Servicios Web

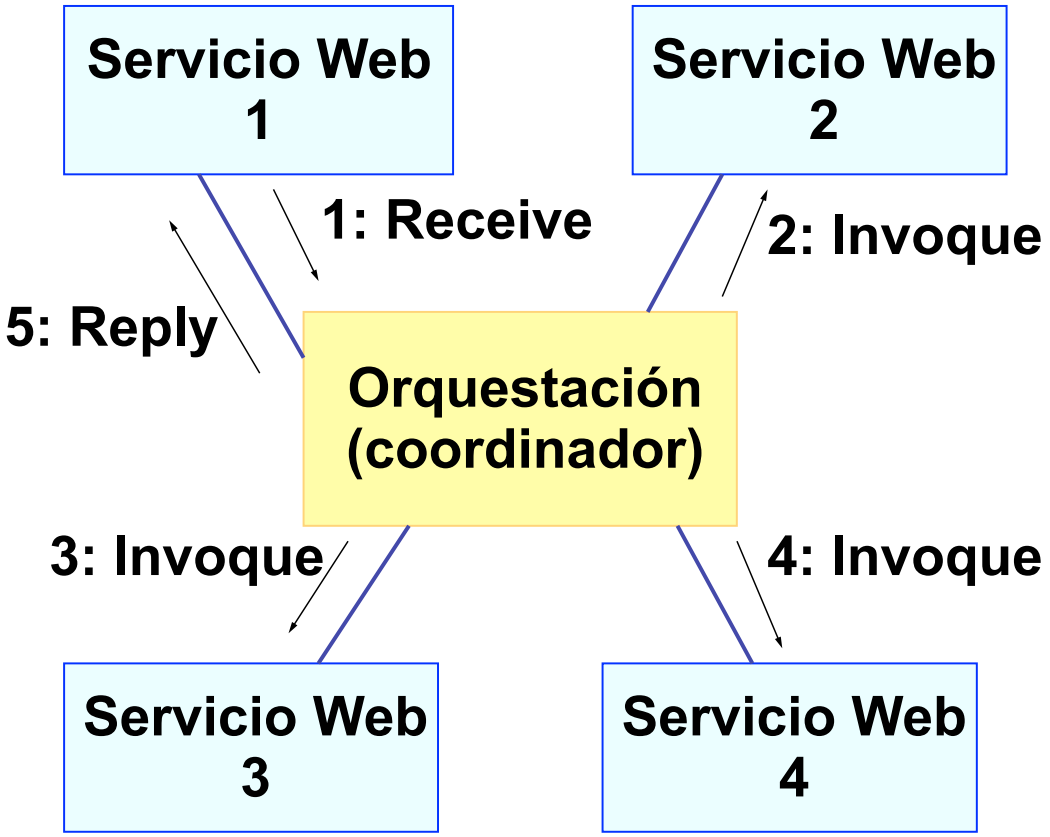
Sesión 3: Orquestación de servicios: BPEL

Puntos a tratar

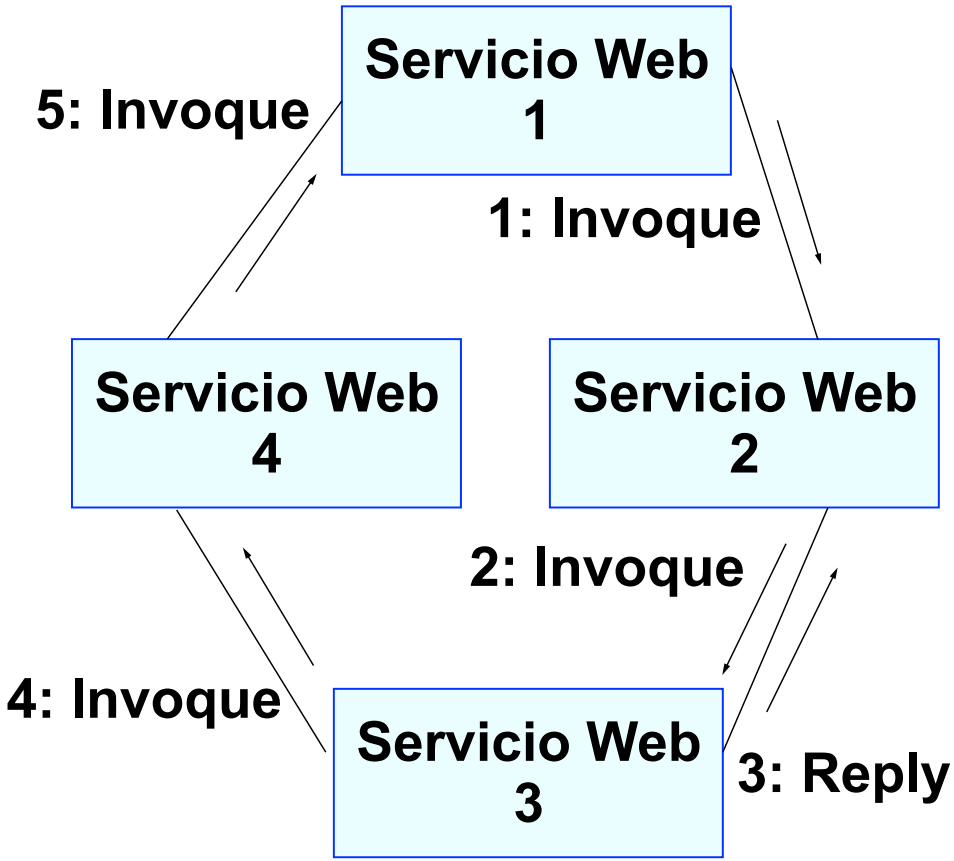
- Orquestación vs. Coreografía
- El lenguaje BPEL
- Estructura de un proceso BPEL
- Pasos para desarrollar un proceso BPEL
- Despliegue y pruebas de un proceso BPEL
- Creación y ejecución de casos de prueba

Orquestación vs. Coreografía

- ORQUESTACIÓN



- COREOGRAFÍA



¿Por qué orquestar servicios Web?

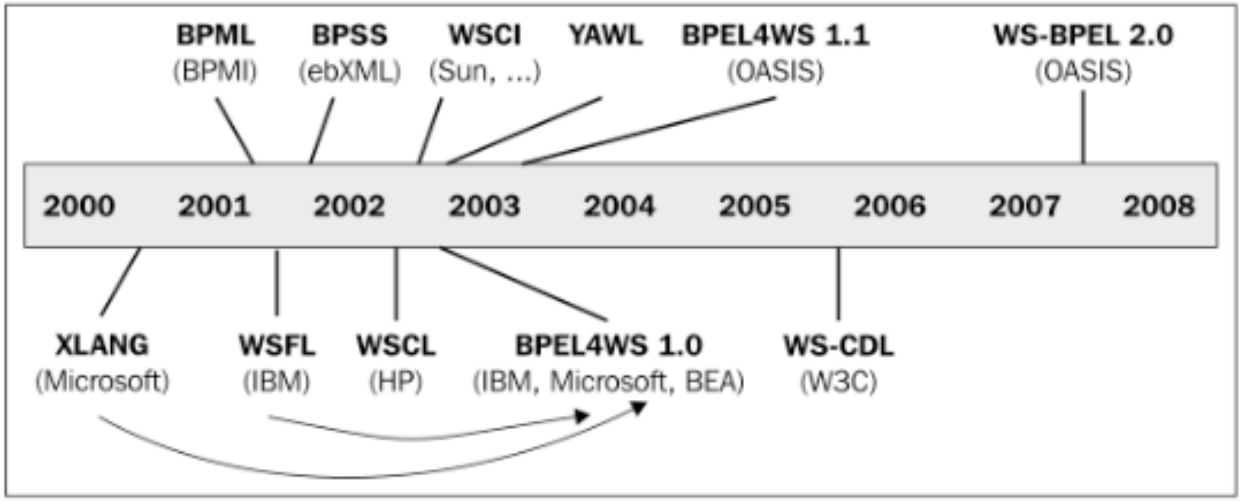
- Los servicios Web como tecnología común para proporcionar puntos de integración entre las aplicaciones
  - Modelo de interfaces que permite integrar las aplicaciones independientemente de su origen
  - Descubrimiento de los servicios en tiempo de ejecución
  - Bajo acoplamiento
- Orquestación de servicios Web como aproximación abierta, basada en estándares para crear procesos de negocio de alto nivel

¿Por qué orquestar Servicios Web con BPEL?

- BPEL es un lenguaje estándar para la integración y automatización de procesos
- Menores costes de mantenimiento
- Menores costes de soporte
- Amplía el grupo de desarrolladores
- BPEL proporciona soporte para:
  - Elevados tiempos de ejecución
  - Compensación
  - Reacción ante eventos
  - Modelado de actividades concurrentes
  - Modelos con estado

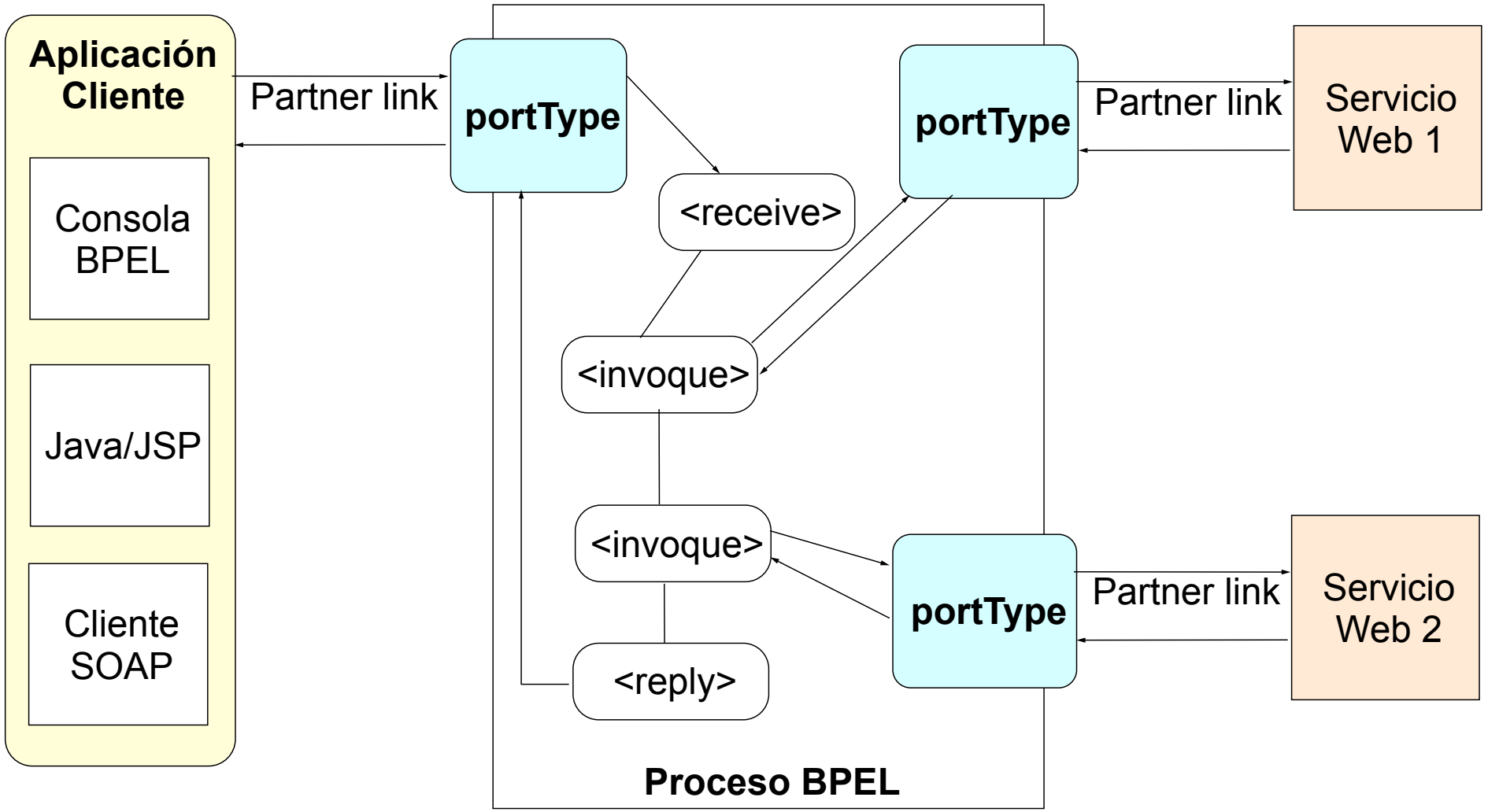
El lenguaje BPEL

- BPEL: **B**usiness **P**rocess **E**xecution **L**anguage
- Lenguaje basado en XML que soporta la tecnología de servicios Web
- Diseñado para definir procesos de negocio
- BPEL es la convergencia entre WSFL (IBM) y XLANG (Microsoft) → BPEL4WS

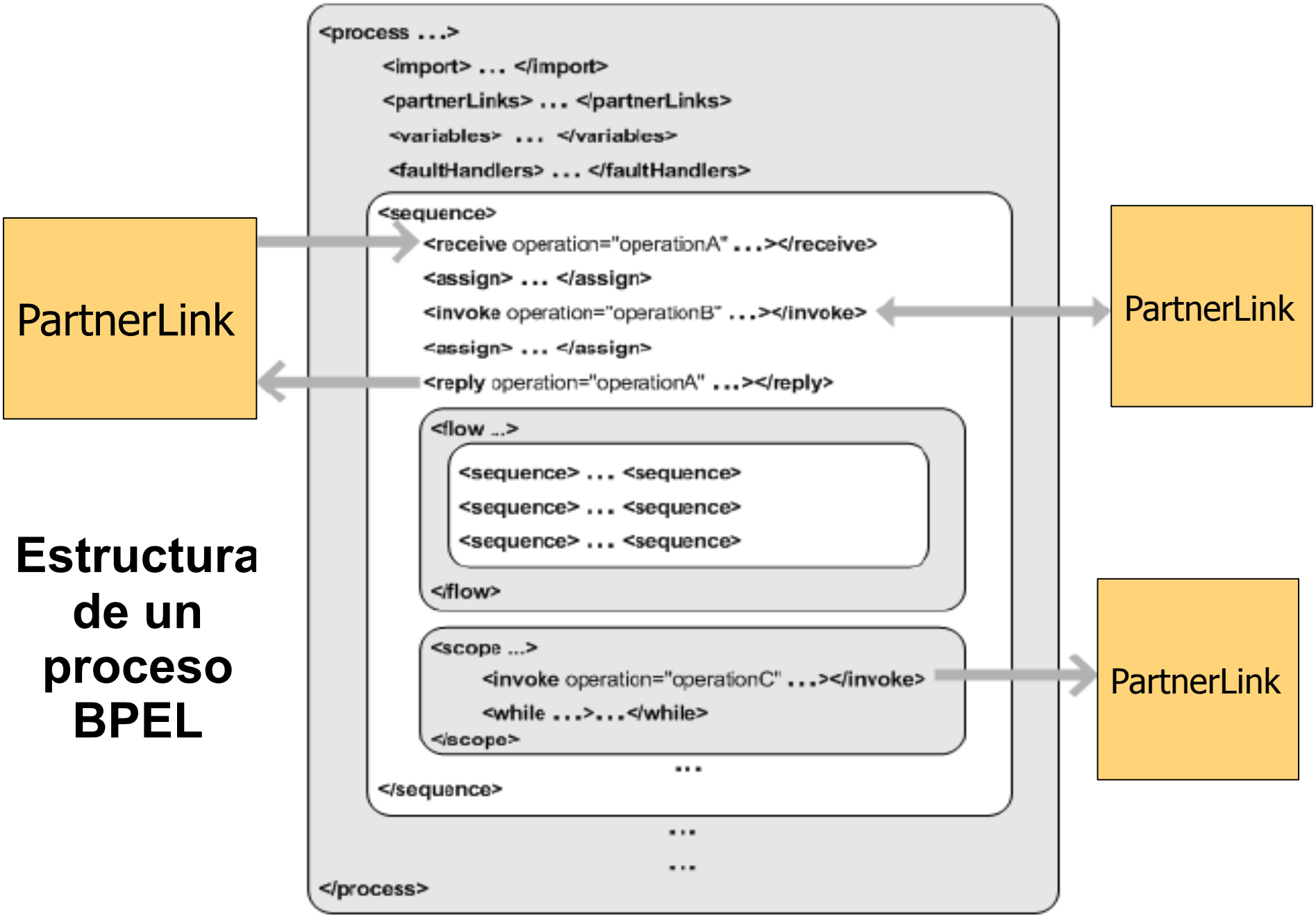


- BPEL puede utilizarse dentro de una empresa y entre empresas

Esquema de un proceso BPEL

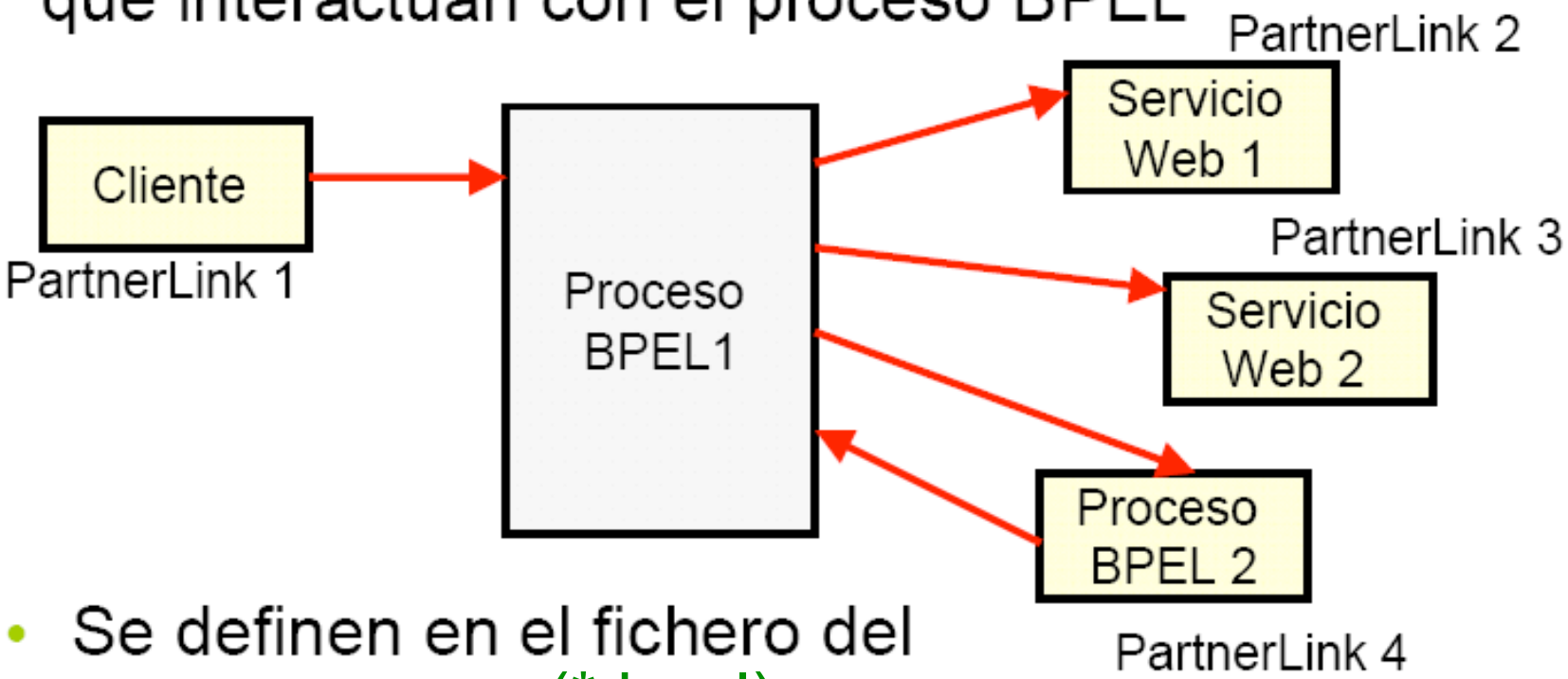


Estructura de un proceso BPEL



Partner Links

- Definen los enlaces con las diferentes "partes" que interactúan con el proceso BPEL



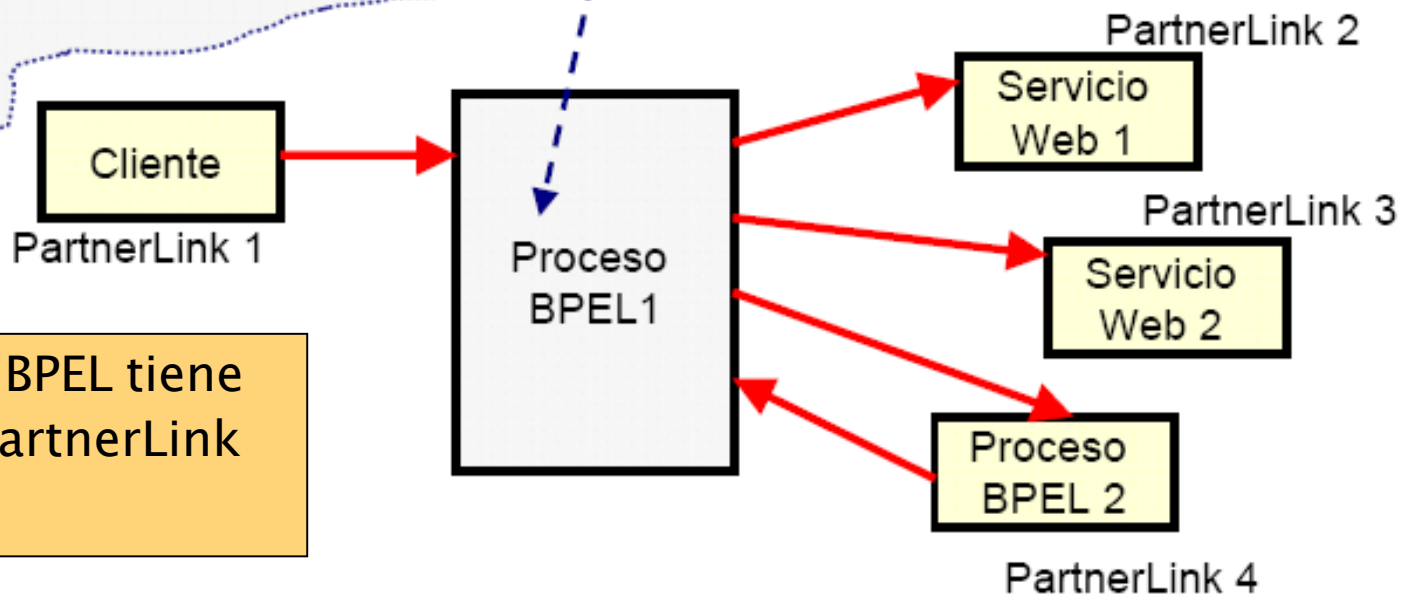
- Se definen en el fichero del proceso BPEL (*.bpel)

Sintaxis definicion de Partner Links

```
<partnerLinks>
    <partnerLink name="nombrePartnerLink1"
                 partnerLinkType="definidoEnWSDLdel Partner"
                 myrole="rolDelProcesoBPEL"
                 partnerRole="rolDelPartner"/>
    <partnerLink name= ... />
    ...
</partnerLinks>
```

Define el tipo de interacción entre dos partners

Los roles se definen en el partnerLinkType

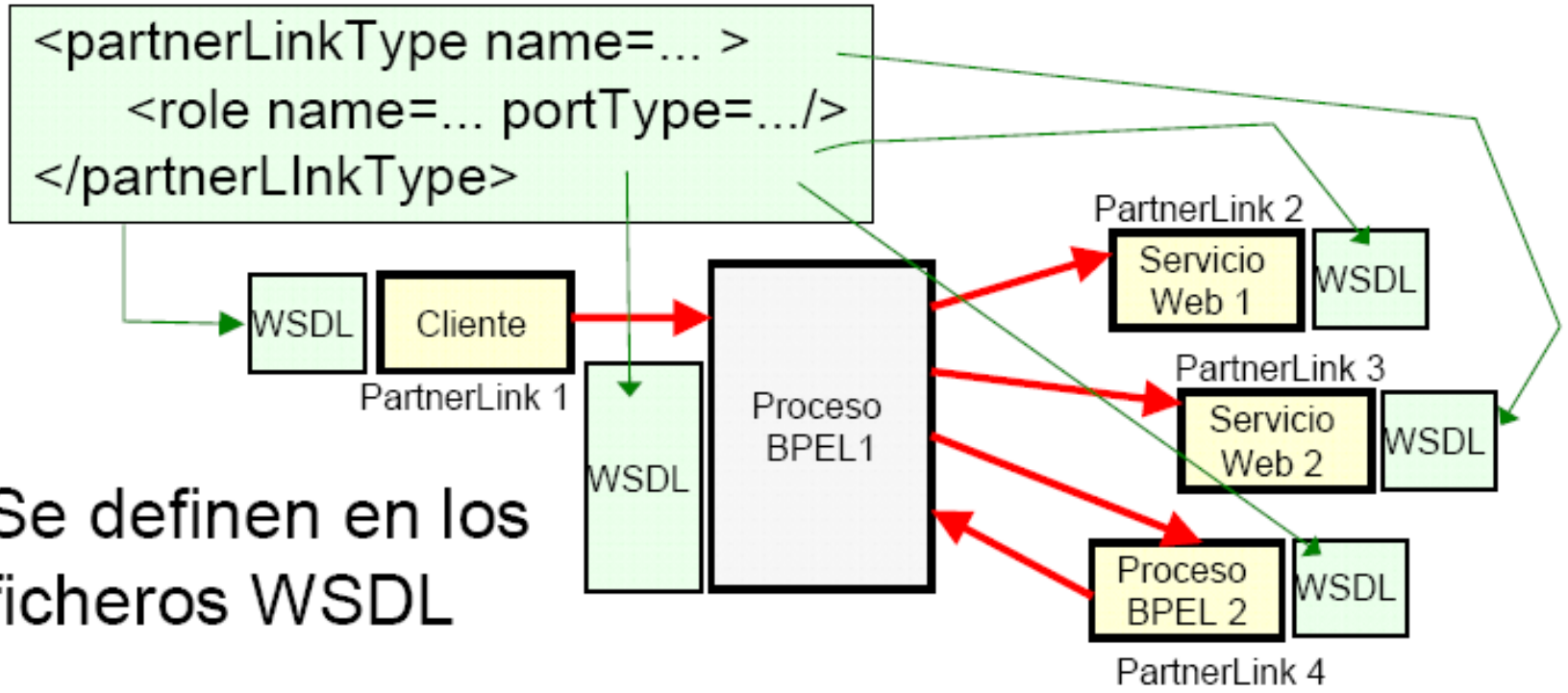


Cada proceso BPEL tiene al menos un PartnerLink cliente

PartnerLinkType

Declara cómo interactúan las dos partes y lo que cada parte ofrece

- Especifica la relación entre dos servicios, definiendo el rol que cada servicio implementa
  - Se pueden especificar un rol o dos
  - <partnerLinkType name=... >
    <role name=... portType=.../>
    </partnerLInkType>
- Se definen en los ficheros WSDL



Ejemplo

- Supongamos un servicio Bpel denominado *saludo.bpel*

```
<!-- Extracto de Saludo.wsdl -->
<partnerLinkType name="MyPartnerLinkType">
   <role name="ProveedorServicioSaludo"
      portType="SaludoPortType"/>
   </role>
</partnerLinkType>
```

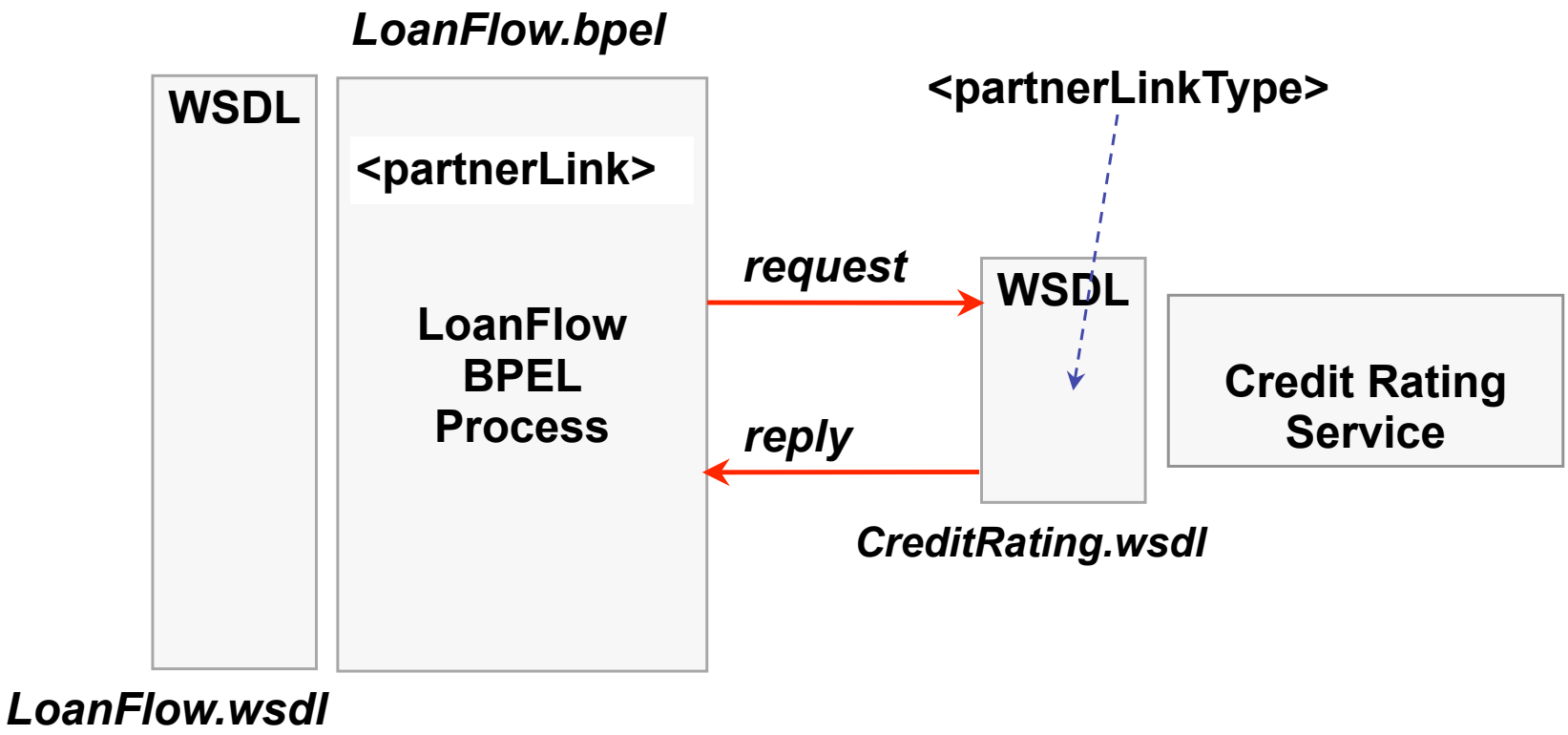
Define la relación entre el proceso bpel y el cliente del proceso bpel

```
<!-- Extracto de Saludo.bpel -->
<partnerLinks>
   <partnerLink name="cliente"
        partnerLinkType="MyPartnerLinkType"
        myRole="ProveedorServicioSaludo"/>
</partnerLinks>
```

El proceso bpel implementa el portType=”SaludoPortType”

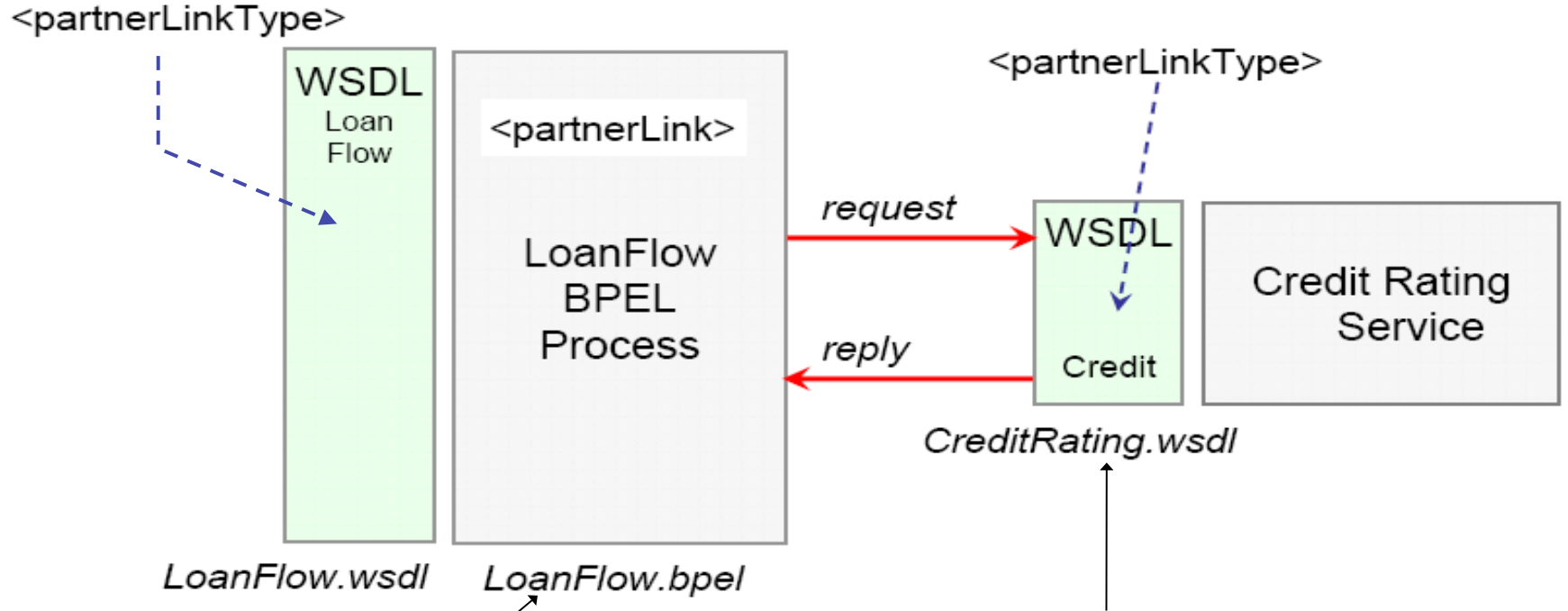
```
Cada partnerLinkType se define en el fichero WSDL:
(a) del proceso BPEL, en el caso de que describa la interacción del cliente
    con el propio proceso BPEL, o
(b) del servicio Web al que invoca dicho proceso BPEL
```

PartnerLink Types y PartnerLinks.Ejemplo



- El proceso *LoanFlow* hace una petición al servicio *Credit Rating*

PartnerLink Types y PartnerLinks.Ejemplo




```
<partnerLinks>
    <partnerLink
        name="creditRatingService"
        partnerLinkType=" CreditService "
        partnerRole="ProveedorDelCredito"/>
</partnerLinks>
```

```
<plnk:partnerLinkType name="CreditService">
    <plnk:role name="ProveedorDelCredito">
        <plnk:portType
            name="tns:CreditRatingService" />
    </plnk:role>
</plnk:partnerLinkType>
```

Variables

- Declaración

Los atributos messageType, type y element son excluyentes

```
<variables>
      <variable name="nombreVar"
                  messageType="qname"
                  type="qname"
                  element="qname" />
</variables>
```

- Asignación

```
<assign>
      <copy>
            <from variable="ncname" part = "ncname"/>
            <to variable="ncname" part = "ncname"/>
      </copy>
</ assign >
```

Actividades (I)

- **Primitivas**: representan construcciones básicas
  - **<receive>**: bloquea al proceso que la invoca
  - **<reply>**: devueve una respuesta
  - **<invoque>**: invoca a un servicio Web
  - **<assign>**: asigna un valor a una variable
  - **<wait>**: suspende al proceso un cierto tiempo
  - **<throw>**: para indicar fallos y excepciones

Actividades (II)

- **Estructuradas**: permiten combinar las actividades primitivas
  - **<sequence>**: las actividades se invoca en forma de secuencia ordenada
  - **<flow>**: las actividades se ejecutarán en paralelo
  - **<if>**: las actividades se ejecutan en función de una condición
  - **<while>**: definición de bucles
  - **<pick>**: hace que el proceso espere la llegada de algún evento y en función de él elija un camino de entre varios alternativos

Pasos para crear un proceso de negocio con BPEL

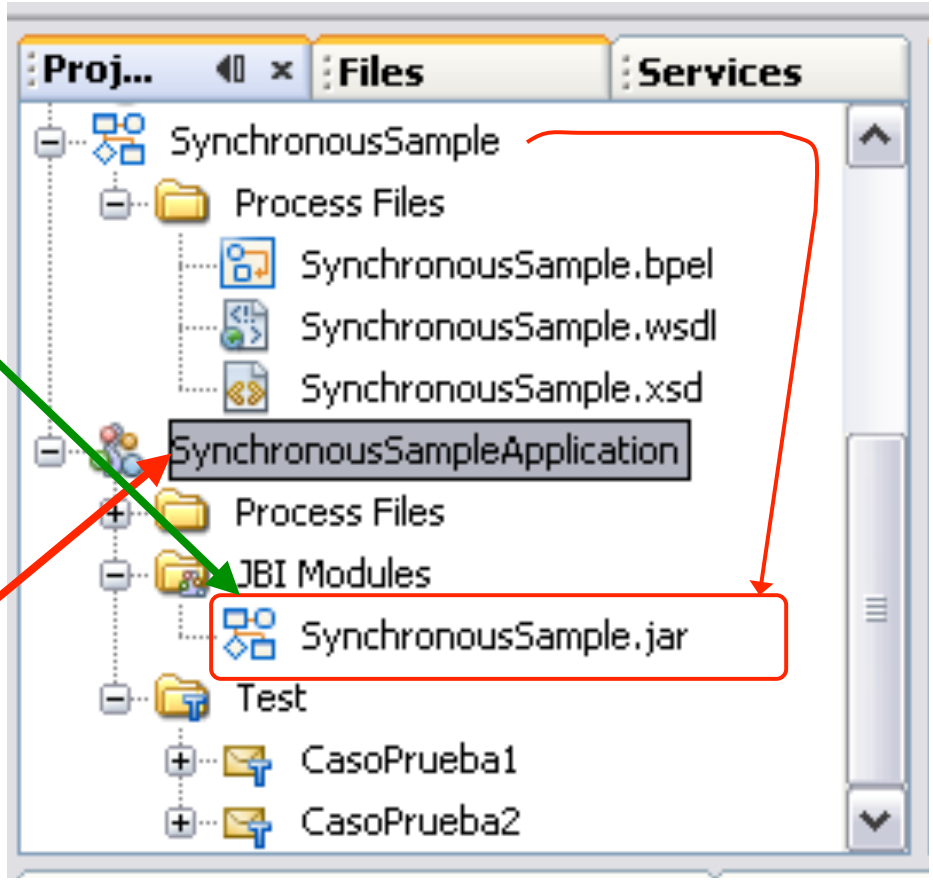
- Conocer los servicios Web implicados
  - Familiarizarnos con los *port types* de los *Web partners*
- Definir el WSDL del proceso BPEL
  - Definir el *partner link type* del cliente
- Desarrollar el proceso BPEL
  - Definir los *partner links*
  - Declarar las variables
  - Escribir la definición de la lógica del proceso

Pasos para crear un proceso de negocio con BPEL y *Netbeans*

- Crear un proyecto BPEL
  - Proceso BPEL (*.bpel)
  - Fichero WSDL (*.wsdl)
  - Fichero de esquema (opcional) (*.xsd)
- Crear un proyecto *Composite Application*
- Añadir el módulo BPEL como un módulo JBI en la *Composite Application*
- Arrancar el servidor de aplicaciones
- Desplegar el proyecto *Composite Application* en la BPEL *service engine*
- Crear y ejecutar los casos de prueba

Composite Application Project

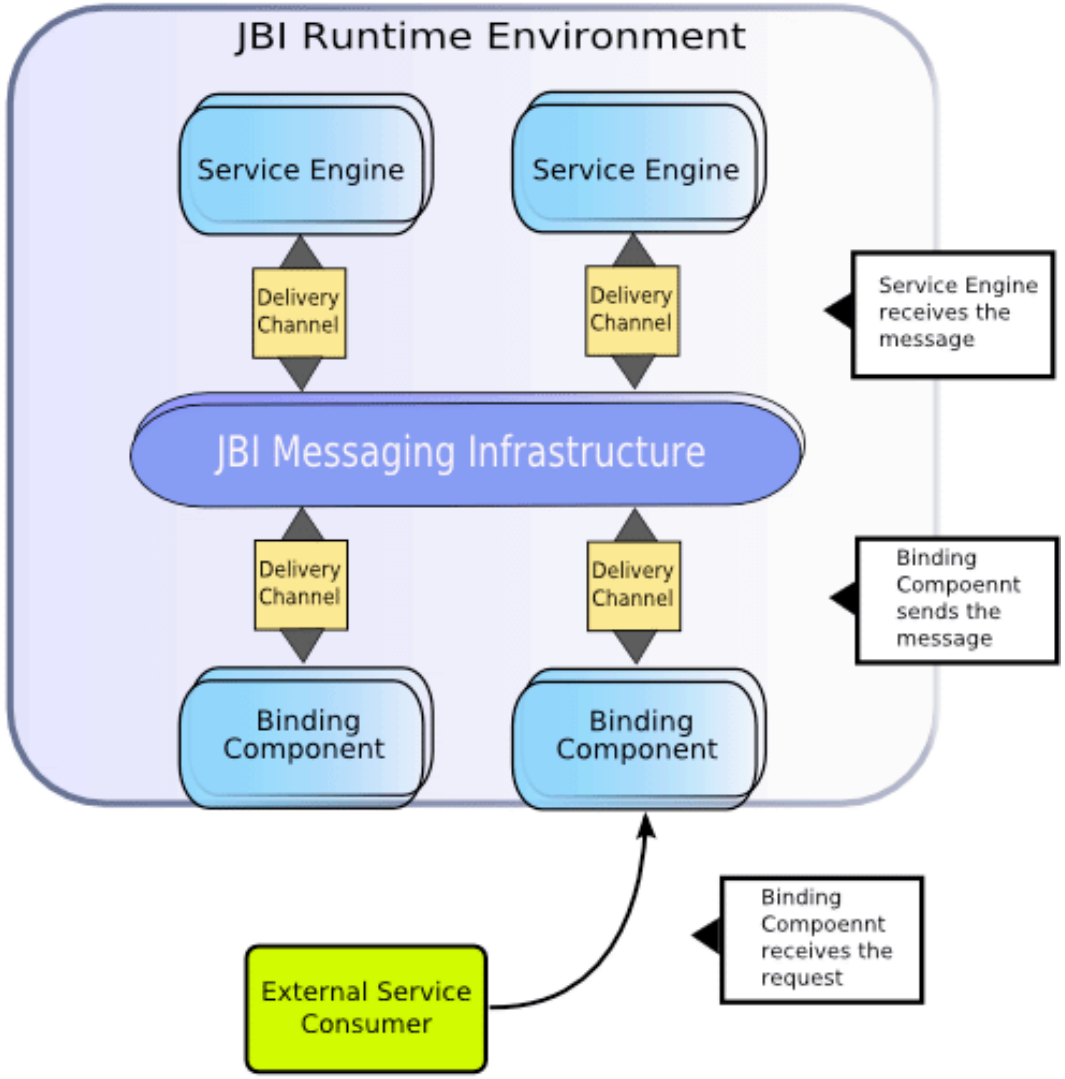
- Se utiliza para crear un ensamblado de servicios (***Service Assembly***) que puede desplegarse en el servidor de aplicaciones como un componente JBI.
- Un proyecto BPEL no es directamente desplegable. Primero debemos añadir dicho proyecto BPEL, como un módulo JBI, en un proyecto ***Composite Application***.



- A continuación podremos desplegar el proyecto *Composite Application* en la máquina de servicios BPEL.

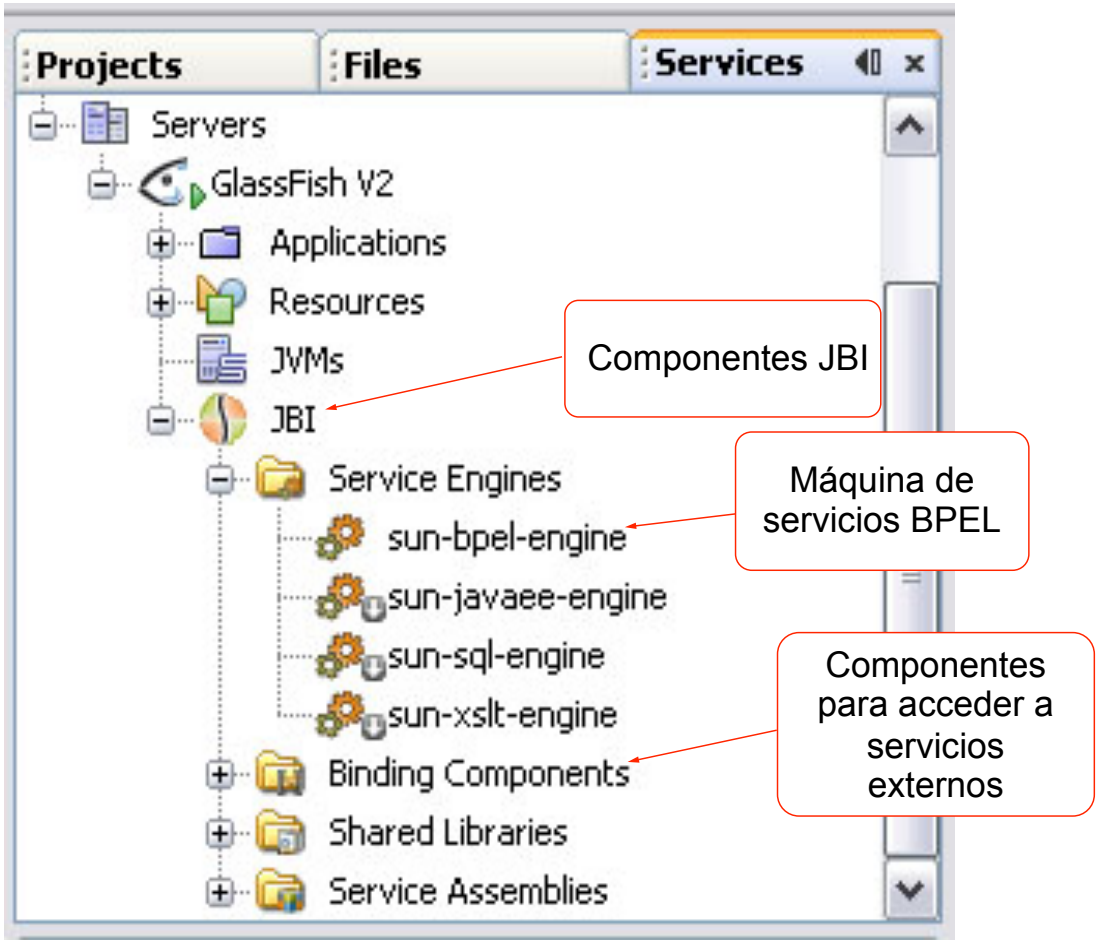
Entorno de ejecución JBI

- Los componentes que suministran o consumen servicios dentro del entorno JBI son referenciados como máquinas de servicios (***Service Engines***)



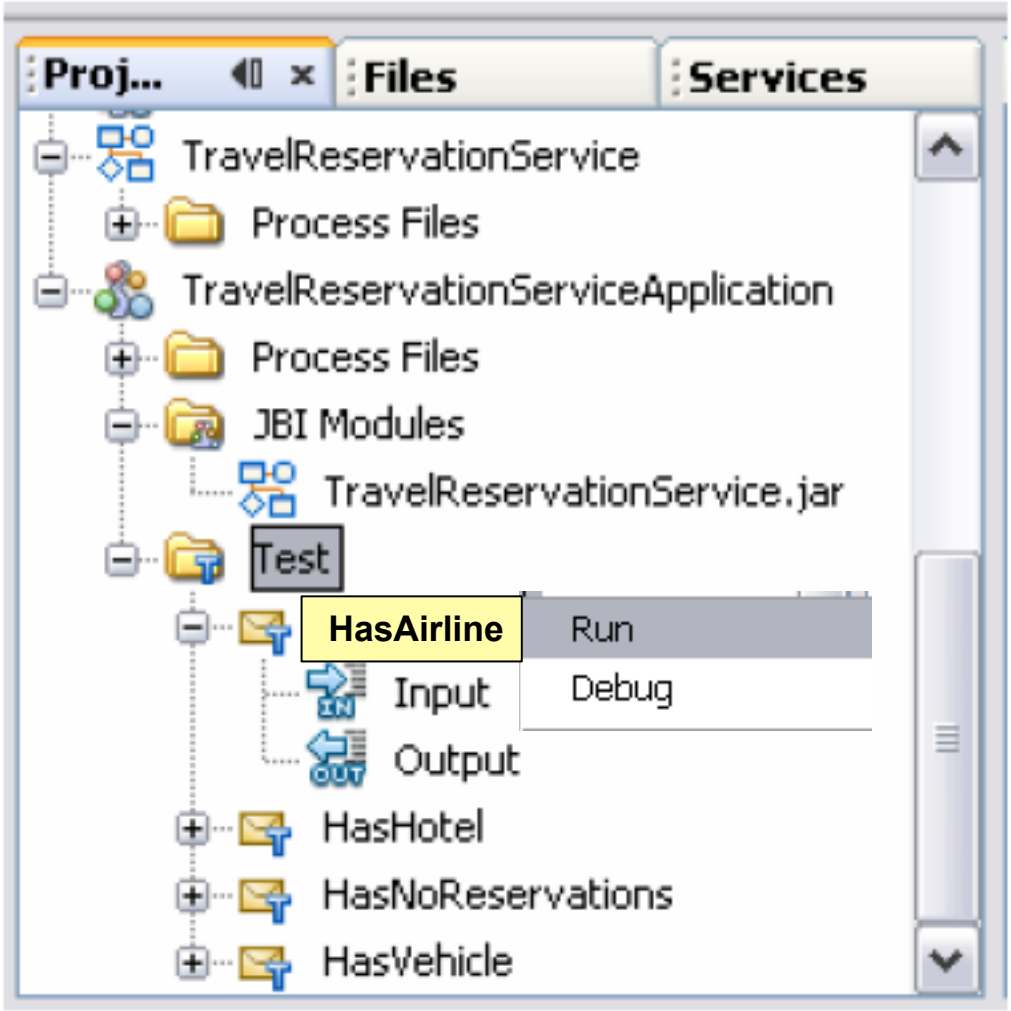
Máquina de servicios BPEL

- Es un componente JBI que proporciona servicios para ejecutar procesos de negocio desarrollados con WS-BPEL 2.0.
- La máquina de servicios BPEL arranca juntamente con el servidor de aplicaciones



Creación y ejecución de Pruebas

- Añadimos un caso de prueba y lo enlazamos con una operación BPEL
- Determinamos las propiedades de la prueba
- Modificamos las entradas de las pruebas: fichero ***Input.xml***
- Ejecutamos la prueba: ésta queda registrada en el fichero ***Output.xml***



¿Preguntas...?