Sistemas Distribuidos

Grado en Ingeniería Informática

Arquitecturas Orientadas a Servicios

Departamento de Ingeniería Informática
Universidad de Cádiz

Curso 2013 – 2014

Indice

Sección 1 | Introducción

Arquitecturas Orientadas a Servicios (I)

Servicio

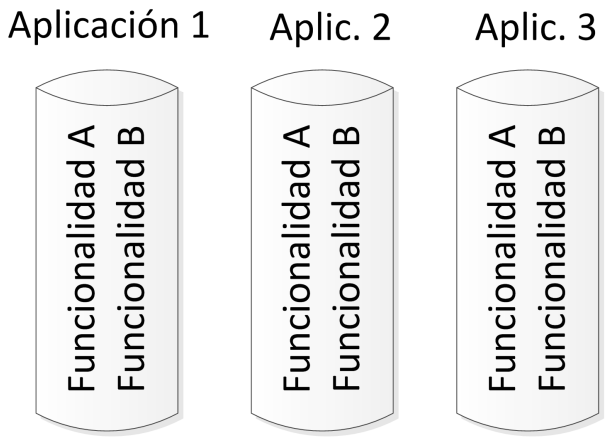
- Un “contrato” (prestaciones al usuario).
- “Funcionalidad concreta” que puede descubrirse y que describe qué puede hacer y cómo interactuar con él.

Arquitectura Orientada a Servicios o *Service-Oriented Architecture* (SOA)

- Arquitectura software que define un modelo desacoplado de los servicios para soportar los requisitos de los procesos de negocio.
- Proporcionan “funciones” que pueden ser reutilizables por distintos clientes (sólo necesitan conocer la descripción del servicio).

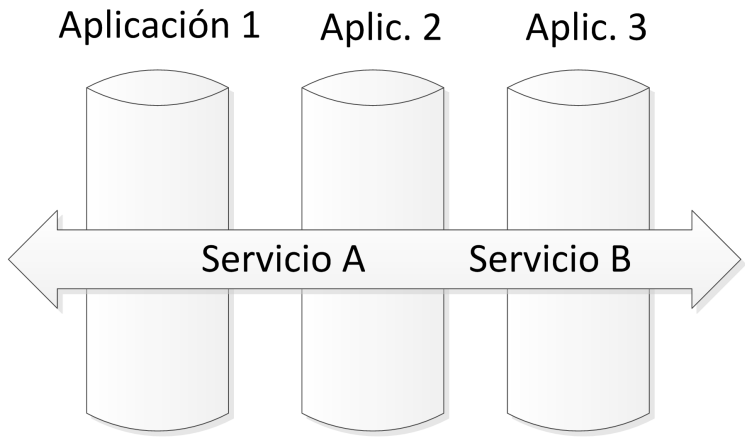
Arquitecturas Orientadas a Servicios (II)

El enfoque silo



Funcionalidades comunes duplicadas

El enfoque SOA



Funcionalidades comunes compartidas a través de servicios

Servicios Web (I)

Definición (W3C, 2004)

*Un **sistema software** diseñado para ofrecer **interacción máquina-a-máquina** sobre una red. Tiene una **interfaz** descrita en un formato procesable por la máquina: **WSDL**. Otros sistemas interactúan con el servicio Web en una forma prescrita por su descripción utilizando **mensajes SOAP**, normalmente transmitidos usando HTTP con una serialización XML y con otros estándares Web relacionados.*

- El uso de WS implica:
  1. Publicar los WS: estándares WSDL (*Web Services Description Language*), XML (*eXtensible Markup Language*) y SOAP (*Simple Object Access Protocol*).
  2. Componer los WS:
     - Orquestación: estándar WS-BPEL (*Business Process Execution Language*).
     - Coreografía: WS-CDL (*Web Services Choreography Description Language*).

Servicios Web (II)

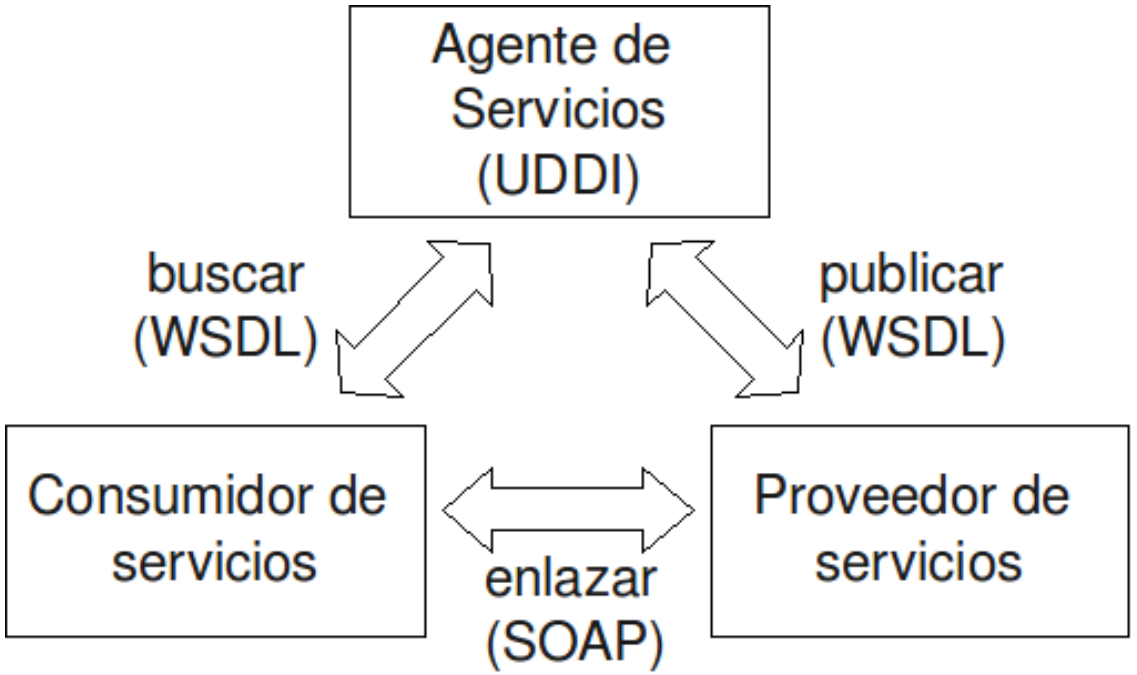
WSDL

- Lenguaje basado en XML utilizado para describir la interfaz de SW.
- Describe qué pueden hacer, dónde se encuentran, qué tipo de datos esperan y en qué formato.
- Sólo se describe el SW, no su implementación $\Longrightarrow$ reduce los problemas de compatibilidad entre SW.

SOAP

- Protocolo simple y extensible basado en XML, estandarizado por el W3C.
- Utilizado para el intercambio de mensajes en un entorno distribuido.

Servicios Web (III)



Orquestación vs Coreografía (I)

Orquestación

- Existe un control centralizado que dirige las actividades de los distintos WS participantes.
- Cada WS recibe el pedido de ejecución de un servicio controlador (podrá devolver una respuesta del trabajo realizado).
- Un orquestador es un tipo de servicio particular que coordina los servicios mediante invocaciones a los mismos.

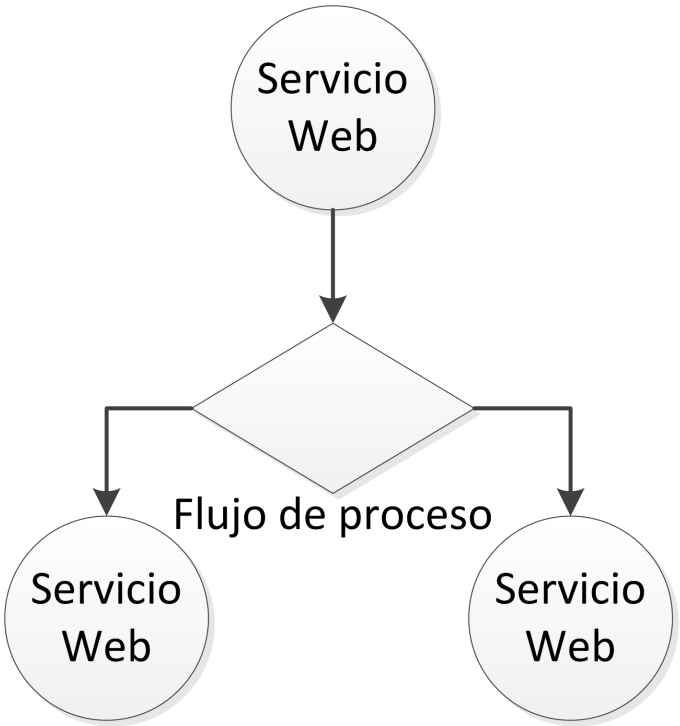
Coreografía

- No posee un coordinador central.
- Los WS participantes se organizan para llevar a cabo la tarea.
- Perspectiva neutral para cualquier participante.
- La ejecución y el control central son responsabilidad de los WS participantes.

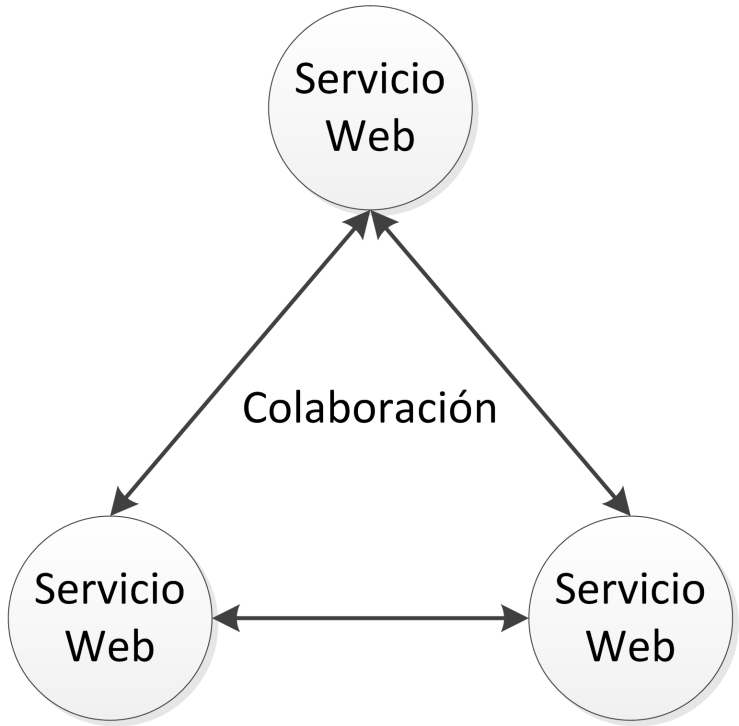
Orquestación vs Coreografía (II)

Orquestación	Coreografía
Más restrictiva	Menos restrictiva
”Semáforo”	”Rotonda”
WS-BPEL	WS-CDL
Ejecución de proceso de negocio	Colaboración entre actores
Visión detallada	Visión más abstracta
Flujo de control (proceso)	Varios flujos de control
Reglas de unión y trabajo colaborativo	Interoperabilidad
Composición de nuevos servicios	
Posee director (motor WS-BPEL)	

Orquestación vs Coreografía (III)



Orquestación



Coreografía

Sección 2 | Características de WS-BPEL

Características de WS-BPEL (I)

- Sintaxis estándar definida en XML.
- Puede mantener temporalmente estructura de datos en variables.
- Las actividades básicas incluyen la invocación de un servicio remoto, recepción de mensajes, y asignación de estructuras XML a variables.
- Proporciona constructores condicionales y de time-out.
- Proporciona condiciones de excepción.
- Ofrece coordinación entre las partes para determinar si han finalizado con éxito o no.
- Define un modelo y una gramática para describir el comportamiento del proceso de negocios basado en la interacción entre un proceso y sus partners.
- La interacción de un proceso con cada uno de sus partners se realiza a través de interfaces de Servicios Web (partner links).

Características de WS-BPEL (II)

- Define la manera de coordinar interacciones de servicios múltiples con estos partners.
- Posee un mecanismo de concurrencia.
- Ofrece exclusión mutua cuando se comparten variables que son accedidas por flujos concurrentes.

Sección 3 | Relación de WS-BPEL con WSDL

Relación de WS-BPEL con WSDL (I)

- Un proceso y sus *partners* se modelan como servicios web (relación *peer-to-peer*).
- La definición de un proceso WS-BPEL provee/usa uno o más servicios web.
- Un documento WSDL define servicios como colecciones de *endpoints* de redes.
- La definición abstracta de los *endpoints* y mensajes se separa de la red concreta o del enlace del formato de datos, permitiendo la reutilización de las definiciones abstractas:
  - Mensajes: descripciones abstractas de los datos a intercambiar.
  - Tipos de puertos: colecciones abstractas de operaciones.

Relación de WS-BPEL con WSDL (II)

- El protocolo concreto y las especificaciones del formato de datos para un tipo de puerto particular constituye un enlace reutilizable.
- Un puerto se define por la asociación de la dirección de la red con un enlace reutilizable.
- Una colección de puertos define un servicio.

Relación de WS-BPEL con WSDL (III)

- Un documento WSDL utiliza los siguientes elementos para definir un servicio de red:
  - types: ofrecen definiciones de tipos de datos usados para describir los mensajes intercambiados.
  - message: representa una definición abstracta de los datos que se transmiten. Un message está compuesto de partes lógicas, cada una asociada con una definición en algún sistema de tipos.
  - portType: conjunto de operaciones abstractas. Cada operación hace referencia a un mensaje de entrada y a mensajes de salida.
  - binding: especifica un protocolo concreto y especificaciones de formato de datos para las operaciones y mensajes definidos por una portType particular.
  - port: especifica una dirección para un binding, que define un endpoint de comunicación simple.
  - service: se usa para agrupar un conjunto de ports relacionados.

Relación de WS-BPEL con WSDL (IV)

- Los procesos WS-BPEL usan definiciones WSDL.
- Los ficheros WSDL contienen el namespace, partner link types, port types, operations y messages necesarios para definir las actividades de proceso.
- Los ficheros WSDL son necesarios para crear una definición en WS-BPEL que sea válida y ejecutable.
- Cada interacción entre servicios puede ser vista como una comunicación entre partners.
- Esta interacción se describe con la ayuda de los partner links, que son instancias de conectores que especifican en los port types los procesos ofrecidos y requeridos de un partner a otro.

Sección 4 | Descripción de WS-BPEL 2.0

Descripción de WS-BPEL 2.0 (I)

Estructura (resumen) de un proceso WS-BPEL

1. Declaración de relaciones con los agentes externos (*partners*).
2. Declaración de variables.
3. Declaración de manejadores.
4. Descripción del comportamiento del proceso.

WS-BPEL 2.0 (II)

Ejemplo práctica 1 (hola mundo)

```
<bpel:process name="HelloWorld" <!-- ... -> >

    <bpel:partnerLinks> ← 1. Declaración de partners
        <bpel:partnerLink name="client"
                          partnerLinkType="tns:HelloWorld"
                          myRole="HelloWorldProvider"
                          />
    </bpel:partnerLinks>
    <bpel:variables> ← 2. Declaración de variables
        <bpel:variable name="input"
                       messageType="tns:HelloWorldRequestMessage"/>
        <bpel:variable name="output"
                       messageType="tns:HelloWorldResponseMessage"/>
    </bpel:variables>
    <bpel:sequence name="main"> ← 4. Comportamiento del proceso
        <!--  ...  -->
```

WS-BPEL 2.0 (III)

Elementos

- Todo se estructura en actividades: básicas y estructuradas.
- Las actividades pueden, a su vez, tener tanto atributos como contenedores.

WS-BPEL 2.0 (IV)

Ejemplo

```
<flow>← Actividad estructurada
  <links>← Contenedor
    <link name="comprobarVuelo-A-reservarVuelo"← Atributo />← Elemento
  </links>
  <invoke name="comprobarVuelo" ... >← Actividad básica
    <sources>← Contenedor
      <source linkName="comprobarVuelo-A-reservarVuelo"← Atributo />← Elemento
    </sources>
  </invoke>
  <invoke name="comprobarHotel" ... />
  <invoke name="comprobarAlquilerCoche" ... />
  <invoke name="reservarVuelo" ...>
    <targets>← Contenedor
      <target linkName="comprobarVuelo-A-reservarVuelo" />← Elemento
    </targets>
  </invoke>
</flow>
```

Sección 5 | Estructura de un proceso WS-BPEL 2.0

Estructura de un proceso (I)

- WS-BPEL es un lenguaje basado en XML.
- Su espacio de nombres es: `http://docs.oasis-open.org/wsbpel/2.0/process/executable`.
- El elemento raíz es `<process>` y debe definirse, como mínimo, el atributo `name` (el nombre del proceso).
- Adicionalmente, pueden especificarse los atributos:

  queryLanguage Especifica el lenguaje de consulta que utiliza el proceso para la selección de nodos en las asignaciones. Normalmente, XPath.

  expressionLanguage Determina el lenguaje de las expresiones usadas en el elemento `<process>`. Normalmente, XPath.

  suppressJoinFailure Especifica si el error `joinFailure` será suprimido para todas las actividades. El valor por defecto es “no”. Puede sobrecargarse por cada actividad.

Estructura de un proceso (II)

exitOnStandardFault Si el valor de este atributo es “yes”, entonces el proceso deberá terminar inmediatamente (como si se alcanzara una actividad `<exit>`), cuando ocurra un fallo estándar distinto de `bpel:joinFailure`. Si el valor es “no” (por defecto), entonces el proceso podrá manejar un fallo estándar usando un manejador de fallos.

- Podría añadirse un elemento `<extensions>` al proceso dentro del cual se especifiquen tantas `<extension>` como sea necesario. Estos elementos deben tener dos atributos:

  namespace El espacio de nombres de la extensión.

  mustUnderstand Especifica si el motor de ejecución debe entender la extensión, o puede ignorarla.

Estructura de un proceso (III)

- Además de las extensiones, un proceso puede contener los siguientes elementos en su primer nivel:

  `<import>` Importación de otros ficheros: declaraciones WSDL, XSD...

  `<partnerLinks>` Declaración de los enlaces a otros servicios (agentes externos).

  `<variables>` Declaración de variables globales.

  `<correlationSets>` Vincula un mensaje con una instancia concreta del proceso.

  `<faultHandlers>` Manejadores de fallos.

  `<eventHandlers>` Manejadores de eventos.

  Actividad La actividad principal del proceso.

Actividades básicas (I)

`<assign>` Se usa para asignar valores a las variables.

`<compensate>` Se usa sólo dentro de un manejador de fallos, de compensación o de terminación para comenzar la compensación.

`<compensateScope>` Se usa sólo dentro de un manejador de fallos, de compensación o de terminación para comenzar la compensación de un actividad `<scope>` específica.

`<empty>` No hace nada, pero puede utilizarse para sincronizar.

`<exit>` Provoca la salida inmediata del proceso.

`<invoke>` Invoca un servicio Web.

`<receive>` Espera la llegada de un mensaje en una operación específica.

`<reply>` Devuelve un mensaje en respuesta a un mensaje recibido previamente.

Actividades básicas (II)

`<rethrow>`	Se usa sólo dentro de un manejador de fallos para relanzar el fallo.
`<throw>`	Lanza un fallo.
`<validate>`	Valida una o más variables.
`<wait>`	Provoca la parada del proceso durante un periodo de tiempo específico o hasta que se alcanza una fecha límite.

Actividades estructuradas (I)

`<sequence>` Contiene una o más actividades que son ejecutadas secuencialmente, en el orden léxico en el que aparecen dentro de esta actividad.

`<flow>` Proporciona concurrencia mediante la ejecución de todas sus actividades hijas en paralelo. También introduce enlaces de sincronización para hacer que una actividad espere por otra, o salte su ejecución por completo.

`<scope>` Permite la declaración de los recursos locales, como las variables y los agentes externos, y una actividad principal a ejecutar en este entorno local. También incluye manejadores de compensación, de fallos, de terminación y de eventos.

`<pick>` Permite elegir uno de varios eventos (mensajes entrantes o alarmas temporizadas) y continuar por la ruta de ejecución seleccionada.

Actividades estructuradas (II)

`<if>` Proporciona un comportamiento condicional. Se toma la primera rama cuya `<condition>` se evalúa a *true*, y se ejecuta la actividad que contiene. En caso contrario, se tomará la rama `<else>` si existe.

`<forEach>` Es un constructor de repetición especial para ejecutar múltiples instancias de la misma actividad `<scope>`.

`<while>` Proporciona la ejecución repetida de una actividad contenida. La actividad contenida se ejecuta mientras que la `<condition>` booleana se evalúa a *true* al comienzo de cada iteración.

`<repeatUntil>` Proporciona la ejecución repetida de una actividad contenida. La actividad contenida se ejecuta hasta que la `<condition>` booleana se evalúe a *true*.

Bibliografía (I)

M.P. Papazoglou
Web Services & SOA: Principles and Technology.
Pearson – Prentice Hall, enero 2012.

OASIS.
Web Services Business Process Execution Language 2.0
http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, abril 2007.

W3C.
Extensible Markup Language (XML)
http://www.w3.org/XML/, enero 2012.

W3C.
SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)
http://www.w3.org/TR/soap12-part1/, abril 2007.

Bibliografía (II)


W3C.
Web Services Choreography Description Language Version 1.0
http://www.w3.org/TR/ws-cdl-10/, noviembre 2005.


W3C.
Web Services Description Language (WSDL) 1.1
http://www.w3.org/TR/wsdl, marzo 2001.