	INGENIERÍA EN INFORMÁTICA – PLAN 2003 DISEÑO AVANZADO DE SOFTWARE – 10º CUATRIMESTRE	
	APUNTE DE ARQUITECTURA ORIENTADA A SERVICIOS	VERSIÓN: 1.3 VIGENCIA: 19-09-2007

ARQUITECTURA ORIENTADA A SERVICIOS

¿Qué es SOA?

La arquitectura orientada a servicios es un paradigma para organizar y utilizar capacidades distribuidas que pueden estar controladas bajo diferentes propietarios e implementadas bajo diferentes tecnologías.

La arquitectura orientada a servicios define la base para que un conjunto de servicios independientes puedan colaborar entre sí dando lugar a procesos de negocio más complejos.

Es decir,

Una arquitectura débilmente acoplada diseñada para resolver las necesidades del negocio de una organización.

Es importante destacar que SOA es un tipo de arquitectura de software, teoría, no está ligado a ninguna tecnología concreta. De hecho puede implementarse un Servicio SOA con .NET o con Java. En definitiva, se puede aplicar SOA con cualquier tecnología que permita desarrollar servicios interoperables.

Un SOA no requiere necesariamente el uso de los servicios Web, estos son, para la mayoría de las organizaciones, el acercamiento más simple para poner en ejecución una arquitectura débilmente acoplada. La definición no pone el foco en la tecnología sino sobre resolver las necesidades de la organización. En términos más simples, un SOA puede parecer nada más que una mejora de los servicios Web (o de otras tecnologías). Puede haber algunas capacidades comunes tales como registración y autenticación, pero para la mayor parte, el SOA para una organización será absolutamente diferente al SOA utilizado en otra. Muchos analistas de la industria han confundido el concepto de la arquitectura orientada a servicios con puestas en práctica orientadas a servicios. Esto ha llevado solamente a la confusión asociada a SOA y a sus conceptos relacionados.

Las razones de la aparición de SOA son básicamente las siguientes:

- (1) La Integración entre aplicaciones y plataformas es difícil.
- (2) Existen sistemas heterogéneos (diferentes tecnologías).
- (3) Existen múltiples soluciones de integración, independientes y ajenas unas a otras y normalmente las integraciones han sido siempre muy costosas.

Se necesitaba un planteamiento estándar que aporte:

- (1) Arquitectura orientada a servicios.
- (2) Basada en un "bus común de mensajería".
- (3) Estándares y especificaciones de servicios (WS-*) para todas las plataformas.

En años recientes, la visión de la Tecnología de Información (IT – **I**nformation **T**echnology), de lo que puede y debe hacer una empresa, así como el papel del profesional IT dentro de la organización, ha llegado a ser cada vez más sofisticada y ambiciosa. Un resultado fascinante de la Tecnología de Información es el planeamiento de la arquitectura de la empresa (EAP – **E**nterprise **A**rchitecture **P**lanning). EAP es el proceso de diseñar y de implementar soluciones IT para resolver las metas y las expectativas gerenciales y, al mismo tiempo, asegurar la continuidad y la eficacia de los sistemas.

Los servicios Web están haciendo un impacto importante en el campo de EAP. Permiten mejorar la alineación entre procesos tecnológicos y de negocio, promesa de llevar a una nueva fase EAP son: los estándares basados SOA. Debido a su naturaleza abierta, el SOA tiene el potencial de entregar mejoras extensas en control de costos IT, agilizar el negocio y eficacia del proceso del mismo.

Algunos definen la arquitectura de la empresa (EA – **E**nterprise **A**rchitecture) como el diseño de un modelo corporativo que incluye la cultura corporativa, mercados, geografía, tecnología, y capital humano. Esto se puede conocer como "arquitectura del negocio". Para nuestros propósitos, el EA significará el diseño, el planeamiento, y la ejecución de los sistemas IT de una empresa.

Para ver cómo el EA afecta el papel y las circunstancias de la tecnología en el negocio, debemos primero examinar el concepto de "arquitectura" en sí mismo. La arquitectura es el proceso de diseñar edificios de modo que respondan a su propósito previsto. La EA es el proceso de diseñar los sistemas IT para "construir" estructuras IT que satisfagan las necesidades del negocio. La EA no tiene ningún propósito más alto que permitir las metas de negocio de la compañía.

La arquitectura orientada a servicios permite crear un proceso que continuamente puede ser actualizado y optimizado y la Tecnología de Información es la estrategia de negocio que sirve como fuerza impulsora en la compañía.

	INGENIERÍA EN INFORMÁTICA – PLAN 2003	
	DISEÑO AVANZADO DE SOFTWARE – 10º CUATRIMESTRE	
	APUNTE DE ARQUITECTURA ORIENTADA A SERVICIOS	VERSIÓN: 1.3 VIGENCIA: 19-09-2007

Esto es así ya que cada elemento de un SOA puede ser movido, ser substituido, y ser modificado. Porque cada uno de los servicios existe en una manera débilmente acoplada, pueden ser montados y ser vueltos a montar en diferente formas por diversos propósitos según las circunstancias. La capacidad de crear procesos y usos compuestos de estos servicios, combinados con la reusabilidad y los estándares basados en la interoperabilidad, crean una arquitectura flexible al cambio.

Podemos ver el SOA como los bloques del Lego. Puede moverse y configurarse de nuevo según nuestra voluntad.

Por todo lo antes expuesto, es primordial que las comunicaciones entre aplicaciones (Servicios) sean independientes de la plataforma (.NET, Java, etc.), de los lenguajes, de los objetos e incluso de los mecanismos de llamada y protocolos de transporte. Estos objetivos son los que conforman SOA.

SOA "ve el mundo" de una forma distinta, todas las aplicaciones deben ser servicios autónomos donde se definan fronteras explícitas y se asuma la heterogeneidad y colaboren plataformas dispares (por ejemplo comunicar aplicaciones .NET con aplicaciones Java). Para esto, en lo que ya es la "implementación de SOA en el mundo real" es imprescindible hablar un mismo "idioma", es decir, compartir el mismo formato de datos (XML) y los mismos esquemas XML (esquema de mensajes de comunicación SOAP) y basarse en protocolos de comunicación estándar.

La "Orientación a Servicios" se diferencia de la "Orientación a Objetos" primeramente en cómo define el término "aplicación". El "Desarrollo Orientado a Objetos" se centra en aplicaciones que están construidas basadas en librerías de clases interdependientes. SOA, sin embargo, hace hincapié en sistemas que se construyen basándose en un conjunto de servicios autónomos. Esta diferencia tiene un profundo impacto en las asunciones que uno puede hacer sobre el desarrollo.

Beneficios

Los beneficios de SOA para una organización se plasman a dos niveles distintos: al del usuario corporativo y a nivel de la organización IT.


Desde el punto de vista de la empresa, SOA permite el desarrollo de una nueva generación de aplicaciones dinámicas que resuelven una gran cantidad de problemas de alto nivel, fundamentales para el crecimiento y la competitividad. Las soluciones SOA permiten entre otras cosas:

- (1) *Mejorar la toma de decisiones:* Al integrar el acceso a los servicios e información de negocio dentro de un conjunto de aplicaciones dinámicas compuestas, los directivos disponen de más información y de mejor calidad (más exacta y actualizada). Las personas, procesos y sistemas que abarcan múltiples departamentos pueden introducirse de forma más directa en una panorámica unificada, lo que permite conocer mejor los balances de costes y beneficios que se producen en las operaciones de negocio que se realizan a diario. Y al disponer de mejor información en un tiempo menor, las organizaciones pueden reaccionar de manera más ágil y rápida cuando surgen problemas o cambios.
- (2) *Mejorar la productividad de los empleados:* Un acceso óptimo a los sistemas y la información y la posibilidad de mejorar los procesos permiten a las empresas aumentar la productividad individual de los empleados. Estos pueden dedicar sus energías a los procesos importantes, los que generan valor añadido y a actividades de colaboración, semiestructuradas, en vez de aceptar las limitaciones y restricciones impuestas por los sistemas de IT rígidos y monolíticos. Más aún, puesto que los usuarios pueden acceder a la información en los formatos y modalidades de presentación (Web, cliente avanzado, dispositivo móvil), que necesitan, su productividad se multiplica en una gran cantidad de escenarios de uso, habituales o nuevos.
- (3) *Potenciar las relaciones con clientes y proveedores:* Las ventajas de SOA trascienden las fronteras de la organización. Los beneficios que ofrece SOA trascienden los límites de la propia organización. Los procesos de fusión y compra de empresas se hacen más rentables al ser más sencilla la integración de sistemas y aplicaciones diferentes. La integración con socios comerciales y la optimización de los procesos de la cadena de suministro son, bajo esta perspectiva, objetivos perfectamente asequibles. Con SOA se puede conseguir mejorar la capacidad de respuesta a los clientes, habilitando por ejemplo portales unificados de servicios. Si los clientes y proveedores externos pueden disponer de acceso a aplicaciones y servicios de negocio dinámicos, no solamente se permite una colaboración avanzada, sino que se aumenta la satisfacción de clientes y proveedores. SOA permite flexibilizar los procesos críticos de compras y gestión de pedidos, habilitando modalidades como la subcontratación de ciertas actividades internas- superando las restricciones impuestas por las arquitecturas de IT subyacentes, y con ello consiguiendo un mejor alineamiento de los procesos con la estrategia corporativa.

SOA contribuye también a documentar el modelo de negocio de la empresa y a utilizar el modelo de negocio documentado para integrar en él y dar respuesta a las dinámicas de cambio que se produzcan y optimizarlo de acuerdo con ellas.

Desde el punto de vista de los departamentos IT, la orientación a servicios supone un marco conceptual mediante el cual se puede simplificar la creación y mantenimiento de sistemas y aplicaciones integradas, y una fórmula para alinear los recursos IT con el modelo de negocio y las necesidades y dinámicas de cambio que le afectan.

- (1) *Aplicaciones más productivas y flexibles:* La estrategia de orientación a servicios permite a la Tecnología de Información conseguir una mayor productividad de los recursos IT existentes, como pueden ser las

	INGENIERÍA EN INFORMÁTICA – PLAN 2003	
	DISEÑO AVANZADO DE SOFTWARE – 10º CUATRIMESTRE	VERSIÓN: 1.3 VIGENCIA: 19-09-2007
APUNTE DE ARQUITECTURA ORIENTADA A SERVICIOS		

aplicaciones y sistemas ya instalados e incluso los más antiguos, y obtener mayor valor de ellos de cara a la organización sin necesidad de aplicar soluciones de integración desarrolladas ex profeso para este fin. La orientación a servicios permite además el desarrollo de una nueva generación de aplicaciones compuestas que ofrecen capacidades avanzadas y multifuncionales para la organización con independencia de las plataformas y lenguajes de programación que soportan los procesos de base. Más aún, puesto que los servicios son entidades independientes de la infraestructura subyacente, una de sus características más importantes es su flexibilidad a la hora del diseño de cualquier solución.

- (2) *Desarrollo de aplicaciones más rápido y económico:* El diseño de servicios basado en estándares facilita la creación de un repositorio de servicios reutilizables que se pueden combinar en servicios de mayor nivel y aplicaciones compuestas en respuesta a nuevas necesidades de la empresa. Con ello se reduce el coste del desarrollo de soluciones y de los ciclos de prueba, se eliminan redundancias y se consigue su puesta en valor en menos tiempo. Y el uso de un entorno y un modelo de desarrollo unificados simplifica y homogeneiza la creación de aplicaciones, desde su diseño y prueba hasta su puesta en marcha y mantenimiento.
- (3) *Aplicaciones más seguras y manejables:* Las soluciones orientadas a servicios proporcionan una infraestructura común (y una documentación común también) para desarrollar servicios seguros, predecibles y gestionables. Conforme van evolucionando las necesidades de negocio, SOA facilita la posibilidad de añadir nuevos servicios y funcionalidades para gestionar los procesos de negocio críticos. Se accede a los servicios y no a las aplicaciones, y gracias a ello la arquitectura orientada a servicios optimiza las inversiones realizadas en IT potenciando la capacidad de introducir nuevas capacidades y mejoras. Y además, puesto que se utilizan mecanismos de autenticación y autorización robustos en todos los servicios, y puesto que los servicios existen de forma independiente unos de otros y no se interfieren entre ellos, la estrategia de SOA permite dotarse de un nivel de seguridad superior.

¿Cómo se resuelven los retos de SOA?

Embarcarse en un proyecto de SOA supone tener que resolver una serie de retos, tanto a nivel organizativo como técnico, y estos retos pueden convertirse en verdaderas barreras insuperables si se ha partido de la idea de que SOA es el remedio para toda clase de males.


Para que las iniciativas de adopción de SOA tengan un fin satisfactorio, hay que asegurarse de que se cumplen una serie de condiciones indispensables:

- (1) *Definir claramente los objetivos del negocio:* El primer paso a la hora de adoptar SOA es identificar con claridad los problemas o retos empresariales más prioritarios. Cuando más precisa sea esa formulación, más fácilmente se podrá delimitar la dirección y el alcance de cualquier proyecto SOA. Disponer de una visión y un rumbo claro desde el principio hará mucho más fácil la ejecución de procesos cuya esencia es la integración de múltiples funciones.
- (2) *Definir claramente el alcance del proyecto SOA:* El objetivo de cualquier proyecto SOA no debe consistir en renovar de forma indiscriminada y masiva toda la infraestructura IT. Este tipo de megaproyectos fracasan a la hora de implementarlos porque cuando por fin se ha conseguido crear la solución, las condiciones del negocio suelen haber cambiado tanto que los problemas que ahora deben resolverse ya no tienen mucho que ver con aquellos que se pretendían resolver cuando se inició el proyecto. El objetivo real de cada iniciativa SOA debe ser responder a necesidades concretas de negocio y crear soluciones en pasos discretos, incrementales e iterativos.
- (3) *Evitar introducir SOA sin motivos reales que lo justifiquen:* La adopción de SOA no debe considerarse una necesidad tecnológica, sino organizativa; debe responder a las necesidades de la organización. Si la introducción de SOA solamente responde al puro gusto por disponer de SOA y se empiezan a crear servicios sin un significado de negocio claro, sin la granularidad adecuada o con demasiadas interconexiones, el resultado será una implementación excesivamente compleja, inmanejable y tremendamente costosa.
- (4) *Gestionar el proceso:* Los servicios y aplicaciones se corresponden con procesos y las salidas de información deseadas a través de las diversas áreas funcionales de la organización. Puesto que representan procesos compartidos, es necesario que se les asigne un propietario para que puedan inventariarse y gestionarse a fin de garantizar que cumplen en todo momento con las directivas corporativas y responden adecuadamente a las necesidades que los justifican.

¿Qué es un servicio?

Un "servicio" es simplemente un programa con el que otros programas interactúan mediante mensajes. Un conjunto de servicios instalados/desplegados sería un sistema.

Los servicios individuales se deben de construir de una forma consistente (disponibilidad y estabilidad son cruciales en un servicio). Un sistema agregado/compuesto por varios servicios se debe construir de forma que permita el cambio, el sistema debe adaptarse a la presencia de nuevos servicios que aparezcan a lo largo del tiempo después de que se hubieran desplegado o instalado los servicios y clientes originales. Y dichos cambios no deben romper la funcionalidad del sistema.

	INGENIERÍA EN INFORMÁTICA – PLAN 2003	
	DISEÑO AVANZADO DE SOFTWARE – 10º CUATRIMESTRE	
	APUNTE DE ARQUITECTURA ORIENTADA A SERVICIOS	VERSIÓN: 1.3 VIGENCIA: 19-09-2007

Así pues, un servicio es una unidad de trabajo bien definida, que acepta mensajes de entrada y produce mensajes de salida. La funcionalidad del servicio es expuesta al exterior mediante una interfaz pública bien definida, llamada "contrato".

Un servicio provee una función específica, generalmente una función del negocio. Un servicio puede proveer una función discreta simple, tal como convertir un tipo de concurrencia en otra, o puede ejecutar un conjunto de funciones de negocio relacionadas. Los servicios que ejecutan un conjunto de funciones de negocios, a diferencia de una función simple, se dicen que son "coarse grained" (grano grueso). Se pueden utilizar conjuntamente múltiples servicios de una manera coordinada. El servicio agregado, o compuesto, se puede utilizar para satisfacer un requerimiento del negocio más complejo. De hecho, una manera de mirar a una SOA es como una estrategia para conectar aplicaciones (expuestas como servicios) de tal manera que se puedan comunicar (y sacar provecho) entre sí. En otras palabras, una arquitectura orientada a servicio es una manera de compartir funciones (generalmente de negocio) de una manera generalizada y flexible.

El concepto de una SOA no es nuevo. Las arquitecturas orientadas a servicios han estado siendo utilizadas por años. Lo que distingue una SOA de otras arquitecturas es el acoplamiento débil (loose coupling). Acoplamiento débil significa que el cliente de un servicio es esencialmente independiente del servicio. La manera en que un cliente (el cual puede ser otro servicio) se comunica con éste no es dependiente de la implementación del servicio. Es de resaltar que esto significa que el cliente no tiene que conocer demasiado acerca del servicio que utiliza. Por ejemplo, el cliente no necesita saber en qué lenguaje se encuentra codificado el servicio, ni sobre qué plataforma se ejecuta. El cliente se comunica con él de acuerdo a una interfaz bien especificada, y luego deja en manos de la implementación del servicio la ejecución del procesamiento necesario.

¿Cómo los servicios encapsulan la lógica?

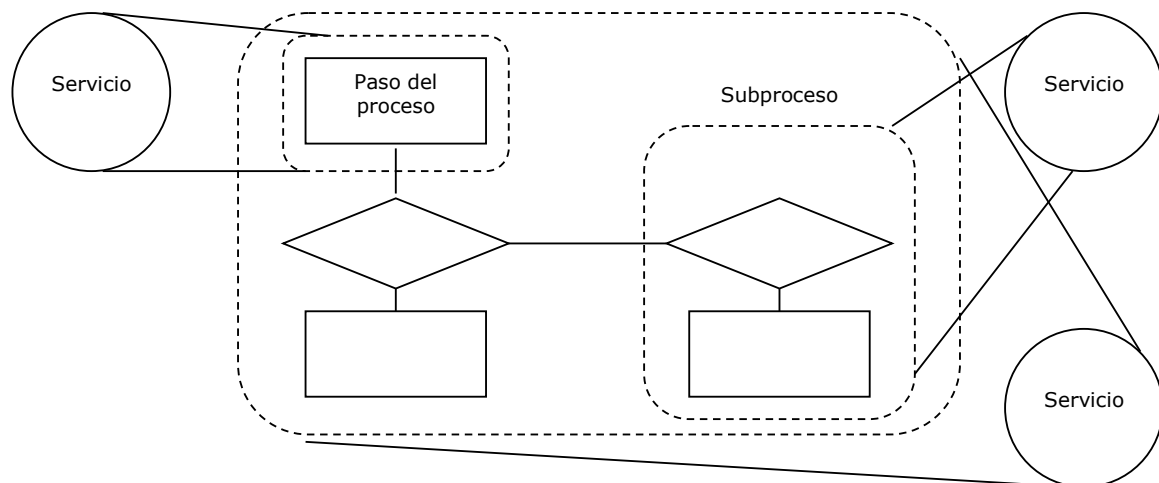
Para mantener su independencia, los servicios encapsulan la lógica dentro de un contexto distinto. Este contexto puede ser una tarea específica de un proceso, un subproceso, o un proceso de negocio.

El contexto abordado por un servicio puede ser grande o pequeño. Por lo tanto, el tamaño y el alcance de la lógica representada por el servicio pueden variar.

Además, la lógica de un servicio puede abarcar la lógica proporcionada por otros servicios. En este caso, uno o más servicios se componen en una colección.

Por ejemplo, las soluciones de automatización en empresas son típicamente una aplicación de un proceso de negocio. Este proceso se compone de la lógica que dictan las acciones realizadas por la solución. La lógica se descompone en una serie de medidas que se ejecutan en la secuencia predefinida de acuerdo con las normas y condiciones en tiempo de ejecución.

Como se muestra en la figura, la construcción de servicios de una solución de automatización puede consistir en que cada servicio pueda encapsular una tarea efectuada por un paso o un subproceso compuesto por una serie de pasos. También, un servicio puede encapsular la lógica de todo el proceso. En los dos últimos casos, el ámbito más amplio representado por los servicios que puede abarcar la lógica encapsulada, son otros servicios.

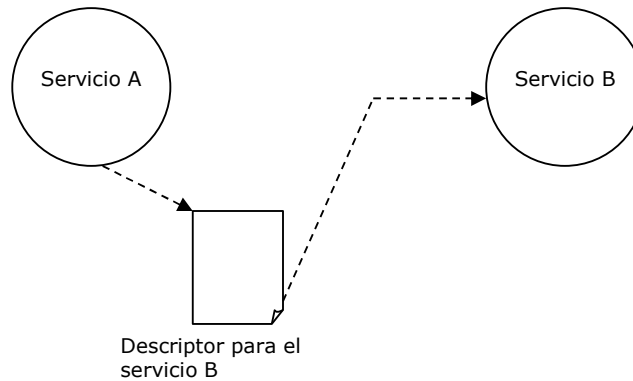


¿Cómo se relacionan los servicios?

Dentro de SOA, los servicios pueden ser utilizados por otros servicios u otros programas. A pesar de todo, la relación entre los servicios se basa en el entendimiento de la interacción de estos, los servicios deben ser conscientes de sí. Este conocimiento se logra mediante la utilización de descriptores de servicios.

	INGENIERÍA EN INFORMÁTICA – PLAN 2003	
	DISEÑO AVANZADO DE SOFTWARE – 10º CUATRIMESTRE	
	APUNTE DE ARQUITECTURA ORIENTADA A SERVICIOS	VERSIÓN: 1.3 VIGENCIA: 19-09-2007

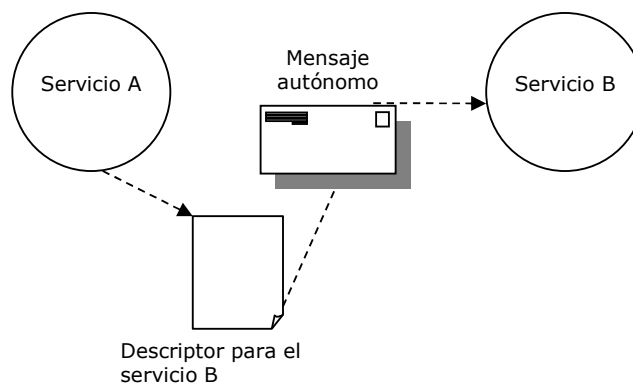
Un descriptor es un formato básico en donde se establece el nombre del servicio y los datos esperados y devueltos por el mismo. La forma en que utilizan los servicios, los descriptors se traduce en una relación clasificada como vagamente unida. Por ejemplo, la figura ilustra un servicio A que es consciente del servicio B porque el servicio A posee el descriptor del servicio B.



Para que los servicios interactúen y logren algo significativo, deben intercambiar información. Un framework de comunicación capaz de preservar su relación vagamente unida, por lo tanto, es necesario. Uno de ellos es el framework de mensajería.

¿Cómo se comunican los servicios?

Después que un servicio envía un mensaje, pierde el control de lo que ocurre con ese mensaje. Esa es la razón por la que se requiere que los mensajes existan como "unidades independientes de comunicación". Esto significa que los mensajes, al igual que los servicios, deben ser autónomos. A tal efecto, los mensajes pueden ser equipados con la suficiente inteligencia para manejarse dentro de la lógica de procesamiento.



Los servicios que proporcionan descriptors de servicios y la comunicación a través de mensajes forman una arquitectura básica. Hasta el momento, esta arquitectura parece similar a las anteriores arquitecturas distribuidas en el apoyo de la mensajería y en la separación de la interfaz de procesamiento de la lógica. Lo que distingue a la arquitectura orientada a servicios es la manera en que sus tres componentes básicos (servicios, descriptors y mensajes) están diseñados.

¿Cómo se diseñan los servicios?

Tal como ocurre con la orientación a objetos, la orientación a servicios se ha convertido en un enfoque de diseño que introduce los principios comúnmente aceptados que rigen el posicionamiento y el diseño arquitectónico de nuestros componentes.

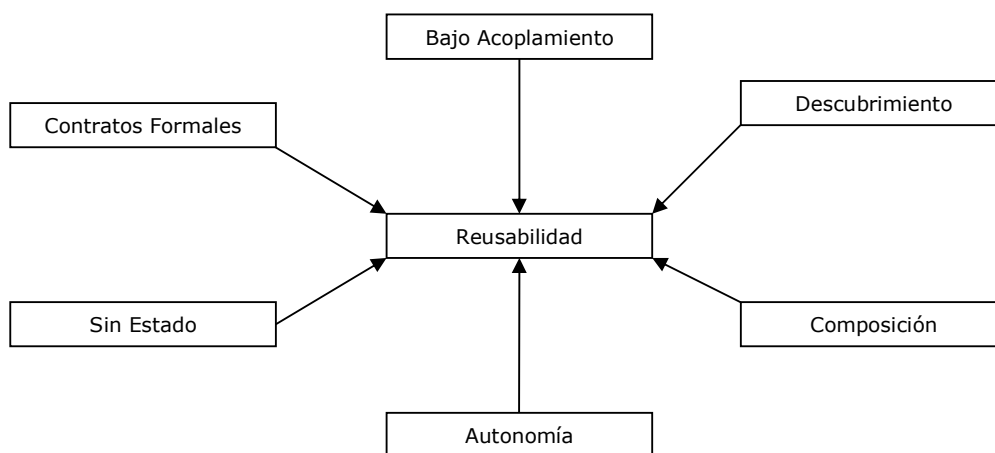
La aplicación de los principios de la orientación a servicios para el procesamiento resulta en estándares que definen la lógica necesaria para dicho procesamiento. Cuando una solución se compone de unidades orientadas a servicios en su lógica de procesamiento, se convierte en lo que nos referimos a una solución orientada a servicios. Algunos de los principios que determinan si una aplicación es "SOA Compliant":

	INGENIERÍA EN INFORMÁTICA – PLAN 2003	
	DISEÑO AVANZADO DE SOFTWARE – 10º CUATRIMESTRE	
	APUNTE DE ARQUITECTURA ORIENTADA A SERVICIOS	VERSIÓN: 1.3 VIGENCIA: 19-09-2007

- (1) *Los servicios deben ser reusables*: Todo servicio debe ser diseñado y construido pensando en su reutilización dentro de la misma aplicación, dentro del dominio de aplicaciones de la empresa o incluso dentro del dominio público para su uso masivo.
- (2) *Los servicios deben proporcionar un contrato formal*: Todo servicio desarrollado, debe proporcionar un contrato en el cual figuren: el nombre del servicio, su forma de acceso, las funcionalidades que ofrece, los datos de entrada de cada una de las funcionalidades y los datos de salida. De esta manera, todo consumidor del servicio, accederá a este mediante el contrato, logrando así la independencia entre el consumidor y la implementación del propio servicio. En el caso de los Servicios Web, esto se logrará mediante la definición de interfaces con WSDL.
- (3) *Los servicios deben tener bajo acoplamiento*: Es decir, que los servicios tienen que ser independientes unos de otros. Para lograr ese bajo acoplamiento, lo que se hará es que cada vez que se vaya a ejecutar un servicio, se accederá a él a través del contrato, logrando así la independencia entre el servicio que se va a ejecutar y el que lo llama. Si conseguimos este bajo acoplamiento, entonces los servicios podrán ser totalmente reutilizables.
- (4) *Los servicios deben permitir la composición*: Todo servicio debe ser construido de tal manera que pueda ser utilizado para construir servicios genéricos de más alto nivel, el cual estará compuesto de servicios de más bajo nivel. En el caso de los Servicios Web, esto se logrará mediante el uso de los protocolos para orquestación (WS-BPEL) y coreografía (WS-CDL).
- (5) *Los servicios deben de ser autónomos*: Todo servicio debe tener su propio entorno de ejecución. De esta manera el servicio es totalmente independiente y nos podemos asegurar que así podrá ser reutilizable desde el punto de vista de la plataforma de ejecución.
- (6) *Los servicios no deben tener estado*: Un servicio no debe guardar ningún tipo de información. Esto es así porque una aplicación está formada por un conjunto de servicios, lo que implica que si un servicio almacena algún tipo de información, se pueden producir problemas de inconsistencia de datos. La solución, es que un servicio sólo contenga lógica, y que toda información esté almacenada en algún sistema de información sea del tipo que sea.
- (7) *Los servicios deben poder ser descubiertos*: Todo servicio debe poder ser descubierto de alguna forma para que pueda ser utilizado, consiguiendo así evitar la creación accidental de servicios que proporcionen las mismas funcionalidades. En el caso de los Servicios Web, el descubrimiento se logrará publicando los interfaces de los servicios en registros UDDI.

Cuando se desarrollan aplicaciones SOA es muy útil y necesario tener en cuenta siempre estos principios, ya que nos van a dar las pautas necesarias para tomar ciertas decisiones de diseño complejas.


Como se habrá podido observar, una característica muy importante de los **Principios de la Orientación a Servicios**, es que todos ellos se interrelacionan. El siguiente gráfico muestra la interrelación de los diferentes principios:



Como se puede observar en el gráfico, **el objetivo de la orientación a servicios es obtener software totalmente reutilizable** a través de un conjunto de técnicas y principios como los descriptos anteriormente.

¿Cómo se construyen los servicios?

Como mencionamos anteriormente, el término "orientación a servicios" y diversos modelos de SOA existían antes de la llegada de los servicios Web.

	INGENIERÍA EN INFORMÁTICA – PLAN 2003 DISEÑO AVANZADO DE SOFTWARE – 10º CUATRIMESTRE	
	APUNTE DE ARQUITECTURA ORIENTADA A SERVICIOS	VERSIÓN: 1.3 VIGENCIA: 19-09-2007

Sin embargo, los servicios Web se han manifestado como un avance tecnológico muy adecuado y con éxito para SOA, son, para la mayoría de organizaciones, la aproximación más simple para implementar una arquitectura desacoplada (ó débilmente acoplada). Hace algunos años, este desacoplamiento se realizaba utilizando otras tecnologías como CORBA y DCOM, o incluso otras aproximaciones basadas en documentos, como EDI para la integración B2B (Business to Business). De hecho, muchas de estas tecnologías son ampliamente utilizadas, y están siendo complementadas, sustituidas o extendidas con la utilización de servicios Web.

En la actualidad, todos los principales vendedores de plataformas apoyan la creación de soluciones orientadas a servicios, y la mayoría deben hacerlo con el entendimiento de que el apoyo en SOA se basa en el uso de servicios Web, si bien, como se dijo anteriormente, se reconoce plenamente que el logro de SOA no requiere servicios Web.

Mitos

Existen una gran cantidad de Mitos entorno a SOA, y que es importante entender antes de continuar avanzando en el tema. La siguiente tabla describe algunos de estos mitos, y muestra las realidades que los "desmienten".

Mito	Hecho
SOA es una tecnología	SOA es independiente de la filosofía de diseño de cualquier tendencia del vendedor, del producto, de la tecnología o de la industria. Ningún vendedor ofrecerá siempre un mismo SOA, porque las necesidades de SOA varían de una organización a otra. Comprar tu infraestructura de SOA de un solo vendedor derrota el propósito de la inversión en SOA.
SOA requiere servicios Web	SOA se puede observar vía servicios Web pero los servicios Web no se requieren necesariamente para implementar SOA.
SOA es nuevo y revolucionario	EDI, CORBA y DCOM eran ejemplos conceptuales de orientación a servicios.
SOA asegura la alineación IT y de los procesos de negocio	SOA no es una metodología.
Una arquitectura como SOA reduce riesgos en la puesta en práctica	SOA es como los copos de nieve, no hay dos iguales. Una arquitectura como SOA puede no proporcionar necesariamente la mejor solución para una organización.
SOA requiere un reacondicionamiento completo de los procesos tecnológicos y de negocio	SOA debe ser incremental y construido sobre las inversiones actuales de la organización en la que se aplica.
Necesitamos construir un SOA	SOA es un medio, no una finalidad.

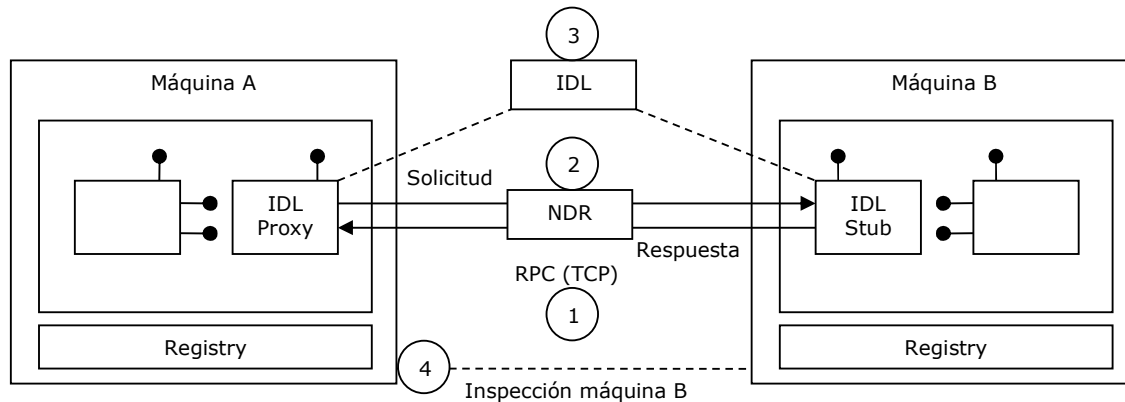
Se debe centrar en entregar una solución, no un SOA. SOA es el medio para entregar una solución y no debe ser una meta final.

Evolución de SOA

La orientación a servicios (SO) es la evolución natural de los modelos de desarrollo de etapas anteriores. A finales de los 80's y principios de los 90's, los modelos de desarrollo estaban basados principalmente en la orientación a objetos; este enfoque se convirtió posteriormente a finales de los 90's en el desarrollo basado en componentes, para convertirse ahora con fuerza en la orientación a servicios. La orientación a servicios mantiene las características de un desarrollo basado en componentes (autodescripción, encapsulación, descubrimiento dinámico y balanceo), pero existe un cambio importante en el nuevo paradigma, y es que se ha pasado de invocar métodos sobre objetos, al intercambio de mensajes entre servicios.

El desarrollo basado en componentes se caracteriza por hacer prácticamente transparente la programación distribuida. Los objetos se pueden distribuir a lo largo de la arquitectura y a ellos se puede acceder tanto si se encuentran localmente como si residen en otra máquina.

Un ejemplo típico de orientación a componentes son las arquitecturas Windows de Microsoft llamadas DCOM (Distributed Component Object Model), actualmente sustituidas por la tecnología .NET. En esta tecnología, cualquier componente podía ser configurado para ser accedido desde otra máquina. Se utilizaba NDR sobre RPC para la comunicación. Este protocolo se caracteriza por ser binario y muy dependiente de la tecnología. Mediante archivos IDLs conseguíamos las llamadas TypeLibs, que también son archivos binarios y propietarios donde se describían los metadatos de los objetos exportados (funciones, parámetros, relaciones, etc.). Por último, como mecanismos para descubrir qué objetos eran expuestos se utilizaba el registro de Windows. Todos los componentes tenían la obligación de registrarse aquí para que un tercero pudiera localizarlo.



1. Protocolo de comunicación – 2. Formato de mensaje – 3. Lenguaje de descripción – 4. Mecanismo de descubrimiento

Esta aproximación a la distribución de componentes de Microsoft no es la única. CORBA o Java RMI son tecnologías de distribución de componentes muy parecidas para otras plataformas que también implementan protocolos binarios y propietarios para la comunicación distribuida de objetos. El principal problema de todas estas tecnologías es que no interoperan entre sí. Los formatos y los conceptos son tan distintos que es muy difícil comunicar un sistema DCOM con otro Java. Además ninguno por separado puede utilizarse en todos los escenarios, porque están ligados fuertemente a una plataforma. DCOM sólo expone objetos COM desarrollados con C++ o Visual Basic y Java RMI sólo expone objetos desarrollados con Java.

En la actualidad, no existen sistemas desarrollados bajo una plataforma única. Los sistemas son heterogéneos y no podemos utilizar ninguna de las tecnologías anteriores para comunicarlos. Es por ello que en las premisas para la definición de la nueva arquitectura basada en servicios, la distribución de aplicaciones tiene que ser independiente de:

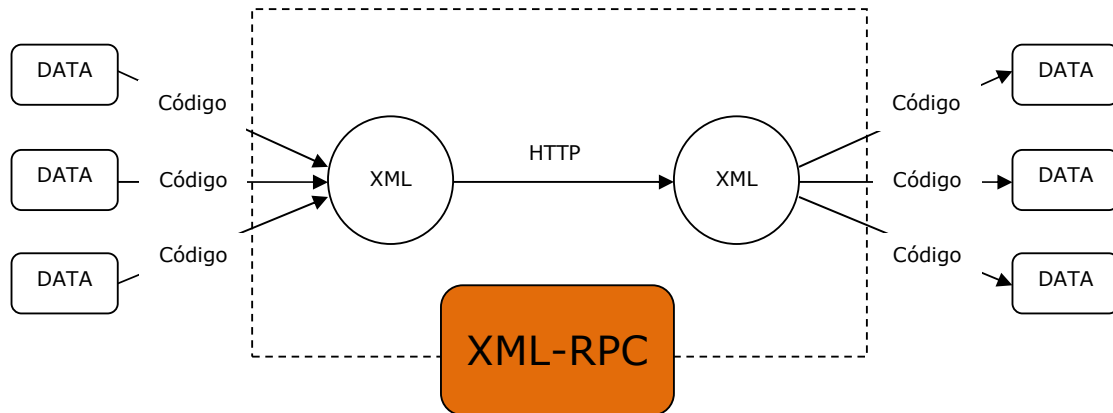
- (1) La plataforma (el sistema operativo y los servidores de aplicaciones).
- (2) El lenguaje de desarrollo (Java, C#, Visual Basic, C++, Cobol, etc.).
- (3) Los objetos, dependiendo de la plataforma y el lenguaje los objetos pueden tener distintas características como polimorfismo, sobrecarga de operadores, tipos de parámetros, etc., por tanto es necesario que no se expongan objetos sino mensajes que son independientes de la implementación. Por ejemplo, si exponemos un objeto desarrollado con C# o C++ que tienen una sobrecarga de operadores ¿cómo vamos a acceder a él desde Java?
- (4) Mecanismos de llamada, evidentemente tiene que ser independiente del paso de parámetros (paso tipo C, tipo Pascal, Java, etc.). Las llamadas tienen que ser sin estado, para permitir el funcionamiento en entornos Internet.

Con todas estas premisas, nace SOA, en la cual una aplicación no expone objetos, sino servicios. Como ya se ha descrito en los párrafos anteriores, un servicio es una unidad completamente autónoma (por ejemplo, dos servicios no deben compartir una base de datos). Cualquier intento de comunicación entre objetos ha sido un fracaso anteriormente, porque la propia naturaleza del objeto lo hace dependiente de la implementación. En SOA no se tiene la funcionalidad de los objetos en la comunicación (por ejemplo herencia, sobrecarga, polimorfismo, etc.). Además, las "fronteras" tienen que ser conocidas y explícitas, es decir, el desarrollador debe de ser consciente de cuándo realiza una llamada remota o local. Como se ha dicho en las premisas, los servicios podrán correr en cualquier plataforma; para conseguirlo, la clave está en el XML. Toda la comunicación se realiza con XML y la definición de los servicios se realiza con esquemas XML (XSD), que son estándares universales y no necesitan la utilización de objetos Proxies.

La primera aproximación tecnológica a SOA está basada en XML-RPC (aparecida en 1998). Nace como solución para la integración de aplicaciones distribuidas, inclusive las de dentro de una compañía. La base de la tecnología es la sencillez, facilidad de aprender y conexión rápida.

XML-RPC está basado en XML y como transporte se utiliza HTTP. Se utilizan mecanismos para poder llevar a cabo llamadas de procedimientos remotos a través de la red.

XML-RPC no puede ser definido como una tecnología para distribución de componentes, pues no maneja objetos, y tampoco provee ningún mecanismo para incluir información de otros vocabularios XML.

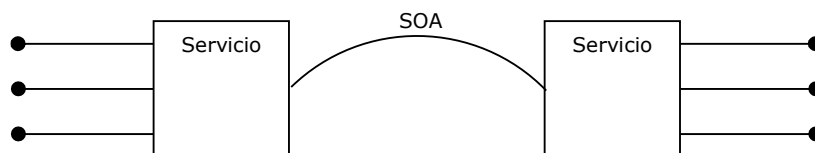


Aunque XML-RPC aporta un cambio en la mentalidad de las arquitecturas de desarrollo basadas en componentes, también conlleva una serie de desventajas:

- (1) Los clientes deben tener conocimiento de las funciones registradas en el servidor.
- (2) No existe ninguna forma de publicar las funciones del servicio ni que el cliente pueda obtener esta información dinámicamente.
- (3) No es válido en el caso de interoperabilidad por colaboración.


La orientación a servicios proporciona un acercamiento evolutivo al software distribuido que facilita la integración y el débil acoplamiento para el cambio. Con el advenimiento de los servicios Web (WS – **Web Service**), la arquitectura ha hecho factible el desarrollo de software orientado a servicios en virtud de la ayuda de las herramientas de desarrollo y de la amplia interoperabilidad de la industria. La orientación a servicios es independiente de la tecnología y de sus patrones arquitectónicos. Desafortunadamente, las ventajas ofrecidas por la orientación a servicios y SOA han sido oscurecidas por la confusión que rodean cada vez más a los términos. Hay tres observaciones importantes alrededor SO:

- (1) *Es evolutivo*: Los principios de la estructura orientada a servicios fue la experiencia en el desarrollo distribuido de aplicaciones en el mundo real. Incorpora SO, conceptos tales como autodescripción, encapsulamiento, y carga dinámica de la funcionalidad en el tiempo ejecución, principios introducidos en los años 80 y los años 90 con el desarrollo orientado a objeto y basado en componentes. Con SO, en vez de usar la invocación del método en una referencia del objeto, se mantiene intercambio de mensajes.
- (2) *No es un producto o una tecnología*: Es un sistema de principios arquitectónicos expresados independientemente de cualquier producto. Apenas pues los conceptos del desarrollo tales como polimorfismo y encapsulamiento son independiente de la tecnología, es la orientación a servicios. Y mientras que los servicios Web han facilitado el desarrollo de servicios orientados, no se requieren usarlos siempre.
- (3) *Es incremental*: Finalmente, la orientación a servicios debe ser un proceso incremental, mejorar el nivel existente en la Tecnología de Información existente.



El bloque fundamental en la construcción de una arquitectura orientada a servicios es un servicio. Un servicio es un programa que se puede ejecutar con intercambios bien definidos de mensajes. Los servicios se deben diseñar, considerando la disponibilidad y la estabilidad. Los servicios se construyen al último mientras que las configuraciones y las agregaciones del servicio se construyen para el cambio. La agilidad se promueve a menudo pues una de las ventajas más grandes de SOA, es una organización de los procesos de negocio puestos en ejecución en una infraestructura débilmente acoplada que es mucho más abierta al cambio que una organización obligada por la aplicaciones monolíticas que requieren semanas para hacer un pequeño cambio en la ejecución.

Sistemas débilmente acoplados dan lugar a procesos de negocio débilmente acoplados, puesto que los procesos de negocio son obligados no más por las limitaciones de la infraestructura subyacente. Los servicios y sus interfaces asociadas deben seguir siendo estables, permitiendo ser configurados de nuevo o reagregados para resolver las necesidades siempre que cambien las reglas de negocio. Los servicios siguen siendo estables confiando en interfaces

	INGENIERÍA EN INFORMÁTICA – PLAN 2003	
	DISEÑO AVANZADO DE SOFTWARE – 10º CUATRIMESTRE	
	APUNTE DE ARQUITECTURA ORIENTADA A SERVICIOS	VERSIÓN: 1.3 VIGENCIA: 19-09-2007

basadas en estándares y mensajes bien definidos por ejemplo que usan los esquemas de SOAP y de XML para la definición del mensaje. Los servicios diseñados para realizar funciones simples, granulares con conocimiento limitado de cómo los mensajes se pasan o se recuperan de él; pueden ser reutilizados mucho más veces dentro de una infraestructura de SOA. La orientación a servicios no requiere necesariamente reescribir funcionalidades. Permite la reutilización IT envolviéndolo en los servicios modulares que se pueden aplicar en cualquier proceso de negocio que se diseñe. Las metas para hacer esto deben ser:

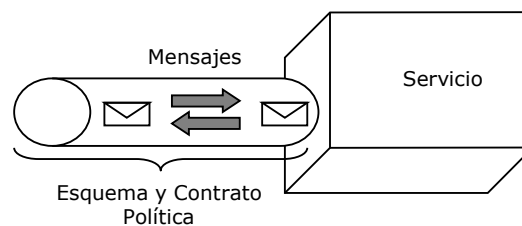
- (1) Conectar con lo que ya está, la gerencia de proceso del negocio, workflows de colaboración, y la divulgación IT existente.
- (2) Extraer más valor de lo que ya está, permitir que aplicaciones existentes sean reutilizadas de nuevas maneras.
- (3) Extender y desarrollar lo que ya está, para crear un soporte a la IT de nuevos procesos funcionales del negocio que se extienden más allá de los límites para lo que fueron diseñadas las aplicaciones existentes.

Una de las ventajas dominantes de la orientación a servicio es el bajo acoplamiento. Ninguna discusión de los servicios Web parece completa sin una cierta referencia a las ventajas de un débil acoplamiento de los puntos finales (aplicaciones) facilitado por el uso de los protocolos del servicio Web. El principio es el de usar un recurso solamente con su servicio publicado y no directamente implementando detrás de él. De esta manera, cambios en la implementación del proveedor del servicio no afecta al consumidor de dicho servicio. Manteniendo un interfaz constante, el consumidor del servicio podría elegir una instancia alternativa del mismo tipo de servicio sin la modificación de las peticiones de la aplicación, aparte de la dirección de la nueva instancia. El consumidor y proveedor del servicio no tienen que tener las mismas tecnologías para la implementación, la interfaz o la integración cuando se utilizan los servicios Web (ambos están limitados sin embargo para utilizar los mismos protocolos del servicio Web).

En resumen,

SOA	Servicios Web (WS)
Encapsulamiento de la lógica	A través de los servicios (con WS)
Relación entre servicios	A través de la descripción de los servicios (con WSDL)
Comunicación de los servicios	A través de mensajes (con SOAP)


Con una Arquitectura Orientada a Servicios la funcionalidad de las aplicaciones se expone a través de una colección de servicios. Estos servicios son independientes y encapsulan la lógica del negocio y sus datos asociados. Los servicios se interconectan vía mensajes con un esquema que define su formato; un contrato que define sus intercambios y una política que define cómo deben ser intercambiados.



Los servicios de una aplicación se diseñan al último con la expectativa que no podemos controlar donde y quien los consume. Esta es una de las diferencias dominantes entre SOA y las arquitecturas tradicionales de una aplicación. Las aplicaciones se diseñaban tradicionalmente para obrar reciprocamente con los usuarios, y la aplicación proporcionaba a la interfaz del usuario, los componentes subyacentes de negocio y almacenes de datos. Mientras que las buenas disciplinas de ingeniería separaron los componentes de negocio de la interfaz, los únicos consumidores de esta lógica de negocio eran las interfaces entregadas como parte de la aplicación. Estos componentes de IGU y de negocio fueron desplegados y versionados tradicionalmente como una sola entidad. Con la orientación a servicios, la funcionalidad del negocio expuesta con el servicio se puede utilizar por cualquier consumidor fuera del control de la aplicación. Estos consumidores pueden ser otros servicios (que incorporan el proceso encapsulado de negocio) o alternativas de interfaces proporcionadas por la misma aplicación. Por lo tanto, los contratos para estos servicios, una vez que estén publicados, deben seguir siendo constantes pues no tenemos ninguna idea de quien los esté consumiendo, ni cuando. Además una aplicación debe tener la flexibilidad de adaptarse a los nuevos servicios ofrecidos después del despliegue. La disponibilidad y la estabilidad de estos servicios por lo tanto se convierten en un factor crítico.

Por lo tanto, cuatro son los elementos esenciales necesarios para la construcción de una Arquitectura Orientada a Servicios:

- (1) *Operación*: Es la unidad de trabajo o procesamiento en una arquitectura SOA.
- (2) *Servicio*: Es un contenedor de lógica. Estará compuesto por un conjunto de operaciones, las cuales las ofrecerá a sus usuarios.

	INGENIERÍA EN INFORMÁTICA – PLAN 2003	
	DISEÑO AVANZADO DE SOFTWARE – 10º CUATRIMESTRE	VERSIÓN: 1.3 VIGENCIA: 19-09-2007
APUNTE DE ARQUITECTURA ORIENTADA A SERVICIOS		

(3) *Mensaje*: Para poder ejecutar una determinada operación, es necesario un conjunto de datos de entrada. A su vez, una vez ejecutada la operación, esta devolverá un resultado. Los mensajes son los encargados de encapsular esos datos de entrada y de salida.

(4) *Proceso de negocio*: Son un conjunto de operaciones ejecutadas en una determinada secuencia (intercambiando mensajes entre ellas) con el objetivo de realizar una determinada tarea.

Por lo tanto, una aplicación SOA estará formada por un conjunto de procesos de negocio. A su vez esos procesos de negocio estarán compuestos por aquellos que servicios que proporcionan las operaciones que se necesitan ejecutar para que el proceso de negocio llegue a buen término. Por último para ejecutar esas operaciones es necesario el envío de los datos necesarios mediante los correspondientes mensajes.

¿Quién define las pautas de SOA?

Mucha gente se pregunta qué organismo es el encargado de estandarizar o por lo menos gestionar SOA. La respuesta es muy sencilla puesto que no hay ningún organismo que pueda hacerlo, ya que SOA es un concepto abstracto, el cual se rige única y exclusivamente por los principios de la Orientación a Servicios.

Pero entonces, ¿No hay manera de controlar la evolución de SOA? ¿Cada fabricante podrá hacer lo que quiera? Estas preguntas tienen una respuesta sencilla. Para comenzar, es necesario dejar un aspecto muy claro. Los Servicios Web, CORBA, MQSERIES, etc., son posibles tecnologías que se pueden utilizar a la hora de implementar una Arquitectura Orientada a Servicios y estas tecnologías sí que están estandarizadas y gestionadas por diversas organizaciones. Por lo tanto, las organizaciones que dirigen el rumbo de SOA, son aquellas que estandarizan las diferentes tecnologías utilizadas para implementar una Arquitectura Orientada a Servicios.

Como ejemplo, para el caso de los Servicios Web. Realmente los Servicios Web están formados por diferentes tecnologías, y por ello son varias las organizaciones que participan en su gestión. Concretamente son tres las organizaciones involucradas:

World Wide Web Consortium: Organismo muy conocido porque es el encargado de la estandarización de HTML y XML (y tecnologías relacionadas). Referente a los Servicios Web, es el encargado de gestionar el protocolo de comunicación de los Servicios Web (SOAP), y el lenguaje de descripción de interfaces (WSDL). Más recientemente, W3C también se ha dedicado a estandarizar algunas de las extensiones WS-* de los Servicios Web. Concretamente se encarga de WS-CDL (Web Services Choreography Description Language) y de WS-Addressing.

El W3C se caracteriza por ser muy formales y rigurosos a la hora de definir y gestionar tecnologías y protocolos, ofreciendo siempre las mejores garantías.

OASIS: Anteriormente conocida como SGML Open, cambió su nombre para redirigir sus acciones de SGML hacia XML. Esta organización es bastante conocida por gestionar dos tecnologías muy conocidas. Es el encargado de desarrollar el estándar UDDI para el registro de Servicios Web, y también se encarga de gestionar la especificación ebXML es cual es un estándar para el intercambio de datos entre aplicaciones B2B. Actualmente también se encarga de desarrollar extensiones WS-* para los Servicios Web. Estas extensiones son WS-BPEL creada para la orquestación de Servicios Web y WS-Security para todos los aspectos relacionados con la seguridad.

Web Services Interoperability: Este organismo es reciente (apareció en 2002), y su principal objetivo es asegurar que se utilizan los estándares adecuados y **no** definirlos ni desarrollarlos. Por ello, este organismo ha definido un documento llamado perfil básico (Basic Profile), en el cual se indican cuales son los estándares que se deberían utilizar para diseñar arquitecturas interoperables. Es decir, este documento es utilizado como mecanismo para generar arquitecturas SOA "compliant". Actualmente también se han preocupado por un aspecto tan importante como la seguridad, y han publicado un perfil de seguridad básico (Security Basic Profile) cuya finalidad es la misma que el anterior perfil, pero relacionado con la seguridad. Además esta organización ha prometido seguir publicando perfiles para distintos aspectos relevantes de las arquitecturas.

Estrategias de implementación

Estrategia 1: 12 pasos para el desarrollo de un SOA

No hay regla rápida en cuanto a cómo una construye una Arquitectura Orientada a Servicios en una organización. A continuación se definen 12 pasos que ayudan a definir un SOA:

- (1) Entender los objetivos del negocio y definir el éxito del proyecto.
- (2) Definir el dominio del problema.
- (3) Entender toda la semántica de la aplicación dentro del dominio.
- (4) Entender todos los servicios disponibles dentro del dominio.

	INGENIERÍA EN INFORMÁTICA – PLAN 2003 DISEÑO AVANZADO DE SOFTWARE – 10º CUATRIMESTRE	
	APUNTE DE ARQUITECTURA ORIENTADA A SERVICIOS	VERSIÓN: 1.3 VIGENCIA: 19-09-2007

- (5) Entender todas las fuentes de información disponible del dominio.
- (6) Entender todos los procesos del dominio.
- (7) Identificar y catalogar todos las interfaces fuera del dominio (los servicios e información simple).
- (8) Definir los nuevos servicios e información limitada a esos servicios.
- (9) Definir los nuevos procesos, así como los servicios y la información limitada a esos procesos.
- (10) Seleccionar la tecnología.
- (11) Desplegar la tecnología de SOA.
- (12) Probar y evaluar.

Paso 1: Entender los objetivos del negocio y definir el éxito del proyecto

Dentro de un negocio es fundamental para el éxito, aplicar tecnología que agregue valor a los objetivos de ese negocio facilitando la eficacia del mismo. Es decir, la tecnología debe ayudar a mejorar. Así, es muy importante para definir los objetivos incluir las metas para el éxito del negocio. Se requiere la interconexión de la gente y sistemas para determinar la información que permitirá resolver el problema de la integración de la aplicación que se definirá correctamente al analizar, modelar y refinar. Solamente entonces se podrá encontrar una solución apropiada.

Paso 2: Definir el dominio del problema

Se debe definir el alcance del SOA dentro de una empresa. La mayoría de los SOA se implementaron dentro de una determinada porción de negocio, hoy es requerido aplicarlo en todos los procesos de la empresa. Los pequeños éxitos conducen en un cierto plazo a grandes éxitos estratégicos, y por ello se necesita las líneas de demarcación al principio del proyecto para proporcionar un foco y entender mejor lo que se pretende lograr.

Paso 3: Entender toda la semántica de la aplicación dentro del dominio

No se puede ocupar de la información que no se entiende, incluyendo la información limitada al comportamiento (servicios).

Así, es extremadamente importante que se identifique todos los metadatos y semánticas de la aplicación del dominio, para tratar correctamente los datos.

La comprensión de la semántica de la aplicación establece la manera y la forma de las cuales una aplicación en particular se refiere a las características del proceso del negocio. Entender la semántica de la aplicación garantiza que no habrá contradicción en la información cuando la aplicación se integre con otras aplicaciones, esto representa uno de los desafíos principales al crear un SOA.

Paso 4: Entender todos los servicios disponibles dentro del dominio

Los servicios proporcionan comportamiento así como información. Aunque hay información que no tiene ningún impacto en el comportamiento. Se debe estar interesado en servicios que abarquen comportamiento e información durante este paso.

Es importante dedicar tiempo a validar supuestos sobre los servicios, incluyendo:


- (1) Donde existen
- (2) El propósito del servicio
- (3) La información que limita al servicio
- (4) Dependencias, por ejemplo, si es un servicio compuesto
- (5) Ediciones de seguridad

Lo importante es la recopilación de toda la información que se tenga disponible sobre los servicios, incluyendo lo que hacen, información pasada a un servicio, información que viene de un servicio, etc.

Paso 5: Entender todas las fuentes de información disponible del dominio

Después, es importante definir interfaces con la información simple. Las cuales pueden hacer una de dos cosas: Consumir la información o producir información. Lo importante es entender:

- (1) Donde existen
- (2) La estructura en el intercambio de la información
- (3) Restricciones de integridad

	INGENIERÍA EN INFORMÁTICA – PLAN 2003	
	DISEÑO AVANZADO DE SOFTWARE – 10º CUATRIMESTRE	VERSIÓN: 1.3 VIGENCIA: 19-09-2007
APUNTE DE ARQUITECTURA ORIENTADA A SERVICIOS		

(4) Dependencias

(5) Ediciones de seguridad

Paso 6: Entender todos los procesos del dominio

Se necesita definir y enumerar todos los procesos del negocio que existen dentro del dominio, automatizado o no. Esto es importante porque, ahora que sabemos qué servicios están disponibles, se deben definir los mecanismos de alto nivel para la interacción, incluyendo procesos de alto, medio y bajo nivel. En muchos casos, estos procesos tienen que todavía automatizarse o se automatizan solamente parcialmente.

Por ejemplo, si un arquitecto en la integración de la aplicación necesita entender todos los procesos que existan dentro de dicha aplicación. Entonces, el arquitecto incorporará los procesos y determinará el propósito del proceso, que lo posee, qué hace exactamente, y la tecnología que emplea. Estos procesos están limitados más adelante a los nuevos procesos o metaprocesos.

Se debe también considerar la noción de compartido contra procesos privados. Algunos procesos son privados, y no se comparten con entidades exteriores (o, en algunos casos, no se comparten con otras partes de la organización). Otros procesos se comparten, para automatizar cosas interempresa. Los procesos privados y compartidos pueden existir en el mismo espacio.

Paso 7: Identificar y catalogar todas las interfaces fuera del dominio (los servicios e información simple)

Se necesita identificar todas las interfaces exteriores para sistemas que forman parte del dominio del problema.

Paso 8: Definir los nuevos servicios e información limitada a esos servicios

Se debe definir todos los nuevos servicios que componen el SOA; éstos formarán parte de una de tres categorías. Primero están los servicios expuestos fuera de sistemas existentes. El segundo tipo de servicios es servicios compuestos, los cuáles son servicios en sí mismos que se componen de muchos otros servicios. Finalmente, los servicios que se diseñan para proveer servicio en aplicaciones nuevas.

Paso 9: Definir los nuevos procesos, así como los servicios y la información limitada a esos procesos

Se debe entender la mayor parte de que es necesario para definir nuevos procesos, así como la relación entre ellos y los procesos existentes, y automatizar procesos que previamente no estaban automatizados. Los nuevos procesos deben ser definidos para automatizar las interacciones de servicios así como los flujos de información a automatizar para un acontecimiento particular del negocio o los sistemas de acontecimientos.

Paso 10: Seleccionar la tecnología

La selección de la tecnología es un proceso difícil que requiere tiempo y mucho esfuerzo. Crear criterios para la tecnología y los productos, el entender las soluciones disponibles, y después emparejar los criterios a esos productos. En muchos casos se requiere un proyecto experimental para probar con que tecnología trabajar.

Paso 11: Desplegar la tecnología de SOA

En este punto, se considera que se ha entendido todo lo que se necesita saber, se tienen nuevos servicios y procesos definidos, se tiene una tecnología apropiada, por lo cual, es el momento de construir.

Paso 12: Probar y evaluar

Para asegurar la prueba apropiada, un plan de prueba tendrá que ser puesto en práctica.

Estrategia 2: Pasos progresivos propuestos por Microsoft

Antes de que un desarrollador escriba la primera línea de código es imprescindible identificar cuáles son los principales elementos motrices de la empresa de cara al proyecto SOA y las dependencias existentes entre el propio negocio y las tecnologías que lo soportan. Ignorar el contexto empresarial puede dar origen a un proyecto donde la infraestructura SOA se implante sin motivo o donde las inversiones realizadas no tengan un correlato adecuado con las necesidades y prioridades de la propia empresa.

Se aplican generalmente dos tipos de estrategia para implementar SOA: la denominada “descendente” (“top-down” en inglés) y la “ascendente” (o “botton-up”). Ambas tienen sus propios puntos débiles que pueden poner en riesgo el éxito del proyecto. Muchas organizaciones que han intentado poner en marcha una infraestructura SOA aplicando el enfoque top-down han descubierto después que cuando la infraestructura por fin se ha puesto en servicio, está desconectada de las necesidades reales del negocio. Y a la inversa, un enfoque ascendente puede también fracasar porque puede originar una implementación caótica de servicios creados sin tener en cuenta los objetivos de la organización.

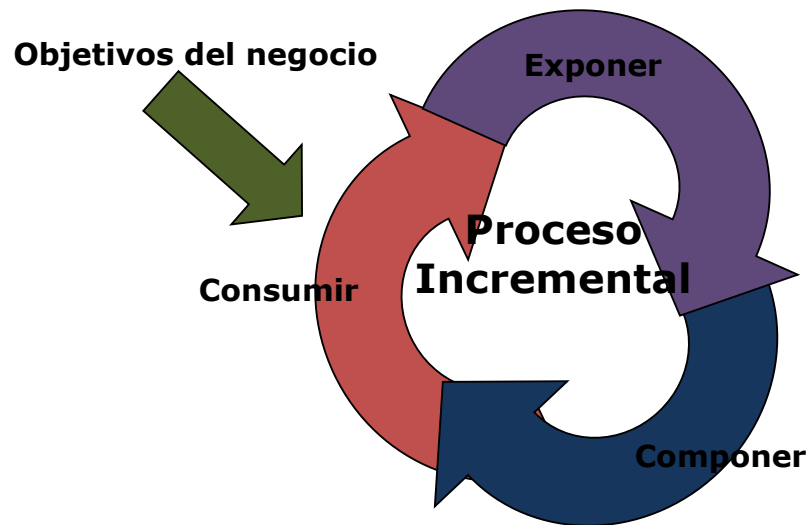
Existe una metodología híbrida, denominada de “término medio” (“middle-out”) que es una síntesis equilibrada de estas dos anteriores.

Los condicionantes principales de la empresa y la visión estratégica se emplean en primer lugar para establecer con claridad el rumbo y las prioridades del proyecto.

	INGENIERÍA EN INFORMÁTICA – PLAN 2003	
	DISEÑO AVANZADO DE SOFTWARE – 10º CUATRIMESTRE	
	APUNTE DE ARQUITECTURA ORIENTADA A SERVICIOS	VERSIÓN: 1.3 VIGENCIA: 19-09-2007

Basándose en ellos, se inicia un proceso iterativo de múltiples pasos orientados a crear pequeños fragmentos de funcionalidades de alto nivel, y en cada iteración se entrega a la organización una nueva aplicación dinámica que se utiliza para generar retorno de la inversión.

Esta estrategia tiene la ventaja de una rápida puesta en valor y genera resultados de negocio en todos sus pasos incrementales e iterativos, facilitando un correcto alineamiento de los recursos IT con las condiciones de negocio, aunque varíen estas con el tiempo.



Una vez que los principales condicionantes del negocio están claramente definidos, se puede comenzar el proceso de implementación.

Partiendo de una visión y unas prioridades claramente definidas, cada proyecto de implementación es un paso progresivo con creación ("exposición") de nuevos servicios, agregación ("composición") de dichos servicios dentro de procesos más amplios, y puesta de estos agregados a disposición de los usuarios ("consumo") dentro de la empresa.

Exposición

La fase de exposición de esta metodología SOA se centra en generar los servicios necesarios a partir de las aplicaciones y datos disponibles. La creación de servicios puede ser de grano fino (un servicio individual que se corresponde con un proceso de negocio individual, como puede ser por ejemplo "insertar código de productos"), o de grano grueso (múltiples servicios que van juntos para realizar una serie de funciones de negocio relacionadas entre sí, como "procesar un pedido").

La fase de exposición viene también muy condicionada por la forma en que se implementan los servicios. La funcionalidad de los recursos IT subyacentes puede hacerse disponible de forma directa (nativa) si esas aplicaciones ya son, por sí mismas, compatibles con los servicios Web o pueden hacerse disponibles como servicios Web utilizando algún adaptador.

Composición

Cuando los servicios ya están creados se pueden combinar en servicios de mayor nivel de complejidad, aplicaciones o procesos de negocio multifuncionales. Puesto que los servicios son entidades independientes entre sí y también con respecto a la infraestructura IT en la cual se basan, pueden combinarse y reutilizarse con la máxima flexibilidad. Y según van evolucionando los procesos de negocio, las reglas y prácticas internas pueden ajustarse sin las restricciones impuestas por las limitaciones que afectan a las aplicaciones de base.

Consumo

Después de crear una nueva aplicación o proceso de negocio, la funcionalidad resultante se pone a disposición (consumo) por parte de usuarios finales o de otros sistemas IT. Al crear aplicaciones compuestas que consumen estos servicios y procesos, la organización dispone ahora de aplicaciones dinámicas que permiten mejorar la productividad y la visión interna del rendimiento de la empresa. Los usuarios pueden consumir los servicios compuestos utilizando distintos medios, como pueden ser portales Web, clientes avanzados, aplicaciones varias y dispositivos móviles.

BIBLIOGRAFÍA

- [Erl05] Thomas Erl (2005). "Service-Oriented Architecture: Concepts, technology and design" Prentice Hall. USA
- [Linthicum04] David Linthicum (2004). "12 Steps to Implementing a Service Oriented Architecture" Grand Central Communicate. USA
- [Microsoft06] Microsoft Corporation (2006). "SOA in the Real World" Microsoft Corporation. USA
- [Pulier06] Eric Pulier, Hugh Taylor (2006). "Understanding Enterprise SOA" Manning. USA