Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

# DECISIONES DE DISEÑO PARA CREAR UNA ARQUITECTURA ORIENTADA A LOS SERVICIOS, POR MEDIO DE SERVICIOS WEB

**Juan Mauricio Luca Chavez**
Asesorado por el Ing. Rodolfo Estuardo Arriaga Herrera

Guatemala, marzo de 2009

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA


FACULTAD DE INGENIERÍA

# DECISIONES DE DISEÑO PARA CREAR UNA ARQUITECTURA ORIENTADA A LOS SERVICIOS, POR MEDIO DE SERVICIOS WEB

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA

FACULTAD DE INGENIERÍA

POR:

**JUAN MAURICIO LUCA CHAVEZ**

ASESORADO POR EL ING. RODOLFO ESTUARDO ARRIAGA HERRERA

AL CONFERÍRSELE EL TÍTULO DE

**INGENIERO EN CIENCIAS Y SISTEMAS**

GUATEMALA, MARZO DE 2009

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

## NÓMINA DE JUNTA DIRECTIVA

| | |
|---|---|
| DECANO | Ing. Murphy Olympo Paiz Recinos |
| VOCAL I | Inga. Glenda Patricia García Soria |
| VOCAL II | Inga. Alba Maritza Guerrero de Lòpez |
| VOCAL III | Ing. Miguel Ángel Dávila Calderón |
| VOCAL IV | Br. Jose Miguel Milton De León Bran |
| VOCAL V | Br. Isaac Sultan Mejía |
| SECRETARIA | Inga. Marcia Ivónne Véliz Vargas |

## TRIBUNAL QUE PRACTICÓ EL EXÁMEN GENERAL PRIVADO

| | |
|---|---|
| DECANO | Ing. Murphy Olympo Paiz Recinos |
| EXAMINADOR | Inga. Virgina Victoria Tala Ayerdi |
| EXAMINADOR | Ing. Marlon Antonio Pérez Turk |
| EXAMINADOR | Ing. Víctor Hugo de León Barrios |
| SECRETARIA | Inga. Marcia Ivònne Véliz Vargas |

## HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

# DECISIONES DE DISEÑO PARA CREAR UNA ARQUITECTURA ORIENTADA A LOS SERVICIOS, POR MEDIO DE SERVICIOS WEB,

tema que me fuera asignado por la Dirección de la Escuela de Ingeniería de Ciencias y Sistemas, en febrero 2008.

Juan Mauricio Luca Chavez

## AGRADECIMIENTOS A:

A Dios gracias, por que sin él nada puedo y nada soy.

**Mi familia** Mi madre Irma Isabel, por ser siempre la luz de esperanza en mi vida.

Mis hermanas Araceli, Evelyn y Saida, por ser siempre un apoyo constante en mi vida.

Mi padre Juan Manuel, por enseñarme el valor del trabajo.

A mis sobrinas Katy, Julisa y Fátima, por ser siempre motivo de inspiración.

Mis tíos y tías, mis abuelos, mi abuela Cruz, que en paz descanse, por darme siempre su apoyo.

**Mi novia**

Alejandra, gracias por tu apoyo invaluable que estuvo allí cuando más lo necesitaba.

## DEDICATORIA:

**A mis compañeros universitarios**

Porque el esfuerzo rindió fruto.

**A mis primos, primas, sobrinas**

Para motivarlos a alcanzar sus metas y sobrepasarlas.

# ÍNDICE GENERAL

# 3. SERVICIOS WEB Y SOA

## 4. APLICACIONES WEB Y SOA BASADO EN SERVICIOS WEB

# ÍNDICE DE ILUSTRACIONES

## FIGURAS



## TABLAS

# GLOSARIO

| | |
|---|---|
| **API** | Interfaz que sirve para poder interactuar con una aplicación. |
| **BluePrint** | Término en el idioma Inglés que hace referencia a un plan detallado de acciones. |
| **Byte** | En su definición más sencilla un byte es la representación de información en forma de carácter con fines computacionales, sirven para la representación de datos y medir el tamaño de la información. |
| **Cookie** | Denominación que se le da algunos archivos que son almacenados en las máquinas del visitante de una página de Internet. Es utilizado para mantener configuraciones como sesiones de usuario al abrir el correo, configuración visual u otro. |
| **CRM** | Término que hace referencia a los sistemas informáticos que están orientados a dar satisfacción a las necesidades de los Clientes, o por su siglas en Inglés "Customer Ralationchip Management" |
| **Direccionamiento** | Capacidad de poder identificar la ubicación de algo por medio de una dirección. |
| **E-Procurement** | Denominación que se le da a los sitios en Internet que dan acceso a los usuarios que buscan compradores/vendedores de productos/servicios. |
| **Firewall** | Programa de software o equipo físico que tiene como fin el filtrado del tráfico de información hacia adentro o hacia fuera de una red de trabajo de computadoras. |

| | |
|---|---|
| **FTP** | Es un protocolo de comunicación que permite el intercambio de archivos (File transfer Protocol) entre computadoras conectadas a una misma red basada en TCP. |
| **Hardware** | Equipo de cómputo físico. |
| **Hipertexto** | Hipertexto, en informática, es el nombre que recibe el texto que en la pantalla de una computadora conduce a su usuario a otro texto relacionado. La forma más habitual de hipertexto en documentos es la de hipervínculos o referencias cruzadas automáticas que van a otros documentos.[1] |
| **HTML** | Es el lenguaje de marcado predominante para la construcción de páginas Web. Por su traducción Lenguaje de marcas de hipertexto, éste lenguaje es utilizado para describir la estructura y el contenido en forma de texto de las páginas Web. |
| **Integración** | Capacidad de acoplamiento de un sistema de información con otro sistema de información. |
| **Interoperabilidad** | Capacidad de poder comunicarse entre sistemas realizados con distintas tecnologías. |
| **MIME** | Es un estándar de criptografía de clave pública y firmado de correo electrónico. |
| **OASIS** | OASIS (Organization for the Advancement of Structured Information Standards) es un consorcio internacional que guía el desarrollo, convergencia, y la adopción de estándares para negocios en Internet. |

[1] Definición obtenida de http://es.efactory.pl/Hipertexto

| | |
|---|---|
| **Parseo** | Dícese del proceso de analizar una secuencia de símbolos con el objetivo de determinar su estructura gramatical con respecto a una ya dada con una sintaxis establecida. |
| **PDAs** | Los Asistentes digitales personal son dispositivos digitales de bolsillo con sistemas operativos y funcionalidades como agenda, acceso a Internet y otros. |
| **RFC** | Son una serie de documentos técnicos, en los cuales se detallan todos los aspectos relacionados a la tecnología de la que se sirve el Internet. |
| **Ruteo** | Encaminamiento o forma de guiar algo hacia un destino determinado. |
| **SGML** | Lenguaje de marcado generalizado, es un sistema para la organización y etiquetado de documentos sin la imposición de algún tipo de etiquetas como el HTML. |
| **SLA** | Por sus siglas en Inglés "Nivel de aceptación del Servicio", se denomina así al contrato del servicio donde el nivel del servicio es definido. Usualmente se utiliza el término al momento de de la entrega de la aceptación del contrato del servicio. |
| **Software** | Programas computacionales con determinado fin. |
| **TCP-IP** | Protocolos de comunicación más importantes sobre los cuales se el Internet, Protocolo de Control de transmisión y Protocolo de Internet. |
| **TI o IT** | Tecnologías de la información |

| | |
|---|---|
| **UDDI** | Son las siglas para denominar "Descripción Universal, Descubrimiento e Integración", es utilizado para el registro de servicios Web, el objetivo de UDDI es la publicación de servicios Web para ser accedidos por los mensajes SOAP y dar paso a documentos WSDL, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con dichos servicios Web.[2] |
| **Vistas** | Abstracciones que identifican las partes de un sistema. |
| **W3c** | Consorcio de la triple w “World Wide Web”. |
| **w3c** | Consorcio de la triple w “World Wide Web” encargado de desarrollar y aprobar estándares que serán utilizados en la Web. |
| **Web** | Dícese de lo referente o lo relacionado con el Internet. |
| **WSEL** | Es un formato de XML para describir características no operacionales de servicios finales, tales como calidad de servicio, costo o propiedades de seguridad. Por sus siglas en inglés "Web Service EndPoint Languaje". |

[2] Obtenida de http://es.wikipedia.org/wiki/UDDI

# RESUMEN

La arquitectura de software se puede decir que es el diseño a nivel mas alto para un sistema o solución de software, por lo que en la primera parte del presente documento se trata acerca de los conceptos referentes a la arquitectura de software y de cómo la importancia de una orientación hacia los servicios en el diseño de soluciones de software promueve una arquitectura de software que se oriente hacia los servicios.

Dado que la arquitectura orientada a los servicios se puede considerar como una nueva forma de diseñar soluciones de software basados en esta arquitectura. En la segunda parte se da una descripción de lo que es SOA, la evolución que se ha tenido para llegar a ella y los patrones relacionados con esta, así como las decisiones que se deben tomar a la hora de pensar en una arquitectura orientada a los servicios. También se hace mención de las capas que deberían de componer una arquitectura que se oriente a los servicios.

Por último, se realiza una descripción de lo que son los servicios Web y de cómo estos se pueden utilizar para llevar a cabo algunos conceptos de SOA, se describen todos los aspectos importantes relacionados a los servicios Web tales como sus especificaciones y algunos estándares aunque no en una forma profunda, así como los protocolos que dan soporte a las características de una arquitectura orientada a los servicios. También se da un vistazo a algunos sistemas que interactúan por medio de servicios Web, haciendo notar en esta interacción que lo servicios Web proveen de las características para implementar los servicios de una SOA, promoviendo la integración e interoperabilidad entre sistemas.

# OBJETIVOS

## General

Identificar las decisiones de diseño involucradas en el proceso de diseño y desarrollo de una arquitectura orientada a los servicios SOA basada en servicios Web y como estas decisiones pueden ser usados para el diseño y desarrollo de aplicaciones Web para Internet. Definir lo que es un servicio Web y cómo estos son los componentes que dan pie a la integración con otras aplicaciones, identificando los estándares y protocolos involucrados en dicha integración.

## Específicos

1. Comprender el por qué la necesidad de una arquitectura orientada a los servicios en aplicaciones de software, identificando las tendencias y razones de esta necesidad.

2. Identificar las decisiones de diseño y posibles capas para el diseño y desarrollo de una arquitectura orientada a los servicios.

3. Un entendimiento detallado de lo que es un servicio, y de cómo la arquitectura orientada a estos provee la plataforma para prestar dichos servicios en forma Web.

4. Comprender como una arquitectura orientada a los servicios basada en servicios Web provee de los argumentos necesarios para una integración e interoperabilidad entre aplicaciones más eficaz y eficiente.

# INTRODUCCIÓN

Actualmente en Guatemala, la globalización y la fusión de empresas así como un acelerado crecimiento en el uso de Internet y el intercambio de información a nivel comercial y gubernamental, son factores clave que hay que tomar en cuenta al momento de desarrollar soluciones de software, dichas soluciones deben proveer un modelo de negocio que se adecue a las necesidades cambiantes de la empresa y su entorno.

Estas soluciones de software deben tomar en cuenta las interacciones con otras empresas, sus clientes, y hasta con la competencia, interacciones que les provean de funcionalidades que generen ventaja y diferenciación que permitan alcanzar los objetivos organizacionales.

Por lo tanto, si las empresas desean extraer el beneficio máximo de sus aplicaciones que serán publicadas en Internet, se debe pensar en una forma de crear aplicaciones con una arquitectura adecuada a dichas necesidades. Por lo tanto es aquí donde surge la idea de crear aplicaciones con una arquitectura orientada a los servicios basada en servicios Web, puesto que esta provee de las bases para la interoperabilidad e integración de aplicaciones con la ayuda de los servicios Web.

Dicha integración por medio de la interoperabilidad de protocolos y estándares para servicios Web proveerá a las empresas de la información adecuada en el momento deseado, con datos en tiempo real que sean de importancia para las empresas, esto a su vez ayudara a reducir los costos de integración, debido a que la integración será independiente de la plataforma en que estén las aplicaciones, ya que los servicios Web se basan en protocolos abiertos como lo son XML y SOAP que funcionan sobre el protocolo de Internet mas utilizado como lo es el HTTP.

# 1. ARQUITECTURA Y ORIENTACIÓN A LOS SERVICIOS

## 1.1. Arquitectura de software

La arquitectura de software es a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones.[3]

## 1.2. ¿Qué es la arquitectura de software?

Existen muchas definiciones de arquitectura del software y al parecer ninguna de ellas ha sido totalmente aceptada. Ampliamente se puede aceptar que la arquitectura del software es el diseño de nivel más alto de la forma que en la estructura de un sistema está basada, la cual tiene la responsabilidad de:

- Definir los módulos principales
- Definir las responsabilidades que tendrá cada uno de estos módulos
- Definir la interacción que existirá entre dichos módulos:
- Control y flujo de datos
- Secuenciación de la información
- Protocolos de interacción y comunicación
- Ubicación en el hardware

[3] Paul Clements. "A Survey of Architecture Description Languages". Proceedings of the International Workshop on Software Specification and Design, Alemania, 1996

La arquitectura del software aporta una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores del diseño.

La definición oficial de Arquitectura del Software de la IEEE Std. 1471-2000 dice textualmente así: "La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución". [4]

Entonces una arquitectura de software es un modelo abstracto reusable que define la estructura para construir un sistema. Su definición envuelve decisiones de diseño que se traducen en atributos de calidad como:

- Rendimiento
- Seguridad
- Costo al hacer un cambio
- Confiabilidad
- Usabilidad

La arquitectura representa la clave para comprender, organizar y comunicar un sistema, además, permite implantar conceptos como la reusabilidad y facilita la evolución de la solución.

### 1.3. Productos resultantes de la arquitectura de software

El objetivo principal de la arquitectura del software es aportar elementos que ayuden a la toma de decisiones, y al mismo tiempo proporcionar

[4] L. Bass et al., "Software Architecture in Practice," Addison-Wesley, 1998 M. Fowler, "Public versus Published Interfaces," IEEE Software, Vol. 19, No. 2, March/April 2002, páginas 18-19

conceptos para un lenguaje común que permita la comunicación entre los equipos que participen en un proyecto. Para conseguirlo, la arquitectura del software construye abstracciones materializadas en formas de diagramas o "bluePrints".

***No se han definido estándares para definir la forma o el lenguaje que se utilizará para la realización de tales diagramas o "blueprints". Esta necesidad condujo a la existencia de un acuerdo en la necesidad de realizar una organización de éstas abstracciones en vistas, tal y como se hace al diseñar un edificio. La cantidad y tipos de vistas difiere en función de cada tendencia arquitectónica.***

## 1.4. Las vistas de la arquitectura de una aplicación

Se puede decir que la arquitectura que se define para una aplicación, no es más que una de las formas posibles de su Arquitectura corporativa u organizacional.

Dado que una aplicación pertenece a una organización y dicha aplicación posee una arquitectura que describe la forma en que ésta está estructurada y las funciones que realiza, la idea es describir la estructura de los "sistemas/aplicaciones" con modelos que reflejen la visión que tienen todos aquellos interesados en la organización de tal forma que se puedan utilizar para planificar y tomar decisiones de mejor forma.

Dado lo anterior, entonces para una organización de cualquier índole se puede describir su arquitectura de sistemas de información desde distintos puntos de vista, a continuación se listan algunos:

- **Negocio:** Describe el funcionamiento interno del negocio central de la organización.

- **Aplicación:** Muestra las aplicaciones de la organización, su funcionalidad y relaciones.
- **Información:** Describe la información que maneja la organización y cómo está ligada a los circuitos de trabajo.
- **Tecnología:** Describe la estructura de hardware y software base que da soporte informático a la organización. [5]

Se debe hacer hincapié sobre estas perspectivas, las cuales representan las mismas perspectivas de implicados en la organización o empresa, por lo tanto habrá quienes estén interesados en el **negocio** y su funcionamiento, las "**aplicaciones/sistemas**" y sus relaciones, los **datos** que finalmente se volverán en la **información** que manejará la organización/empresa y finalmente la **tecnología** (hardware y software de base) que sirve de plataforma para las aplicaciones/sistemas.

Se pueden mencionar otras posibles vistas arquitectura de una aplicación:

- **Vista conceptual**
- **Vista lógica**
- **Vista física**
- **Vista de implementación**

Cuando se desarrolla una aplicación es de un valor muy grande realizar una descripción de las vistas y plasmarlo en un documento de arquitectura. Para estos fines la notación que más se utiliza es el Lenguaje Unificado de Modelado o UML de sus siglas en inglés.

[5] Tomado de UML y Diagramas de estados http://www.informatizate.net/articulos/arquitectura_de_software_20031006.html

Quizá uno de los modelos más conocidos es el "4+1" de Philippe Kruchten, vinculado al Rational Unified Process (RUP), que define cuatro vistas diferentes:[6]

- **Vista lógica/diseño**: describe el modelo de objetos.
- **Vista de proceso**: muestra la concurrencia y sincronía de los procesos.
- **Vista física/implementación**: muestra la ubicación del software en el hardware.
- **Vista de desarrollo**: describe la organización del entorno de desarrollo.

Se puede crear una **quinta** vista, consistente en la elaboración de escenarios basados en una selección de las vistas ya existentes.

## 1.5. Ejemplos de arquitectura

Para dar una idea de lo anteriormente expuesto, en el anexo 1 se muestran dos diagramas de arquitectura en un entorno J2EE, ambos diagramas están disponibles en Designing Enterprise Applications with the J2EE Platform, Second Edition.[7]

[6] Basado en "Usabilidd y Arq. De Software" http://alzado.org/articulo.php?id_art=355
[7] Anexo tomado de http://alzado.org/articulo.php?id_art=355

## 1.6. Orientación hacia los servicios

Esta orientación describe una nueva arquitectura para conectar sistemas y está basada en tres simples conceptos:

- Un **servicio** es un programa con el cual otros sistemas/programas interactúan usando mensajes.
- Un **cliente** es un programa que hace utilizables los servicios para las personas.
- Un **sistema conectado** es una colección de servicios y clientes interconectados.

Un servicio es un simple programa que permite al cliente interactuar con él, intercambiando mensajes XML. Los servicios publican su "contrato" a través de una definición WSDL (Web service description language, lenguaje descriptor del servicio Web). El mensaje XML que es intercambiado está definido en el esquema y contrato requerido por el servicio Web. Los servicios pueden estar separados a través de una larga distancia geográfica, múltiples autoridades de confianza y distintos ambientes de ejecución. Las interfaces entre los servicios siempre son explicitas y anónimas.

## 1.7. Los servicios web

Los servicios Web son funciones de negocio auto contenidas que operan en Internet. Son de gran importancia porque permiten a los sistemas de empresas, organizaciones y gobiernos, interactuar entre sí más fácilmente.

Actualmente las empresas necesitan estar más cerca entre proveedores, compañeros, clientes y con todo aquel involucrado que entre en juego en su cadena de valor. La dinámica de las alianzas es muy fuerte, y no se puede desaprovechar oportunidades de negocio, por estas razones es necesario

que los sistemas de las compañías se conecten más rápidamente y de manera natural, sin la necesidad de construir grandes interfaces de conversión y transmisión.

En el caso del e-procurement una entidad gubernamental o ministerio puede comunicarse con otros sistemas para requerir información sobre RRHH por ejemplo, solicitar órdenes de compra, solicitar pagos, solicitar información de contribuyentes etcétera, con la gran ventaja de que lo pueden hacer en cualquier momento desde cualquier parte del mundo.

## 1.8. Estándares y servicios web

Los servicios Web están escritos y codificados siguiendo especificaciones estándares estrictas para trabajar y comunicarse entre si y con componentes similares.

Los servicios Web son pequeñas unidades de código de programación, diseñadas para manejar una limitada cantidad de tareas y funciones específicas. Usan XML (Extensible markup language) como protocolo básico de comunicación. Son independientes del sistema operativo donde se estén ejecutando y del lenguaje de programación. Conectan entre sí aplicaciones, sistemas y dispositivos electrónicos, y finalmente exponen a sus usuarios y consumidores sus métodos y su funcionalidad.

Se basan en una arquitectura orientada a los servicios SOA. Allí se puede encontrar al cliente (quien necesita o requiere servicios), el Service Broker (cuando el cliente necesita encontrar un proveedor, la ubicación es provista por un service broker, quien opera usualmente como repositorio de servicios, éste le devuelve un documento que permite al cliente encontrar el servicio y comunicarse con él), y el Service Provider (es quien definitivamente provee el servicio).

## 1.9. Protocolos y Estándares de Internet que usan los servicios web

**HTTP** (Hypertext Transfer Protocol) es el estándar (por W3C) de la world wide Web para comunicación sobre Internet.

**XML** (eXtensible Markup Language) es el protocolo estándar para registrar, transportar e intercambiar datos, también estandardizado por W3C.

**SOAP** (Simple Object Access Protocol) es un protocolo independiente de plataforma y lenguaje que permite a los programas comunicarse vía el estándar de Internet http. (w3c).

**WSDL** (Web services description language) es un lenguaje basado en xml que se usa para definir los servicios Web y describir como se acceden a los mismos. WDSL es una propuesta de ariba, IBM y Microsoft para describir webservices para w3c XML Activity sobre xml protocols.

**UDDI** (Universal description, discovery and integration) es un servicio de directorio donde se registran y buscan servicios web construidos por distintos proveedores. Se trata de un registro público donde realizar consultas sobre los servicios y su ubicación.

## 1.10. Importancia de la orientación hacia los servicios

La orientación a los servicios, en comparación con arquitecturas distribuidas como .NET o J2EE, refleja más de cerca procesos y relaciones del mundo real. Por lo tanto, la orientación a los servicios representa de una manera más natural la forma de modelar y construir software que resuelva procesos de negocio en el mundo real.

Cuando diversas partes del sistema son desarrolladas y operadas por diversas organizaciones, por ejemplo, en escenarios de B2B (Bussines to bussines), SOA llega a ser extremadamente importante. Los sistemas con una arquitectura orientada a los servicios se diseñan para soportar el cambio.

El hecho que los servicios se pueden programar en cualquier lenguaje usando cualquier herramienta de desarrollo, y que pueden correr en diversas plataformas los hace convenientes para la integración e interoperabilidad entre aplicaciones.

Sin la ayuda una arquitectura orientada a los servicios SOA, se debería exponer el contrato en los términos del lenguaje que fue elegido para poner los servicios en ejecución. Esto claramente representa algunos desafíos a los clientes, por ejemplo el cliente debe tener sistemas específicos en determinado lenguaje y debe entender el contrato público expuesto por el proveedor del servicio con respecto al lenguaje en que éste fue publicado, etc.

Afortunadamente con un SOA no se requiere a que los proveedores de servicios publiquen alguna implementación específica en un lenguaje determinado para los clientes, por esta razón, los clientes no tienen que preocuparse con un lenguaje determinado.

## 1.11. ¿Por qué la necesidad de una Arquitectura orientada a los servicios SOA?

El SOA (service oriented architecture por sus siglas en inglés o arquitectura orientada a los servicios) puede ser la primera arquitectura pensada a largo plazo, en la cual se asume que la única constante es el cambio.

Esto se deriva de los conceptos de la reutilización y la interoperabilidad, y de cómo éstos forman parte fundamental de la integración de aplicaciones por medio de servicios Web.

El aspecto principal que se considera de una SOA es su flexibilidad, dado que esto lo hace considerarla la primera arquitectura pensada a largo plazo que toma en cuenta el paradigma que: "la única constante es el cambio". Dado que SOA es en esencia básicamente un conjunto de servicios acoplados débilmente, los cuales se pueden reemplazar debido a que puede construirse nuevamente de forma relativamente barata. El acoplamiento débil hace posible que se adapten cambios en la arquitectura y así poder adecuarlos a los tradicionales; sólidos pero inestables.

En una arquitectura SOA, se puede reemplazar un servicio por otro similar sin tomar en cuenta la utilización de la misma tecnología, ya que lo importante es que se mantengan las interfases, y éstas interfaces están definidas por estándares que son universales para la creación/consumo de servicios.

Ésta flexibilidad implícitamente significa que se debe tener la capacidad de potenciar los activos existentes, los sistemas legados y las bases de

datos y así poder unir todas las soluciones de la empresa extendiéndolas dentro del SOA, en lugar ser reemplazadas.

Esto resulta en la habilidad de poder evolucionar en una forma rápida y eficiente. En otras palabras, significa adaptarse en forma orgánica según las demandas o necesidades del negocio, esto es realmente lo nuevo.

Como segundo aspecto se puede pensar sobre la importancia del negocio, ya que SOA es la expresión en de un nivel en cual se plasman de forma significativa las necesidades del negocio. Esto es clave, ya que una función principal de las tecnologías de la información es alinear y colaborar en los procesos de negocio.

La utilización de aplicaciones Web con un SOA genera una ventaja competitiva debido a la integración con otras aplicaciones por medio de los servicios Web.

El diseñar con una arquitectura orientada a los servicios SOA en mente provee la habilidad de integrar sistemas de forma más rápida, así como organizar procesos con base en la disponibilidad de ciertos que pueden ser publicados o consumidos, de continuar con la tendencia de cambio hacia una arquitectura SOA, se tendrá que tener en mente una nueva forma de desarrollar con dicha arquitectura en mente. Construir aplicaciones pensando en SOA requiere cambiar el paradigma de resolver problemas de forma aislada. La reutilización de código también forma parte de SOA, por lo tanto los desarrolladores la deberán tener en mente al momento de crear las aplicaciones.

Las características en los procesos de creación de arquitecturas para los negocios y el advenimiento de SOA, ha promovido que los "parthners" puedan colaborar mutuamente en la creación de procesos vinculados a sus negocios por medio de las tecnologías de la información. Esto ha venido a crear nuevos retos alentando a las compañías a lograr alcanzarlas, planteando así la interacción de aplicaciones informáticas realizadas en distintas plataformas que garanticen una integración e interoperabilidad con otros sistemas informáticos, y esto podrá ser logrado solamente en aquellas donde el asociarse con otras compañías es valorado basándose en las cualidades una arquitectura.

# 2. ARQUITECTURA ORIENTADA A LOS SERVICIOS SOA

## 2.1. Evolución de SOA

Los servicios de una arquitectura SOA son visibles al consumidor del servicio y sus componentes subyacentes son transparentes. Mientras que para el proveedor del servicio, el diseño de componentes, su exposición y la administración reflejan la arquitectura y las decisiones del diseño que permiten hacer servicios en SOA.

Realizar estas decisiones requiere de una comprensión de los componentes de SOA, el modelar una SOA implica identificar, clasificar, especificar, y tener una visión de la estructura de los servicios y componentes de SOA.

Una breve revisión del desarrollo orientado a objetos (OO) y de CBD (desarrollo basado en componentes) puede ayudar a entender la evolución de SOA

El mundo de objetos comenzó con la introducción de lenguajes de programación orientados a objetos y se afirmo al modelar técnicas más nuevas para el diseño, después con la maduración de los métodos de análisis y diseño para (OOAD object oriented analisys and design). Los servicios de infraestructura, las herramientas, las plataformas de desarrollo, los patrones, y las arquitecturas de referencia siguieron, el desarrollo basado en componentes (CBD component-based development) sobrevino, ofreciendo un nuevo acercamiento al diseño, la construcción, la puesta en práctica, y la evolución de las aplicaciones de software.

Las aplicaciones eran montadas usando componentes de una variedad de fuentes y los componentes se escribían en diversos lenguajes de programación y diversos ambientes de desarrollo, y corrían en diversas plataformas. Tal y como con el desarrollo de orientado a objetos (OO), la infraestructura de los servicios, las herramientas, las plataformas de desarrollo, y los patrones maduraron para hacer CBD (desarrollo basado en componentes) una realidad.

OO y CBD fueron pensados para crear software para economías de escala en los cuales se requería la reutilización en el desarrollo del software. El logro de estas economías de escala ya sea puesto en producción o en el desarrollo de aplicaciones de software fue evasivo.

SOA, con algunas mejoras sobre el desarrollo de OO y de CBD, proporciona una oportunidad incluso mayor para la reutilización. SOA se convierte en un nuevo mandato para alcanzar economías de escala en la ingeniería de software.

La evolución de la arquitectura orientada a los servicios SOA, la proliferación del Internet y los esfuerzos por una estandarización de XML condujo a la necesidad de definir y describir lo que es un servicio publicado por un proveedor de servicio, dicho servicio puede ser localizado e invocado por un solicitante de servicio. La descripción del servicio puede ser implementada por N cantidad de abastecedores o proveedores de servicio, en la cual cada uno puede ofrecer distintas cualidades de servicio, basados en requisitos técnicos según el área de disponibilidad, funcionamiento, escalabilidad y seguridad. La invocación de dichos servicios puede ser a través de Internet o intranet en una connotación distribuida y estándares Web como WSDL (Lenguaje de descripción de servicios Web) y SOAP (protocolo simple de acceso a objetos).

Los estándares y las tecnologías para servicios Web sirven como base para dar pie a la arquitectura orientada a los servicios SOA. La infraestructura de servicios en áreas tales como administración de servicios, seguridad de servicios, es requerida para que las economías de escala hagan posible que SOA se convierta en una realidad.

SOA es el diseño de una arquitectura de libre unión de servicios para permitir que las aplicaciones expongan su lógica del negocio y que esta sea flexible, permitiendo así la interoperabilidad. Por lo tanto, SOA consiste de un sistema compuesto por servicios del negocio alineados de tal manera que estos apoyan una realización de final-a-final end-to-end de forma flexible y dinámicamente re configurable gracias a la utilización de interfaces basadas en descripciones para dichos servicios. La ventaja inherente a esta forma de comunicación o por descripción de servicios, es el des emparejamiento del proveedor del servicio como del solicitante del servicio, gracias a los estándares y protocolos abiertos. Por otra parte, colecciones o servicios individuales pueden ser combinados para producir “nuevos” servicios compuestos por otros servicios, aprovechando así la reutilización.

Por lo tanto el desarrollo orientado a los servicios que SOA permite, se podría decir que es una evolución del desarrollo basado en componentes y el desarrollo orientado a objetos. Sin embargo, esta evolución solo puede ser potente con la fusión adecuada de los negocios y la tecnología de la información, y que esto se vea reflejado en la flexibilidad de enlazar a un consumidor de un servicio con un proveedor de servicio, tomando en cuenta las necesidades de calidad de servicio, la estabilidad de las interacciones entre estos y la descripción del servicio así como una invocación adecuada con

protocolos y estándares previamente definidos para la arquitectura, basados en la flexibilidad.

SOA promueve el acercamiento entre los negocios y la tecnología de la información, basándose en la reutilización de servicios, haciendo hincapié en la flexibilidad que permita modelar una comunicación final-a-final end-to-end en la implementación de procesos de negocio.

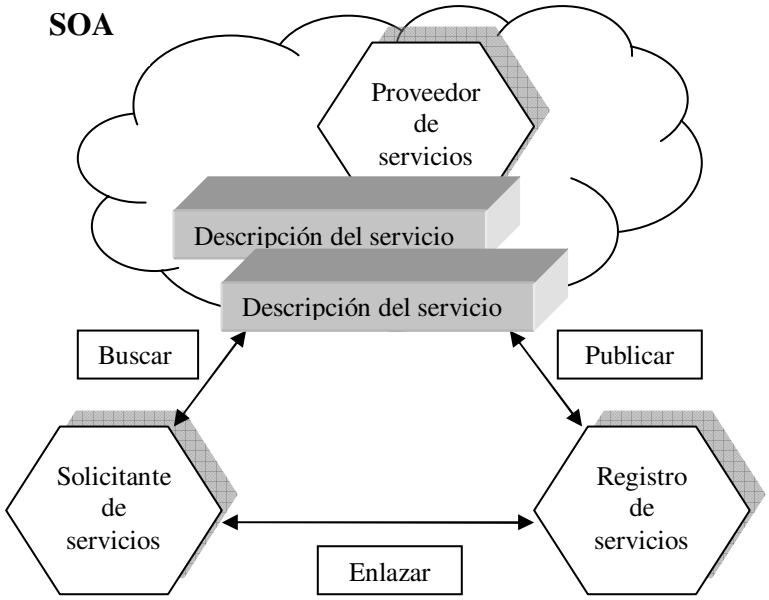
## 2.2. Descripción de SOA

SOA es una metodología de diseño, la cual está orientada a maximizar la reutilización de servicios de las aplicaciones para aumentar la adaptabilidad y eficacia de estas.

En SOA las funcionalidades objetivo del software se representan como servicios, proporcionando así una manera estándar en que las aplicaciones de software puedan interactuar, dicha representación a través de servicios permite la reusabilidad de estos en otras aplicaciones.

Entonces, se podría decir que SOA no es mas que un concepto en el cual la clave esta en la interacción de los servicios, por lo tanto, SOA está compuesto por varios actores, los cuales se ilustran en la imagen de abajo.

**Figura 1. Actores de una arquitectura SOA**




Entonces, toda arquitectura orientada a los servicios tendrá en su forma básica un esquema como el indicado arriba, el cual está compuesto básicamente por un proveedor de servicio, un solicitante de servicio y un registro de servicios y algunas acciones entre ellos.

A continuación se da una breve explicación de los actores de una arquitectura orientada a los servicios, y las acciones asociadas a cada uno de ellos.

**2.2.1. Proveedor de servicio:** Se denomina así a aquel que tiene la responsabilidad de crear una descripción del servicio que se pondrá a disposición para consumir, se deberá publicar la descripción del servicio en uno o más registros de servicios, dicho servicio deberá permitir sea invocado por uno o más solicitantes.

**2.2.2. Solicitante de servicio:** es aquel que tiene la responsabilidad de buscar/encontrar la descripción del servicio que desea consumir ya sea de uno o más registros de servicios. Es responsable de utilizar adecuadamente lo descrito por servicio para enlazar o invocar el servicio que desea consumir.

**2.2.3. Registro de servicios:** éste es el responsable de anunciar las descripciones de servicios Web publicadas por los proveedores de servicios, también debe permitir a los solicitantes de servicios buscar en la colección de descripción de servicios contenidos en el registro de servicios. Una vez encontrada la información, el servicio del registro no es necesario y el resto de la interacción se da directamente entre el solicitante del servicio y el proveedor de servicio[8]

La arquitectura orientada a los servicios SOA, provee de argumentos para hacer frente a la interoperabilidad e integración de aplicaciones, que son aspectos clave que se deben afrontar en las organizaciones de la tecnología de la información IT que desarrolla soluciones de software para Internet, es por tal razón que la arquitectura orientada a los servicios basadas en estándares de servicios Web ha emergido como una solución para afrontar dichos aspectos clave.

## 2.3. Componentes de una SOA

Los componentes de una arquitectura SOA están relacionados directamente con los actores y las acciones que estos realizan.

[8] Basado en la definición tomada de http://fisica.uson.mx

### 2.3.1. Transporte

El componente del transporte representa los protocolos y formatos basados en estándares que son utilizados para la comunicación adecuada con un servicio. El formato de datos establece y especifica los tipos de datos y los formatos de corriente de bytes que suelen codificar datos dentro de mensajes. El protocolo de "cable" especifica el mecanismo usado para el empaquetado de datos codificados en los mensajes. El protocolo de transferencia especifica la semántica de aplicación que controla la transferencia de los mensajes, el protocolo de transferencia realiza la transferencia real.

### 2.3.2. Descripción

El componente de descripción representa el lenguaje en el que el servicio es descrito, dicha descripción provee de la información necesaria para realizar el enlace adecuado con un servicio. El lenguaje proporciona el medio para indicar el contrato de un servicio, dicho contrato incluye las operaciones o funciones que el servicio realiza, los parámetros y los tipos para dichos parámetros, así como el mensaje de respuesta.

### 2.3.3. Descubrimiento

El componente de descubrimiento, representa los mecanismos para registrar, anunciar y encontrar un servicio con su descripción respectiva. Estos mecanismos de descubrimiento pueden ser realizados ya sea en tiempo de compilación y/o de ejecución.

## 2.4. Patrones de diseño

### 2.4.1. ¿Qué es un patrón de diseño?

Es un documento escrito el cual describe una solución de forma general para un problema de diseño que ocurre repetidamente en distintos escenarios o proyectos. Los patrones usan una descripción formal de un problema, su solución, y cualquier factor que pueda afectar dicha solución.

### 2.4.2. Patrones de diseño

Los siguientes patrones de diseño están orientados hacia la reducción del tráfico de mensajes, ya que ésta es la forma en que una arquitectura orientada a los servicios basada en servicios Web se comunica con otras aplicaciones.

#### 2.4.2.1. Patrón proxy

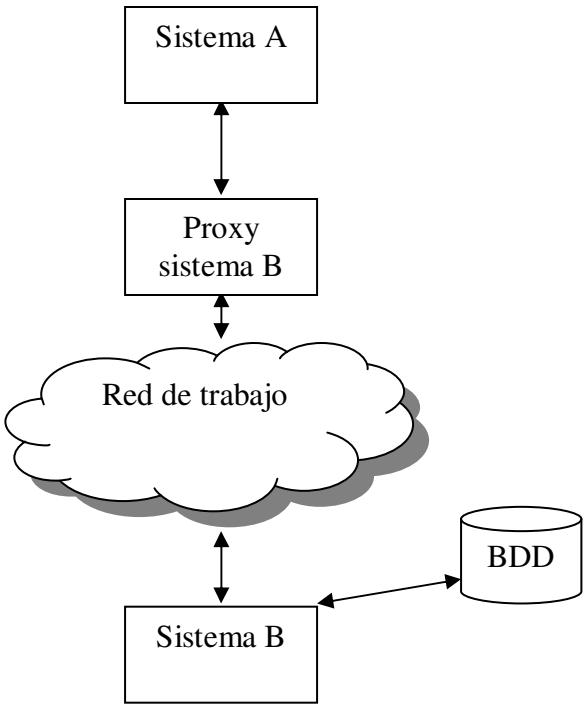
El proveedor de servicio provee de un Proxy o enlace al servicio para el consumidor de éste. El consumidor del servicio ejecuta la petición llamando una función del API en el Proxy, el Proxy encuentra un contrato y una referencia para el proveedor del servicio en el registro, luego ajusta el formato del mensaje de la petición y ejecuta la petición en nombre del consumidor.

El servicio del Proxy es una entidad de conveniencia para el consumidor del servicio. No se requiere que el desarrollador del servicio del consumidor escriba o implemente el software necesario para tener acceso al servicio directamente. El Proxy puede realizar mejoras en el desempeño almacenando en cache (memoria temporal) referencias remotas y datos. Cuando un Proxy almacena en cache una referencia,

las llamadas a los subsecuentes servicios no requerirán adicionales llamadas al registro, almacenando contratos de servicio en forma local el consumidor de servicios reduce el número de saltos en la red requeridos para ejecutar un servicio.

Además, los Proxys pueden mejorar el funcionamiento eliminando llamadas de la red en conjunto realizando algunas funciones localmente. Para los métodos de los servicios que no requieren datos de servicio, todo el método se puede poner en ejecución localmente en el Proxy**.**

**Figura 2 Patrón Proxy**



Entre algunos beneficios del patrón Proxy se pueden mencionar la transparencia de los servicios Web hacia los consumidores, mientras que entre sus desventajas se pueden enumerar las siguientes:

- Sobrecarga en la utilización de red
- Pobre concurrencia
- Alto acoplamiento
- Pobre reutilización
- Pobre mantenimiento

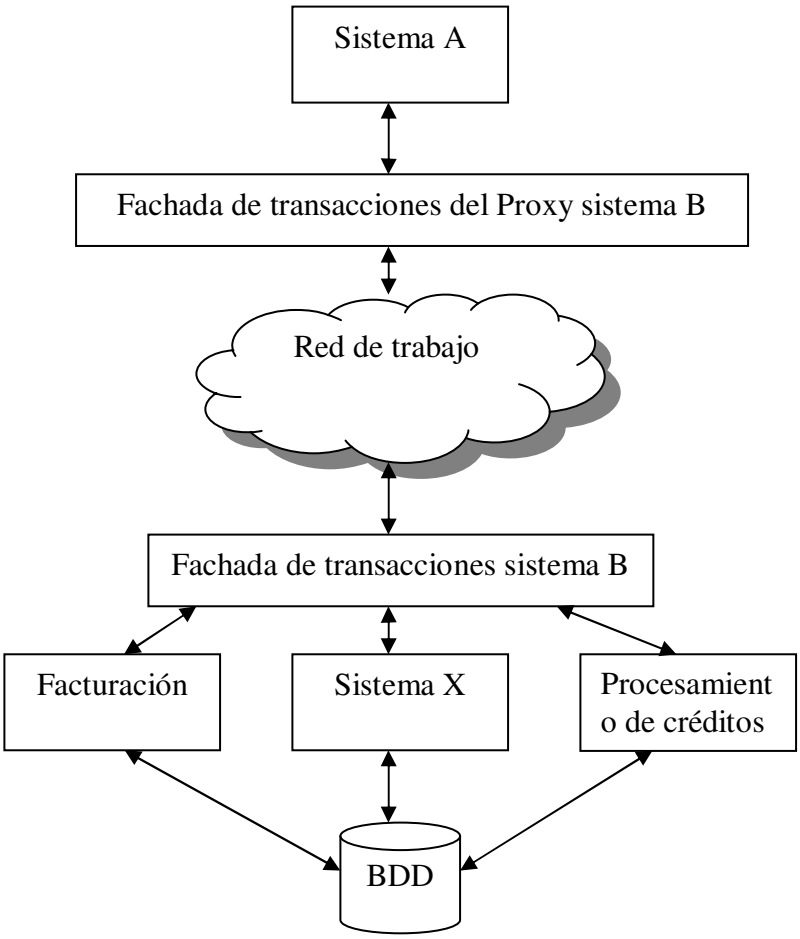
#### 2.4.2.2. Patrón de fachada

Los módulos se deben diseñar desde un punto de vista externo, al momento de diseñar se debe tomar el punto de vista de un usuario del módulo que se diseña. Lo que será expuesto al usuario de este módulo se llama una fachada o la interfaz del servicio.

El patrón del diseño de fachada simplifica y unifica la interfaz de un subsistema coherente y posiblemente autónomo. Una fachada es un punto de entrada simplificado de un API, lo cual hace que un conjunto de clases sea más fácil de utilizar.

Para la implementación de la fachada y su puesta en ejecución, la única cosa que es necesario hacer es agrupar todas las características de un módulo del negocio y exponer a este grupo como una lista de métodos para así no extender esos métodos en muchas clases.

**Figura 3. Patrón de fachada**




Entre algunos de los beneficios del patrón de fachada se pueden mencionar los siguientes:

- Baja sobrecarga de la red de trabajo
- Integridad en las transacciones
- Poco acoplamiento
- Buena reutilización
- Buen mantenimiento

Algunos de los problemas que se pueden presentar con este patrón al ausentarse la fachada de transacciones son:

- Múltiples llamadas en la red
- Alto uso de la red de trabajo
- Pobre desempeño

#### 2.4.2.3. Patrón DTO

DTO (objeto de transferencia de datos) es un patrón en el cual se pueden pasar más de un objeto en una llamada remota, objetos que llevan datos en ellos para así reducir las llamadas remotas a los servicios encapsulando la respuesta de un servicio.

Los objetos de datos contienen solamente datos, no así ninguna lógica del negocio. Se puede pensar en ellos como estructuras de datos simples que transportan datos desde una capa a otra. Por supuesto, esta definición puede adaptarse a ciertas situaciones, a veces se puede asociar comportamientos mínimos a esos objetos.

Los objetos de los datos deben respetar algunas reglas por ejemplo:

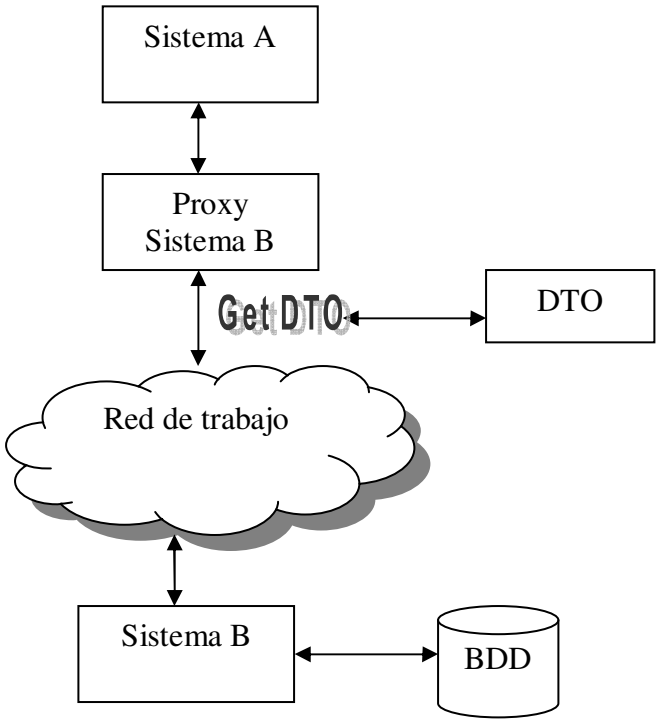
- **Deben ser serializables en XML**, que se puedan representar en un archivo de dicho tipo.
- **Deben ser independientes de cualquier plataforma de implementación** (un objeto de datos proveniente de un servicio que es implementado en JAVA deberá ser invocado o consumido desde .NET como ejemplo de plataformas distintas. La serialización de XML garantiza este punto. En este ejemplo, ambos lados tienen un objeto, respectivamente en el .NET para el

cliente y en Java para el servicio, que es la encarnación del mismo esquema de XML).

- **Deben ser independientes de la fuente u origen de datos**. La mejor manera de cerciorarse de que este punto es respetado es no incluir ningún código persistente en los objetos de datos.

Los objetos de los datos tienen muchos sinónimos, tales como entidad de negocio, objeto del valor u objeto de transferencia de datos (DTO). Recalcando que un DTO contiene solamente una agregación de datos que se transferirán, y no lógica del negocio.

**Figura 4. Patrón DTO**



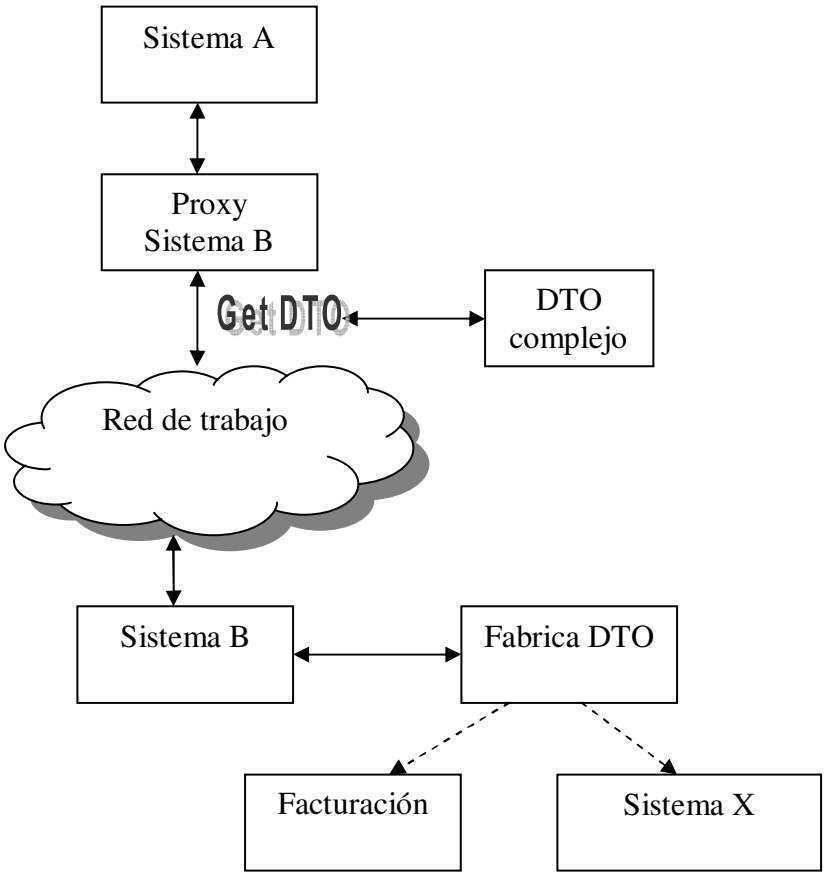
Entre algunos de los beneficios del patrón DTO se pueden mencionar las siguientes:

- Un solo tiempo en transferencia de datos
- Menor sobrecarga de la red de trabajo
- Mejor desempeño

#### 2.4.2.4. Patrón de fábrica

El patrón del método de fábrica delega la responsabilidad de la creación de clases a las subclases externas a la aplicación del cliente, permitiendo al cliente diferir enlazarse a un servicio Web particular hasta el momento de corrida.

**Figura 5. Patrón de fábrica**

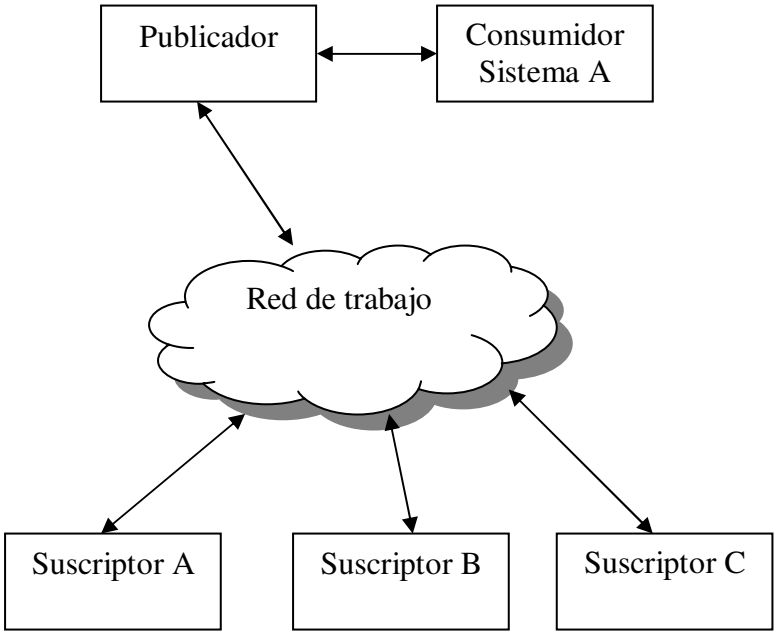



Entre algunos de los beneficios del patrón de fábrica se pueden mencionar, un tiempo de transferencia de datos menor (denotado por la línea punteada en la figura 05), menor sobrecarga de la red de trabajo y un mejor desempeño.

### 2.4.2.5. Patrón publicador/suscriptor

El patrón del Publicador/Suscriptor es donde un mensaje viene en un servicio notificando lo que espera escuchar (como respuesta) para los "mensajes que un publicador difunde a sus consumidores de sus suscriptores". Una aplicación cliente envía un mensaje de "suscripción" acerca de quién es y donde puede recibir estas "respuestas" del publicador. Media vez en la aplicación cliente sea instalado físicamente, se puede correr, y espera que el publicador de servicios genere las respuesta y las envíe de vuelta al consumidor.

**Figura 6. Patrón Publicador/Suscriptor**



## 2.5. Decisiones de diseño y arquitectura de SOA

Algunas de las decisiones de de diseño de la arquitectura son esenciales para que SOA alcance sus metas y ventajas. Las decisiones de arquitectura de SOA se relacionan generalmente con las opciones de las tecnologías, y con los productos necesarios para resolver con la calidad deseada las características de los servicios requeridos por los procesos de negocio. Las decisiones de diseño se concentran en las decisiones que deben ser tomadas acerca de los servicios, los cuales son la clave para una arquitectura orientada a los servicios.

Algunas de las decisiones clave son las siguientes:

Decisiones para el diseño de la identificación de servicios son esenciales para el éxito de SOA. Tal y como la proliferación de objetos no permitió que las organizaciones realizaran economías de escala, dado que la reutilización se limitaba en gran parte debido a la carencia de la utilidad de dichos objetos para los negocios debido a la proliferación de estos, se da el mismo caso para la proliferación de los servicios Web. Muchos que apoyan a SOA y los servicios Web se han dado cuenta que la proliferación de los servicios Web no hace eco para el modelo SOA, debido a que estas opciones de diseño se hacen durante el análisis y el diseño (la orientación a los servicios).

Las decisiones de la realización de la arquitectura y del diseño son críticas para que un servicio proporcione las bases críticas para la extensibilidad y la capacidad de mantenimiento.

Las opciones del diseño de granularidad para los servicios se deben emparejar al nivel de la reutilización y de la flexibilidad requerida dado el contexto, los servicios grandes granulados pueden ayudar a encapsular cambios en los servicios técnicos menos granulados, de tal forma que estos

puedan tender a cambiar con más frecuencia que la interfaz de nivel superior del negocio. La flexibilidad será el criterio dominante para la encapsulación más bien que simplemente la encapsulación de la función.

A continuación se describen de una forma mas concreta los factores de mayor importancia o retos que se deben afrontar en el diseño de una arquitectura orientada a los servicios SOA.

### 2.5.1. Identificación de servicios

Desafío

Identificar correctamente los servicios y determinar los correspondientes proveedores de servicio es el primer paso crítico para realizar el diseño de una arquitectura orientada a los servicios. Actualmente, funciones de negocio similares bien podrían ser proveídas por sistemas múltiples dentro de la empresa.

Acercamiento

Hay dos maneras de tratar este desafío; **racionalización y consolidación** de servicios.

Racionalización de servicios

La racionalización de los servicios implica un análisis cuidadoso de todos los sistemas y usos que proporcionan las funciones del negocio. Con la racionalización de los servicios, la funcionalidad del negocio apoyada por los sistemas menos frecuentemente accedidos puede ser implementada con aquellos que son más frecuentemente accedidos, haciéndolo de esta manera se puede forzar una entrega de servicios en forma mas consistente.

Consolidación de servicios

La consolidación del servicio implica la redefinición de todos los casos del servicio en una versión consolidada que apoye el súper conjunto de todos las interfaces expuestas por los casos individuales. El servicio redefinido y consolidado es proporcionado por todos las aplicaciones individuales en forma constante.

Por ejemplo, en un depósito de catálogos de productos es alcanzado por tres servicios separados. Estos servicios se dedican a recuperar los subconjuntos predefinidos de la información disponible sobre un producto. Para consolidar estos servicios, hay un solo servicio que trabaja con el catálogo entero de los productos. Este servicio contiene todos los segmentos de la información empleados por los servicios individuales antes de la consolidación. Los consumidores trabajan selectivamente con la porción del catálogo que es de su interés. La consolidación del servicio es así una manera eficaz de dinamizar los servicios que apoyan la misma función de negocio.

### 2.5.2. Ubicación de servicios

Desafío

Los servicios funcionan generalmente sobre un sistema específico de las entidades de negocio que son residentes dentro de un sistema dado el registro de estas. Este sistema de registro es una localización ideal para que los servicios se ejecuten. Sin embargo, las soluciones para arquitecturas distribuidas pueden dar lugar a que los datos de negocio se separen a través de múltiples aplicaciones y pueden generar múltiples instancias del sistema de registro para la misma entidad de negocio. La sincronización de los datos entre los dos sistemas se convierte en un requerimiento clave. ¿Dónde debería ser situado o ubicado el servicio en tales escenarios?

Acercamiento

Hay tres maneras de solucionar este desafío; el enrutamiento contenido-basado, enrutamiento de servicios repositorio-basado, y la replicación back-end.

Enrutamiento contenido-basado, este acercamiento encamina las solicitudes entrantes para el servicio al sistema apropiado de registro. Tal solución apoya la transparencia de la localización para los consumidores del servicio: El algoritmo que determina dónde donde un servicio es dado o proveído no tiene que ser expuesto a los consumidores del servicio. Ambos sistemas de registro soportan una instancia del servicio y ambas instancias del servicio sirven como puntos de entrada lógicos para la petición dada.

Enrutamiento de servicios repositorio-basado, es una variación del acercamiento del enrutamiento contenido-basado descrito arriba. En el enrutamiento de servicios repositorio-basado, mientras que el perfil del servicio del cliente ejecuta la misma regla de negocio que hace en el enrutamiento contenido-basado, él levanta la información en el depósito de servicios para dirigir la petición a la región apropiada. Esto hace más fácil cambiar la lógica del enrutamiento, en caso fuera necesario. Las peticiones pueden ser redireccionadas hacia una región diferente simplemente actualizando la información en el repositorio de servicios, sin cambiar las reglas del negocio dentro del servicio del cliente.

Replicación Back-end, este acercamiento promueve las capacidades de las conexiones intrínsecas entre las aplicaciones para poder acceder a la información desde el repositorio físico que contiene la información requerida. Así, ambas instancias del sistema de registro funcionan como un punto de entrada lógico para acceder la información distribuida a lo largo de ambos sistemas. El servicio puede ser ejecutado en cualquier sistema, la

localización física de los datos que son operados es transparente al servicio mismo. El mismo perfil de servicio del cliente se ejecuta en la instancia del sistema de registro que es más cercano al servicio. En caso de que la información de otro repositorio regional se requiera, las capacidades intrínsecas de replicación de los datos y la tecnología detrás del repositorio de datos se emplean para traer los datos relevantes.

### 2.5.3. Definición de dominios de servicios

Desafío

Clasificar servicios en dominios lógicos simplifica la arquitectura, reduciendo el número de componentes que se tratarán. Tales agrupaciones pueden ser apalancadas por múltiples razones arquitectónicas, tales como el balanceo de de la carga, control de acceso, simulación de Proxy y vertical u horizontal partición de la lógica del negocio. Sin embargo, es a menudo un desafío serio para las unidades del negocio y los centros de tecnología dentro de una empresa, el llegar a un consenso en una definición apropiada de los dominios del servicio. ¿Cuál sería un buen agrupamiento lógico de los dominios de servicios?

Acercamiento

Se pueden adoptar acercamientos múltiples para definir dominios de servicio. La tabla I muestra un ejemplo de distribución de las aplicaciones y plataformas a través de diversas unidades de negocio. Este ejemplo será utilizado para definir las características salientes de cada acercamiento presentado en esta sección.

**Tabla I Ejemplo de la distribución de una aplicación.**

| Unidad de negocio | Plataforma de la capa primaria | Aplicación |
|---|---|---|
| Prestamos caseros | UNIX | SAP |
| Banca en línea | Windows | Siebel |
| Centros bancarios | UNIX | PeopleSoft |
| Servicios de seguros | Windows | SAP |
| Prestamos de consumidor | Linux | Oracle |
| Prestamos corporativos | UNIX | IBM DB2 |

Dominios funcionales.

Los dominios funcionales se basan en las funciones del negocio que son servidas por un sistema de servicios. Los propietarios de los procesos de negocio dentro de la empresa son los que mejor situados están para definir y para segregar las funciones del negocio, y por lo tanto los dominios del servicio. Con tales agrupaciones, los propietarios del proceso de negocio para un dominio dado, pueden tener control autónomo de los servicios dentro de ese dominio. Mientras los propietarios del proceso de negocio se aseguren de que los servicios especificados dentro de sus dominios respectivos estén proporcionados al resto de la empresa, se tiene control completo sobre la arquitectura y la implementación de los servicios.

En el ejemplo arriba, hay tres dominios funcionales del servicio: Préstamos, banca y seguros. Los servicios contenidos dentro de estos dominios puede que tengan que ir a través de plataformas distintas y de distintas aplicaciones finales para procesar las peticiones entrantes específicas a sus dominios. Sin embargo, los procesos del negocio servidos por los servicios son similares dentro de un dominio dado, sin importar el la aplicación o la plataforma en los cuales los servicios se están ejecutando.

1. **Préstamos,** el dominio de servicios de préstamos son proporcionados típicamente dentro del contexto de publicar y administrar a los consumidores y a las entidades corporativas. Este dominio de servicios incluye ambos, préstamos de hipoteca para comprar residencias así como préstamos para comprar activos, con excepción de residencias. Los servicios pueden incluir creaciones de préstamo, amortización del préstamo y el cálculo del pago mensual**.**

2. **Banca,** el dominio de servicios de banca, asociados típicamente con la banca a través de múltiples medios como el Internet, ATM, y centros financieros. Entre sus posibles servicios incluye la apertura de cuentas, generación de balance de cuentas, transferencias de fondos entre cuentas.

3. **Seguros,** el dominio de servicios de seguros contiene servicios que son únicos en la industria de seguros. Los servicios posibles incluyen computo de alto nivel, operaciones de búsqueda de historial medico y procesos de demandas.

Dominios basados en tecnología

Los dominios de servicios funcionales que atraviesan plataformas múltiples, plantean el desafío intrínseco de mantener el paso con el estado de cada plataforma tecnológica en cualquier momento dado. Los vendedores tienden a interpretar estándares de la industria de una manera que favorezca a su solución y que fuerce a otras compañías a depender de su arquitectura, hardware y/o software. La especificación de dominios de servicios basados en tecnología permite el uso eficiente y eficaz de las capacidades únicas de esa tecnología.

En el ejemplo dado arriba, los dominios de servicios se podrían clasificar en las plataformas de UNIX, LINUX y WINDOWS. Los servicios de la infraestructura tales como registro de errores, monitoreo de transacciones y captura de eventos son buenos candidatos para tales servicios. Estos son dependientes de la plataforma en que se están ejecutando y son típicamente independientes de los procesos de negocio que guían los dominios de servicio funcionales.

Dominios basados en aplicaciones

El concepto de la integración de aplicaciones entre empresas surgió como una manera en que las compañías eliminen la necesidad de reemplazar sistemas existentes. Hoy, las empresas tienen múltiples aplicaciones que necesiten integrar con el mismo sistema de registro para procesar, empaquetar y presentar la misma información de diversas maneras.

Los dominios basados en aplicaciones permite la agrupación de los servicios proporcionados en un sistema dado. Tal acercamiento facilita la administración y el mantenimiento de los servicios puesto que el sistema subyacente es el mismo para todos los servicios dentro del dominio.

En el ejemplo arriba, SAP, SIEBEL, PeopleSoft, IBM DB2 y ORACLE son buenos candidatos a dominios de servicios basados en aplicaciones. Algunos de los servicios de la muestra que se pueden contener dentro de estos dominios se listan abajo.

**SAP**

- Pago a cuentas
- Cuentas financieras

**PeopleSoft**

- Adición de empleados
- Actualizaciones de compensación

**Oracle**

- Replicación de datos
- Representación de roles basados en cuentas

### 2.5.4. Empaquetado de servicios

Desafío

En una arquitectura orientada a los servicios, los sistemas de una empresa deben exponer su funcionalidad como servicios. Los sistemas construidos para facilitar la integración pueden ayudar a hacer esto de una manera más fácil, en los sistemas principales o "mainframe" con herencia esto es más difícil. Cuando estos sistemas fueron construidos, sirvieron como aplicaciones monolíticos que contenían todas las reglas del negocio y procesaban la lógica implicada. Tal información estaba distribuida a través de los múltiples sistemas ligados entre si.

Una arquitectura orientada a los servicios promueve que los servicios sean autos contenidos o autónomos, sin el conocimiento del contexto de otros servicios. Los programas principales "mainframe" están profundamente interconectados con el contexto y el conocimiento de este ¿Cómo tales programas principales serán re-empaquetados en servicios independientes y autónomos?

Acercamiento

Podemos utilizar un acercamiento en tres pasos para tratar este desafío. El acercamiento implica el definir de las áreas lógicas del negocio dentro de la solución principal “mainframe”, asignar conjuntos de programas a estas áreas del negocio, y entonces diseñar una solución que se junte libremente entre el conjunto de programas. Estos pasos se detallan abajo**:**

**1. Definición del área del negocio**. En este paso, establecemos áreas funcionales del negocio. Podemos utilizar programas llamados mapas y organigramas de procesos para definir estas áreas de negocio. Podemos también definir las áreas por relaciones entre los programas en los sistemas principales. [Los programas sobre sistemas principales “mainframe” tienden a ser ligados a otro. Hay procesos subyacentes comunes del negocio detrás de tales relaciones de programa-a-programa]

**2. Asignación de programas.** Teniendo Identificadas las áreas de negocio, asignamos programas individuales a un área de negocio dada. Talvez sea necesario realizar una re ingeniería para aquellos programas que no se orientan fácilmente a una determinada área de negocio. Tal forma de agrupar los programas se alinea bien con el concepto del dominio de servicios discutido anterior.

**3. Libre unión/integración.** En este punto, aunque los programas hayan asignado a áreas de negocio identificadas, todavía se enlazan entre sí mismos. En este paso final, se reemplaza esa fuerte o marcada relación con un acercamiento o unión en una forma más libre. Para hacerlo así, redefinimos las interfaces del programa principal o “mainframe”, de modo que otras aplicaciones puedan reutilizarlos; los programas proporcionan las mismas salidas y aceptan las mismas entradas en la forma en que fue hecho originalmente. Este proceso de la redefinición proporciona una

oportunidad excelente de asegurarse de que estos programas sirven a la empresa en su totalidad más que el solo uso en el aislamiento y el propósito para el cual fueron creados originalmente. Este acercamiento también encamina a los programas existentes hacia una arquitectura orientada a los servicios, colocándolos como servicios usados por consumidores de servicios externos e internos.

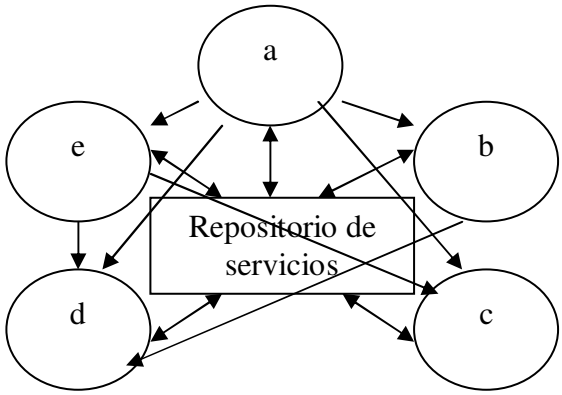
### 2.5.5. Orquestación de servicios

Desafío

Un servicio dado existe porque hay por lo menos un consumidor del servicio, que realiza peticiones a dicho servicio. En algunos escenarios, sin embargo, un servicio puede que invoque a muchos otros servicios para satisfacer la petición o "request" original del consumidor del servicio. Los escenarios simples implican un servicio dado que amplía la petición original a unos o más servicios. Sin embargo, escenarios complejos pueden implicar la invocación recurrente de servicios múltiples y, en algunos casos extremos, invocación interdependiente de servicios múltiples, que podrían dar lugar a un callejón sin salida o ínter bloqueo entre estos.

Por ejemplo, para comprar/vender un boleto aéreo, se necesita que los siguiente servicios sean ejecutados a) buscar cliente, b) obtener precios, c) chequear disponibilidad, d) chequear horario, e) recibir pago, construir en cada servicio la capacidad de orquestarse u organizarse puede resultar en un escenario complejo, ilustrado en la figura 2.1.

**Figura 7. Escenario de servicios sin orquestación.**



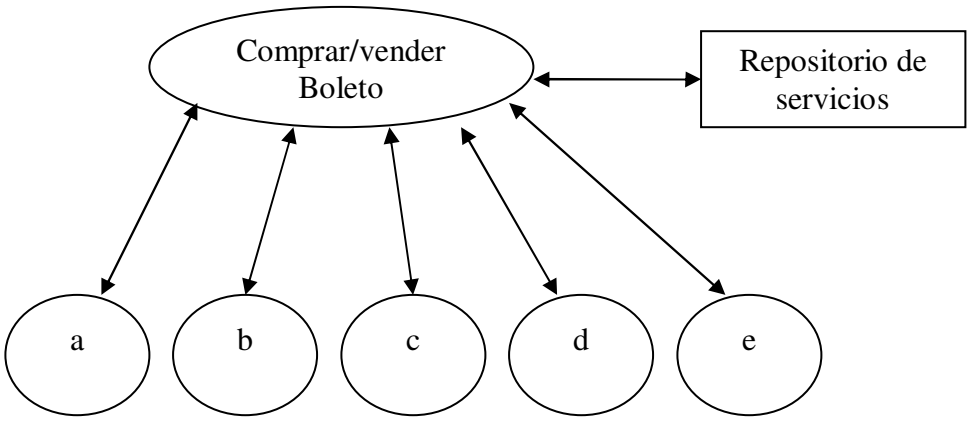

¿Cómo debería ser orquestada tal composición de servicios?

Acercamiento a la administración del proceso del negocio

En este acercamiento se mantiene el servicio individual en su forma simple, los servicios no tienen la capacidad de orquestarse u organizarse para realizar la invocación del resto de servicios requeridos para satisfacer las peticiones.

Por lo tanto, dicha capacidad y lógica debe ser colocada dentro de la capa de procesos del negocio. Los procesos del negocio son los responsables de invocar a cada servicio requerido, de tal modo proporcionando así el servicio compuesto al consumidor que lo solicitó originalmente. Entonces, los procesos del negocio se convierten en un caso especializado de un servicio compuesto.

**Figura 8. Escenario con servicios orquestados.**




En la figura 8 se ilustra el proceso de negocio de la compra/venta de un boleto, dicho proceso contiene la lógica procesal de los pasos individuales que se ejecutarán. El proceso del negocio de la compra/venta del boleto descubre los servicios constituidos para dicho proceso con un solo acceso al repositorio de servicios y orquestra posteriormente los pasos apropiados en la secuencia adecuada.

### 2.5.6. Ruteo de servicios

Desafío

Las arquitecturas orientadas a los servicios deben proporcionar transparencia de localización a los consumidores de los servicios, los consumidores de servicios tienen que enviar una petición o “request” a cualquier servicio situado en cualquier dominio de servicios. Al mismo tiempo, acceder al repositorio de servicios antes de cada invocación de un servicio puede ser un proceso intensivo, por lo tanto ¿Cómo puede SOA proporcionar transparencia de ubicación y al mismo tiempo también aseguran niveles de desempeño aceptables del sistema?

Acercamiento

Se puede solucionar el desafío del ruteo o encaminamiento de servicios de dos maneras.

Servicios inteligentes

Usando este acercamiento, construimos información de la localización para todos los servicios en cada servicio individual. Esto elimina algunos de los saltos requeridos pero resulta en la sobrecarga del servicio. Dependiendo de la frecuencia en la cual los servicios o sus localizaciones experimentan cambios, este acercamiento puede generar mantenimiento intensivo. Además, tal acercamiento no se apega con la arquitectura de libre unión de los servicios. Sin embargo, apoya una solución de alto rendimiento.

Enrutadores

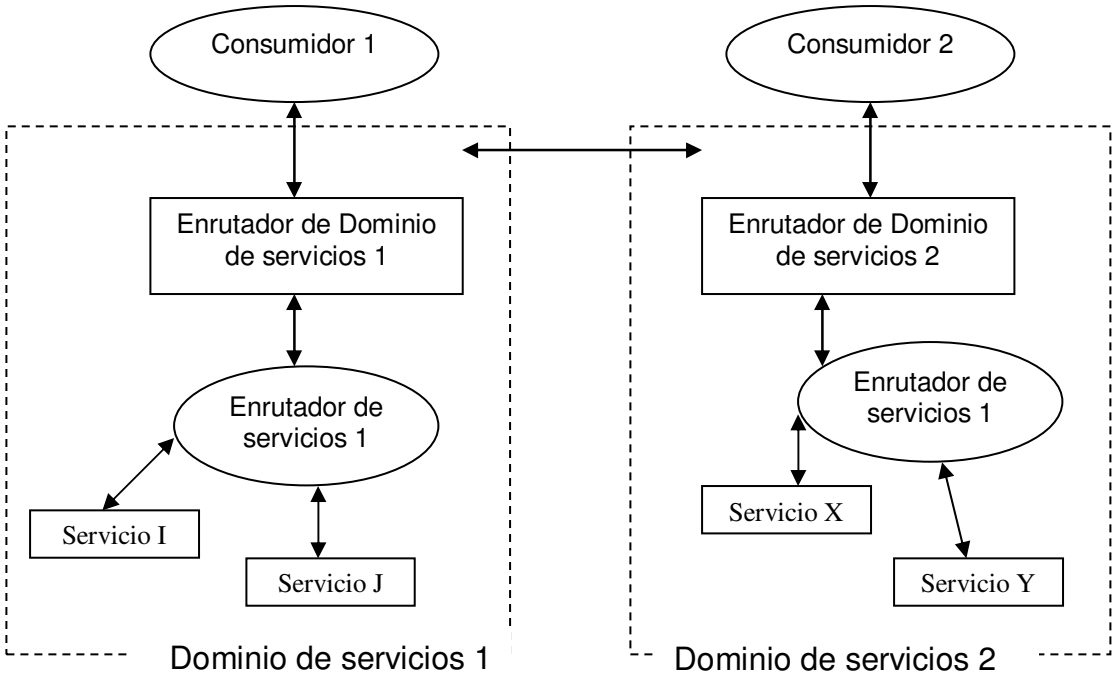
La otra forma de realizar esto es colocar la lógica del enrutamiento de los servicios individuales en un componente de enrutamiento. Este componente de enrutamiento puede estar en dos niveles, ya sea como un dominio de servicios o como un solo servicio.

1. Enrutador de dominio de servicios: un enrutador de dominio de servicios, utiliza de forma inteligente la información que posee acerca de todos los dominios de servicios. Al momento de la recepción de una petición, el enrutador determina, si puede responder a la petición dada, utilizando alguno de los servicios de los cuales el posee información, de ser así, procesa la petición. De lo contrario, pasa dicha petición hacia el dominio de servicios apropiado que pueda atender la petición.

2. Enrutador de servicios: un enrutador de servicios es utilizado dentro de un dominio de servicios para dirigir las peticiones entrantes, al servicio apropiado dentro del dominio. Solamente aquellas peticiones que pueden ser atendidas dentro del dominio de servicios dado son pasadas al enrutador de servicios. El enrutador de servicios reduce la carga de la información acerca de la localización de los servicios individuales.

Los enrutadores de dominios de servicios y los enrutadores de servicios, son más aplicables a los dominios de servicios que contienen un número significativo de servicios. Cuando existen solamente algunos servicios, los servicios inteligentes son una opción más viable. En la figura 9 se ilustra el concepto del enrutamiento en un dominio de servicios y el enrutamiento de servicios.

**Figura 9. Enrutadores de dominio de servicios y enrutadores de servicios.**



### 2.5.7. Gobierno de servicios

Desafío

Sin importar la manera en que los dominios de servicios están definidos dentro de una empresa, hay varios acercamientos filosóficos y técnicos para crear nuevos servicios y modificar servicios existentes. ¿Quién debe supervisar, definir y autorizar los cambios al conjunto existente de servicios dentro de una empresa? ¿Quién debe poseer el aprovisionamiento y el mantenimiento de estos servicios?

Acercamiento

Este desafío se puede tratar de una forma eficaz estableciendo un cuerpo de gobierno interno. Los modelos múltiples de gobierno son posibles. Estos modelos se explican abajo.

Gobierno central.

Con un gobierno central, el cuerpo que gobierna tiene representación de cada dominio de servicios, así como de los partidos independientes que no tienen responsabilidad directa. Debe también haber representación de las diversas unidades de negocio y de los expertos del tema que se pueden comunicar con los componentes tecnológicos claves de la solución. El cuerpo de gobierno central en su totalidad realiza algo más que la simple adición y la eliminación de servicios, así como los cambios a los servicios existentes antes de autorizar su implementación.

Por lo tanto, el gobierno central es responsable de establecer y de hacer cumplir las pautas arquitectónicas orientadas a los servicios, así como los estándares a través de la empresa.

El cuerpo de gobierno central es también responsable de comunicar esos estándares a las unidades de negocio, a los equipos arquitectónicos y a los equipos de la tecnología.

Gobierno distribuido

Con gobierno distribuido, cada unidad de negocio tiene control autónomo sobre cómo proporcionar los servicios dentro de su propia organización. El gobierno distribuido asigna un acercamiento funcional del dominio por mandato del servicio. Un comité de arquitectura de servicios puede proporcionar pautas de alto nivel y los estándares para la implementación, pero ese comité no debe autorizar cambios a la infraestructura existente de servicios dentro de una unidad de negocio.

### 2.5.8. Adopción de estándares

Los estándares de mensajes, obligan a la estandarización basada en un conjunto de elementos de datos y formatos de mensajes. Sin embargo, en un nivel individual de elementos de datos, estos estándares son bastante flexibles, de tal manera que se puedan adaptar en cualquier contexto o área específica de negocio de la empresa. Como resultado deferentes unidades de negocio dentro de la misma empresa pueden conformarse con el mismo estándar en distintas maneras. Adicionalmente, estos estándares proveen la creación de elementos de datos propios.

Dado que la tarea de mantener los estándares no es algo que se realice de forma simple, surge la siguiente interrogante ¿Cómo es posible obligar la adopción de un solo estándar a lo largo de una empresa? como posible respuesta a esta interrogante surge:

Acercamiento al gobierno de la Metadata

Los depósitos de Metadata apoyan la representación consistente de las entidades clave de negocio. Estas representaciones son un súper conjunto de la información, distribuida a través de múltiples sistemas dentro de una empresa u organización. Los diccionarios de datos así como modelos lógicos y físicos de datos son entradas clave en la definición y el mantenimiento de los depósitos del metadata.

El equipo de gobierno de la metadata debe ser un grupo enfocado dentro del modelo central del gobierno explicado anteriormente. El gobierno de Metadata debe ser desempeñado en el nivel de la empresa. Es decir, si una empresa adopta un modelo distribuido del gobierno para el mantenimiento de servicios en general, debe adoptar un modelo central de gobierno para la metadata.

## 2.6. Capas de SOA

Se puede tener una vista abstracta de SOA representada como una arquitectura parcialmente distribuida en capas, capas de servicios compuestos que se alinean con los procesos de negocio. Dicha representación se esboza al final de la sección 2.6

La relación entre servicios y componentes es que los componentes empresariales de escala (componentes grandes o componente pequeños de negocio) realizan los servicios y son responsables de proporcionar su funcionalidad y de mantener su calidad de servicio.

Los flujos de los procesos de negocios se pueden apoyar por una coreografía de estos servicios, expuestos por aplicaciones compuestas. Una arquitectura de integración soporta el ruteo o encaminamiento, la

mediación y la traducción de estos servicios sus componentes y flujos usando una interfaz de servicio empresarial (ESB enterprise service bus). Los servicios desplegados se deben supervisar y administrar para garantizar la calidad del servicio y de la adherencia a los requisitos no funcionales.

### 2.6.1. Sistemas operacionales

La capa de sistemas operacionales, consiste en una capa que esta formada por aplicaciones hechas a la medida ya existentes dentro de la organización, aplicaciones que son utilizadas en ciertas áreas de negocio que tienen valor para la empresa. Dichas aplicaciones o sistemas son conocidos como sistemas heredados (legacy systems), entre estos se pueden mencionar software CRM (Customer realationship manager) existente y paquete de aplicaciones ERP (Enterprise resource proccesing) o aplicaciones antiguas orientadas a objetos, u otra que sea de importancia para la empresa. La estructura en capas de una arquitectura SOA puede apoyar sistemas existentes e integrarlos utilizando técnicas de integración orientadas a los servicios.

### 2.6.2. Componentes empresariales

La capa de componente empresariales, es la capa de los componentes empresariales que son responsables de realizar la funcionalidad de los servicios expuestos, así como de mantener el QoS o calidad de de los servicios (Quality of services) expuestos.

Estos componentes especiales son un sistema manejado y gobernado que representa los objetivos de la empresa y que se fundamentan en el nivel de las unidades de negocio de la empresa, Entonces estos componentes son responsables de asegurar el uso y las mejores prácticas arquitectónicas con respecto a los servicios. Esta

capa utiliza típicamente tecnologías contenido-basadas tales como servidores de aplicaciones para implementar los componentes, administración de la carga de trabajo, asegurar una alta disponibilidad, y el balanceo de la carga.

### 2.6.3. Servicios

La capa de servicios es la capa en la cual residen los servicios que la empresa elige implementar para luego exponer. Estos servicios pueden ser descubiertos dinámicamente, se pueden realizar enlaces con ellos de una forma estática para después ser invocados o posiblemente ser coreografiados en una composición de servicios. Esta capa de exposición de los servicios también provee del mecanismo para tomar otros componentes de utilidad para la empresa, componentes específicos de alguna unidad de negocio, en algunos casos componentes de proyectos específicos, para exponer así un subconjunto de sus interfaces en forma de descripciones de servicios.

Por lo tanto, los componentes de la empresa proporcionan la realización de los servicios en tiempo de corrida, utilizando la funcionalidad proporcionada por sus interfaces. Las interfaces entonces son exportadas hacia fuera como descripciones de los servicios en esta capa, donde estos servicios son expuestos para un posterior uso. Dichos servicios pueden existir en el aislamiento o como servicios compuestos.

### 2.6.4. Composición de procesos de negocio

En la capa de composición de procesos de negocio o capa de coreografía, es en la cual las composiciones y las coreografías de los servicios expuestos en la capa 3 son definidas.

Los servicios se unen a través de la orquestación o coreografía con un flujo adecuado, esto para formar un bloque que parezca único y así actuar juntos como una sola aplicación. Estas aplicaciones apoyan casos específicos de uso y procesos de negocio. Como ejemplo de herramientas visuales para la composición del flujo, se pueden mencionar "IBM WebSphere Business Integration Modeler o Websphere Application Developer Integration Edition", estas se pueden utilizar para el diseño del flujo de la aplicación.

Entonces esta capa es una evolución de la composición de servicios hacia flujos o coreografías de servicios para actuar como una aplicación.

### 2.6.5. Presentación o acceso

Aunque esta capa está generalmente fuera de alcance para las discusiones alrededor de un arquitectura SOA, gradualmente está llegando a ser más relevante. Lo represento aquí porque hay una convergencia de aumento de estándares, tales como servicios Web para la telefonía móvil y otras tecnologías, que buscan apalancamiento en los servicios Web y en las interfaces de las aplicaciones o a nivel de presentación.

Entonces se puede pensar en esta capa como una futura capa, la cual se necesitara tomar en cuenta para futuras soluciones, además es importante notar que SOA separa la interfaz de usuario de los componentes, y que se necesitará proveer de una solución punto a punto end-to-end desde un canal de acceso a un servicio o a una composición de servicios**.**

### 2.6.6. Integración

Esta capa permite la integración de servicios a través de la introducción de un sistema de capacidades confiables, tales como encaminamiento o enrutamiento inteligente, mediación de protocolo, y otros mecanismos de la transformación, descritos a menudo como el ESB enterprise service bus.

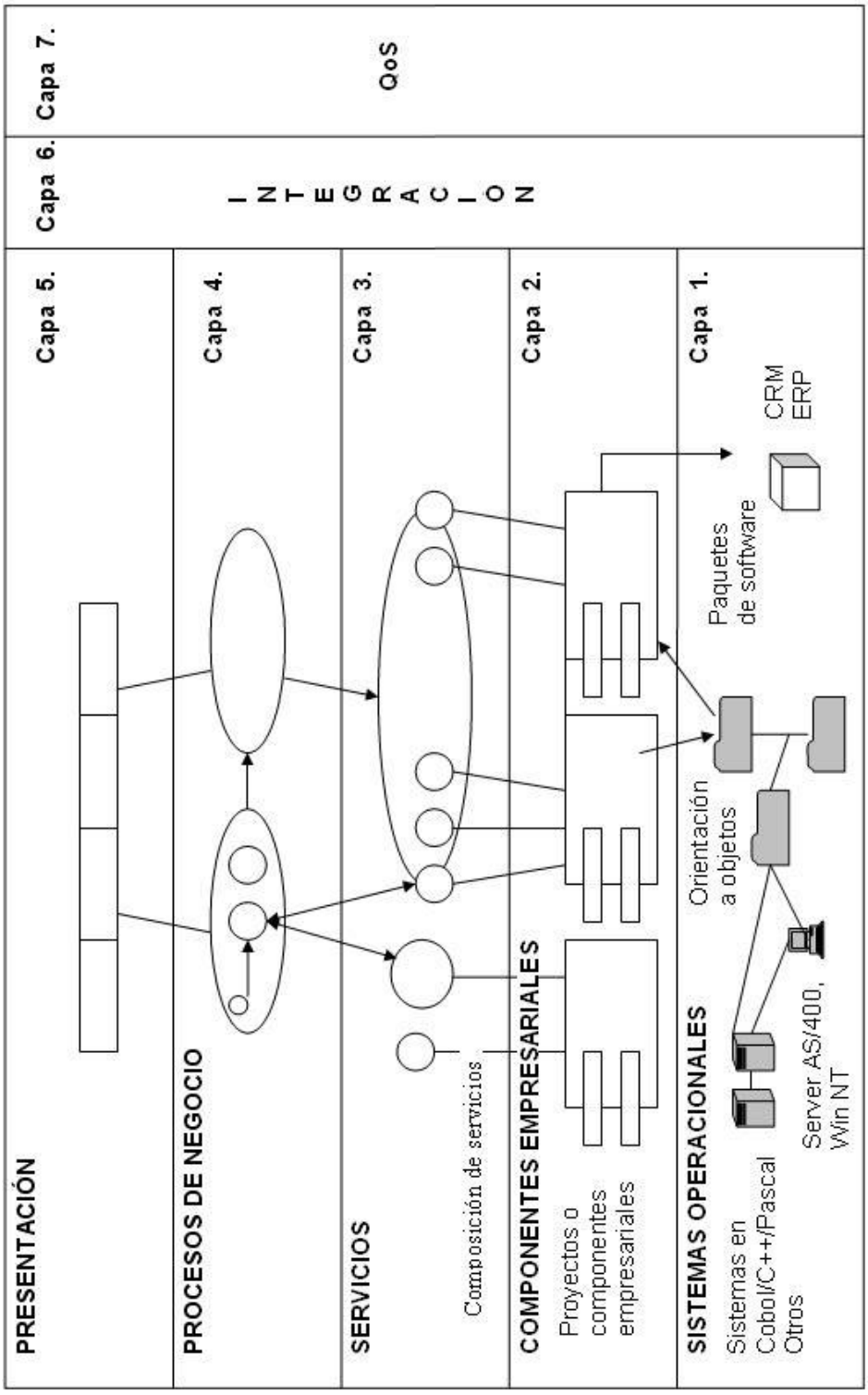
El lenguaje para la descripción de servicios por sus siglas en inglés WSDL especifica un enlace, el cual incluye una localización en donde se proporciona el servicio. Por otra parte un ESB proporciona un mecanismo independiente de la localización para la integración

### 2.6.7. Representación de las capas SOA

A continuación se bosqueja una imagen lo como las capas explicadas anteriormente se unen para formar una arquitectura orientada a los servicios SOA, se adiciona una capa mas denominada QoS.

La capa siete QoS (Calidad de servicios) asegura la calidad de los servicios a través de mecanismos para detectar y responder a funcionamientos no adecuados por medio de herramientas para monitorear el desempeño de las aplicaciones SOA, incluyendo todos los estándares de administración de servicios Web.

**Figura 10.Capas de una arquitectura orientada a los servicios.**



# 3. SERVICIOS WEB Y SOA

La arquitectura orientada a los servicios y los servicios Web se han vuelto parte del lenguaje diario para las tecnologías de la información en lo que se refiere al desarrollo de aplicaciones para Internet, de a poco ésta idea esta siendo aceptada como un estilo arquitectónico para la creación de aplicaciones o soluciones empresariales. En tal contexto, lo que da soporte a dicha arquitectura como lo son los servicios, en éste caso los servicios Web, se han convertido altamente críticos para asegurar una implementación satisfactoria de una arquitectura orientada a los servicios.

## 3.1. ¿Qué es un servicio web?

No se ha llegado a un consenso formal para una definición de lo que es un servicio Web, sin embargo existe una concepción amplia, cada compañía vinculada al desarrollo y creación de infraestructura paral la construcción de servicios Web, tales como IBM, Microsoft, o Sun Microsystems tienen su propia definición.

Una definición funcional de un servicio Web es: un servicio Web es un componente de software independiente de plataforma e implementación y que posee las siguientes características:

- Descrito usando un lenguaje de descripción de servicio.
- Publicado en un registro de servicios.
- Descubierto a través de un mecanismo estándar
- Invocado a través de un API declarado, usualmente sobre una red.
- Compuesto con o por otros servicio.

Cabe recalcar que no obligatoriamente los servicios Web deben estar publicados sobre Internet, ya que dicho servicio puede estar publicado sobre una red de computadoras no necesariamente pública como una Intranet (tipo de red local para una empresa); inclusive los servicios pueden ser consumidos/invocados desde un proceso interno de un mismo sistema operativo.

Otro punto que cabe recalcar y que es de suma importancia, es que la implementación y publicación de los servicios Web no son de importancia para quien los consume, ya que el servicio Web se encuentra publicado por medio de una interfaz API para el usuario final, API que provee del mecanismo para su consumo e invocación.

## 3.2. Pilas de interoperabilidad de servicios web

En la actualidad se pueden encontrar muchas tecnologías vinculadas a los servicios Web, tales como XML, SOAP, WSDL, UDDI, y otros, los mencionados anteriormente se detallan en la sección 3.5.

En la propuesta enviada por IBM y Microsoft al W3C en Marzo de 2001, se proponen organizar los servicios Web en 3 pilas:

- La pila del cable
- La pila de descripción
- La pila de descubrimiento [9]

### 3.2.1. La pila del cable

Ésta pila es la representación en la cual están involucradas las tecnologías encargadas de enviar los mensajes del solicitante de un servicio hacia el proveedor del servicio.

[9] Física México, http://www.fisica.uson.mx/carlos/WebServices/WSOverview.htm , Segundo trimestre 2005

Ésta pila se basa principalmente en los protocolos de red. Protocolos que sirven de base para los servicios Web, tales como HTTP, HTTPS, SMTP, FTP, y otros, conocidos como protocolos de "Cable" de la Internet.

Para el intercambio de mensajes se utiliza XML, para especificar los datos embedidos se utilizan Esquemas basados en de XML. Estos esquemas se pueden configurar de una forma específica o pueden ser regidos por un conjunto de reglas o sintaxis previamente establecida tal y como lo hace SOAP.

Por encima de las capas utilizadas como protocolos de Red y cifrado de los datos se encuentra la capa donde residen los mensajes XML. La envoltura de estos mensajes que se intercambian en formato XML es realizada por SOAP.

Existe un concepto por encima de la envoltura de SOAP, éste concepto provee del mecanismo para extender la envoltura de SOAP denominándose encabezado de SOAP.

**Tabla II Pila de interoperabilidad de cable**

<table>
<tr><td>Encabezados de SOAP</td><td>Extensiones de la Envoltura</td><td rowspan="4">Seguridad</td><td rowspan="4">Manejo</td><td rowspan="4">Calidad de Servicio</td></tr>
<tr><td>SOAP</td><td>Mensajes XML</td></tr>
<tr><td>XML y SOAP</td><td>Codificación de Datos</td></tr>
<tr><td>HTTP(S), SMTP, FTP, etc.</td><td>Protocolos de Red</td></tr>
</table>

### 3.2.2. La pila de descripción

La pila de la descripción es el elemento clave en una arquitectura orientada a los servicios, ya que esta provee información propia del servicio que se esta publicando. Debido a que se deben describir claramente todas las características de los servicios se requieren de varios niveles que describan cada aspecto importante de éste para que sea consumido de la mejor forma por quién lo solicite.

La descripción de los servicios Web, se basa en el XML, y la definición de los datos contenidos dentro de esta descripción se forman a partir de un esquema que rige sin importar la tecnología el contenido del servicio.

Los niveles que siguen en la pila de interoperabilidad son los de "implementación de servicios" e "interfaz de servicios". Estos son descritos con WSDL (Lenguaje de descripción de Servicios Web). Dado que los lenguajes de definición de interfaces o "IDL" no son suficientes., se debe indicar por encima del WSDL, información adicional utilizando WSEL (Web Service Endpoint Language), que se encuentra en desarrollo (IBM).[10]

Por encima en la pila de descripción del servicio, se deberá pensar en la capa de integración u orquestación de los servicios. A grandes rasgos no se puede decir que no es más que la unión de varios servicios relacionados entre sí que finalmente son acoplados en uno solo. Para poder realizar esto se han propuesto estándares como el WSFL (Web Service Flow Language) de IBM y el Xlang de Microsoft.[11]

[10] Física México, http://www.fisica.uson.mx/carlos/WebServices/WSOverview.htm , Segundo trimestre 2005
[11] Ver nota No.11 línea superior.

**Tabla III Pila de descripción**

| | |
|---|---|
| **WSFL/XLANG** | Orquestación de Servicios |
| **WSEL** | Descripción de Punta |
| **WSDL** | Interfaz de Servicios |
| **WSDL** | Implementación del Servicios |
| **Esquema XML** | XML |

### 3.2.3. La pila de descubrimiento

Para descubrir un servicio Web es necesario tener acceso a éstos por medio de un directorio conocido o utilizando un navegador que sea capaz de buscar en los repositorios dónde los proveedores publican sus servicios Web para que sean consumidos.

**Tabla IV Pila de descubrimiento**

| | |
|---|---|
| **UDDI** | Directorio |
| **ADS/DISCO** | Inspección |

El nivel más alto de la pila representa la “inspección”, la inspección es un método en el cual conocen previamente las características del servicio, para éste método IBM trabaja con ADS y Microsoft usa .NET DISCO.

El nivel de directorio es el que se encargará de buscar y descubrir servicios que estén vinculados a los procesos de negocio de las empresas, tal descubrimiento se hace posible con la utilización del estándar de "Descripción universal de Descubrimiento e Integración" UDDI.

### 3.2.4. Integrando las pilas de interoperabilidad

Dado que no todos los servicios Web no están basados en las tecnologías previamente descritas, no es un pre-requisito para considerarlos como tal. Esto hace que la definición de lo que es un servicio Web se haga mucho más amplia, ya que no es solamente la utilización de las tecnologías lo que lo hará finalmente un servicio si no la forma en que será publicado y consumido.

Lo que si se puede asegurar es que la clave de los servicios Web es la interoperabilidad, ya que este es un factor clave en la creación de aplicaciones con una arquitectura orientada a los servicios.

## 3.3. SOA basadas en servicios web

Los servicios Web son sistemas de software diseñados para apoyar la interacción ínter operable de maquina-a-maquina o servidor a servidor sobre una red. Por tal razón los servicios Web son candidatos óptimos para la creación de arquitectura SOA. Esta interoperabilidad se da gracias a un sistema de estándares abiertos basados en XML, tales como WSDL, SOAP, y UDDI. Estos estándares proporcionan un acercamiento común para definir, publicar, y consumir servicios Web.

Entre las plataformas existentes para poner en desarrollo los servicios Web se pueden mencionar, el paquete 1,5 **(Sun's Java Web Services Developer Pack WSDP 1.5), Java 2 Platform, Enterprise Edition (J2EE) 1.4,** [12]ASP .NET, estas entre otras plataformas se pueden utilizar para desarrollar servicios Web para poner SOA en ejecución.

Estas plataformas deben permitir construir y desplegar servicios Web y acoplarlos con la estructura de las tecnologías de información, en cualquier plataforma de servidores de aplicación. Además deberán proporcionar las herramientas que se necesitan para construir rápidamente, probar, y poner en producción dichos servicios, con la finalidad de que estos sean consumidos por los clientes que tengan aplicaciones que funcionan en plataformas Java, .NET-basadas o no-Java, .NET-basadas, por poner un ejemplo. Entonces las aplicaciones podrán interactuar como clientes ellos mismos de servicios Web, y podrán comunicarse con otros servicios de Web, sin importar la plataforma en que estos se ponen en ejecución

### 3.3.1. Descripción de servicios por esquema

Distinto a otros sistemas anteriores, el modelo de servicios Web no funciona sobre la noción de tipos compartidos entre aplicaciones que requieren implementaciones comunes. Sin embargo, los servicios interactúan recíprocamente basados solamente en contratos (WSDL/BPEL4WS para el comportamiento y procesamiento de mensajes) y los esquemas (WSDL/XSD para la estructura del mensaje). Esto permite al servicio describir la estructura de mensajes que puede enviar y/o recibir. La separación entre la estructura y el comportamiento, simplifica la integración en ambientes heterogéneos

[12] http://java.sun.com/developer/technicalArticles/WebServices/soa/

Además, esta información describe de forma suficiente la interfaz del servicio, de modo que la integración con una aplicación con SOA basada en servicios Web no requiere de un ambiente compartido de ejecución para crear los mensajes en lo que se refiere a su estructura o comportamiento.

La arquitectura orientada a los servicios asume un ambiente completamente distribuido donde es un poco difícil más no imposible, propagar cambios de esquema y/o contrato a todos las partes que han encontrado o que utilizan un servicio. Entonces la orientación a los servicios implica que los contratos y el esquema deben seguir siendo compatibles luego de alguna modificación en estos.

Por esa razón, las tecnologías para describir los contratos y esquemas diseñados para utilizarse en el diseño de arquitecturas orientadas a los servicios permiten más flexibilidad en sus interfaces que otras interfaces tradicionales como en las orientadas a objetos. En detalle, los servicios utilizan características tales como elementos XML, extensiones de esquema y encabezados opcionales SOAP para desarrollar servicios de manera que no entren en conflicto con aplicaciones ya realizadas. Estas características son la clave en la composición de los servicios Web.

### 3.3.2. Compatibilidad de servicios

Los diseños basados en procedimientos y orientados a objetos comparan típicamente la compatibilidad de tipos con la compatibilidad semántica. La orientación a los servicios proporciona un modelo más rico para determinar la compatibilidad. La compatibilidad estructural se basa en contratos (utilizando WSDL) y esquemas (XSD), y ambos pueden ser validados. Por otra parte el advenimiento de las políticas para

servicios Web, provee el análisis automatizado para garantizar y asegurar la compatibilidad entre los servicios. Esto se hace basado en aserciones explícitas de las capacidades y requisitos en forma de declaraciones de políticas de servicios Web.

Usando políticas para servicios Web (WS-Policy), los servicios describen sus capacidades y requisitos en una forma de expresión legible. Esto permite a los servicios seleccionarse entre ellos basados en "como" o "con que calidad" estos entregan sus contratos.

Las aserciones de la política para servicios Web son identificadas por el grupo de nombres globales únicos, cuyo significado es consistente en tiempo y espacio sin importar que aserción de servicio sea aplicada. Las aserciones de la política pueden también tener parámetros que califiquen la interpretación exacta de la aserción.

### 3.3.3. Flexibilidad de servicios

Uno de los conceptos base de la arquitectura orientada a los servicios (SOA) es el enlace flexible de los servicios. Modelos más tradicionales basados en procesos, componentes y objetos atan componentes junto con sus referencias (indicadores). Una SOA se apoya en un descubrimiento de forma más dinámica de las instancias de los servicios, proveídos por una interfaz clara que provee la semántica al servicio que el solicitante espera

En sistemas basados en proceso u orientados a objetos, un solicitante encuentra típicamente un servidor basado en los tipos, él exporta o comparte un espacio de nombre. En un sistema de SOA, los

solicitantes pueden buscar registros tales como UDDI( por sus siglas en ingles Universal Description, Discovery, and Integration) para un servicio.

1. Esto se puede llamar compatibilidad de mensajes con los requerimientos de un solicitante. La compatibilidad puede ocurrir a través de WSDL o mensajes con esquemas XML bien conocidos.

2. Entonces, esto apoya las características del servicio que el solicitante requiere. Por ejemplo, el solicitante puede desear características como de seguridad o transacciones.

El acoplamiento débil con respecto a la implementación de servicios que permite realizar implementaciones alternativas de comportamiento, puede ser utilizado para tratar una gran gama de requerimientos del negocio. Por ejemplo, las implementaciones alternativas pudieran corresponder a vendedores (plataformas .NET J2EE etc.) alternos en una cadena de abastecimiento permitiendo así una respuesta más rápida a las condiciones de mercado cambiante. Semejante a la implementación alternativa pudiera estar distribuida geográficamente permitiendo así una tolerancia a fallos o desastres.

## 3.4. Especificación de servicios web

En esta sección se pretende dar una descripción de las especificaciones para los servicios Web.

### 3.4.1. Transporte y mensajes

Por ejemplo, si alguien le envía una carta escrita en francés pero usted estaba esperando una llamada telefónica en inglés, no se podrá realizar la comunicación de ninguna manera debido a que la telefonía y mensajería son formas distintas de comunicación además del lenguaje

indicado en cada uno de estos francés-inglés también es diferente. Entonces la interoperabilidad de de los servicios Web enfrenta el mismo problema, esto se trata proporcionando un sistema de tecnologías comunes para el transporte y mensajes**.**

Por otra parte, para asegurar que estas tecnologías sean efectivas en la práctica, IBM, Microsoft, y otros crearon la organización de la interoperabilidad de los servicios Web o "WS-I". Recientemente la WS-I (http://www.ws-i.org) lanzo un perfil básico que documenta formalmente los mecanismos para la los servicios Web ínter operables en transporte y mensajes.

#### 3.4.1.1. Transporte

El transporte incluye protocolos como http o https (protocolo de transferencia de hipertexto), SMPT (protocolo simple de transferencia de correos), este conjunto de especificaciones define los mecanismos de comunicación base para movilizar la información entre los servicios Web.

El HTTP, HTTPS, y SMPT son ejemplos en este grupo. Las implementaciones de servicios Web adicionalmente pueden soportar otros protocolos de transferencia, pero es crítico proporcionar estándares y soporte para estos protocolos, para garantizar una interoperabilidad eficiente y eficaz**.**

#### 3.4.1.2. Mensajes

En estas especificaciones se definen los mecanismos ínter-operables para la codificación de los mensajes de los servicios Web que serán transportados. La capa de transporte mueve bloques de

"bytes" entre los servicios. Esto es útil solamente si los participantes pueden convertir los “bytes” en estructuras de datos que las aplicaciones puedan procesar.

El grupo de especificaciones de mensajes define como dar un formato correctamente a los mensajes. El XML y las definiciones de esquemas de XML proporcionan el mecanismo para llegar a la aceptación en la estructura de los mensajes (datos). SOAP (Protocolo de acceso simple a objetos) define la codificación estándar para representar mensajes de XML con información en “bytes” que los servicios intercambian sobre la capa de transporte.

#### 3.4.1.3. Direccionamiento de servicios web

Los mensajes y las respuestas ambos van hacia alguna parte y vienen de en alguna parte. El direccionamiento de los servicios Web proporciona una aproximación independiente del transporte e ínter operable, para identificar a los emisores y receptores de los mensajes. El direccionamiento de servicios Web también proporciona una aproximación más detallada para identificar elementos dentro de un servicio que envía o recibe un mensaje.

La mayoría de los sistemas que usan servicios Web, codifican el destino para un mensaje de un servicio Web con un URL que es colocado en el nivel de transporte de HTTP. El destino para la respuesta es determinada por la dirección de transporte de retorno. Esta aproximación se basa en modelo básico navegador-servidor de HTTP.

La nueva tendencia es que el origen y el destino de la información no son la parte del mensaje en sí. Esto puede causar varios problemas. La información puede ser extraviada si una conexión de transporte termina (por ejemplo, si la respuesta toma un tiempo largo y los tiempos de conexión se caen “time out”) o si el mensaje es remitido por un intermediario tal como un cortafuego “firewall”.

El direccionamiento de los servicios Web provee un mecanismo para colocar el destino, el origen y otras direcciones importantes, dentro del mensaje del servicio Web, en poco tiempo se espera que el direccionamiento de servicios Web separe esta información relevante para el de cualquier capa y de cualquier modelo de transporte.

El direccionamiento de servicios Web además permite a un remitente indicar a donde una respuesta debe entrar de manera transporte-independiente. La respuesta a un mensaje podría no ir necesariamente al remitente. En el HTTP por ejemplo, sin la dirección del servicio Web es imposible especificar que la respuesta se debe enviar a alguna parte.

Estos realces al modelo de los mensajes permiten a los servicios Web ser utilizados para apoyar muchos escenarios de negocio. Por ejemplo, ciertas tareas de las actividades bancarias requieren la revisión humana para la aprobación en ciertos pasos. Hay generalmente muchas instancias activas de las tareas en cualquier punto en el tiempo. El direccionamiento de los servicios Web proporciona un mecanismo general para asociar mensajes entrantes o de salida a tareas específicas.

El mecanismo que el servicio utiliza es transparente para aquellos que usan el servicio por medio de una referencia de punto final.

### 3.4.2. Descripción

Las especificaciones del transporte y de los mensajes permiten que los servicios Web se comuniquen con mensajes. ¿Pero cómo los participantes saben cuáles son los mensajes? ¿Cómo un servicio Web documenta o describe los mensajes que envía y que recibe? Utilizar un servicio Web requiere una comprensión de los mensajes que el servicio Web consumirá, que producirá finalmente y la interfaz para el servicio. Entonces las especificaciones de la descripción de un servicio Web permiten a un servicio expresar su interfaz y capacidades.

Además de interoperabilidad de los mensajes, estas especificaciones también permiten interoperabilidad entre herramientas de desarrollo. Las especificaciones de la descripción proporcionan un modelo estándar que permite que diversas herramientas de diversos vendedores apoyen el desarrollo de los programadores. De la misma manera que los servicios Web aíslan a socios de la implementación y la infraestructura, las especificaciones de la descripción para los servicios Web aíslan a los socios de opciones de las herramientas de desarrollo.

En la descripción de servicios Web el idioma descriptivo de los servicios Web (WSDL) y el esquema de XML (XSD) son las especificaciones base en este grupo. El esquema de XML permite que los desarrolladores y los proveedores de servicios definan los tipos de XML para las estructuras de datos que se utilizaran en el intercambio de información. WSDL permite que un servicio Web documente los mensajes que recibe y envía. Es decir las "acciones" o las "funciones"

que el servicio realiza en los términos de los mensajes que recibe y envía.

WSDL proporciona el soporte para una gama interacción de patrones de mensajes. Soporta mensajes unidireccionales o de una vía de entrada que no tienen ninguna respuesta, request/response, y una forma envío con o sin una respuesta. Los últimos dos patrones permiten a un servicio especificar otros servicios lo que el primero necesite.

#### 3.4.2.1. Políticas de servicios web

Las definiciones de WSDL y de XSD no proporcionan a menudo bastante información para llamar un servicio Web. WSDL y XSD definen la sintaxis de la interfaz del servicio pero no expresan la información sobre cómo el servicio entrega su interfaz o lo que espera el servicio del solicitante. ¿Por ejemplo, el servicio requiere seguridad o pone transacciones en ejecución?

Las políticas de servicios Web permiten a un servicio especificar lo que espera de los solicitantes y cómo implementar su interfaz. También es crítico alcanzar la interoperabilidad funcional del servicio en un alto nivel. La seguridad, las transacciones, los mensajes confiables y otras especificaciones requieren esquemas concretos. Éstos permiten que los servicios describan el aseguramiento funcional que esperan de él y proporcionarlo a los solicitantes.

### 3.4.3. Políticas de seguridad

Los servicios han generado mucho entusiasmo en parte debido a su capacidad de funcionar como un puente sobre sistemas dispares con una necesidad de interactuar. Los desarrolladores han producido muchas soluciones completamente funcionales usando las capacidades bases del transporte, los mensajes y la descripción. Sin embargo, para ser aceptados por los desarrolladores que crean soluciones de integración más poderosas, los servicios Web deben proporcionar funcionalidad para asegurar el mismo nivel de aseguramiento de los servicios proporcionados por soluciones más tradicionales. No es suficiente el simple intercambio de mensajes. Las aplicaciones y los servicios residen en los sistemas que proporcionan funciones de alto nivel valiosas, tales como seguridad, confiabilidad, y operaciones transaccionales. Los servicios Web deben proporcionar un mecanismo para la interoperabilidad entre estas funciones.

#### 3.4.3.1. Seguridad en servicios web

La seguridad es el bloque de construcción básico para los servicios Web seguros. Hoy, la mayoría de distribuciones de los servicios Web confían en la ayuda del nivel de transporte para las funciones de seguridad. Los ejemplos son http o https y autentificación básica Estos acercamientos a la seguridad proporcionan el mínimo necesario para la comunicación segura. El nivel de función que proporcionan, sin embargo es menos que lo proporcionado por el middleware existente y los ambientes distribuidos.

La Seguridad en los servicios Web utiliza los modelos existentes de seguridad (Kerberos, X509, etc.). Las especificaciones en concreto definen cómo utilizar los modelos existentes de una manera ínter operable. Los servicios Web no pueden ser seguros sin políticas de seguridad para servicios Web.

#### 3.4.3.2. Confianza de servicios web

La seguridad recae en relaciones predefinidas de confianza. Kerberos funciona porque los participantes "confían en" el centro de distribución de llaves del Kerberos. PKI funciona porque los participantes confían en las autoridades de certificación de la raíz o root. La confianza de los servicios Web define un modelo extensible para la creación y verificación de relaciones de confianza.

El concepto clave es un servicio simbólico de la seguridad (STS security token service). Un STS es un servicio distinguido en la Web pública, intercambia y valida símbolos “tokens” de la seguridad. La confianza de los servicios Web permite a éstos modificar y aceptar en qué servidores de seguridad "confiar".

#### 3.4.3.3. Conversación segura de servicios web

Algunos escenarios de servicios Web involucran solamente un pequeño y esporádico intercambio de algunos mensajes. La seguridad de los servicios Web apoya fácilmente este modelo. Otros escenarios implican una duración larga, conversaciones de multi-mensajes entre los servicios. La seguridad de los servicios Web también soporta este modelo, pero la solución no es óptima.

Hay dos óptimas utilizaciones de la seguridad de los servicios Web en estos escenarios:

1.- Utilización repetida de operaciones criptográficas de un costo de cómputo elevado tales como validación de llave pública.

2.- Envío y recepción de muchos mensajes usando las mismas llaves criptográficas, proporcionando información que pudiera permitir ataques de fuerza bruta "rompan el código".

Por estas razones, protocolos como HTTP/S utilizan llaves públicas para realizar una negociación simple que define llaves específicas de conversación. Este intercambio de llaves permite una implementación de seguridad más eficiente y además disminuye la cantidad de información cifrada con un conjunto específico de llaves.

Las conversaciones seguras en los servicios Web proveen un apoyo similar al de la seguridad en la navegación segura. Los participantes a menudo utilizan la seguridad de SW con llaves publicas para empezar una “conversación” o “sesión”, y utilizan conversaciones seguras de SW para llegar a un acuerdo sobre las llaves específicas inicio de sesión y para cifrar información**.**

## 3.5. Protocolos y estándares de comunicación para servicios web

Los protocolos y estándares explicados a continuación son los más relevantes para los servicios Web, dichos protocolos y estándares fueron utilizados a lo largo de las secciones anteriores en este capítulo, por lo cual se amplía la explicación para estos.

### 3.5.1. HTTP

El HTTP (protocolo de transferencia de hipertexto) es el método primario utilizado para transportar la información sobre el World Wide Web o “WWW”. El propósito original de este protocolo era proporcionar una manera de publicar y de recibir las páginas del HTML.

El desarrollo del HTTP fue coordinado por el consorcio de “www” (World Wide Web Consortium) y por Internet Engineering Task Force, culminando en la publicación de una serie documentos RFCs con las especificaciones para este protocolo , el más notable posiblemente el RFC 2616, que define HTTP/1.1, la versión del HTTP mas común en uso hoy.

El HTTP es un protocolo de petición/respuesta o (request/response) entre los clientes y los servidores. Un cliente de HTTP, tal como un navegador Web, típicamente inicia una petición estableciendo una conexión TCP/IP a un puerto particular en un anfitrión alejado (puerto 80 por defecto). Un servidor de HTTP que escucha en ese puerto espera al cliente envíen una cadena “string” de petición, tal como lo haría "Get/HTTP/1.1" (que solicitaría la página por defecto de ese servidor Web), seguido por un correo como un mensaje del MIME que tiene un número de encabezados de información, que describen aspectos como la petición , seguido de un cuerpo opcional de datos arbitrarios, algunos encabezados son opcionales, mientras otros, mientras que otros (tales como servidor anfitrión) son requeridos por el protocolo HTTP/1.1.

Sobre la recepción de la cadena de petición (y del mensaje, si existiera), el servidor envía de vuelta una cadena o “string” de respuesta, tal como "200 ACEPTADA", y un mensaje propio, el cuerpo de lo que es quizá el archivo solicitado, un mensaje de error u otra información.

El HTTP a diferencia de otros protocolos basados en TCP tales como FTP (protocolo de transferencia de archivos), en el que las conexiones son terminadas una vez una petición particular (o una serie relacionada de peticiones) han sido completadas. Este diseño hace el HTTP ideal para el World Wide Web, donde las páginas regularmente se ligan a páginas en otros servidores. Esto puede plantear de vez en cuando problemas para los diseñadores Web, pues la carencia de una conexión persistente hace necesarios métodos alternativos de mantener la “sesión” de los usuarios. Muchos de estos métodos implican el uso de galletas “cookies” para mantener asociado el usuario a una sesión determinada.

HTTPS es la versión segura del HTTP, usando SSL/TLS para proteger el tráfico. El protocolo utiliza normalmente el puerto TCP 443. El SSL, creado originalmente para proteger el HTTP, es especialmente utilizado en HTTP puesto que puede proporcionar cierta protección incluso si solamente un lado de la comunicación, el servidor, esta autenticado. Éste es el caso típico en transacciones de HTTP sobre el Internet.

### 3.5.2. XML

EL XML o lenguaje de marcado extensible (Extensible Markup Language) es un lenguaje recomendado por el consorcio de la triple w W3C para crear lenguajes especiales de marcado. El XML es un subconjunto de SGML, capaz de describir distintos tipos de datos, su

propósito principal es el de facilitar el compartir texto estructurado e información a través de Internet. Lenguajes basados en XML (como RDF, RSS, MATHML, SIL y SVG) se describen a ellos mismos de una manera formal gracias al XML, permitiendo a los programas modificar y validar documentos en estos lenguajes sin conocimiento anterior de su forma.

XML como todo lenguaje tiene ciertas características que lo hacer ser un lenguaje fuerte y otras que se pueden considerar como debilidades

Algunas de las características que hacen de XML un excelente lenguaje para la transferencia de datos son las siguientes:

- Tiene simultáneamente un formato entendible por humanos y por maquinas.
- Soporta unicote, permitiendo comunicar casi toda información en cualquier lenguaje humano.
- Su habilidad para representar la mayoría de las estructuras de datos de las ciencias computacionales (registros, listas, árboles etc.)
- Su formato auto documentable que describe su estructura, tipos de datos, nombres de campos así como valores específicos.
- Su sintaxis estricta y los requerimientos de parseo que permiten que los algoritmos para validar su sintaxis se mantengan simples, eficientes y consistentes.

Para ciertas aplicaciones, XML puede presentar las siguientes debilidades.

- Su sintaxis es bastantes verbosa y parcialmente redundante. Esto puede afectar la lectura humana y la eficiencia de las aplicaciones, esto genera costos de almacenamiento altos. Además el uso de XML puede afectar donde el ancho de banda es limitado, aunque con compresión de datos se puede reducir notablemente este problema. Este problema es particularmente cierto para aplicaciones multimedia corriendo en teléfonos celulares y PDAs que utilizan XML para describir imágenes y video.
- La sintaxis contiene características no necesarias, las cuales son heredadas de su compatibilidad con SGML.
- Los requisitos básicos del análisis no soportan una gama amplia de tipos de datos, así que el análisis a veces implica trabajo adicional para extraer los datos deseados de un documento XML.
- No facilita el acceso aleatorio o actualización de solo una parte de los datos del documento.
- Mapear XML a un paradigma relacional u orientado a objetos es a menudo incomodo.

### 3.5.3. SOAP

SOAP es un estándar para intercambiar mensajes sobre una red de computadoras, mensajes basados en XML usando normalmente como transporte HTTP. SOAP es la capa fundamental de la pila para los servicios Web, proporcionando un marco de trabajo básico para los mensajes.

Existen diversos tipos de patrones de mensajería en SOAP, pero en gran medida el más común es el patrón de RPC llamada remota a procedimiento, donde un nodo de la red (el cliente) envía un mensaje de petición a otro nodo (el servidor), y el servidor envía inmediatamente un mensaje de respuesta al cliente.

SOAP fue originalmente un acrónimo para protocolo de acceso simple a objetos (Simple Object Access Protocol), pero esto fue botado o descartado en la versión 1.2 de especificaciones de SOAP. Las especificaciones de SOAP son actualmente mantenidas por el XML Protocol Working Group que pertenece al consorcio de la triple w "W3C".

### 3.5.4. WSDL

WSDL o lenguaje de descripción de servicios Web es un formato de XML publicado para describir servicios de Web. La versión V 1,1 no ha sido aprobada por el consorcio de la World Wide Web (W3C), no obstante ahora se está trabajando en una nueva versión 2,0, que será la recomendación (un estándar oficial) del W3C.

WSDL describe la interfaz pública de los servicios Web. Esto es una descripción basada en XML del servicio, de cómo comunicarse con él y de cómo utilizarlo, indicando protocolos y el formato de los mensajes requeridos para interactuar con los servicios listados en el directorio (conjunto de servicios prestados por un servicio Web). Las operaciones soportadas y mensajes son descritos de una forma abstracta, y luego enlazados a una red de trabajo concreta, protocolo y formato para luego ser utilizados.

WSDL es utilizado conjuntamente con SOAP y esquemas XML para proporcionar servicios Web sobre el Internet. Un cliente (programa) que se conecta con el objetivo de consumir algún servicio puede leer el WSDL y así determinar qué servicios tiene publicados el servidor. Cualquier tipo de dato especial usado es embedido en el archivo de WSDL con un formato de esquema XML. El cliente puede entonces utilizar el SOAP para llamar realmente una de las funciones listadas en el WSDL.

### 3.5.5. UDDI

UDDI son las siglas para la descripción universal, descubrimiento e integración (Universal Description, Discovery, and Integration), que es una plataforma independiente, basada en registros XML para negocios en el World Wide Web para listarse en Internet. UDDI es una iniciativa abierta de la industria (promovida por OASIS) que permite a negocios descubrirse y definir cómo interactuar recíprocamente sobre el Internet.

Un registro del negocio de UDDI esta formado por tres componentes principales:

- Páginas blancas – dirección, contrato, e identificadores
- Páginas amarillas – clasificaciones industriales basadas en estándares taxonómicos.
- Páginas verdes – información técnica acerca de los servicios expuestos por el negocio.

Es por eso que UDDI es uno de los estándares medulares para servicios Web. Está diseñado para ser interrogado por los mensajes de SOAP y para proporcionar el acceso a los documentos de WSDL que describen los formatos de los protocolos de enlace y formatos de

mensajes utilizando XML que son requeridos para poder comunicarse con los servicios Web ubicados en un directorio de servicios Web.

# 4. APLICACIONES WEB Y SOA BASADO EN SERVICIOS WEB

En las últimas décadas, los departamentos responsables de manejo de las tecnologías de la información de las empresas han estado construyendo la infraestructura que da soporte en gran medida a los procesos de negocio de las empresas y sus clientes. Alcanzar lo que se ha logrado ha requerido del aprendizaje de los errores y aciertos en la utilización de las tecnologías emergentes, dejando como resultado la existencia de un gran número de aplicaciones con distintas tecnologías que proveen a las empresas de la ejecución de muchas tareas.

Los negocias ahora requieren la creación de aplicaciones de una forma mucho más rápida, con menor presupuesto y de mayor complejidad. Construir dichas aplicaciones en la mayoría de los negocios necesitan reutilizar funcionalidades ya desarrolladas en sistemas anteriores. Cuando se alcanza tal situación se tienen dos soluciones en lo referente a la arquitectura que se desee adoptar.

1.- Reutilizar, Intentar la reutilización de funcionalidades ya desarrolladas en otras aplicaciones. Algo que a la postre se volverá difícil ya que la integración de sistemas se hace difícil si las tecnologías utilizadas para crearlos no fueron diseñadas utilizando estándares que vayan más allá de su plataforma original.

2.- Crear aplicaciones con tecnología nueva, que realicen la misma funcionalidad que se requiere algo así como "reinventar la rueda". Si bien ésta solución infla los tiempos requeridos para desarrollar las nuevas aplicaciones, resulta ser ésta la decisión más segura y menos difícil.

Debido a que la segunda opción resulta ser la más aceptada, se pueden listar algunos de sus efectos en el largo plazo:

- Funciones o procesos del negocio replicados en varias aplicaciones distribuidas en el empresa.
- Se hace difícil migrar las aplicaciones de negocio internas, debido a que las interacciones y dependencias con otros sistemas hacen que al cambiar uno se deban alterar muchos procesos colaterales..
- Derivado de lo anterior, la integración no es clara y puede llegar a ser redundante, la alteración de una aplicación generará muchos puntos de falla y error en otros sistemas.

Con aplicaciones que tengan las características mencionadas anteriormente no se puede pensar en la escalabilidad, por lo que finalmente las aplicaciones tendrán una respuesta lenta y no adecuada a los cambios en las tecnologías.

Por tales razones, el diseñar aplicaciones con una arquitectura orientada a los servicios se vuelve fundamental, ya que esta nos garantiza la interoperabilidad, escalabilidad e integración entre distintos sistemas ya que es una arquitectura pensada en el cambio.

## 4.1. Combinación de aplicaciones y SOA con servicios web

Una estrategia de aplicaciones empresariales debe facilitar la integración de todos las aplicaciones de las empresas, y debe encaminar el desarrollo de aplicaciones basadas en servicios. La creación de estos servicios es con el objetivo de exponer a otros sistemas la ejecución de procesos bien definidos, pensando en que aquellos sistemas que consuman dichos servicios solamente deberán orquestarlos de la forma que mejor se apegue a los procesos que estén realizando,

La exposición de los procesos de negocio como servicios es el factor fundamental de una SOA al momento de pensar en la flexibilidad, proveyendo así la posibilidad de incorporar de forma transparente estos servicios en otras aplicaciones ayudando a cumplir con su funcionalidad. Dando como resulta una Arquitectura SOA en la cual están claramente definidos los servicios que se orquestarán y las tecnologías involucradas en su desarrollo y utilización. Por lo tanto, las aplicaciones Web son ideales para utilizar todos estos conceptos, ya que estas son las que catapultan a los negocios a realizar todas sus operaciones vía Internet con clientes, proveedores y otros actores que sean de relevancia para alcanzar los objetivos de las empresas.

## 4.2. Diseño de servicios web con SOA en mente

Un servicio debe ser totalmente autónomo y por ende toda su funcionalidad debe estar contenida dentro de él para garantizar esa característica, no por tal razón se convierte en un componente aislado ya que existe una interfaz hacia él que garantiza su utilización y una comunicación adecuada con quien lo invoque.

La comunicación por lo tanto debe ser bi-direccional, en ambas direcciones hacia fuera del servicio o hacia adentro ésta es realizada utilizando mensajes. Estos mensajes deben contener toda la información relevante para la utilización adecuada posterior del servicio y así reducir al mínimo las llamadas entre el servicio y la aplicación que lo consume.

Un servicio es la versión más actual, compleja y evolucionada de lo que antes denominaba un “componente distribuido”, pero se diferencia de éste por las siguientes características:

- Menor nivel de acoplamiento respecto de los sistemas que los utilizan.
- Las formas de utilizar el servicio son más transparentes.
- Son diseñados pensando en la reutilización en sistemas que cumplen distintos propósitos, indistintamente de las tecnologías utilizadas.
- Son controlados y administrados de manera independiente.
- Utilización de protocolos de comunicación más abiertos y no vinculados forzosamente con tecnologías exclusivas de una empresa.
- La comunicación es transparente, basándose en sus tecnologías y protocolos de comunicación. Dando como resultado una base para la escalabilidad y tolerancia/recuperación de fallas.

La implementación ideal de un servicio exige resolver algunos inconvenientes técnicos inherentes a su modelo:

- Los tiempos de llamado no son despreciables, gracias a la comunicación de la red, tamaño de los mensajes, etc. Esto necesariamente implica la utilización de mensajería confiable.
- La respuesta del servicio es afectada directamente por aspectos externos como problemas en la red, configuración, etc. Estos deben ser tomados en cuenta en el diseño, desarrollándose mecanismos alternos que eviten tener congeladas o en espera a las aplicaciones y servicios que dependen de él.
- Debe tener la capacidad de poder manejar comunicaciones no seguras, mensajes inesperados, reintentos y mensajes fuera de secuencia, etc.

Según lo anterior, se puede ver que la construcción de un servicio es una tarea mucho mas complicada que la de un simple componente distribuido.

El servicio debe publicar una interfaz (utilizando WSDL) fácilmente localizable en la red. Esta interfaz debe servir como un contrato de servicio. Donde se describen cada una de las funciones que provee, e incluso los niveles de prestación de servicio (SLA). Esta interfaz debe estar claramente documentada de manera que sea muy fácil implementar una conexión.

Un diseño exitoso de una arquitectura basada en servicios debe estar basado en una plataforma de mensajería confiable, que aísle de la implementación funcional muchos de los problemas anteriormente mencionados. Algunas de las responsabilidades de un mecanismo así, incluyen:

- Entrega garantizada de mensajes,
- Enrutamiento de peticiones a un servicio disponible,
- Seguridad del contenido de los mensajes,
- QoS (quality of service) Calidad del Servicio

Entre más de estas características tenga la tecnología elegida, menos problemas tendrá la solución final en operación.

Una comunicación basada en mensajes por lo general implica que no existen sesiones. Por lo tanto, los clientes no guardan estado en el servicio (mayor escalabilidad), y la autenticación se debe dar a nivel de cada mensaje.

Por último, los servicios deben ser diseñados para controlar internamente la transaccionalidad de sus propias operaciones. No es recomendable que una transacción traspase los límites de un servicio.

## 4.3. Casos de estudio de interacción por servicios web

En esta sección se presentan las características de arquitectura, e interacciones entre algunas aplicaciones Web, tales interacciones se llevan a cabo por medio de servicios Web, En algunos casos la integración e interacción de estas aplicaciones es bastante sencilla debido a que la plataforma de desarrollo e implementación es la misma (.NET para ser más específicos), tan bien se ve el caso en el cual las plataformas de desarrollo e implementación son diferentes, es en este caso en el cual WSDL juega un papel importante ya que a través de este se realiza la descripción del servicio y el contrato del servicio Web. La utilización de servicios entre estas aplicaciones Web realza la importancia de crear arquitecturas orientadas a los servicios, ya que éstas garantizan una integración con otras aplicaciones no importando las plataformas tecnológicas utilizadas para la realización de dichos servicios Web.

Los sistemas sobre los cuales se realiza el análisis de interacción por servicios Web se explican a continuación.

Guatecompras, sistema de compras gubernamentales del gobierno de Guatemala, en este sistema se publican todas las adquisiciones para bienes y servicios requeridos por las entidades de gobierno (ministerios, municipalidades etc.) y otras no gubernamentales (ONG's, entidades privadas etc.), así como la sucesión de eventos para dichas compras las cuales pueden finalizar en una adjudicación a uno o N proveedores, o una anulación.

SICOIN sistema de contabilidad integrado, sistema para llevar el control y registro tanto de ingresos como de gastos de las entidades de gobierno.

El tercer sistema pertenece a la SAT, y su inclusión en esta sección es debido a la interacción con Guatecompras ya que por medio de este se realiza el registro de proveedores en Guatecompras, así como la actualización de datos de los proveedores (en este caso los proveedores son personas o entes jurídicos que posean número de identificación tributaria NIT).

### 4.3.1. Arquitectura del sistema Guatecompras

La arquitectura del sistema Guatecompras fue pensada en capas, las cuales tienen como objetivo una separación clara la capa de presentación de datos, otra capa de lógica del negocio, y una para el manejo de la base de datos, estas capas contribuyen a que el sistema sea escalable en el desempeño del mismo y el uso de tecnologías orientadas a componentes.

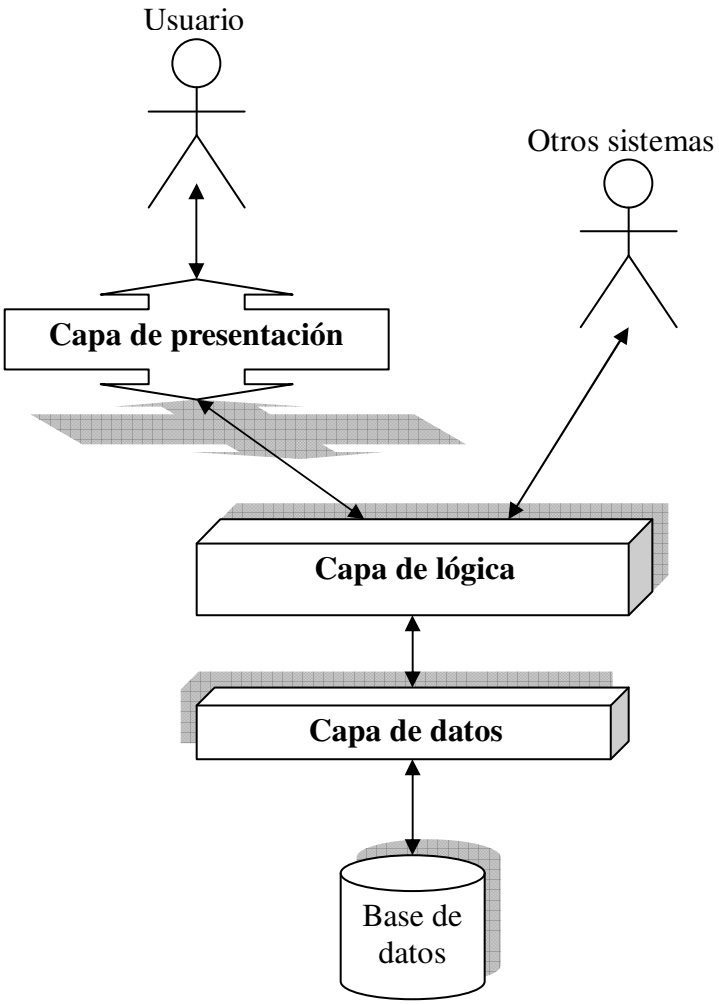
La capa de presentación tiene como objetivo principal la interacción directa con los usuarios del sistema ya que es sólo a través de ésta que los usuarios pueden realizar las acciones sobre el sistema, los usuarios pueden ser desde compradores, contralores, administradores y proveedores (proveedores con NIT previamente registrados vía Bancasat-SAT-Guatecompras)

La capa de lógica de negocio tiene como objetivo principal llevar a cabo las operaciones realizadas por los usuarios que así sean solicitadas por el nivel de presentación, además esta es la capa encargada de la interacción con otros sistemas, cabe mencionar que en esta capa existen

dos niveles, de los cuales uno es para el uso exclusivo de la capa de presentación por lo que nadie más puede interactuar con él y el segundo nivel es para la interacción con sistemas externos, específicamente este nivel sirve para la interacción directa con sistemas como la SAT para la inscripción/actualización de datos de proveedores al registro de proveedores de Guatecompras e incluso con SICOIN y SIGES (Sistema de Gestión de Órdenes de compra). Posteriormente este nivel se ampliara para poder interactuar con otros sistemas como los del IGSS (Instituto Guatemalteco de Seguridad Social) y se estima que también sirva para interactuar con proveedores en alguna etapa posterior que permita crecer a Guatecompras y transformarse y evolucionar en un B2B (Negocio a negocio).

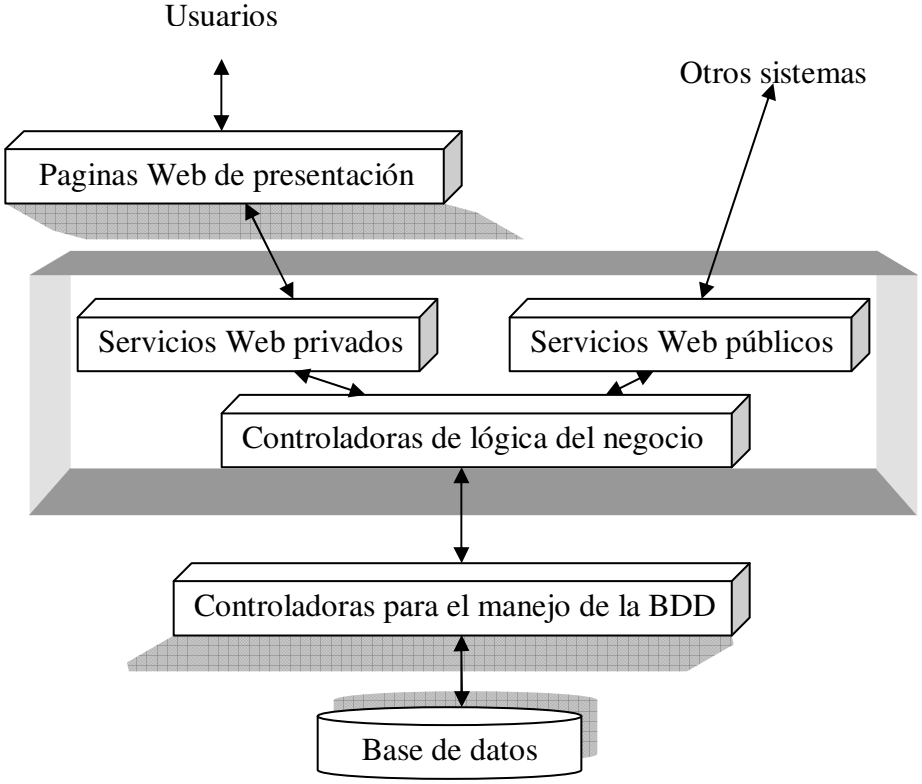
La tercera capa es la que interactúa directamente con la base de datos realizando las consultas, actualizaciones e inserciones requeridas así por la capa de lógica del negocio. A continuación se muestra una imagen que da una visión general de la arquitectura del sistema Guatecompras.

**Figura 11. Vista general de la arquitectura de Guatecompras.**




A continuación se da una vista lógica un poco mas detallada de la arquitectura de Guatecompras, la cual incluye la separación mencionada anteriormente en la capa de lógica del negocio.

**Figura 12. Vista del modelo lógico Arquitectura de Guatecompras.**




Dadas estas dos simples vistas de la arquitectura de Guatecompras se pueden se pueden resaltar algunos aspectos importantes. Dado que esta fuera del alcance de este texto detallar mas otras sub-capas como las de autenticación en la presentación y a nivel de la capa de lógica del negocio, así como el detalle de los tipos de clases controladoras, y otros aspectos técnicos físicos como balanceo de carga, cortafuegos etc.

De la vista del modelo lógico se puede notar una separación importante entre las controladoras de la lógica del negocio y la forma en que estas funcionalidades de la lógica son expuestas o publicadas para el uso de la capa de presentación u otros sistemas, las funciones o

componentes son expuestos como “Servicios Web”, es aquí donde surge la orientación a los servicios por lo que esto recalca la necesidad de exponer los componentes, funciones y operaciones de los sistemas como servicios y que esto debe estar basado en una arquitectura que se preste o sea la ideal para esto, es decir crear aplicaciones con una arquitectura orientada a los servicios ya que esto garantiza un soporte de fondo para la infraestructura de sistemas o aplicaciones Web que prebendan interactuar e integrarse con otros sistemas vía Internet

### 4.3.2. Interacción Guatecompras-Sicoin por medio de servicios web

La interacción Guatecompras-SICOIN se basa en el intercambio de mensajes utilizando XML/SOAP sobre HTTP, es decir por medio de servicios Web. Dicha interacción e integración entre ambos sistemas es de suma importancia para ambos, ya que por medio de estas se realizan operaciones que son de suma importancia en la realización de ciertas actividades para estos, tales como la validación de operaciones en base a la existencia y estatus de un NOG (numero de operación de Guatecompras para identificar compras) que se utiliza para verificar y luego poder solicitar y finalmente pagar a un beneficiario en el sistema de SICOIN, por lo que se requiere validar en Guatecompras el NIT al cual el Sistema de Contabilidad deberá asignar el pago que puede ascender a sumas bastante considerables.

Las plataformas de desarrollo para ambos sistemas es la herramienta .NET de Microsoft, dichas aplicaciones corren sobre servidores con Windows NT 2003, utilizan IIS (Internet Información Server) para atender las solicitudes de paginas y una versión de Framework de .NET versión 1.1 para el caso de SICOIN y versión 2.0 para Guatecompras.

Para llevar a cabo dicha integración entre estos diferentes sistemas, primeramente SICOIN deberá descubrir el servicio Web que este requiere del sistema Guatecompras, luego de ser descubierto este será integrado a la aplicación donde se utilizara el servicio Web de Guatecompras, dicha aplicación podría ser en la capa de presentación o en otro servicio Web perteneciente a SICOIN.

Luego de haber descubierto e integrado el servicio Web de Guatecompras, SICOIN procederá a invocar al servicio Web que le es de utilidad, de esta forma quedaran integrados dichos sistemas y así podrán interactuar por medio del servicios Web a través de solicitudes SOAP desde SICOIN y una respuesta de de Guatecompras a dicha solicitud.

A continuación se muestran una petición y una respuesta SOAP al servicio de consulta para un NOG desde el sistema SICOIN a Guatecompras, cabe hacer mención que toda esta interacción se realiza a través de una red independiente a ambos sistemas como lo es la Internet.

**Figura 13 Mensaje de petición SOAP**

```
<?xml version="1.0" encoding="utf-8" ?>
- <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  - <soap:Body>
    - <ConsultarNOGSICOIN xmlns="http://tempuri.org/">
      - <arrParam>
          <anyType />
          <anyType />
        </arrParam>
      </ConsultarNOGSICOIN>
    </soap:Body>
  </soap:Envelope>
```

Como se puede notar en el mensaje de petición los parámetros que se envían en “arrParam” no se indican para mantener la confidencialidad del uso de dicho servicio Web. El mensaje siguiente es la respuesta si la solicitud de la información es satisfactoria,

**Figura 14 Mensaje de respuesta SOAP**

```
<?xml version="1.0" encoding="utf-8" ?>
- <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  - <soap:Body>
    - <ConsultarNOGSICOINResponse xmlns="http://tempuri.org/">
        <ConsultarNOGSICOINResult>string</ConsultarNOGSICOINResult>
      </ConsultarNOGSICOINResponse>
    </soap:Body>
  </soap:Envelope>
```

### 4.3.3. Interacción Guatecompras-Sat por medio de servicios web

La interacción entre estos sistemas al igual que en la interacción anterior SICOIN-Guatecompras, se da por medio de mensajes utilizando SOAP/XML sobre HTTP, pero la diferencia que cabe mencionar en este caso es que ambos sistemas han sido desarrollados utilizando tecnologías diferentes y ambos sistemas corren sobre sistemas plataformas completamente diferentes.

En este caso se explicará desde el punto de vista de Guatecompras, que es el sistema que utiliza un servicio Web publicado por el sistema de la SAT, dicho servicio Web tiene como objetivo la actualización de datos de los proveedores inscritos en Guatecompras que estén actualizados en la base de datos de SAT, para realizar esto primero se debe acceder a la descripción del servicio Web en el cual aparece la lista de servicios prestados el servicio Web prestado por la SAT, para esto se utiliza la url (dirección Web) del servicio Web,

añadiendo al final los parámetros para solicitar la descripción del servicio Web “?WSDL”, esto dará como resultado la descripción del servicio Web, ilustrado abajo.

**Figura 15 XML de respuesta para descripción un servicio Web**

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Generated by the Oracle9i JDeveloper Web Services WSDL Generator  -->
<!-- Date Created: Mon May 30 16:31:21 CDT 2005  -->
- <definitions name="sat_rtu.ConsultasSIAF" targetNamespace="http://sat_rtu/ConsultasSIAF.wsdl"
    xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://sat_rtu/ConsultasSIAF.wsdl"
    xmlns:ns1="http://sat_rtu/IConsultasSIAFBDD.xsd">
  + <types>
  + <message name="datos_contribuyente0Request">
  + <message name="datos_contribuyente0Response">
  + <message name="datos_comerciales1Request">
  + <message name="datos_comerciales1Response">
  + <message name="datos_facturas2Request">
  + <message name="datos_facturas2Response">
  + <message name="datos_contribuyentes3Request">
  + <message name="datos_contribuyentes3Response">
  + <portType name="ConsultasSIAFBDDPortType">
  + <binding name="ConsultasSIAFBDDBinding" type="tns:ConsultasSIAFBDDPortType">
  + <service name="sat_rtu.ConsultasSIAF">
  </definitions>
```

Luego de que el servicio Web es descubierto, pasa a ser parte de un servicio Web en Guatecompras el cual hace uso del servicio de la SAT. La invocación del servicio Web de la SAT requiere previamente la utilización de un certificado (certificado que es confidencial entre ambos actores) para poder utilizar dicho servicio. Para invocar dicho servicios se envía una petición SOAP indicando el NIT para el cual se desean actualizar los datos, de ser aceptada la petición la SAT devuelve un mensaje SOAP con los datos actualizados para el NIT solicitado.

### 4.3.4. SOA y los casos de estudio

Como se pudo observar en las secciones anteriores la interacción de los sistemas involucrados Guatecompras-SICOIN-SAT se lleva a cabo de servicios Web, dichos servicios deberán estar descritos de una forma que otros sistemas los puedan entender no importando las herramientas y plataformas de desarrollo y de operación para dichos sistemas, no obstante cabe mencionar que la arquitectura del sistema de Guatecompras que es la que se expuso en la sección 4.3.1 no es una arquitectura totalmente orientada a los servicios pues carece de ciertas capas como la de procesos de negocio, ya que en dicha arquitectura ésta no está claramente separada de la capa de Lógica del negocio, aunque esto no se puede considerar como un defecto, ya que la arquitectura de dicho sistema no fue diseñada totalmente orientada a los servicios.

También sobresale que la utilización de SOAP/XML sobre HTTP y otros protocolos, son algunas de las herramientas y estándares fundamentales en lo que a servicios Web se refiere y que dan pie para que los servicios Web sean parte fundamental en el desarrollo de futuras aplicaciones. De tal situación se puede notar que la mejor forma de llegar a soluciones de software que provean de su funcionalidad a otros sistemas es por medio de servicios que deberían estar basadas en una arquitectura orientada a éstos.

## 4.4. Integración con otras aplicaciones

La integración con otros sistemas se ha vuelto de vital importancia en lo que se refiere al desarrollo de soluciones de software, dicha integración debe sobreponerse a las diferencias en las herramientas de desarrollo utilizadas para realizar dichas soluciones, esto trae como resultado la interoperabilidad de sistemas desarrollados y operados por tecnologías distintas. Por lo tanto, la integración resulta ser de suma importancia para

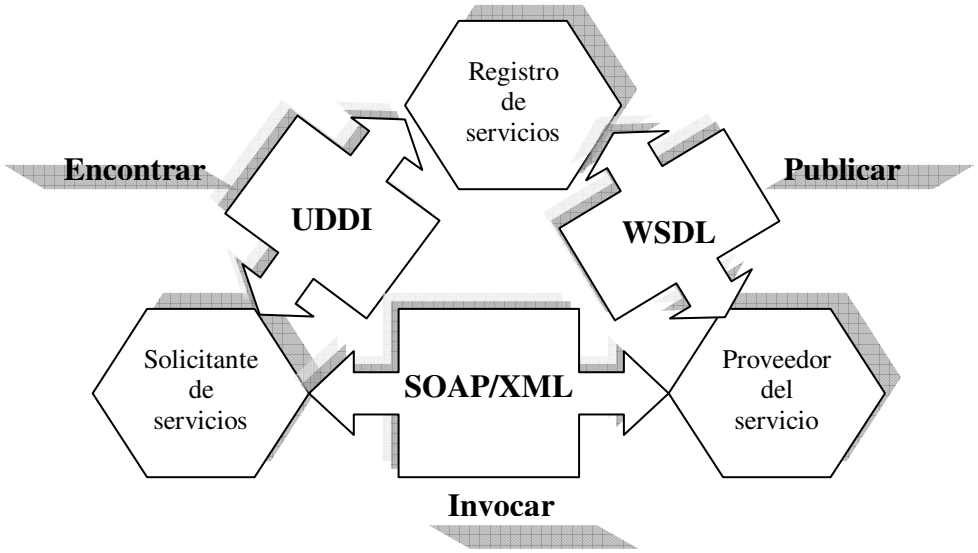
interactuar con múltiples sistemas a través de Internet como lo demuestran los casos de estudio de la sección 4.3. Por tales razones se puede afirmar que SOA surge como un nuevo paradigma en el desarrollo de soluciones de software.

### 4.4.1. Invocación de otros servicios

La invocación de un servicio Web determinado depende en gran parte de la herramienta de desarrollo utilizada para invocar dicho servicio, ya que es esta la que se encargara de realizar la búsqueda de dicho servicios, verificar el contrato que el servicio Web publica para su uso, y luego realizar el enlace o peticiones SOAP a dicho servicio según sea necesario.

Para llegar a una invocación satisfactoria de un servicio Web en una arquitectura SOA se deberán ejecutar ciertos pasos en el orden adecuado, antes que todo este deberá ser publicado por el proveedor del servicio, luego este deberá ser buscado y encontrado por el solicitante del servicio, luego de esto el solicitante ya estará habilitado para poder realizar las llamadas correspondientes al servicio Web indicado realizando las peticiones que el solicitante así desee, lo que dará como resultado una respuesta del proveedor del servicio, en la figura 4.6 se muestran estos pasos, así como también los protocolos y estándares involucrados en cada uno de ellos.

**Figura 16. Invocar servicio Web en SOA**




.

### 4.4.2. Interoperabilidad por medio de SOA

Como se ha visto la interoperabilidad es un factor fundamental para diseñar aplicaciones con una arquitectura orientada a los servicios, según ObjectWatch y un estudio denominado “Interoperabilidad por medio de Arquitectura Orientada a los Servicios (SOA)” en el cual se evalúan experiencias en la implementación de ambientes SOA, también se incluye una guía para evaluar diferentes opciones de esta tecnología.

La interoperabilidad de aplicaciones es el tema más importante para la mayoría de los arquitectos de software en cualquier organización. El uso de SOA basado en tecnologías de Web Services es se podría decir el mejor acercamiento que tenemos en interoperabilidad.

Un análisis basado en un estudio reciente por parte de Jupiter Research, demostró que al elegir una plataforma de desarrollo, la principal preocupación de los tomadores de decisiones en TI (tecnologías de la información) era la interoperabilidad con sus aplicaciones

existentes. Para el 55% de los encuestados, SOAP (Simple Object Access Protocol) y WSDL (Web Service Definition Language), que son la base de los estándares de Web Services, eran los de mayor ayuda para resolver problemas de interoperabilidad.

Dicho análisis toma como ejemplo el desarrollo SOA basado en tecnologías Microsoft. Trata sobre cómo implementar un ambiente SOA heterogéneo de manera exitosa donde se puede sacar provecho de las ventajas de diversas plataformas.

Las fortalezas específicas de Microsoft que resultan de esta evaluación son:

- Bajo costo de adquisición
- Estrecha integración entre estándares de Servicios Web y herramientas de programación
- Amplia gama de tecnologías back-end
- Extraordinario soporte para interoperabilidad con sistemas legados

Sin embargo, este estudio no se concentra en tecnologías Microsoft de manera exclusiva, sino en cómo implementar un ambiente SOA en cualquier plataforma. El documento está dividido en 3 partes principales:

Parte I: Etapa actual, es un vistazo a la Arquitectura Orientada a Servicios (Service Oriented Architecture, SOA) y por qué es considerada tan importante.

Parte II: es un conjunto de casos de estudio que muestran cómo se utiliza SOA en el mundo real.

Parte III: ofrece un análisis que incluye lecciones aprendidas y una evaluación de las tecnologías SOA Microsoft. Los criterios de evaluación presentados en el punto 9.1 pueden ser útiles para la evaluación de cualquier tecnología SOA.

# CONCLUSIONES

1. Dado un ambiente globalizado, en el cual las empresas necesitan soluciones a sus problemas en el manejo de información, la información relevante al negocio es de vital importancia, ya que provee lo necesario para la toma de decisiones con el fin de alcanzar los objetivos de las empresas. El manejo de dicha información recae en los sistemas de información y los departamentos de tecnologías de la información, los cuales diseñan e implementan las soluciones de software que darán una solución al manejo de tal información que finalmente se transformara en un objetivo tangible, es en éste punto donde surge la necesidad de crear soluciones de software que contemplen dicho ambiente globalizado, donde los factores clave para el éxito de los sistemas de información son la integración y la interoperabilidad, factores que se relacionan directamente con el diseño de de aplicaciones con una arquitectura orientada a los servicios, la cual surge como la nueva tendencia en arquitectura para el diseño aplicaciones en un contexto globalizado.

2. La arquitectura orientada a los servicios vista en su nivel de abstracción más sencillo consta de tres actores solicitante del servicio-servicio-proveedor de servicio-, pero para que las soluciones de software puedan llegar a dicho nivel de abstracción es necesario pasar por decisiones de diseño, decisiones que se deben tomar pensando en una arquitectura orientada a los servicios, en la cual se puede argumentar que los servicios son la parte fundamental de tal arquitectura, por lo que se debe pensar en la definición de servicios, identificación, orquestación y otras que garantizan una consecución de los objetivos de la arquitectura orientada a los servicios.

Los servicios son la parte fundamental de SOA, por lo que las capas de SOA garantizan que todo aquello que sea de vital importancia para los procesos del negocio emerjan desde el nivel mas bajo como lo son los sistemas operacionales y estos pasen a ser componentes en otras aplicaciones para luego pasar a ser servicios y así componer procesos de negocio en un nivel más alto, para luego ser presentados, proveyendo de esa forma la integración e interacción con otros sistemas.

3. Los servicios son claramente la parte fundamental de una SOA, por tal razón es que los servicios Web resultan ser una forma de llevar a cabo los servicios requeridos por SOA, ya que estos soportan muchos de los conceptos planteados por una arquitectura orientada a los servicios. Los servicios Web con estándares y protocolos como SOAP/XML, WSDL, UDDI y HTTP entre otros, proveen la plataforma para el la publicación, descubrimiento y enlace con otros servicios (todos estos conceptos de SOA), ya sea entre soluciones de software desarrolladas utilizando tecnologías iguales o diferentes, y operadas sobre idénticos o distintos sistemas operativos, lo cual garantiza una integración e interoperabilidad entre sistemas distintos, alcanzando así la integración e interoperabilidad conceptos clave de SOA. Si bien los servicios Web implementan algunos de los conceptos de SOA, esto no quiere decir que aplicaciones que interactúan por medio de estos sean totalmente aplicaciones con una arquitectura orientada a los servicios, como se vio en los casos de estudio, ya que conociendo el servicio con el cual se desea integrar o interactuar no es necesario realizar un descubrimiento de dicho servicio como se haría en una SOA.

# RECOMENDACIONES

1. Dado que la arquitectura orientada a los servicios apuntala la integración e interoperabilidad de los sistemas de información en un ambiente globalizado es de suma importancia que los departamentos de tecnologías de la información, promuevan políticas que garanticen una consecución adecuada para llevar a cabo el diseño de aplicaciones con SOA, dichas políticas deben involucrarse tanto en el diseño como en el desarrollo e implantación de soluciones de software.

2. Ya que los servicios son la parte fundamental de una SOA, una buena forma de implementar dichos servicios es por medio de servicios Web. Esto promueve que los sistemas de información en las empresas sean diseñados con una orientaron hacia la Web, es decir que estos sistemas no necesariamente deberán estar publicados en Internet, ya que estos se pueden ser accedidos desde una red interna en el área de trabajo, promoviendo así una infraestructura en los sistemas de información que está preparada para dar los pasos necesarios para llegar a ser una SOA.

## BIBLIOGRAFÍA

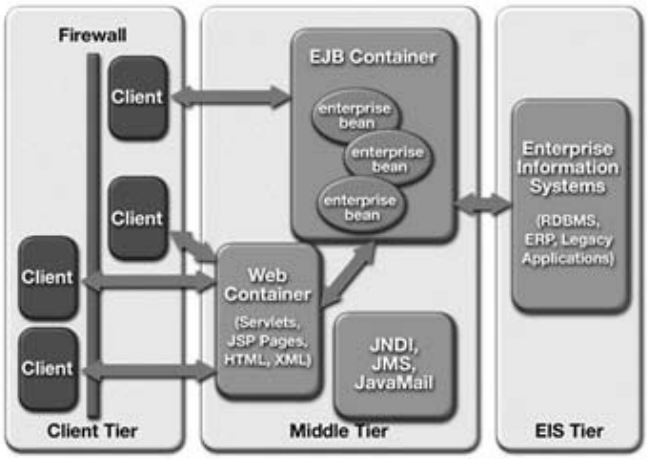
1. Paul Clements.
   "A Survey of Architecture Description Languages". Proceedings of the International Workshop on Software Specification and Design, Alemania, 1996
2. L. Bass et al.,
   "Software Architecture in Practice "Addison-Wesley, 1998 M. Fowler,
   "Public versus Published Interfaces," IEEE Software, Vol. 19, No. 2, March/April 2002, páginas 18-19.
3. Desarrollo Web.
   http://www.desarrolloweb.com/articulos/1622.php?manual=5
   Segundo trimestre 2008.
4. Developer
   http://www.developer.com/java/ent/print.php/10933_1010451_1
   Segundo trimestre 2008.
5. Microsoft español
   http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/intro.asp
   Segundo trimestre 2005
6. Microsoft español
   http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art110.asp
   Segundo trimestre 2005
7. Microsoft español
   http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art143.asp
   Segundo trimestre 2005

8. Information Week
   http://www.informationweek.webservicespipeline.com/trends/47204187
   Segundo trimestre 2005
9. MSDN Microsoft
   http://msdn.microsoft.com/architecture/journal/default.aspx?pull=/library/en-us/dnmaj/html/aj2soaimpc.asp
   Segundo trimestre 2008
10. BEA
    http://dev2dev.bea.com/technologies/soa/index.jsp
    Segundo trimestre 2008
11. Armanhal
    http://armanhal.blogspot.com/2004/06/retos-al-implementar-soa-service.html
    Segundo trimestre 2008
12. Física México
    http://www.fisica.uson.mx/carlos/WebServices/WSOverview.htm
    Segundo trimestre 2005
13. MSDN Microsoft
    http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/wsoverview.asp
    Segundo trimestre 2005.
14. Descargas de Microsoft
    http://download.microsoft.com/download/f/5/f/f5fcd06a-dafd-4f45-ab4d-bdd2da2b2e86/ObjectWatch.pdf
    Segundo trimestre 2005.
15. SOA
    http://java.sun.com/developer/technicalArticles/WebServices/soa/
    2008.

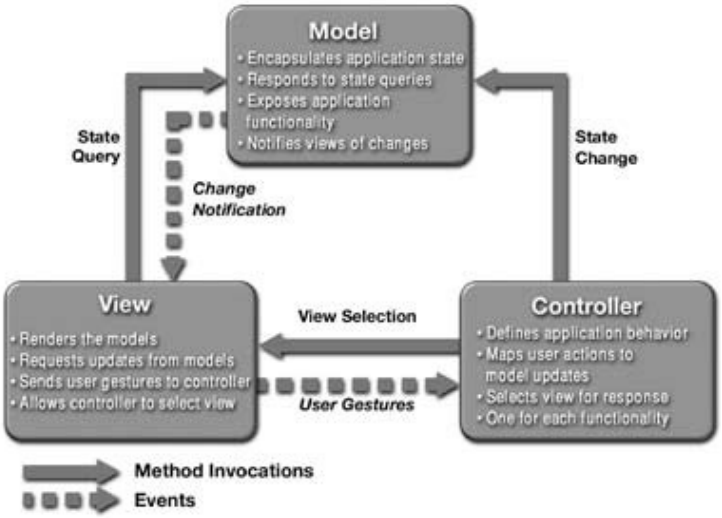
## ANEXO 1. Ejemplos de Arquitectura del Software: J2EE y MVC

Para ilustrar un poco lo que se ha explicado, a continuación se muestran dos diagramas de arquitectura en un entorno J2EE, ambos diagramas están disponibles en Designing Enterprise Applications with the J2EE Platform, Second Edition.

El primer diagrama consiste en una vista lógica que muestra los componentes y servicios típicos de un entorno J2EE.




El segundo diagrama es una vista de proceso que muestra las relaciones entre las capas model, view y controller de la arquitectura MVC bajo J2EE.




[13]

[13] Tomado de http://alzado.org/articulo.php?id_art=355