Service Oriented Architecture

Isaac Gutiérrez Gómez, Salvador Otón Tortosa

Universidad de Alcalá, Departamento de Ciencias de la Computación,
28871 Alcalá de Henares, Spain
igutierrez09@yahoo.es, salvador.oton@uah.es

Abstract. This work present a study of the art of the architectures orientated to services and the Web Services. They present all the components that form the architecture as well as his organization and his application to solve the development of distributed systems based on services. Also there are studied the principal bases of design who are applied in this type of architectures.

Arquitecturas Orientadas a Servicios

Isaac Gutiérrez Gómez, Salvador Otón Tortosa

Departamento de Ciencias de la Computación
Escuela Técnica Superior de Ingeniería Informática
Universidad de Alcalá
28871 Alcalá de Henares
igutierrez09@yahoo.es, salvador.oton@uah.es

Resumen. En este trabajo se hace un estudio sobre el estado del arte de las arquitecturas orientadas a servicios y los servicios Web. Se presentan todos los componentes que forman la arquitectura así como su organización y su aplicación para resolver el desarrollo de sistemas distribuidos basados en servicios. También se estudian los principales patrones de diseño que se aplican en este tipo de arquitecturas.

1 Introducción

A lo largo de los últimos años y sobre todo con la evolución que ha tenido el negocio a través de Internet, las compañías cada vez necesitan más la posibilidad de integración, y existen multitud de sistemas heterogéneos.

La aparición de los servicios Web [1] permite tener una arquitectura común para exponer la funcionalidad de las distintas aplicaciones permitiendo la integración de distintas aplicaciones y sistemas creando nuevos procesos de negocio que permitan crecer a las plataformas tecnológicas, a los departamentos y por supuesto a las compañías.

En cuanto a los servicios Web es importante tener en cuenta que no son los responsables del comportamiento de un servicio, sino que se centran en como se puede acceder a dicho comportamiento que ofrece un servicio.

Antes de continuar veamos la definición que hace la "World Wide Web Consortium (W3C)" [2] de un servicio Web:

"Es un sistema software identificado por medio de una URI [3], que publica interfaces que son definidas y descritas usando XML [4]. Esta definición de interfaz puede ser descubierta por otros sistemas software. Estos sistemas pueden interactuar con el servicio Web de la manera prescrita en la definición, usando XML, utilizando mensajes cubiertos por los protocolos de Internet."

Otro concepto que tenemos que tener claro es el de patrón. Un patrón es simplemente un mecanismo para resolver y tener documentado un problema que ocurre múltiples veces en un determinado contexto. La utilización de patrones permite resolver el problema que dice el patrón a la hora de construir software.

2 Objetivos de una Arquitectura Orientada a Servicio (SOA)

Vamos a contemplar dos objetivos diferentes, desde el punto de vista empresarial y desde el punto de vista de la tecnología.

2.1 Desde el punto de vista empresarial

Cuando una empresa decide hacer uso de una arquitectura SOA [5] es porque tiene objetivos específicos de negocio que cubrir, reducir costes, aumentar ingresos, mejorar la productividad, comunicación inter-empresarial con varias empresas y ajustar los sistemas a los requerimientos del negocio.

También se puede decir que una arquitectura SOA consiste en una forma de modularizar los sistemas y aplicaciones en componentes de negocio que pueden combinarse y recombinarse con interfaces bien definidas para responder a las necesidades de la empresa.

Con el uso de entornos orientados a servicios las empresas pretenden mejorar la interacción con los clientes, partners, proveedores, empleados y también reducir el ROI (Return of Investment) retorno de la inversión, es decir, conseguir una mayor rentabilidad de las inversiones tecnológicas.

Para las empresas se abre un abanico amplio de aplicación, desde la utilización en la cadena de suministro, entornos B2B o servicios de seguridad.

2.1.1 Beneficios para el negocio

- Eficiencia. Transforma los procesos de negocio en servicios compartidos con un menor coste de mantenimiento.
- Capacidad de respuesta. Rápida adaptación y despliegue de servicios, clave para responder a las demandas de clientes, partners y empleados.
- Adaptabilidad. Facilita la adopción de cambios añadiendo flexibilidad y reduciendo el esfuerzo.

2.2. Desde el punto de vista tecnológico

Las arquitecturas SOA pretenden concebir las aplicaciones desde otro punto de vista, una aplicación orientada a servicios combina datos en tiempo real con otros sistemas capaces de fusionar los procesos de negocio.

Las aplicaciones basadas en SOA utilizan tecnología totalmente estándar como es XML y servicios Web para la mensajería. Estándares como SOAP [6], Web Services Description Language (WSDL) [7] y Business Process Execution Language (BPEL)

[8], estandarizan así la compartición de información, el modelo de integración de procesos y la cooperación entre aplicaciones.

Realizando aplicaciones orientadas a servicio se pueden conectar aplicaciones heterogéneas con el aumento de flexibilidad que supone, y un punto muy importante es que permite que las organizaciones interactúen cuando realmente lo requieran, sin necesidad de tener conexiones permanentes.

Como una arquitectura SOA se basa en estándares, el tiempo de aprendizaje de utilización de las tecnologías sobre las que se apoya se reduce drásticamente.

2.2.2. Beneficios Tecnológicos

- Reduce la complejidad gracias a la compatibilidad basada en estándares frente a la integración punto a punto.
- Reutiliza los servicios compartidos que han sido desplegados previamente.
- Integra aplicaciones heredadas limitando así el coste de mantenimiento e integración.
- Beneficios en el desarrollo, ya que las aplicaciones son reutilizables, más fácil de mantener y tienen la capacidad de ampliación de las funcionalidades del sistema, exponiéndolas de una forma segura.

3 Entendiendo la Arquitectura Orientada a Servicio (SOA)

Antes de que aparecieran los servicios Web existían tres técnicas para comunicar aplicaciones:

- Se podía escoger una plataforma particular para ofrecer un servicio, como puede ser la plataforma J2SE con la utilización de RMI (Remote Method Invocation) [9] para implementar el mecanismo de comunicación.
- Se podía utilizar CORBA (Common Object-Request Broker Arquitectura) [10].
- Se podía utilizar un protocolo definido de comunicación particular entre dos aplicaciones.

El objetivo de una arquitectura SOA es proveer de la transparencia en la localización del servicio Web, es decir, de la posibilidad de utilizar un determinado servicio que se encuentre en cualquier lugar, sin la necesidad de tener que modificar el código existente.

En el nivel más alto de la arquitectura orientada a servicio podemos encontrar tres componentes:

- El servicio: Pueden participar cualquier número de servicios. Cada servicio tiene una funcionalidad a la que pueden acceder el resto de servicios y clientes.

– El directorio: El directorio tiene información sobre los servicios y la funcionalidad de estos. Y también tiene la información de cómo se puede acceder a cada servicio.

– El cliente: Usa el directorio para localizar servicios y poder usar su funcionalidad. Un cliente puede ser otro servicio que quiere acceder o utilizar la funcionalidad que aportan otros servicios.

Y también podemos encontrar tres colaboraciones entre los componentes, puede llegar a ser una de las partes más importantes de la arquitectura:

– Localización de servicios: Clientes potenciales de los servicios localizan los servicios por medio del directorio. El directorio aporta a los clientes la información sobre como encontrar un servicio.

– Publicación de servicios: Un componente publica un servicio, haciéndole disponible a los clientes a través del directorio.

– La comunicación entre los servicios y el cliente: El cliente hace peticiones al servicio a través del protocolo de red especificado en la información del servicio que tiene el directorio. El servicio recoge la petición del cliente y le retorna la información pedida.

Lo anterior lo podemos ver en las siguientes figuras:

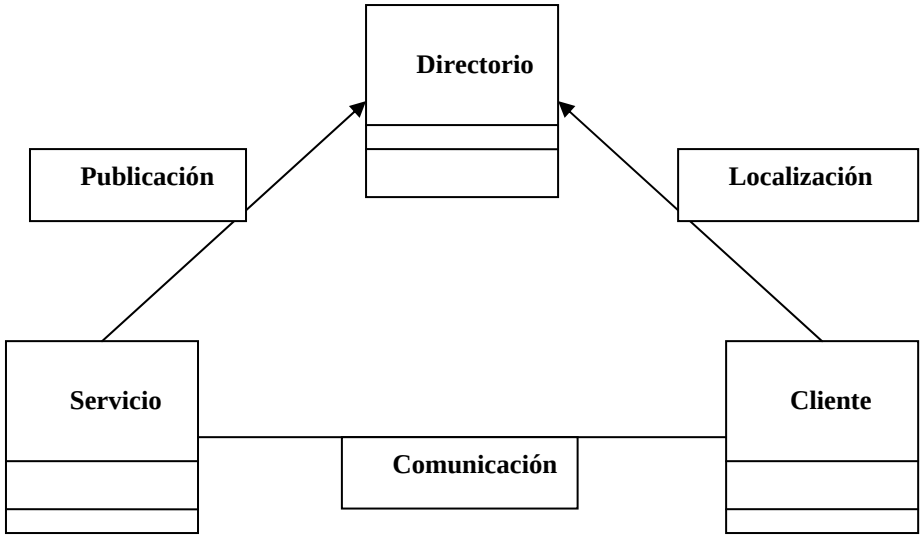

Fig. 1. Componentes de la arquitectura SOA

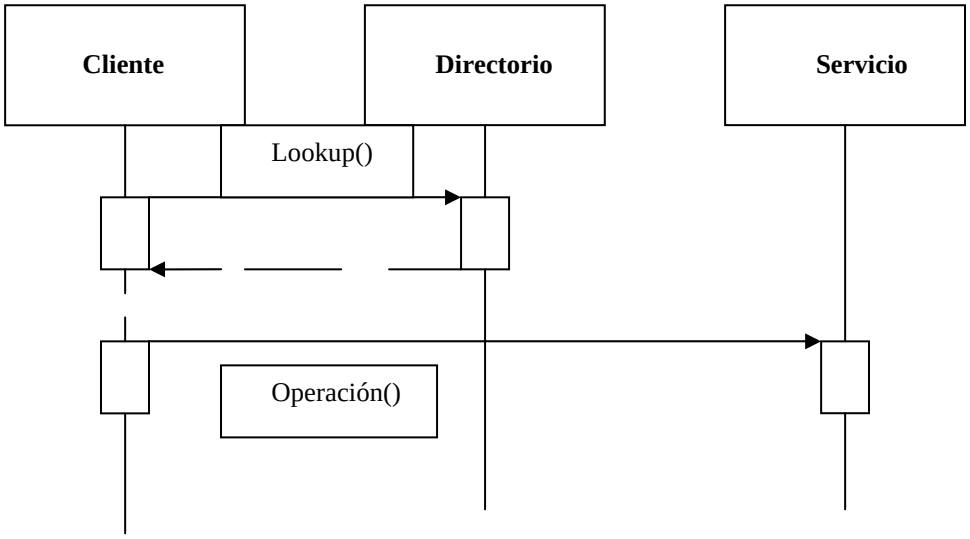

Fig. 2. Diagrama de secuencia de la arquitectura SOA

Normalmente el directorio implementa UDDI (Universal Description, Discovery, and Integration) [11], y todos los componentes se comunican por un protocolo basado en XML (SOAP) y para describir un servicio se utiliza el lenguaje WSDL (Web Service Description Lenguaje).

4 Patrones SOA

También para la construcción de aplicaciones orientadas a servicio utilizando arquitecturas SOA, se han desarrollado una serie de patrones de software, siguiendo la línea de la banda de los cuatro (GoF) [12].

Estos patrones se pueden dividir en cinco categorías:

1. Aprendizaje: Es importante entender el entorno de los servicios Web. Dentro de esta categoría podemos encontrar:

- Service-Oriented Architecture: Es el patrón que forma la arquitectura de los servicios Web como ya hemos visto anteriormente.
- Architecture Adapter. Se puede ver como un patrón genérico que facilita la comunicación entre arquitecturas.

- Service Directory: Este patrón facilita la transparencia en la localización de servicios, permitiendo realizar robustas interfaces para encontrar el servicio que realmente se quiere.

2. Adaptación: Estos patrones son los llamados básicos para conocer el funcionamiento del entorno de los Servicios Web.

En esta categoría nos encontramos:

- Business Object: Un business object engloba a un concepto de negocio del mundo real como puede ser un cliente, una compañía o un producto, por ejemplo, y lo que pretende este patrón es trasladar el concepto de objeto de negocio dentro del paradigma de los servicios Web.
- Business Process: Este patrón se utiliza para tratar con procesos de negocio. En este momento existen dos estándares:
  - Business Process Execution Lenguaje (BPEL) propuesto por Bea Systems, IBM y Microsoft.
  - Business Process Modeling Languaje (BPML) propuesto por el resto de compañías que no están en el grupo anterior como pueden ser WebMethods, SeeBeyond, etc.
- Bussines Object Collection: Con este patrón se pueden realizar composiciones de procesos de negocio
- Asynchronous Bussines Process: Este patrón es la evolución del patrón anterior Bussines Process.

3. Determinando Cambios: Aunque los servicios Web permiten llamadas asíncronas, las implementaciones del servicio pueden estar basados en paso de mensajes, también son importantes los servicios basados en eventos, estos patrones se basan en patrones tradicionales como el Observer o el patrón Publicación/Suscripción.

En esta categoría podemos encontrar:

- Event Monitor: Es un patrón para crear formas efectivas para integrar aplicaciones sin la intervención de otros componentes. El escenario más común donde se utiliza este patrón es para aplicaciones EAI (Enterprise Application Integration).
- Observer Services: Este patrón representa la manera más natural de detectar cambios y actuar en consecuencia.
- Publish/Subscribe Services: Es la evolución del Observer Pattern, mientras que el patrón Observer se base en el registro, el patrón Publish/Subscribe se base en notificaciones, esto permite que distintos servicios puedan enviar la misma notificación.

4. Redefinición: Estos patrones te permiten acceder al comportamiento de un servicio que está implementado en un lenguaje. Ayudan a entender el entorno del Servicio Web y a moldear este entorno de acuerdo con nuestras necesidades.

En esta categoría podemos encontrar:

– Physical Tires: Este patrón ayuda a estructurar mejor la lógica de negocio de los servicios Web, e incluso se puede utilizar para controlar el flujo de negociaciones que puede llegar a producirse utilizando el patrón Publish/Subscribe.

– Connector: Este patrón se suele utilizar con el anterior para resolver los posibles problemas que surgen en la subcripción.

– Faux Implementation: Es una alternativa para resolver los problemas que surgen en la utilización de eventos en los servicios Web. Es simplemente un "socket abierto" que recibe conexiones y aporta las respuestas para los distintos eventos.

5. Creando flexibilidad: Para crear servicios más flexibles y optimizados.
En esta categoría se encuentran:

– Service Factory: Es uno de los patrones más importantes y permite la selección de servicios y aporta flexibilidad en la instanciación de los componentes que crean los servicios Web. Este patrón también se suele utilizar con el patrón Service Cache para aportar una mayor flexibilidad en el mantenimiento de las aplicaciones que utilizan servicios Web, aportando un mayor ROI a las aplicaciones.

– Data Transfer Object: Este patrón aporta rendimiento ya que permite recoger múltiples datos y enviarlos en una única llamada, reduciendo el número de conexiones que el cliente tiene que hacer al servidor.

– Partial Population: Este patrón permite a los clientes seleccionar únicamente los datos que son necesarios para sus necesidades y sólo recuperar del servidor lo necesario. Este patrón además de rendimiento aporta mayor ancho de banda en la red.

Algunos patrones utilizan otros por ejemplo el Business Process Patern usa el Business Object Patern y el Business Object Collection. Y el Service-Oriented Architecture usa los patrones Service Directory y Architecture Adapter.

Referencias

1. Web Services ORG, http://www.webservices.org
2. World Wide Web Consortium (W3C), http://www.w3.org/
3. URI. World Wide Web Consortium (W3C), "Uniform Resource Identifier (URI)", www.w3.org/Addressing/
4. XML. World Wide Web Consortium (W3C), "eXtensible Markup Language (XML)", http://www.w3.org/XML.
5. SOA. "Service Oriented Architecture", http://www.service-architecture.com/
6. World Wide Web Consortium (W3C), "Simple Object Access Protocol (SOAP)": http://www.w3.org/TR/SOAP/ y http://www.develop.com/soap
7. World Wide Web Consortium (W3C), "Web Services Description Language (WSDL)": http://www.w3.org/TR/wsdl.
8. Business Process Execution Language (BPEL), http://www.service-architecture.com/
9. Java Remote Method Invocation (Java RMI), java.sun.com/products/jdk/rmi/

10. Common Object-Request Broker Arquitectura (CORBA), www.corba.org/
11. Universal Description, Discovery and Integration (UDDI): http://www.uddi.org http://uddi.microsoft.com/ y http://www-3.ibm.com/services/uddi/
12. Gamma E., Helm R., Johnson R., Vlissides J.: Patrones de Diseño. Addison Wesley. (2003)