <title> Suave es la noche </title>
    <author> Scott Fitzgerald </author>
    <price currency="euro"> 25 </price>
  </searchresult>
  …
</searchresults>
…
</searchresponse>
```

- /searchresponse/searchresults/searchresult[0]/title

- Consultas XPath. También soportan:
  - Comodines
    - "/searchresponse/*" extrae todos los elementos hijos de searchresponse.
  - Rutas relativas
    - "//searchresult/title" extrae todos los títulos.
  - Condiciones
    - //searchresult[price>20]/title extrae todos los títulos de libros cuyo precio supere el valor 20.
  - Acceso a atributos
    - //searchresult/price[@currency="euro"]selecciona todos los precios de resultados de búsqueda cuya moneda es el euro.
  - Funciones y condiciones complejas
    - Manipulación de cadenas, operaciones aritméticas,…
  - …

- Ejemplos de expresiones Xpath
  - Constantes:
    - Enteros: 100
    - Cadenas: ‟ADOO‟
    - …
  - Funciones:
    - Concatenación de cadenas
      - concat (“cadena 1‟,”cadena2‟)
    - Manipulación de fechas
      - formatDate (“2004-05-03T15:56:00‟, “MMM dd, yyy‟)
    - …

- Assign
  - Elementos
    - Copy (1 o más)
      - From (origen)
      - To (destino)
      - Los tipos de datos deben ser compatibles
  - Ejemplo

```
<assign name="assign“>
   <copy>
    <from>10</from>
    <to variable="searchResponse" part="numberResults" />
    </copy>
</assign>
```

Mensaje en la WSDL

Una parte del mensaje

BPEL Elementos

- Assign
  - From (origen)
    - Variable
    - Parte de un mensaje contenido en una variable
    - Una query (expresión xpath)
      - Constante
      - Xpath sobre el contenido de una variable
      - Funciones Xpath estándar y BPEL

- Assign
  - Fro
    - V
    - F
    - U

Ejemplo

```
<assign name="assign">
        <copy>
        <from>
          <literal>
            <searchresult> … </searchresult>
             <searchresult> … </searchresult>
          </literal>
        </from>
        <to variable=“searchResponse" part=“searchresults"/>
        </copy>
</assign>
```

- Assign
  - Acceso al contenido de una variable
    - Sintaxis: $nombre_variable.nombre_part
    - Ejemplo
      - $data.results/numberResults

- Assign

Permite operaciones matemáticas

```
<assign name="assign">
        <copy>
            <from> $data.results/numberResults + 1</from>
            <to variable="searchResponse" part ="numberResults"/>
        </copy>
</assign>
```

– $data.results/numberResults

Concatenación de cadenas

```
<assign name="assign">
        <copy>
            <from> concat(„Search Results:",$searchResponse.SearchResults)</from>
            <to variable="data" part ="searchResults"/>
        </copy>
</assign>
```

- Sequence
  - Realización de actividades secuenciales

```
<sequence name="main">
    <receive>…</receive>
    <assign>…</assign>
    <reply>…</reply>
</sequence>
```

- Flow
  - Realización de actividades en paralelo
  - Dependencias entre actividades mediante *Links*

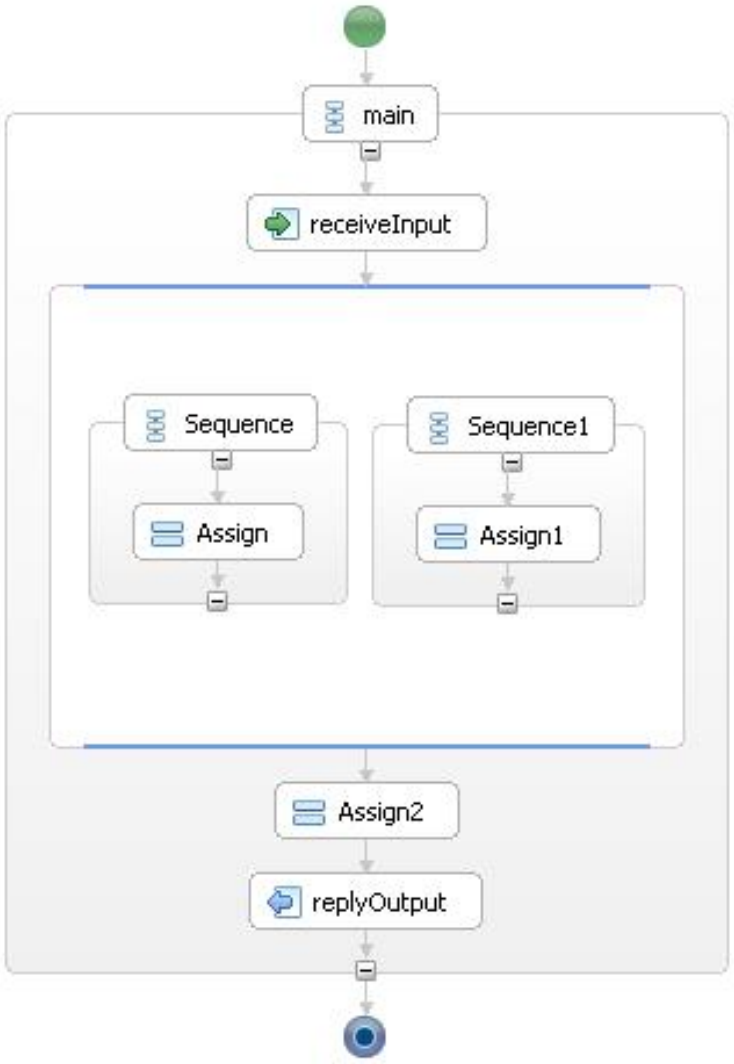
```
<flow name="main">
    <sequence>…</sequence>
    <sequence>…</sequence>
</flow>
```

- Sequence
  - Realizació

```
<sequence na
  <receive>
  <assign>..
  <reply>...<
</sequence>
```

- Flow
  - Realizació
  - Dependen

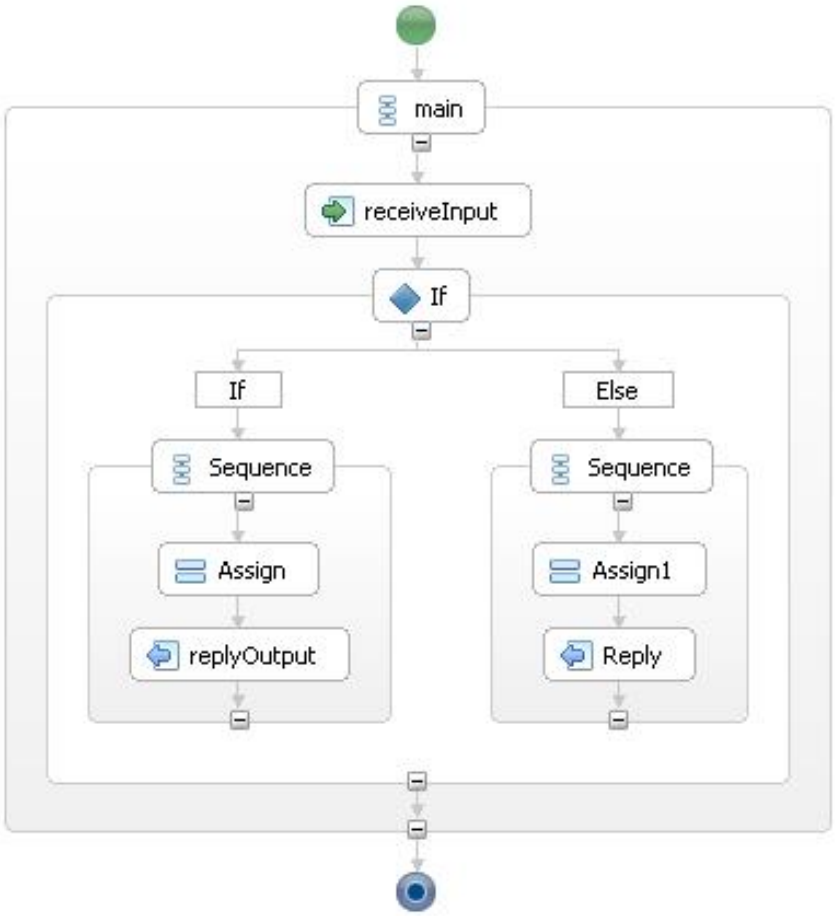
```
<flow name=
  <sequenc
  <sequenc
</flow>
```



- If

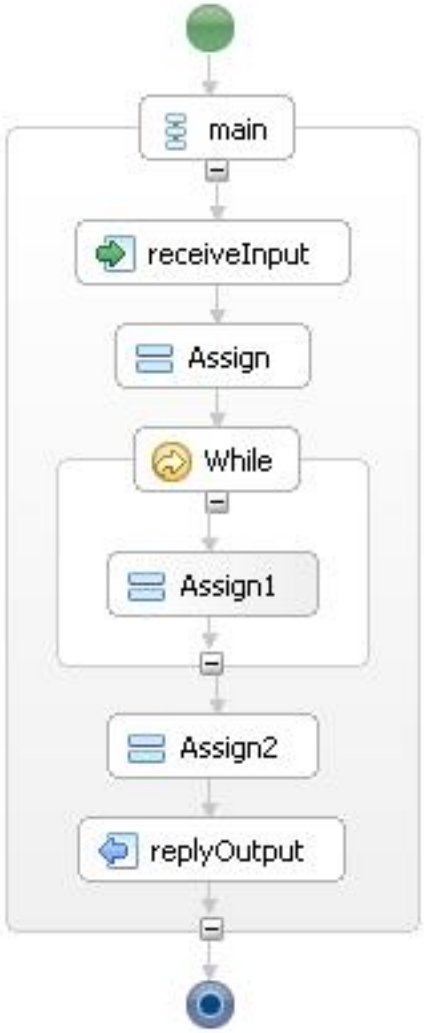
```
<if>
      <condition>$data.part/number > 100</condition>
      <flow>
                              …
      </flow>
     <elseif>
          <condition>$var = true()</condition>
          <sequence>
          …
          </sequence>
     </elseif>
 <elseif>
          <condition>$proveedor1Response.parameters/precio > 0</condition>
          <sequence>
          …
          </sequence>
     </elseif>
      <else>
          <throw faultName="FLT:InvalidValue"/>
     </else>
</if>
```

- If


main
receiveInput
If
If
Else
Sequence
Sequence
Assign
Assign1
replyOutput
Reply

- While

```
<variable name="orderDetails" type="xsd:integer"/>
...
<while>
        <condition>$orderDetails > 100</condition>
        <sequence>
    ...
    </sequence>
</while>
```



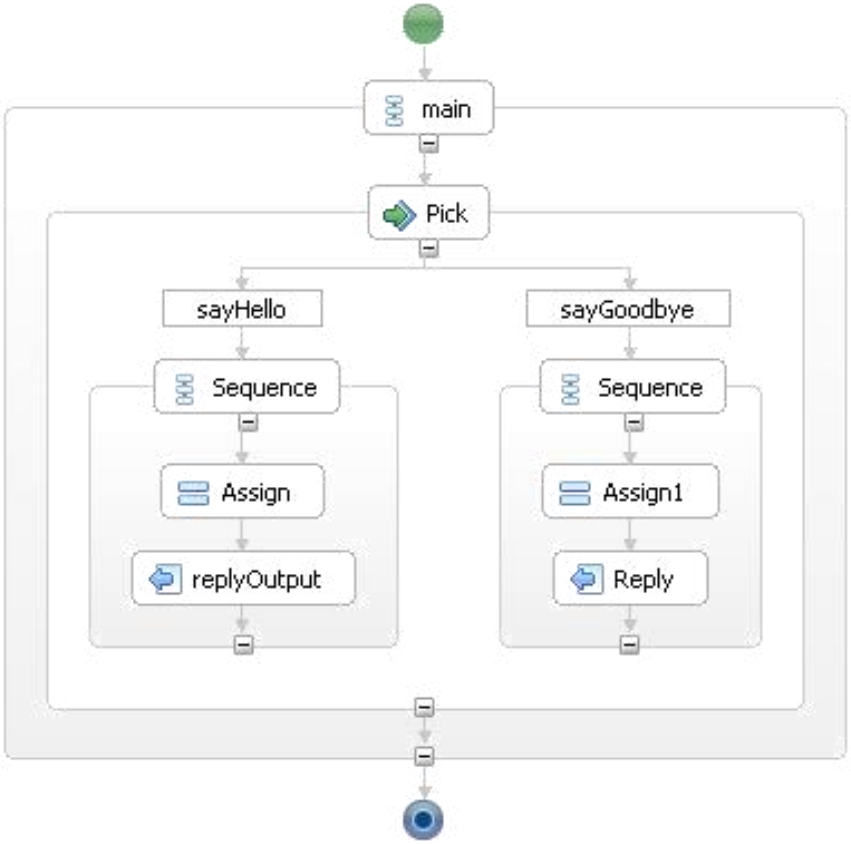
- Pick
  - Actividad orientada a recepción de eventos
  - Fuerza el proceso a que espere hasta que se produzca un evento
  - Tipos
    - OnMessage
      - Similar a la actividad *Reveive* pero espera la llegada de un mensaje
      - La actividad *Receive* es invocada
    - OnAlarm
      - Similar a la actividad *wait*
      - Puede ser
        - » Pro un periodo de tiempo:
        - » Hasta un periodo de tiempo:
  - Puede incluirse un conjunto de ellos pero únicamente se lanzará uno

```
<onAlarm>*
  (
   <for expressionLanguage="anyURI">duration-expr</for>
   |
   <until expressionLanguage="anyURI">deadline-expr</until>
  )
  activity
</onAlarm>
```

- Pic
  - A
  - F
  - Tipos
    - Oı
    - Oı
  - Puede

```
<sequence>
   <invoke partnerLink="Customer" portType="AskPT"
           operation="AskForResponse" ... />
   <pick>
      <onMessage partnerLink="Customer" portType="ResponsePT"
                 operation="ReceiveResponse" ...>
         <assign .../>
      </onMessage>
      <onAlarm>
         <for>'PT5H'</for>
         <!--  Did not receive response within 5 hours -->
         <assign .../>
      </onAlarm>
   </pick>
</sequence>
```

- Pick


main
Pick
sayHello
sayGoodbye
Sequence
Sequence
Assign
Assign1
replyOutput
Reply

- Manejadores
  - Gestión de errores
    - Similar al manejo de excepciones en programación
  - Compensación
    - Tratamiento de transacciones
  - Asociados a un ámbito (*scope*)
    - Similar a un bloque de código en programación

- Herramientas de modelado BPEL
  - Código abierto
    - BPELDesigner
      - Eclipse plug-in
      - http://www.eclipse.org/bpel/
    - Netbeans enterprise pack
      - http://soa.netbeans.org/soa/
    - Jboss
      - RiftSaw BPEL Designer
      - https://community.jboss.org/wiki/RiftSawBPELDesigner

- Motores BPEL
  - Código abierto
    - Apache ODE
      - http://ode.apache.org/
    - Jboss jBPM
      - www.jboss.org/jbpm
    - Jopera
      - www.jopera.org
    - OW2 orchestra
      - http://orchestra.ow2.org
    - WSO2 Business Process Server
      - http://wso2.com/products/business-process-server/

- Algunos inconvenientes
  - Algunas tareas son incómodas
    - Complejidad al manipular datos
    - Código en XML
      - Mayor complejidad que lenguaje de alto nivel
  - Problemas de interoperabilidad
    - Las carencias de BPEL son cubiertas por soluciones propietarias

MTIS

Introducción a la composición de servicios