Composición de servicios

Composición estática

Javier Béjar

ECSDI - 2021/2022 2Q

CS-GEI-FIB

Introducción

- El número de servicios web disponibles en internet crece cada día
  - www.programmableweb.com tiene ∼20.000 WS APIs
- El desarrollo de aplicaciones en entornos abiertos permite aprovechar otros servicios accesibles
- Podemos desarrollar nuevas aplicaciones componiendo servicios
- El esfuerzo de desarrollo de las aplicaciones se ve reducido
- La filosofía de análisis y diseño de aplicaciones debe adaptarse

- El sentido de la programación basada en servicios (o agentes) es poder usarlos como componentes
- Cada servicio individual realiza una (o varias) tarea concreta
- La unión de servicios siguiendo diferentes flujos permite implementar diferentes soluciones
- **Composición:** unión de servicios (o agentes) para obtener una solución/aplicación

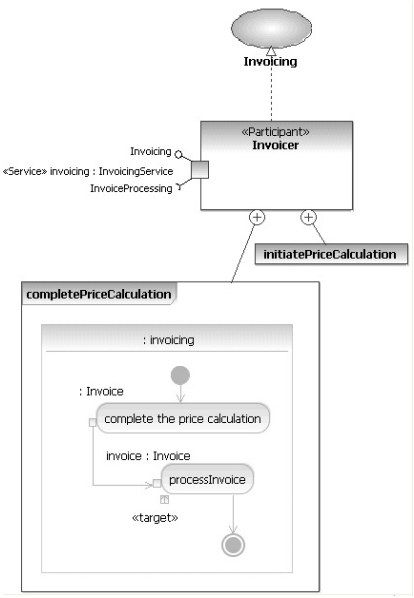
Descripción de Servicios

- Diferentes tecnologías de composición de servicios
  - Ligadas a arquitecturas SOA (eg. SCA)
  - Como lenguajes de modelado para el diseño (SOAML)
  - Como lenguajes de documentacion del APIs web/Microservicios (Swagger/OpenAPI/API Blue-print)
- Están en diferentes procesos de estandarización y maduración
- Están apoyadas por diferentes compañías
- Están dirigidas a diferentes nichos de aplicación

- La unidad es el componente
- Un componente
  - es accesible mediante una interfaz
  - puede referenciar a otros componentes
  - tiene elementos configurables que modifican su comportamiento
  - puede estar formado a su vez por otros componentes
- El flujo del proceso a componer dicta como se han de conectar

- Desacopla la composición de los detalles de invocación
- Es agnóstico al lenguaje/tecnología (C++, java, XML, BPEL)
- Puede utilizar diferentes mecanismos de comunicación
- Puede integrar elementos basados en otras tecnologías (RMI, RPC, CORBA...)
- Varias implementaciones Open Source y propietarias (IBM, Oracle, Red Hat)
- Estándar OASIS (open SCA)

- Service Oriented Architecture Modelling Language (SOAML) es un perfil UML y un metamodelo para la definición de arquitecturas orientadas a servicios
- Extiende UML para poder definir todos los elementos que necesita una arquitectura orientada a servicios



- Como identificar servicios (qué proveen, dependencias)
- Como especificar servicios (capacidades funcionales, protocolos, intercambio de información)
- Definición de proveedores y consumidores
- Políticas de uso y provisión de servicios
- Requerimientos de los servicios y su uso
- Esquemas de clasificación de servicios/organización/restricciones
- Enlace con otros metamodelos UML

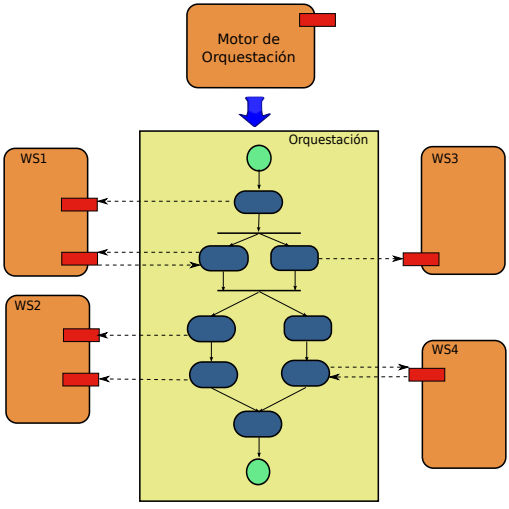
- El elemento principal es el participante
- Los participantes pueden ser proveedores o consumidores
- Los servicios se definen a través de puertos
- Estos definen puntos de provisión y consumo de servicios
- El acceso a los servicios se define a partir de:
  - Una interfaz UML (unidireccional)
  - Una interfaz de servicio (bidireccional)
  - Un contrato de servicio (bidireccional)
- Fases iniciales de implantación (primera versión 2012)

Estrategias de composición

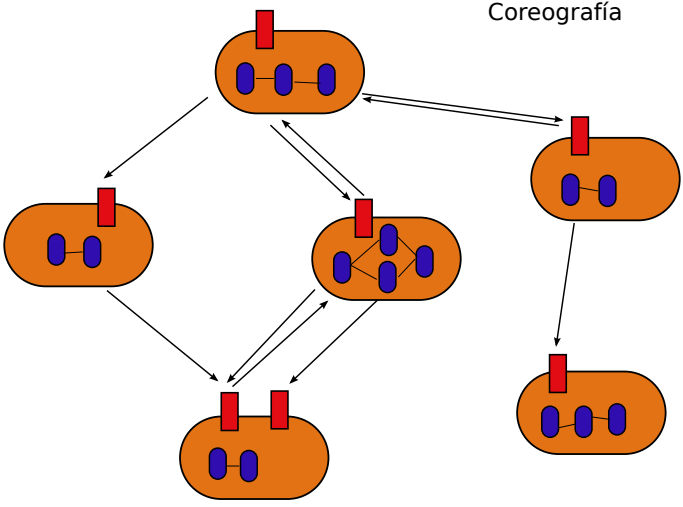
- Estas tecnologías permiten definir cómo son los componentes y cómo se organizarán
- Permiten definir el proceso de negocio especificando:
  - El orden potencial de ejecución de la composición
  - Los datos compartidos por los servicios
  - Qué servicios estarán involucrados y cómo (interacciones/relación)
  - Cómo se tratarán las excepciones (quién/cómo)

- Existen diferentes posibilidades de establecer como la composición se implementará al ejecutarla
  - Para el flujo de ejecución:
    - Prefijado (estático)
    - Generado automáticamente a partir de la tarea a resolver (dinámico)
  - Para los servicios involucrados
    - Servicios prefijados (estáticos)
    - Servicios provistos por un servicio de directorio (coincidencia sintáctica/semántica) (dinámicos)

- Supone que un ente organizador ejecutará la composición (motor de orquestación)
- Define el flujo desde la perspectiva de un único integrante (orquestador)
- Una orquestación define:
  - La lógica de ejecución
  - El orden de las interacciones
  - Cómo los componentes interactuarán al nivel de mensaje


Motor de Orquestación
Orquestación
WS1
WS2
WS3
WS4

- El control está distribuido entre los participantes
- Define el flujo desde la perspectiva de todas las partes
- Cada participante conoce el comportamiento de los otros (con los que colabora)
- Hay una visión común de los elementos del estado
- La coreografía describe la secuencia de mensajes que intercambian múltiples participantes
- Cada participante describe el rol que tiene en la interacción


Coreografía

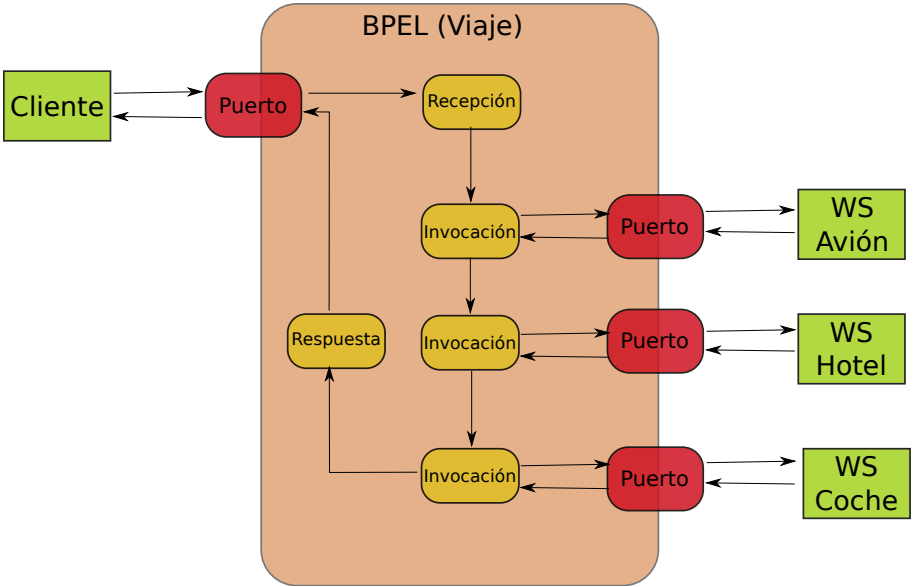
Implementación de composiciones

- La implementación puede ser programática
- Es más ventajoso utilizar **lenguajes de composición**:
  - Independencia del lenguaje de los servicios
  - Adaptación al cambio en el flujo de ejecución del servicio compuesto
- Lenguajes utilizados:
  - WS-BPEL (Web Services Process Execution Language) (OASIS)
  - WS-CDL (Web Services Choreography Description Language) (W3C)

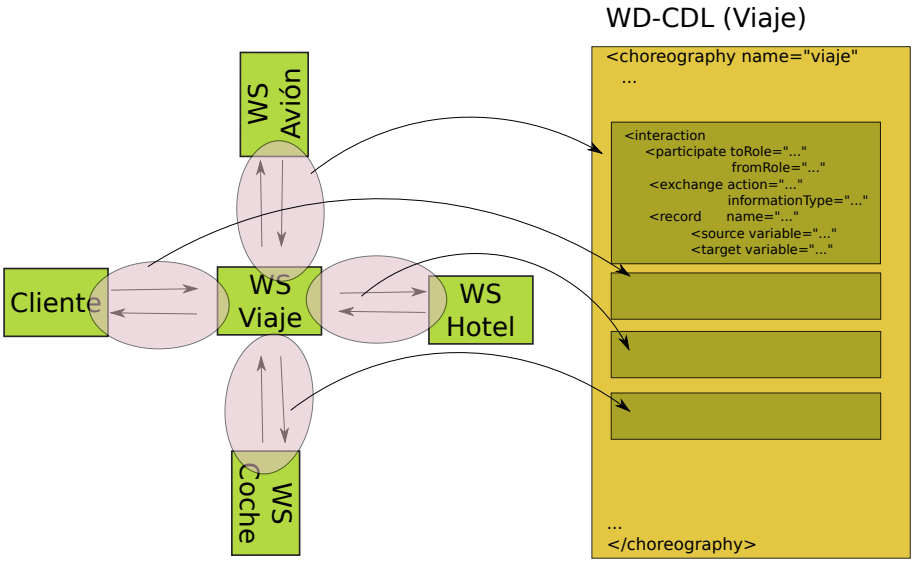
- Descripción mediante WSDL (Web Services Description Language)
- Permite definir:
  - **Procesos ejecutables**: Especificación de los detalles del flujo de ejecución
  - **Procesos abstractos**: Procesos generales que permiten dejar elementos por especificar (a instanciar)
- WS-BPEL define un lenguaje de programación de procesos: invocación, recepción, envío de mensajes, excepciones, secuencias, flujos paralelos, bucles, alternativas...

- Definición del acceso al servicio compuesto desde el exterior mediante WSDL
  - Definición de puertos, operaciones, mensajes...
- El proceso es ejecutado por el motor de orquestación según su definición en WS-BPEL
- Este proceso define
  - cómo y cuándo se ejecutarán los servicios
  - cómo se guardará el estado
  - excepciones, fallos...

1. Una petición inicia el proceso
2. Invocación al servicio de reserva de vuelos (envió/recepción mensaje)
3. Guardar respuesta vuelo
4. Invocación al servicio de reserva de alojamiento (envió/recepción mensaje)
5. Guardar respuesta alojamiento
6. Invocación al servicio de reserva de coche (envió/recepción mensaje)
7. Guardar respuesta coche
8. Generar y enviar mensaje respuesta a la petición


BPEL (Viaje)
Cliente
Puerto
Recepción
Invocación
Puerto
WS Avión
Respuesta
Invocación
Puerto
WS Hotel
Invocación
Puerto
WS Coche

- Se basa en la descripción de los servicios mediante WSDL
- No permite definir procesos ejecutables, es un lenguaje de definición
- Permite definir las interacciones entre los diferentes participantes
  - Cada interacción puede implementarse a partir de mecanismos diferentes (por ejemplo WS-BPEL)
- Permite la interoperabilidad entre servicios al ser independiente de la implementación de las interacciones


WD-CDL (Viaje)
<choreography name="viaje"
...
<interaction
<participate toRole="..."
fromRole="..."
<exchange action="..."
informationType="..."
<record name="..."
<source variable="..."
<target variable="..."
...
</choreography>
WS Avión
Cliente
WS Viaje
WS Hotel
WS Coche

Problemas

- Tienden a obtener composiciones estáticas
- Son sensibles al fallo de los servicios (el fallo de un solo servicio acaba el proceso)
- La complejidad de los procesos los hacen difíciles de diseñar y depurar
- Los servicios son modelados al mismo nivel de abstracción y granularidad independientemente de su complejidad
- Suele haber poco conocimiento del contexto de las interacciones (normas/regulaciones)

- El manejo del flujo de ejecución en entornos abiertos no es trivial
  - No todos los servicios pertenecen a la misma organización
  - No podemos asumir que todas las partes cumplirán su cometido correctamente
  - ¿Debería ser necesario un acuerdo previo entre las partes?
- ¿Tratamiento de servicios esenciales de terceras partes que dejan de funcionar?
- ¿Tratamiento de la evolución de aplicaciones y procesos?

- Los objetivos y procesos de las aplicaciones evolucionan con el tiempo
  - **Diseño**: No podemos prever todas las interacciones
  - **Implantación**: Asumimos una composición dinámica
  - **Gestión**: Cambio/adaptación/evolución deberían ser un elemento natural
- Las tecnologías a usar deberían
  - Permitir adaptarse a un entorno cambiante
  - Permitir la reconfiguración sencilla de los procesos
  - Permitir la adaptación dinámica a las necesidades