Universidad de las Ciencias Informáticas

Facultad 9

Título: Propuesta de desarrollo basada en estándares para aplicar una Arquitectura Orientada a Servicios en la Universidad de la Ciencias Informáticas

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autores:

- Danelys Nieves Hernández
- Donelkys Santana Medina

Tutor:

- Ing. Jorge Infante Osorio

Consultante:

- Ing. Manuel Alejandro Gil Martín

Asesor:

- Lic. Ana Luisa Vigó Mitjans

Ciudad de la Habana, julio 2007

"Año 49 de la Revolución"

El futuro tiene muchos nombres. Para los débiles es lo inalcanzable. Para los temerosos, lo desconocido. Para los valientes es la oportunidad.

Víctor Hugo

DECLARACIÓN DE AUTORÍA

Ciudad de la Habana, Julio del 2007
"Año 49 de la Revolución"

Nosotros, Danelys Nieves Hernández y Donelkys Santana Medina declaramos que somos los únicos autores de la presente investigación y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los 3 días del mes de Julio del año 2007.

Danelys Nieves Hernández

Donelkys Santana Medina

Ing. Jorge Infante Osorio

DATOS DE CONTACTO

Síntesis del Tutor Ing. Jorge Infante Osorio

Profesión: Ing. Informática

Categoría docente: Adiestrado

Años de graduado: 2

Síntesis del Consultante Ing. Manuel Alejandro Gil Martín

Profesión: Ing. Informática

Categoría docente: Especialista de la dirección de informatización y Arquitecto principal del sistema de la intranet.

Años de graduado: 1

Síntesis de la Asesora Lic. Ana Luisa Vigó Mitjans

Profesión: Licenciada en Lengua Inglesa

Categoría docente: Profesora Auxiliar

Años de graduado: 30

AGRADECIMIENTO

A nuestros padres.

Por ser nuestra guía e inspiración, por ser nuestra razón de ser y por aconsejarnos para seguir el camino correcto.

A nuestros familiares.

Por formar parte de nuestras vidas.

A nuestros amigos.

Que lograron que viéramos la universidad como nuestra casa.

Aquellos que demostraron ser amigos en las buenas y en las malas. En especial a la compañera y amiga Eylin Hernández Luque que siempre estuvo allí cuando la necesitamos.

A los tutores y guías.

El Ing. Manuel Alejandro Gil Martín y el Ing. Jorge Infante Osorio que nos apoyaron en todo momento para que esta tesis terminara en tiempo y forma así como a la Lic. Yaqueline Zamora Mora que sin ser miembros de la comisión que respaldaba esta tesis nos brindaron su apoyo a cambio de nada.

A la ***Revolución y a nuestro comandante Fidel Castro Ruz*** que gracias a él este proyecto pasó de un sueño a una realidad.

DEDICATORIA

Quisiera dedicarles esta tesis a mis padres Digna Virgen Hernández Martínez y Jorge Nieves Acosta y a mí hermano Jorge Yannier Nieves Hernández por ser ellos mi fuente de inspiración, por apoyarme en cada momento de mi vida, guiarme por el buen camino y lo más importante porque los quiero con todo mi corazón.

Se la dedico también a quien más que un amigo fue como un padre para mí durante mi estancia en esta universidad, Liuver Reynier Durán Pérez.

A mis abuelas Aleyda Martínez y Rafaela Acosta que siempre estoy es sus recuerdos.

Dedicarle este trabajo a mi pareja y compañero de discusión Donelkys Santana Medina que me supo comprenderme y estuvo siempre a mi lado apoyándome en cada paso y decisión que tomé durante el transcurso de estos cortos pero inolvidables cinco años.

Danelys Nieves Hernández

Quiero dedicar este trabajo de diploma a la persona que más quiero en el mundo, a quien le debo todo lo que soy y seré a partir de ahora, no tengo palabras para agradecerte todo lo que haz hecho por mí y solo puedo decirte que te quiero muchísimo, esta tesis es para mi mamá Mariela Medina Pérez.

También quiero dedicarle esta tesis a toda mi familia, a mi abuela, a mis tías Lucy y Merce, a mi papá, a mi padrastro Rene y por supuesto a quién es mi tío, mi hermano y mi amigo a mi tío Edel. A mis amigos de la universidad y a mis amigos de siempre Liuver, Leonardo, Wilmar y Zudelmis.

A mi compañera de tesis y compañera de la vida a mi novia Danelys Nieves Hernández gracias por estos cinco años maravillosos y por los que vendrán.

Donelkys Santana Medina

OPINIÓN DEL TUTOR

RESUMEN Y PALABRAS CLAVES

Con el fin de conocer cómo se puede poner en práctica la Arquitectura Orientada a Servicios (SOA) en la Universidad de las Ciencias Informáticas (UCI) fue realizada la presente investigación. Trabajo que analiza los principales conceptos relacionados con el objeto de estudio para una mejor compresión por parte del lector. Explica cuáles son las características fundamentales de una SOA abordando en mayor medida en qué es una Arquitectura Orientada a Servicios, cuáles son las ventajas que ella reporta para la universidad y se analizan los principales componentes que utiliza está arquitectura para lograr estandarizar y tener de forma más organizada los servicios que utiliza. Además se estudian los problemas de la UCI que condujeron a utilizar está arquitectura como una vía de solución.

En una SOA los estándares juegan un papel protagónico por tanto fue necesario realizar un estudio de los más usados para conocer en que consiste cada uno y cómo se integran en esta arquitectura. El trabajo culmina con una propuesta de desarrollo que está acompañada por la selección de los estándares que más se ajustan a las necesidades de la universidad, resultado que permite lograr una mayor interoperabilidad entre las aplicaciones y homogenizar la comunicación entre ellas.

Palabras Claves:

- Estándares de SOA.
- Arquitectura Orientada a Servicios.
- Propuesta de desarrollo para una SOA.

ABSTRACT

The University of Informatics Science needs to implement a Service Oriented Architecture (SOA); for this reason, this investigation was carried out. It analyses the main concepts associated with SOA, explains the fundamental attributes of its architecture, its advantages and most important components. Besides, the problems that make the university implement this architecture are exposed in this investigation.

SOA standards play an important role .For this reason, it was necessary to study the most used ones, in order to know their characteristic and how they are compiled into a SOA environment. This research provides a development proposal based on the selection of the standards that can solve the needs of the university. Its result allows achieving and providing more operational facilities among the different applications as well as their internal communication.

Key Words

- SOA Standards
- Services Oriented Architecture
- Proposal to develop SOA

TABLA DE CONTENIDO

INTRODUCCIÓN

Durante mucho tiempo la realización de un software no era de mucha complicación para un desarrollador pues se basaban en proyectos de poca complejidad, se realizaban virtualmente y sin ninguna planificación. Esto fue así hasta que los planes comenzaron a fallar y los costos a aumentar. Los programadores trataban de hacer bien su trabajo y en ocasiones lograban su objetivo. Pero esto solo se logró mientras los softwares fueron pequeños.

La mayoría de los programas eran realizados y utilizados por una misma persona o empresa. Eran ejecutados por ella y si ocurría algún fallo, lo depuraba. Este modo de trabajo que no contenía una documentación y que era realizado en la mente de alguien, le daba al software un entorno personalizado.

Las complicaciones surgen a medida que se hizo necesario realizar softwares más complejo, la búsqueda de técnicas para mejorar la calidad y reducir los costos de producción del software se convirtió en el principal objetivo de los programadores. Todo esto unido al rápido avance tecnológico en el campo de la informática trajo la necesidad del uso de nuevas técnicas para la programación.

Surge entonces la arquitectura de software, la cual facilitaría la creación de aplicaciones mejor estructuradas y pensadas, la comprensión y el manejo de las mismas. La arquitectura brinda conceptos, aporta elementos que ayudan a la toma de decisiones y un lenguaje común que permite la comunicación entre los desarrolladores.

Pero la reducción de los costos no era lo único que sería un problema para los productores de software. La globalización y el negocio aceleraban el paso del cambio. La globalización conduce a la petición feroz de software por parte de las compañías, pues estas, miran a sus competidores para sacar ventaja de lo que tienen.

Las demandas y los requisitos de los usuarios cambian rápidamente conducido por la competencia y la abundancia de la información de los productos disponible en Internet. La necesidad de buscar un modo de tener las aplicaciones lista para el cliente, pero al mismo tiempo preparadas para los posibles cambios que puedan pedir sin estar en la obligación de reescribir totalmente la aplicación, trajo consigo que la arquitectura evolucionara (Ver Anexos 1) tratando de darle solución tanto a las demandas del mercado de aplicaciones como a los negocios de las empresas. De esta manera se lograba que las aplicaciones fueran vistas como una entidad, separándose en capas lógicas que permitían a los desarrolladores trabajar sobre una de ellas sin necesidad de tocar las restantes.

La arquitectura en dos capas fue la solución inmediata para lograr separar las partes lógicas de las aplicaciones, las ventajas fueron considerables, pero la flexibilidad fue mucho mayor al lograr obtener la arquitectura en tres capas, donde la interfaz de usuario, las reglas y/o lógica del negocio y el acceso a datos se encontraban lógicamente distribuidas.

Igualmente fue incorporado como patrón arquitectónico de diseño y para lograr la adaptación a los continuos cambios, el Modelo-Vista-Controlador (Ver Anexos 2), pero este al igual que las restantes arquitecturas trabajaban para resolver problemas a nivel de aplicación.

En estos momentos más que trabajar la arquitectura a nivel de aplicación se requiere ver los procesos de negocio de la empresa completa de forma global. Las organizaciones están viviendo un cambio de mentalidad a la hora de pensar en la tecnología de la información. Una organización lleva a cabo sus tareas mediante la realización de distintos tipos de procesos y estos a su vez generan datos que por supuesto deben ser procesados. Pero son los procesos los que definen a la organización y por tanto la mayor atención va dirigida a ellos y no a los datos que generan.

El objetivo principal de las empresas es conseguir agilidad y ventajas competitivas, siendo capaces de adaptarse a los continuos cambios que se producen en el mercado que operan. Estos cambios suponen siempre una modificación de los procesos de negocio de la organización implicada. De ahí que se puede conseguir una mayor agilidad y capacidad de innovación si las organizaciones consiguen cambiar la arquitectura de sus sistemas informáticos, orientándolas hacia los procesos que habitualmente realizan, y extrayendo la gestión de estos procesos en una capa independiente. Es decir, aplicar una Arquitectura Orientada a Servicios (SOA o Service-Oriented-Architecture).

SOA es una arquitectura que permite desglosar las aplicaciones en términos de servicios, resultando ser más flexibles y por tanto son más fáciles de modificar antes los cambios del negocio. Utilizando esta arquitectura se puede realizar la reutilización de los servicios, pues en la mayoría de las empresas la mayor parte de los procesos de negocio ya están implementados, obteniéndose una reducción del costo de desarrollo y mantenimiento.

Otra de las ventajas es que al reutilizar los servicios hay mayor calidad y reducción de riesgos en las aplicaciones pues estos servicios ya han sido probados y en muchos casos mejorados. El especialista encargado de llevar a cabo la tarea de programar se ve aliviado de la carga de trabajo pues se evita implementar funciones que ya están realizadas.

En la Universidad de las Ciencias Informáticas (UCI) se manejan innumerables procesos de negocio y van en aumento. La frecuencia con que se realizan los cambios en los procesos de negocio provoca constantes actualizaciones. Por tanto la tendencia de realizar aplicaciones basadas en una Arquitectura Orientada a Servicios (SOA), es cada vez más vigente. Sin embargo, ¿se puede asegurar de que la Universidad de las Ciencias Informáticas se encuentra preparada para aplicar esta nueva arquitectura?

Evidentemente, no. Tener un personal capacitado y disponible para ayudar en esta esfera es muy difícil en estos momentos. Para los especialistas de la universidad la Arquitectura Orientada a Servicios es completamente nueva, un campo desconocido que tienen que incursionar.

Compañías de prestigio en la industria del software, basadas en su experiencia, han definido un grupo de estándares a utilizar para implementar una SOA. Por tanto esta investigación está enmarcada en lo que a continuación se expone:

Problema a Resolver:

¿Cómo desarrollar una Arquitectura Orientada a Servicios en la Universidad de las Ciencias Informáticas, que se ajusten a las necesidades de la universidad, basada en una selección de los estándares existentes?

Objeto de estudio:

La Arquitectura Orientada a Servicios (SOA).

Campo de acción:

Estándares para desarrollar una SOA en la Universidad de las Ciencias Informáticas (UCI).

Objetivo general:

Elaborar una propuesta de desarrollo basada en estándares para aplicar una Arquitectura Orientada a Servicios en la Universidad de las Ciencias Informáticas.

Idea a defender:

Una propuesta de desarrollo basada en estándares para aplicar una Arquitectura Orientada a Servicios en la Universidad de las Ciencias Informáticas, servirá de ayuda a los programadores y arquitectos para lograr la misión de dirigir, organizar, coordinar, chequear, diseñar y definir los procesos que rigen la vida de la universidad.

Tareas específicas a desarrollar:

- Estudio sobre la evolución del estado del arte.
- Análisis de los componentes de una SOA.
- Análisis de la seguridad de los servicios en SOA.
- Análisis de las características de los procesos de la UCI.
- Estudio de los principales estándares existentes de una SOA.
- Selección de los estándares que pueden utilizarse dadas las características de la universidad.
- Elaboración de una propuesta de desarrollo para implementar una SOA en la UCI.

Métodos de investigación científica:

Métodos Teóricos:

- Analítico-Sintético: Este método permite extraer lo esencial de la bibliografía consultada de Arquitectura Orientada a Servicios.
- Inductivo-Deductivo: Facilita el análisis de los elementos generales a elementos más particulares.
- Análisis Histórico Lógico: Permite deducir lógicamente cómo ha evolucionado la arquitectura.

Métodos Empíricos:

- Observación: Se observaron cómo trabajaban las aplicaciones en la Universidad de las Ciencias Informáticas (UCI) para sacar conclusiones que permitieran comprender cómo modelar el proceso de negocio de manera global.
- Entrevista: Se realizaron entrevistas al personal de la universidad con conocimiento del funcionamiento de la UCI para recopilar información y establecer acuerdos.

Resultado Esperado:

Obtención de una propuesta de desarrollo de una Arquitectura Orientada a Servicios (SOA) para la Universidad de las Ciencias Informáticas, basada en estándares ajustados a los procesos actuales de la universidad, que brinde además la posibilidad de redefinirlos e incorporar nuevos procesos al negocio actual, facilitando la comunicación entre los mismos.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

La Universidad de las Ciencias Informáticas presenta un grupo de problemas que necesita resolver, pero para darle solución a los mismos precisa realizar un movimiento significativo a la hora de organizar su negocio. Movimiento que requiere que perdure con el tiempo, pues las prestaciones que brindan suelen estar sujetas a constates cambios.

Por tanto como respuesta a estas demandas, se ha propuesto utilizar una Arquitectura Orientada a Servicios (SOA) con fines organizativos y que propicie aplicaciones flexibles y reusables. Para comprender qué es una Arquitectura Orientada a Servicios es necesario realizar un estudio de sus características y analizar un conjunto de conceptos que están relacionados directamente con la comprensión de la misma.

Por estas razones el capítulo siguiente centra su atención en los principales conceptos relacionados con el objeto de estudio. En las principales características de la Arquitectura Orientada a Servicios y también analiza los problemas que presenta la Universidad de las Ciencias Informáticas que condujo a la realización de esta investigación.

1.2 Conceptos asociados al dominio del problema

La Arquitectura Orientada a Servicios es una nueva tendencia que se está poniendo en práctica y es completamente nueva para la Universidad de las Ciencias Informáticas, para lograr entenderla es necesario primeramente revisar algunos conceptos que permiten tener una mejor visión y comprender la información que se transmite.

Son numerosas las definiciones que se han establecido de Arquitectura de Software y ninguna de las existentes es respaldada por la totalidad de los arquitectos o expertos en el tema. Pero muchas de estas son aprobadas y reconocidas a nivel mundial.

- **Arquitectura de Software**

  Una definición reconocida es la de Paul Clements: 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 diferencia del detalle inherente a la mayor parte de las abstracciones (CARLOS BILLY REYNOSO).

Otra definición de Arquitectura de Software aparece en el documento de IEEE Std 1471-2000, adoptada también por Microsoft, que dice así:

- "La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución" (CARLOS BILLY REYNOSO).

De los conceptos antes expuestos, se puede inferir que la Arquitectura de Software establece los fundamentos para que los desarrolladores y miembros de un proyecto trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema. Es una guía para la implementación de la estructura de un software de alto nivel. Muestra como resultado el ensamblado de un número de elementos arquitectónicos que satisfacen la funcionalidad y los requerimientos de software, garantizando la viabilidad del proyecto en el menor tiempo posible.

- **Estándar:** Dícese de lo que sirve como tipo, modelo, norma, patrón o referencia(Diccionario Terra).

  Se puede concluir de lo antes citado que estándar es un modelo o patrón trazados por una organización para tener una norma por donde regirse a la hora de trabajar.

- **Servicios**: Un servicio debe ser una aplicación completamente autónoma e independiente. A pesar de esto, no es una isla, porque expone una interfaz de llamado basada en mensajes, capaz de ser accedida a través de la red. Generalmente, los servicios incluyen tanto lógica de negocio como manejos de estado (datos), relevantes a la solución del problema para el cual fueron diseñados. La manipulación del estado es gobernada por las reglas de negocio(ALEJANDRO GUINEAS DE SALAS; SERGIO JORRIN).

- **Servicios:** Un servicio representa una función de negocio claramente definida que puede ser invocada remotamente mediante protocolos de comunicación estándar. Los servicios se definen mediante interfaces explícitas que son totalmente independientes de la implementación del servicio. Los servicios deben poder ser invocados utilizando protocolos de comunicación estándar que enfatizan la interoperabilidad e independencia de ubicación (HUILBERT AALBERS).

De lo antes mencionado, se puede decir que un servicio es una aplicación que cumple con un determinado requisito del negocio y que es capaz por sí sola de satisfacer esta función. Puede ser invocada a través de la red por otras aplicaciones que necesitan beneficiarse de su funcionalidad.

- **Servicio Asincrónico:** Permite manejar comunicación no directa entre los participantes, puede activarse automáticamente o de forma directa por cada participante. Estos servicios comparten los datos sin la necesidad de que los participantes se encuentren conectados en ese momento.

- **Proveedor de Servicios:** Entidad de software que implementa una especificación de servicio (BILLY REYNOSO). Es decir, es un servicio que brinda prestaciones a otras aplicaciones o servicios.

- **Consumidor de Servicios:** Entidad de software que llama a un proveedor de servicio. Tradicionalmente se le llama “cliente”. Puede ser una aplicación final u otro servicio (BILLY REYNOSO).

- **Localizador de Servicio:** Tipo específico de proveedor de servicio que actúa como registro y permite buscar interfaces de proveedor de servicio y sus ubicaciones (BILLY REYNOSO).

- **Servicio Intermediario**: Tipo específico de proveedor de servicio que puede pasar requerimientos de servicios a otros proveedores de servicios (BILLY REYNOSO).

- **Servicio Web**: Es un servicio que se comunica con sus clientes a través de protocolos y tecnologías estándares. Estos servicios Web estándares son implementados en las plataformas y productos de los mayores vendedores de software posibilitando a los clientes y servicios comunicarse a través de una amplia gama de plataformas y entornos de desarrollo. La versatilidad de los servicios Web los ha colocado como el método más eficiente para implementar una SOA (ED ORT 2005).

Para entender mejor este concepto se puede inferir que un **Servicio Web:** Es la tecnología utilizada para desarrollar servicios en la red. Recibe este nombre cualquier segmento de código que define un servicio usando estándares de interfaz Web para comunicarse con otro software sin importar la plataforma o entorno de desarrollo de ambos. Comunicación que se logra siempre y cuando se trabaje bajo estándares. Algunos desarrolladores prefieren referirse a los servicios Web como una URL programable.

- **Mensaje**: Son los encargados de encapsular los datos de entrada y de salida que se envían los consumidores y proveedores de servicios.

- **Componente:** Este término hace referencia a una de las partes de la solución total, como por ejemplo los componentes de software compilados (los ensamblados de Microsoft.NET, páginas Web, entidades empresariales, agentes de servicios). Un componente puede ser una entidad o reglas del negocio que trabajan en conjuntos para entregar las funciones del negocio.

- **Lógica de Negocio:** Es la parte del sistema que se encarga de las tareas relacionadas con el proceso de un negocio (ejemplo ventas, control de inventario, contabilidad). Son rutinas que realizan entradas de datos, consultas a los datos, generación de informes y más específicamente todo el procesamiento que se realiza detrás de la aplicación visible para el usuario (Wikipedia 2007).

Se puede decir que la **lógica de negocio** colecciona todas las decisiones, cálculos y operaciones que la aplicación debe llevar a cabo para no perder de vista lo que se quiere obtener.

- **Proceso de Negocio:** Un proceso de negocio es la codificación de reglas y prácticas que integran el negocio. Desde la perspectiva de SOA un proceso de negocio incluye personas, servicios de negocio, adaptadores y un grupo de actividades de manejo de procesos que manipulan los flujos de trabajo entre todas las partes implicadas (JUDITH HURWITZ 2007).

Del concepto antes expuesto se puede inferir que un **proceso de negocio** es un conjunto de tareas lógicamente relacionadas llevadas a cabo para lograr un resultado del negocio definido. Este puede ser parte de un proceso mayor que lo contenga o puede incluir otro proceso. Puede ser visto como el flujo de trabajo que efectúan las tareas de una organización. Son la base operativa de una empresa y el éxito de la misma depende en gran medida de la eficiencia con que sean gestionados.

- **Acoplamiento:** Hay dos modos de ver el acoplamiento suele verse como débil o fuerte. El acoplamiento débil es cuando hay poca dependencia entre el productor y el consumidor, es decir, no están atados directamente el uno con el otro sino que abstractamente están conectados virtualmente por un canal llamado publicar/suscribir o canal punto a punto (Point-to-Point), esta conexión es a través de mensaje. El productor no necesita saber cuál aplicación está recibiendo un mensaje, así como el consumidor no necesita saber quién lo está enviando, sabe solamente que está recibiendo un mensaje y actúa sobre él. Mientras que el acoplamiento fuerte es lo contrario.

- **Middleware:** Es un módulo intermedio que actúa como conductor entre sistemas permitiendo a cualquier usuario de sistemas comunicarse con varias fuentes de información u otros sistemas que se encuentran conectadas por una red (PABLO ALVEZ ;PATRICIA FOTI ;MARCO SCALONE 2006).

Para estar claro de qué es exactamente un middleware se puede agregar que es una infraestructura típica diseñada para facilitar la integración. Es una capa intermedia que se agrega en la arquitectura que se esté utilizando. En ocasiones el middleware se aloja en el cliente Web. Es utilizado para monitorear las comunicaciones entre contenidos y plataformas, en cada caso aporta las funcionalidades necesarias. Se busca con el middleware resolver problemas de compatibilidad o construir nuevas funciones no estándar de una manera coherente, flexible y sistemática.

- **Proceso de negocio abstracto:** Protocolo de negocio que especifica el intercambio de mensajes entre diferentes partes sin revelar el comportamiento interno de ninguna de ellas (coreografía de servicios) (ADOLFO R. DE SOTO; EVA CUERVO FERNÁNDEZ 2006).

- **Proceso de negocio ejecutable:** Especifica el orden de ejecución entre un número de actividades que lo constituyen, las partes involucradas, los intercambios de mensajes entre estas partes y los mecanismos de manejo de fallos y excepciones (orquestación de servicios) (ADOLFO R. DE SOTO; EVA CUERVO FERNÁNDEZ 2006).

- **Itinerario de un mensaje:** Es un metadato que viaja junto al mensaje y proporciona la lista de direcciones siguientes a alcanzar por el mensaje. El itinerario es un conjunto de instrucciones que le dice al framework de ejecución del ESB a qué sistema se tiene que enviar el mensaje y este viaja de sistema en sistema a través del bus.

1.3 Objeto de Estudio

1.3.1 ¿Qué se entiende por Arquitectura Orientada a Servicios?

SOA es una de esas cuestiones cuya definición muestra únicamente un pálido reflejo de su significado real, especialmente porque cada experto en SOA le dará una definición ligeramente distinta. Algunos conceptos que a continuación se muestran ayudan a comprender SOA.

- El concepto SOA hace referencia a un enfoque de arquitectura cuyo objetivo es la creación de sistemas a partir de servicios autónomos. Con SOA, la integración pasa a ser una reflexión previa más que una idea posterior. Probablemente, la solución final esté formada por servicios desarrollados en distintos lenguajes de programación y se aloje en plataformas diferentes con numerosos modelos de seguridad y procesos empresariales (JOHN EVDEMON 2005).
- SOA: es una metodología o una estrategia en la cual las aplicaciones hacen uso de los servicios disponibles en la red. Mediante SOA se pueden desarrollar aplicaciones que utilicen servicios o que se comporten como un servicio para que otros accedan a ellas o ambos (ED ORT 2005).

- SOA es una arquitectura de aplicación en la cual todas las funciones se definen como servicios independientes con interfaces invocables bien definidas, que pueden ser llamadas en secuencias definidas para formar procesos de negocio (BILLY REYNOSO).

De los conceptos expuestos se infiere que SOA es una arquitectura que fomenta la creación de aplicaciones que se adaptan fácilmente a los inevitables y frecuentes cambios del negocio. La Arquitectura Orientada a Servicios es tanto un marco de trabajo como un marco de implementación para el desarrollo de los procesos de negocio de una organización, es utilizada con fines organizativos. Propone desglosar las funcionalidades principales de las aplicaciones en servicios autónomos y luego orquestarlos para formar nuevas aplicaciones de negocio. Estos servicios se encuentran disponibles en la red y presentan interfaces estándares bien definidas que permiten el flujo de mensajes entre proveedores y consumidores.

1.3.2 ¿Por qué utilizar SOA?

Esta arquitectura está encaminadas a liberar el negocio de las restricciones de tecnología, rompiendo las cadenas que ellos mismos se han creado, pues el negocio centra su atención más en él y menos en la tecnología, al resultar esta invisible a la capa del negocio deja de ser un obstáculo. Por eso se dice que SOA separa la lógica del negocio de la lógica computacional.

Al tener separada la lógica del negocio, SOA posibilita que se agilice y sea más fácil el desarrollo y el despliegue de los sistemas que desempeñan los objetivos de una empresa. Las empresas actuales están muy vinculadas a la habilidad que esta posea para adaptar sus tecnologías a los frecuentes cambios y desafíos del negocio. Una SOA permite que las tecnologías evolucionen a la par del negocio.

Antes de que surgiera SOA los desarrolladores hablaban de interfaz, clases, tablas, consultas y los directivos de negocio solo hablaban de procesos de negocio y para entonces, lograr un entendimiento entre ellos era muy difícil. Mientras con SOA un servicio de negocio para ambas partes es lo mismo. La posibilidad de que ambas partes tantos los directivos de negocio como los desarrolladores hablen un lenguaje común posibilita que puedan entenderse y tomar decisiones juntos.

SOA también permite la reutilización, trabaja sobre qué es lo que tengo y cómo lo puedo estructurar para reutilizarlo, pero además se cerciora que los cambios en un futuro sean simples, seguros y rápidos. Brinda negocios más flexibles y tecnologías más confiables, sostenibles, extensibles y manejables.

Es cierto que son muchas las razones por la cual utilizar SOA pero no se puede olvidar cuales son sus características más importantes:

- Reusabilidad: SOA logra un mayor grado de reusabilidad que contribuye a disminuir el tiempo de desarrollo y el costo de cualquier proyecto. Esto se logra mediante la reutilización de los servicios, o sea, cada aplicación se desarrolla como un servicio que puede ser usado por otras aplicaciones para satisfacer nuevos requisitos de negocio. El mayor inconveniente presentado en otras arquitecturas era la incompatibilidad que se genera en aplicaciones desarrolladas en diferentes plataformas y lenguajes de programación. En SOA la única característica del servicio que el cliente necesita conocer es la interfaz pública.
- Interoperabilidad: Uno de los objetivo de SOA es la comunicación entre el cliente y el servicio sin importar sobre que plataforma o sistema operativo corra cada uno, esto significa amplia interoperabilidad. Para que el objetivo pueda alcanzarse es necesario que ambos utilicen formas estándares de comunicación entre ellos. Los servicios Web de SOA brindan protocolos y tecnologías altamente aceptadas y utilizadas que son independientes de plataformas, lenguajes y sistemas. Estos protocolos y tecnologías trabajan a través de firewalls, facilitando a los socios de negocio compartir sus servicios.
- Escalabilidad: Los servicios en SOA presentan un acoplamiento débil (Loosely-coupled) por tanto las aplicaciones que los utilizan tienden a ser más escalables. La poca dependencia que existe entre el cliente y el servicio posibilita atender un mayor número de pedidos. Los servicios en SOA tienden a ser orientados a documentos, asincrónicos y de granulación gruesa. La granulación gruesa ofrece una amplia gama de funciones y no una sola función, es decir, se puede procesar una petición completa y no una sola operación de la petición como en los servicios de granulación fina. Al ser orientados a documentos permiten documentos como entrada, en vez de un valor numérico o un objeto. Un servicio asincrónico no obliga al cliente a esperar a que se procese el pedido.

- Flexibilidad: Los servicios en SOA son orientados a documentos, asincrónicos y de acoplamiento débil, posibilitando desarrollar aplicaciones más flexibles y fáciles de modificar ante los cambios del negocio. Lo contrario ocurre con las arquitecturas de acoplamiento fuerte (tightly-coupled) donde los componentes están fuertemente acoplados y es muy complicado modificarlos.
- Costos eficientes: Los sistemas desarrollados en SOA son menos costosos porque la integración entre el cliente y el servicio es más flexible, por tanto, es más fácil su mantenimiento y su modificación. También facilita la reutilización de funciones que significa un mayor ahorro. En otras arquitecturas que presentan componentes fuertemente acoplados, modificar uno de ellos significa tener que modificar el resto, todo lo contrario de SOA.

La principal ventaja es la evolución desde una tecnología de software basada en una estructura monolítica hacia una flexible compuesta por servicios. Estos bloques que se componen para armar una solución, responden a las necesidades del cliente, deja de ser una solución genérica para convertirse en algo hecho a la medida.

Al observar la estructura antigua del software, se asemeja a un bloque armado, funcional pero estático. ¿Qué pasaría si cambia el día de mañana algunas de las funcionalidades que presenta? Si el software es estático e inflexible. Pues inevitablemente para mejorar es necesario cambiarlo todo.

La belleza de SOA radica en que incluso ella se puede implementar usando estas aplicaciones inflexibles, transformando pedazos de esos programas y expresándolos luego como servicios que soportan la realización de procesos de negocio de principio a fin de una forma dinámica y reconfigurable utilizando descripciones de servicios basadas en interfaces.

De esta manera SOA le da un nuevo sentido al antiguo software e incluso le da nuevos usos para que se mantengan vigentes y puedan ser reutilizados, pues en caso que se incorporen nuevas necesidades al proceso de negocio se pueden desacoplar los servicios y volverlos a acoplar según la situación que se presente o simplemente agregar al flujo de trabajo algunos servicios que respondan a las nuevas demandas.

Con SOA cada servicio se desarrolla con la intención de aportar valor al negocio de la organización. SOA permite que componentes ejecutables (servicios Web) puedan ser invocados por otros programas que actúan como clientes o consumidores de servicios. Al ser invocados estas aplicaciones son tratadas como caja negra, pues un desarrollador no necesita saber la lógica interna del programa, basta con que conozca la entrada que requiere, la salida que produce y cómo se invoca para su ejecución.

1.3.3 SOA no es un Servicio Web

Muchos de los desarrolladores aún confunden el concepto de SOA con servicios Web. Es importante destacar que SOA es una visión de arquitectura que ha sido el resultado de la evolución tecnológica y de las necesidades del negocio. Mientras que los servicios Web son algunos de los elementos que permite su implementación.

Vean a SOA como la arquitectura con principios organizativos y a los servicios Web como los bloques que se necesitan para su construcción, estos son la base de la arquitectura, es solo una parte de lo que ella puede hacer.

SOA se encarga de dividir la lógica del negocio de la lógica computacional. Los servicios Web implementan lógica computacional, serían los que proveen una forma estandarizada para integrar aplicaciones, como un medio para que las empresas se comuniquen entre sí, sin necesidad de conocer como funciona cada una.

1.3.4 Conociendo SOA por dentro.

La Arquitectura Orientada a Servicios no es tan simple de entender como muchos piensan, basta con darle una mirada por dentro para cambiar de opinión, la forma en que está formada requiere un poco de tiempo para entenderla.

SOA está compuesta por componentes (Ver Anexos 3):

- **El Registro de SOA:** Contiene información descriptiva de los servicios y referencia la localización de los componentes de SOA.
- **El Servicio Intermediario:** Realiza todas las conexiones entre los componentes, utilizando la información del registro SOA.

- **El Supervisor de SOA:** Monitorea que los servicios se ejecuten con el nivel de aceptación esperado. Controla el tráfico de los mensajes entre los componentes.
- **El Motor del Flujo de Trabajo (Workflow):** Automatiza la ejecución de las operaciones dentro de un proceso de negocio.
- **El Bus de Servicio de Empresa (ESB):** Logra la integración de los servicios, garantizando que los mensajes fluyan en ambos sentido entre los componentes de una aplicación SOA.

Registro SOA.

El registro SOA es como un catálogo electrónico que guarda información sobre los servicios (quién lo hizo, quién puede cambiarlo, cómo puede ser usado, a quién se le permite acceder a la información, dónde se encuentra), además publica los componentes reutilizables para que puedan ser usados por otras entidades. Es extremadamente importante porque actúa como un punto central de referencia de SOA, contiene la información que referencia que componentes son soportados, definiendo el dominio de la arquitectura.

El registro tiene tres funciones fundamentales:

- Publicar y permitir el descubrimiento de los servicios de negocio.
- Coleccionar y manejar los metadatos de los servicios.
- Gobernar el uso de los servicios de negocio.

Tipo de información que puede ser almacenada en el registro:

- La descripción de los servicios que brinda el componente.
- La descripción de la interfaz o interfaces del componente.
- Dónde se localiza el componente.
- Especificación del nivel de servicio definido por el componente.
- Una definición de los derechos de uso del componente (quién tiene qué tipo de acceso bajo qué condiciones)
- Una definición de las reglas de seguridad que gobiernan el uso del componente.

Trabaja sobre dos reglas:

- La primera regla radica en el entorno operacional. Donde provee información acerca de los componentes que se están ejecutando o están disponible para ser usados. Está información es importante para el servicio intermediario.
- La segunda es arraigada en el mundo de los programadores y los analistas de negocio. Para estos actúa como referencia ayudándolos a seleccionar componentes y a conectarlos para construir aplicaciones compuestas y procesos de negocio. También registra información de cómo se conecta un componente con otro.

En una SOA se manejan dos conceptos que son muy similares y tienden a confundir. El registro y el repositorio de SOA, en muchos casos tratan ambos conceptos como si fuera uno solo.

Sin embargo un **repositorio** es un lugar para almacenar y en SOA no es muy distinto, todos los componentes usados para construir una aplicación están almacenados en este sitio. El repositorio contiene versiones e información histórica importante de cada uno de los servicios. Este es estático, solamente cambia en caso de que se adicione algo nuevo, por lo demás es estable y no interviene en la parte dinámica de SOA.

A través de un vínculo cerrado el registro y el repositorio actualmente proveen diferentes funciones. La principal función del registro es publicar y posibilitar el descubrimiento de los servicios y gobernar en tiempo real el uso de estos. Si los servicios llegan a ser modificados el registro siempre mantiene la pista de ellos actualizada. Por tanto se mantiene atendiendo las operaciones dinámicas de la SOA.

Servicio Intermediario (Ver Anexos 4)

Cuando un componente desea conectarse con otro, lo hace a través del servicio intermediario, este lee la información del registro de SOA buscando el componente que ha sido solicitado y le devuelve cómo puede comunicarse y dónde se encuentra localizado el componente que necesita.

Este actúa como una aguja hilando un componente a otro en un proceso de negocio. Utiliza la información de los componentes disponible en el registro y enhebra los componentes para el motor de flujo de trabajo o motor de procesos.

En el mundo de las aplicaciones de acoplamiento débil el servicio intermediario es quien se encarga de la comunicación entre una aplicación y otra, una importante tarea. Orquesta las conexiones entre componentes, de hecho, orquesta la conexión para el proceso de negocio completo.

Supervisor de SOA

El supervisor de SOA actúa como controlador de tráfico y como punto central de toda la orquestación en SOA. Este es activado cada vez que un servicio es operado dentro del entorno SOA, durante una función de negocio mientras los componentes pasan datos e instrucciones a otros, al mismo tiempo el motor de flujo de trabajo está circulando datos, los agentes supervisores están enviando información al supervisor.

Su principal responsabilidad es asegurar el nivel de aceptación de los servicios. Utiliza los reportes de los agentes para saber exactamente que está pasando. Por tanto, está en posición de saber cuando un servicio está funcionando mal o cuando hay algún fallo.

El papel del supervisor de SOA es cerciorarse de que la plataforma por debajo del ambiente de SOA trabaje de una manera constante y fiable. La meta es crear un ambiente en donde todos estos componentes trabajen juntos para mejorar el flujo del proceso del negocio. Todos estos servicios se requieren para acoplar componentes sin relación de la tecnología como si fueron diseñados para trabajar juntos.

La ambición del negocio es expandirse y es ahí donde el trabajo del supervisor de SOA se expande también porque al haber mayor cantidad de aplicaciones conectadas tiene que mantener un mayor número de servicios.

¿Cómo funciona el supervisor?

Cuando se inicia un proceso de negocio el servicio intermediario envía un mensaje al supervisor para solicitar el inicio de un servicio. El supervisor se comunica con el registro para obtener los detalles del todo el proceso de negocio así que puede comenzar el proceso de monitorización para monitorear cada uno de los componentes. Delega esta función de monitorear a SLA Monitoring (Services Level Agreement) y este componente activa los agentes que se encuentran en cada uno de los componentes. El SLA envía regularmente boletines de rendimiento al supervisor y pueden ser presentados en tiempo real en la consola. En caso de que ocurra algún problema el supervisor llama a varios servicios (como el manejador de fallos) que hacen todo lo posible para resolver el problema (Ver Anexos 5).

Motor del Flujo de Trabajo (Workflow)

El motor de flujo de trabajo surge como respuesta al problema del modelado de negocio dentro de una organización. Este se encarga de automatizar la secuencia en que se ejecutan las tareas durante el desarrollo de un proceso.

Los procesos de negocio han cambiado y evolucionado, en un principio se ejecutaban en una o varias aplicaciones internas de la empresa. En la actualidad, han rebasado las fronteras de la empresa, abarcando otras empresas que en ocasiones se encuentran geográficamente distribuidas.

Cada una de estas empresas contiene sus propios procesos que pueden ser heterogéneos y complejos. Para representar este tipo de proceso de negocio se necesita una tecnología más potente que el motor de flujo de trabajo, que sea capaz de controlar las conversaciones de larga duración entre las entidades involucradas en el proceso de negocio, que controle y gestione los diferentes hilos de ejecución, ejecución paralela, control de errores, compensación de transacciones y soporte datos XML. Para satisfacer esta necesidad surge la Gestión de Procesos de Negocio o Manejador de Procesos de Negocio (BPM, siglas en Inglés).

Gestión de Proceso de Negocio (BPM)

Esta metodología es utilizada tanto desde el punto de vista tecnológico como de gestión. Visto desde la perspectiva de gestión es un enfoque estructurado que emplea métodos, políticas y métricas para gestionar y optimizar de manera continua las actividades y procesos de una organización.

Mientras que desde las perspectivas tecnológicas, agrupa una serie de herramientas software para modelar, ejecutar y monitorear los procesos de negocio. Además estandariza las interfaces entre los componentes tecnológicos, reduce los costos de integración de los sistemas y provee la lógica de negocio a nivel de proceso interconectando servicios reutilizables.

También permite encadenar los procesos, asegurando la mejora continua de los mismos y ganando en eficiencia. Logra la modificación rápida dando respuesta inmediata a las demandas y reduce los costos de mantenimiento.

¿Cómo se integra BPM en una Arquitectura SOA?

Todo proceso consta de un conjunto de tareas ejecutadas en un flujo controlado. Un BPM debe dar soporte de modelación, ejecución y monitorización a estos procesos de negocio. Para realizar cada una de estas acciones el BPM utiliza un grupo de estándares que son retomados en el capítulo 2.

Las soluciones BPM de modelado de procesos deben proveer las siguientes funcionalidades:

- Capturar procesos ya existentes de manera estructurada mediante algún tipo de notación y representar las relaciones existentes entre ellos.
- Definir nuevos procesos o modificar los ya existentes.
- Simulación de parámetros de procesos (tiempo de ejecución, costos) en función de las variables independientes del mismo.
- Automatización de la documentación del proceso modelado en formato fácilmente exportable.
- Plantillas de procesos predefinidos.

Para la ejecución de los procesos de negocio utiliza un motor de ejecución de procesos y propone un estándar para realizar el lenguaje de ejecución, este estándar es el encargado de convertir los diagramas del modelado en código ejecutable.

En la monitorización de procesos se obtiene información del negocio con el fin de identificar patrones de utilización, situaciones de riesgos, indicadores de desempeño entre otras acciones que permiten al negocio aprender de los continuos contratiempos rebelando cambios para mejorar su efectividad.

¿Cómo integra BPM la orquestación y la coreografía?

El BPM surge apoyado a la Arquitectura Orientada a Servicios para resolver problemas de distribución y heterogeneidad de los procesos. De ellos se obtiene un nuevo concepto denominado composición de procesos de negocio. La composición permite modelar el proceso de negocio y maximizar las potencialidades que brinda SOA a través de la integración de datos y aplicaciones.

La composición de procesos de negocio implica encontrar un mecanismo que posibilite que dos o más servicios cooperen entre sí para resolver requerimientos que van más allá del alcance de sus capacidades individuales (PABLO ALVEZ ;PATRICIA FOTI ;MARCO SCALONE 2006).

Los requerimientos que deben cumplir que van más allá de sus capacidades son:

- Interacciones asincrónicas con servicios: Posibilita crear procesos que transcurran durante largo período de tiempo.
- Interacciones simultáneas entre servicios: Introduce el procesamiento en paralelo lo que provoca un considerable aumento de la eficiencia.
- Manejo de excepciones: Tiene en cuenta la ocurrencia de los errores y provee mecanismos para manejarlos.
- Integridad transaccional: Deshace las acciones ya efectuadas por un proceso de larga duración en caso de que ocurra algún fallo.

Para resolver este problema los servicios se organizan en un flujo de trabajo mediante la orquestación y la coreografía definiendo un proceso de negocio.

La orquestación y la coreografía describen cómo crear procesos de negocio mediante la composición de servicios Web. **Orquestación** se refiere a un proceso de negocio ejecutable que interactúa con servicios Web internos o externos. Es decir, ella trata con el ordenamiento de la ejecución de los servicios Web en un flujo de procesos. Esta interacción ocurre a través de mensajes e incluye la lógica de negocio y el orden de ejecución de las tareas. La orquestación tiende a ser considerada una fuerza coordinadora.

A continuación se presenta un ejemplo: se quiere realizar una reservación de una habitación en un hotel a un cliente (proceso de negocio) consiguiendo el más económico pero a la vez que se ajuste a sus necesidades.

Para conocer cuál de los hoteles de la ciudad donde se encuentra el cliente cumple con las condiciones, la agencia deberá utilizar varios servicios de otras compañías para comparar condiciones y elegir la más adecuada. En el caso de la orquestación se refiere al orden en que se realizará ese proceso, primero encuentra el hotel, después busca la habitación. Se refiere además a los criterios que sigue la compañía para descartar los hoteles y las habitaciones o los criterios de calidad para la comparación de los hoteles ofertados y las habitaciones. La propia compañía se encarga de establecer cómo se llevará a cabo el proceso (lógica del proceso de negocio) (Ver Anexos 6).

La orquestación de servicios Web debe ser flexible, dinámica y adaptable para descubrir los cambios que necesite el negocio. La separación de la lógica de proceso de los servicios Web garantiza flexibilidad. El BPM logra esta separación, es aquí donde entra a funcionar el motor de ejecución o de orquestación que se mencionó a inicio de este epígrafe este motor a su vez utiliza un lenguaje de ejecución que es el que permite definir la lógica de orquestación entre los diferentes servicios Web.

La **coreografía** representa procesos de negocio abstractos que indican el orden del intercambio de mensajes entre aplicaciones. Normalmente los intercambios de mensajes ocurridos entre los servicios Web. Permite a cada una de las partes involucradas describir su parte en la interacción. Puede verse como un proceso público y no ejecutable. Público porque define un comportamiento común que todas las empresas participantes deben conocer, y no ejecutable porque es más bien un protocolo de negocio que dicta reglas para que las empresas implicadas puedan interactuar entre sí.

En el ejemplo puesto anteriormente la coreografía sería el orden con que se intercambian los mensajes entre la agencia y las compañías hoteleras. Es la secuencia de mensajes válidos entre los participantes que invocan el servicio Web.

Bus de Servicio de Empresa (ESB)

SOA propone la integración de diversas aplicaciones que en ocasiones se encuentran geográficamente distribuidas, separadas por cortafuegos (firewalls) o por políticas de seguridad. Para lograr este objetivo es necesario hacer uso de una infraestructura, de ahí que esta arquitectura introduce un tipo de middleware llamado Bus de Servicio de Empresa (ESB) (Ver Anexos 7).

Un ESB es una plataforma basada en estándar que combina mensajerías, servicios Web, transformación de los datos y enrutamientos inteligentes para conectar y coordinar confiablemente la interacción de diversas aplicaciones a través de empresas extendidas[1] con integridad transaccional. Puede ser utilizado en cualquier proyecto de integración, de ahí que una organización puede tener su propio control y autonomía local en un proyecto de integración individual y a su vez puede unirse a un proyecto mucho más grande.

En un bus de servicio de empresa si un segmento deja de funcionar no compromete el funcionamiento de los demás, pues los ESB pueden configurarse de manera que existan otros servicios, rutas y vías alternativas si esto sucede, esto hace que sea una infraestructura para ser usada en régimen de alta disponibilidad.

El ESB cumple con un grupo de características, que son importantes para lograr la penetrabilidad de las empresas que hacen uso de los servicios que brinda una SOA, estas características lo hacen un componente importante pero no imprescindible, entre ellas se encuentran:

- Es diseñado para actuar de intermediario entre los componentes SOA, las interfaces de los servicios y los procesos de negocio.
- Puede verse como un servicio intermediario, no para buscar en el registro los nuevos servicios disponibles sino para establecer la conexión entre procesos.

[1] Una empresa extendida es una organización y sus socios de negocio que se encuentran separados por límites del negocio o límites físicos. En este tipo de empresa, las aplicaciones pueden estar separadas geográficamente, por cortafuegos corporativos o incluso por política de seguridad interdepartamentales.

- Aplica una Arquitectura Orientada a Servicios (SOA) que soporta las interacciones entre aplicaciones de colaboración que se basan en estándares mejorados de mensajería XML y servicios Web. Lo que hace posible las interacciones entre áreas y entre unidades de negocio. Permite definir las interacciones con socios de negocio en términos empresariales concretos y sólidos en lugar de hacerlo en interfaces de aplicaciones débiles y complejas.
- Debe ofrecer el eje central de comunicación a nivel empresarial necesario para conectar aplicaciones de manera fiable a través de varios dominios geográficos, administrativos o de seguridad. Es capaz de conectar servicios basados en distintos lenguajes de programación, sistemas operativos, entornos de ejecución y protocolos de comunicación.
- Es diseñado para ser versátil, puede conectarse a varios tipos de navegadores, repositorios de metadatos, registros e interfaces de cualquier tipo de aplicaciones. Pues trata siempre de adaptarse a lo que el sistema que pide ser atendido sea capaz de manejar. Para este propósito el ESB tiene una amplia variedad de adaptadores.
- Reduce de manera considerable el tiempo de implementación y el costo total de propiedad de los proyectos de integración pues brinda soporte a los métodos y mecanismos estándares para interconectar y desarrollar aplicaciones a través de las empresas.
- Elimina la necesidad de programar el enrutamiento en el código de las aplicaciones o establecer relaciones rígidas entre dispositivos pues el ESB automatiza el enrutamiento de las transacciones empresariales en base a contenidos de documentos y reglas empresariales XML.
- Proporciona flexibilidad y permite una administración y escalabilidad de los servicios a nivel independiente, con el objetivo de lograr la mayor eficiencia operativa posible. La transparencia de ubicación permite que los servicios se actualicen, muevan o reemplacen sin necesidad de modificar los códigos de las aplicaciones. Incluyen la capacidad de configurar, implementar y administrar los servicios de manera centralizada.
- Enfoca la arquitectura basada en la utilización de eventos como disparadores que inician la distribución de un mensaje que informa a varios receptores acerca de un evento para que realicen las acciones pertinentes.

Una arquitectura basada en la utilización de eventos tienen las siguientes características:

- Desacopladas: la aplicación o servicio que envía el mensaje no sabe quienes están suscritos a ese tipo de mensaje. La relación se establece por el mensaje, pero no entre las aplicaciones.
- Mensajería publicar/suscribir: Esta característica de la arquitectura basada en eventos consiste en que un sistema publica información acerca de un evento determinado para que otro sistema que se haya suscrito y esté autorizado a recibir este mensaje pueda recibirlo y actuar en consecuencia.
- Asíncrona: Soporta mensajería asíncrona donde la información se envía sin que se espere una respuesta inmediata o se necesite mantener una conexión activa entre los dos sistemas mientras se espera la respuesta.

Mensaje Orientado a Middleware (MOM)

En una Arquitectura Orientada a Servicios también hay que resaltar que todas las diferentes piezas del software se comunican enviándose mensajes. Los mensajes son esenciales, deben ser entregados con rapidez y garantizar que lleguen a su destino. Un ESB también se encarga del trabajo con la mensajería de las empresas introduciendo un nuevo concepto, el Mensaje Orientado a Middleware (MOM).

Un MOM es una parte dominante en el ESB que proporciona el enmascaramiento en la red de los canales virtuales que ESB usa para enrutar los mensajes. Es un concepto que envuelve el paso de los datos entre dos aplicaciones usando un canal de comunicación que transporta unidades autocontenidas de información (mensajes). En un entorno de comunicación basado en MOM, los mensajes se envían y reciben de forma asincrónica. La coordinación de mensajes se realiza mediante un concepto denominado enrutamiento basado en itinerarios.

1.3.5 Calidad de los Servicios

La calidad de los servicios en SOA es uno de los aspectos que más preocupa a la hora de aplicar esta arquitectura. Para mantener la calidad de los servicios en SOA hay que tener en cuenta un grupo de conceptos asociados que con su cumplimiento impiden que se provoque fallas en los servicios y estos son:

- Política: Es un conjunto de reglas bajo las cuales, un proveedor de servicio hace que el servicio esté disponible para el cliente.

- Administración: Conjunto de atributos que podrían aplicarse para manejar los servicios proporcionados o consumidos.
- Transacción: Conjunto de atributos que podrían aplicarse a un grupo de servicios para devolver un conjunto de datos consistente.
- Seguridad: Es un conjunto de reglas que pueden aplicarse para la identificación, autorización y control de acceso a los consumidores de servicios.

Todos estos conceptos se tienen en cuenta a la hora de implementar una SOA pero el más importante de todos es la seguridad por tanto la presente investigación se enfoca más a la seguridad de los servicios para dar cumplimiento así a unas de las tareas específicas a desarrollar planteadas al inicio de la investigación.

Dentro de una empresa se manipulan una cierta cantidad de servicio que deben estar disponibles cuando se necesiten, ser consistentes, manejados de forma correcta y que solo pueden acceder a los servicios el personal autorizado, al no ser, que sean públicos para cualquier cliente.
Por lo que se hace necesario establecer reglas de seguridad que limiten las peticiones entre los consumidores de servicios y los proveedores, para dejar claro qué servicio puede ser accedido y por quién.

Seguridad de los Servicios
En la SOA obtener la seguridad de los servicios es un tanto más complicada, pues no solamente se tiene que lograr la autenticidad de los usuarios sino también de los sistemas. Cada uno de los servicio tiene que estar seguro que la respuesta que envía va encaminada a un servicio con autorización a recibir dicha información.

Para lograr esto es necesario estudiar cómo se comporta un sistema informático. Hay cuatro componentes fundamentales que conforman un sistema informático y uno de ellos es enteramente humano: el usuario del sistema. Los otros tres componentes son el software, los datos y el hardware que permite almacenar los datos y que corra el software (Ver Anexos 8).

Ahora bien, visto desde una perspectiva de seguridad, ¿cómo se debe manejar está interacción entre los componentes? Pues primeramente habría que preguntar ¿quién está solicitando un pedido? Cuando el usuario se autentifica para entrar en el sistema, usa un nombre y una contraseña, que pasan por un proceso de autenticación.

La autenticación de los usuarios puede ser débil o robusta

La autenticación débil utiliza la contraseña o procedimientos relativos, tales como, hacer una pregunta específica cuya respuesta solo la conoce una persona. Para este tipo de autenticación cualquier persona que conozca del tema le resulta muy fácil saltarse las barreras de seguridad, pero muchas autenticaciones son de este tipo y puede ser vista como buena para muchos propósitos.

La autenticación robusta se basa en tener algo que es único, puede ser una tarjeta que es usada personalmente. Si esa tarjeta se pierde o es robada solamente se denuncia y ya se le suspenden todos los derechos. Este tipo de autenticación robusta es la más común, pero se están reemplazando gradualmente, pues una persona tiene muchas cosas que son únicas como: la cara, la voz, las huellas digitales, la impresión de la palma de la mano y la retina ocular. La autentificación robusta es, como las palabras pueden sugerir, muy difícil de romper.

¿Puedes dejar hacer lo que te piden?

Si ya se conoce con certeza quién esta haciendo la petición lo único que hay que hacer es ver si se le puede permitir hacer lo que está solicitando. Pero descubrir, si este usuario tiene autorización para hacer lo que está pidiendo, no es tan sencillo.

En ocasiones el usuario invoca tener acceso a ciertos datos haciendo uso solamente de algunos recursos de la computadora, este proceso suele ser muy simple, las complicaciones surgen cuando para darle respuesta a la petición del usuario se necesita conectar con varias computadoras, con otros sistemas y hacer uso de la información almacenada en varios sistemas de base de datos.

Aunque se puede manejar la situación, cabe la posibilidad que el usuario intente hacer algo que él no tenga autorización para hacer. Por tanto hay que establecer qué derechos tiene un usuario sobre cada una de las acciones que esté autorizado a realizar, para eso está el sistema de manejo de identidades.

El sistema de manejo de identidades (Ver Anexos 9)
Maneja los derechos y permisos que el usuario tiene, así que el usuario puede hacer solo lo que está autorizado a hacer. El punto es que provee un servicio de seguridad que puede abarcar la red, incluso redes múltiples, esto es muy utilizado en SOA porque ella verdaderamente necesita tales servicios, aunque el software de manejo de identidades no fue diseñado pensando en SOA se ajusta muy bien a sus necesidades.

En la situación de esta arquitectura que maneja numerosos componentes ¿cómo podemos realizar esto? Un usuario puede que no tenga autorización para utilizar una determinada aplicación pero quiere acceder a un componente particular de la misma. No se puede dar una contraseña de esa aplicación porque entonces tendrá acceso a todos los componentes que la conforman y puede realizar actividades ilegales en ellos. Y en caso de que no se le comunique con claridad a la aplicación quién es ese usuario, si existe cualquier problema no se tiene constancia de quién tuvo acceso y quién no.

Por tanto para poner SOA en ejecución hay que tener un manejador de identidades en ejecución. Es recomendable crear una política de seguridad y guardarla en el registro SOA. Para evitar todo tipo de percances, debe de otorgarse una credencial para todos los componentes.

La credencial se implementa mediante un token[2] de seguridad creado por el manejador de identificaciones (que contiene la credencial del usuario, incluye la identificación y los derechos de acceso) y que se encuentra encriptado, el token solo puede ser leído por los componentes involucrados. El mismo es enviado por el servicio intermediario a todos los componentes que el usuario tiene acceso. Así cada componente puede saber “quién esta haciendo qué”.

[2] Símbolo o Etiqueta que contiene información del cliente del servicio.

Autenticando software y datos

En la Arquitectura Orientada a Servicios un usuario puede ser (y es a menudo) el software que atraviesa varias computadoras y que accede a varios datos. En el mundo de la informática todo es posible, cualquier persona puede hacer uso inapropiado de la información a la que tenga acceso sin previa autorización, lo mismo ocurre con los software, estos pueden hacer uso de los servicios que estén a su alcance y manipularlos, provocando que la información deje de ser segura.

Mediante el manejador de identificaciones puede estar seguro de que conoces la identidad de cualquiera que este haciendo cualquier cosa en un sistema ¿pero qué se puede hacer para el software? Para el software podemos hacer uso de las **Certificaciones Digitales**. La certificación digital es virtualmente imposible de forzar, así cuando se recibe un software con la certificación digital se puede saber con certeza de quién proviene y determinar si esta fuente es confiable o no.

Una certificación digital contiene:

- Nombre (de una compañía o persona).
- Un número de serie.
- La fecha de expiración del certificado.
- Una copia de la clave pública de encriptación del proveedor.
- La firma digital de la autoridad competente.

La autoridad competente en este caso es una tercera parte independiente y confiable que se encarga de chequear las credenciales de las compañías y le otorga una certificación para usar. De hecho, la autoridad competente es quién asegura que ese nombre (compañía o persona) tiene esa clave pública.

El esquema de cifrado PKI (Public key Infrastructure) es utilizado en innumerables situaciones de seguridad y también proporciona una manera de validar que los datos vengan de una fuente confiable y no hayan sido interceptados, ni cambiados. En este tipo de cifrado, el emisor del mensaje calcula un digesto en base al mensaje. Este es encriptado mediante una clave que solo el que emite el mensaje conoce (clave privada) y es añadido como parte del mensaje junto a una clave que el receptor necesita para desencriptarlo (clave pública). El receptor del mensaje recalcula el digesto y lo compara con el recibido. Si los valores no coinciden entonces el mensaje fue alterado.

Usando el ESB para realizar auditorias

Las defensas de seguridad de SOA consisten primeramente en la autenticación, si se conoce con certeza quien de nuestros usuario está trabajando en cada momento, si se conoce con seguridad que los software, tanto internos como externos, así como los datos que recibes son auténticos, solo se necesita una cosa más para estar seguro de que no te pueden estafar y es que los usuario utilicen los software de forma legítima.

Realizando auditorias se puede prevenir ataques internos, pero para realizar las mismas se necesita de una persona confiable, si el contrato de los trabajadores de la empresa no fue lo suficientemente riguroso puede traer problemas. Todas las acciones que se realizan en un sistema dejan rastros de intervención de ejecución que describen qué fue lo que se hizo y cuándo se hizo. Los rastros de intervención de la ejecución son como cámaras de vigilancias de los sistemas informáticos.

En realidad tienes que confiar en alguien para realizar las auditorias. Un problema con las auditorias en SOA es que las operaciones con los servicios de negocio se dividen a través de varios componentes. El uso del Bus de Servicio de Empresas (ESB) puede resolver el problema de las auditorias, pues el ESB puede guardar un rastro de intervención de todos los mensajes que son enviados.

Por tanto para tratar todos estos riesgos no se pueden olvidar estos tres aspectos fundamentales que garantizan la seguridad en SOA:

- Manejador de identificaciones.
- Autenticación de Software y datos.
- Las auditorias.

1.3.6 Problema y Situación Problémica

En estos momentos, en la Universidad de las Ciencias Informáticas (UCI) se manejan innumerables procesos de negocio y van en aumento entre los cuales se encuentran: el directorio, comedores, identificación, servicios generales, servicios telemáticos, gestión tecnológica, control de acceso, portal de la intranet, reservaciones y transporte entre otros que se encuentran en actual funcionamiento en la universidad. Para almacenar toda esta información existen tres bases de datos fundamentales que son UCI trabajadores[3], Akademos[4], Asset[5]; de las tres mencionadas se realiza una recopilación de la información común y se almacenan en la base de datos Ciudadanos, todas ellas se encuentran relacionadas entre sí e intercambian información constantemente.

Debido a los continuos cambios que existen en la universidad, las cuatro bases de datos se encuentran inmersas en una lluvia de actualizaciones diarias. Este proceso de actualizaciones se realiza a través de un sistema que corre automáticamente, haciendo copias de la información de las tres primeras mencionadas para la de Ciudadanos, trayendo consigo réplica de la información y además que dicha operación se puede realizar solo una vez al día, sobre todo en las noches que el tráfico en la red es mínimo para poder transferir todo ese volumen de información sin que se produzca congestión en la red. Cualquier cambio que se realice durante el día en cualquiera de las bases de datos no será actualizado en las demás hasta la noche.

[3] Contiene información de personas que trabajan en la UCI, pero no pertenecen a la misma, sino a un área externa y contiene la información de los eventuales (personas que trabajan por tiempo limitado).

[4] Contiene los datos personales de los estudiantes y la gestión académica de los cursos y las asignaturas.

[5] Contiene los datos de los trabajadores que son plantillas de la UCI y la información de las áreas pertenecientes a la UCI.

Muchos de las softwares que funcionan en la universidad necesitan información de estas bases de datos. Cada uno de ellos se conecta directamente, violando el principio[6] de que ninguna aplicación externa debe tener acceso directo a la misma. Este intercambio directo produce réplica de las funcionalidades de las aplicaciones ya que todas tienen que implementar consultas que generalmente son iguales. Cuando se hace algún cambio en una base de datos hay que emplear mucho tiempo en cambiar dichas consultas porque existe gran dependencia.

Las facultades de la universidad tienen un perfil y se especializan en diferentes plataformas y lenguajes de programaciones, las mismas han desarrollado proyectos utilizando servicios Web pero esto no es suficiente porque los desarrolladores no han utilizado estándares de comunicación por lo que muchas aplicaciones no pueden utilizar estos servicios porque están desarrolladas en diferentes lenguajes y no son compatibles o simplemente no saben que existe porque no ha sido publicada su ubicación.

La falta de interoperatividad entre las aplicaciones en funcionamiento impide la reutilización de funciones y no existe comunicación ni prestación de servicios entre las mismas. Al desarrollar una nueva aplicación se encuentra que alguna de las funcionalidades han sido implementadas por aplicaciones antes desarrolladas y entonces se presenta una disyuntiva, copiar el código o volver a implementar esas funcionalidades. Pero en muchas ocasiones solo se dispone de la segunda opción pues quien desarrolló la aplicación no se encuentra en la universidad o no está programada en el mismo lenguaje.

Esta situación que existe actualmente en la UCI provoca que no se brinden los servicios con la calidad y la rapidez requerida. Cuando ingresa cualquier persona a la universidad no puede pasar al comedor con el solapín hasta el día siguiente y cuando se da baja sigue apareciendo como personal de la universidad, incluso en el directorio de búsqueda todavía aparecen personas que hace tiempo no pertenecen al centro.

[6] El principio de las bases de datos es que las aplicaciones deben verlas como cajas negras, es decir solamente deben ser capaz de saber cómo utilizar lo que ellas brindan y no ver su contenido como tal.

1.3.7 Fundamentación del Objetivo

Debido a los problemas antes mencionados surge la necesidad de realizar una propuesta de desarrollo que comprenda cómo llevar a cabo una Arquitectura Orientada a Servicios en la universidad, la propuesta debe tener en cuenta la utilización de estándares y sobre todo la seguridad de los servicios que serán obtenidos. Esto permite agilizar el trabajo de los desarrolladores, pues solamente van a tener que implementar el resultado de un estudio exhaustivo realizado basándose en los problemas y necesidades de los procesos de negocio de la UCI y que además está propuesta se apoya en las mejores prácticas de la SOA que hasta la actualidad se hayan realizados en las diferentes compañías que la han llevado a cabo (IBM, OASIS, SUN, BEA, Microsoft).

1.4 Conclusiones Parciales

- El manejo de las aplicaciones en la Universidad de las Ciencias Informáticas no es óptimo, pues las características cambiantes de sus procesos y la heterogeneidad en los lenguajes utilizados y las plataformas de desarrollo la obligan a buscar otra alternativa para desarrollarlos.
- El estudio teórico de las Arquitectura Orientada a Servicios, facilitó el aprendizaje de la información imprescindible que se precisa para la comprensión de los capítulos posteriores.

CAPÍTULO 2 ESTÁNDARES DE SOA

2.1 Introducción

Hasta ahora los desarrolladores habían empleado sus propias tecnologías de gestión de procesos de negocio, de diseño y de implementación para sus aplicaciones. Con la adopción de la Arquitectura Orientada a Servicios (SOA) como forma más eficiente para la creación de aplicaciones más interoperables y flexibles, la utilización de estándares se ha convertido en un factor decisivo a la hora de desarrollar software. De ahí que, debido al papel protagónico que juegan estos estándares fue preciso dedicarles el presente capítulo.

2.2 Lenguaje de datos

Uno de los objetivos de SOA es desglosar las aplicaciones en servicios y luego orquestarlos para formar un proceso de negocio. Para lograr que varios servicios puedan establecer comunicación y trabajen juntos, es necesario que su contenido sea entendido por todos los que intervengan en el negocio. SOA utiliza un estándar como base para el intercambio, el Lenguaje Extensible de Marcado (XML).

2.2.1 Lenguaje Extensible de Marcado (Extensible Markup Language o XML)

El lenguaje Extensible de Marcado es adoptado como estándar para el intercambio de datos a través de la red. Utiliza etiquetas para señalar y describir el contenido de un documento. Un documento XML puede ser enviado a otras aplicaciones que pueden entenderlo haciendo un seguimiento de las etiquetas.

Cada documento tiene asociado un esquema que define las reglas de la gramática, es decir, especifica que etiquetas son permitidas en el documento, la estructura de esas etiquetas, así como qué tipo de datos son esperados.

El autor de un documento XML puede crear sus propias etiquetas y especificar las características de las mismas en el esquema XML para que puedan ser entendidas por otras aplicaciones.

Durante la elaboración de un documento XML cada etiqueta que se abre tiene obligatoriamente una de cierre, si contiene múltiples etiquetas se van cerrando de adentro hacia afuera.

Como resultado XML es adoptado como el lenguaje de datos de los servicios Web. Durante la descripción de los demás estándares en epígrafes posteriores se puede ver que la mayoría de ellos están basados en XML.

XML es un lenguaje utilizado para la elaboración de documentos que puedan ser leídos por otras aplicaciones. Esas otras aplicaciones en ocasiones necesitan seleccionar alguna información puntual dentro del documento y para este tipo de acciones XML provee dos lenguajes estándares.

2.2.1.1 Lenguaje de Caminos XML (XPath o XML Path Language)

Este Lenguaje se utiliza para direccionar partes de un documento XML. Provee manipulación de cadenas, números y booleanos. Modela un documento como árbol de nodos que pueden ser nodos elementos, nodos atributos y nodos texto.

XPath utiliza expresiones para manipular objetos que pertenecen a los siguientes tipos de elementos:

- Conjunto de nodos (una colección desordenada de nodos sin duplicados).
- booleano (verdadero o falso).
- número (un número en punto flotante).
- cadena (una secuencia de caracteres).

XPath utiliza un número de funciones definidas para realizar el direccionamiento y para navegar a través de un documento. Estas funciones pueden recibir cero o más atributos y devuelven un único atributo y tanto los valores de entrada como los de salida pertenecen a alguno de los cuatro tipos básicos que se vieron anteriormente.

2.2.1.2 Lenguaje de Plantilla Extensible para Transformación (XSLT o eXtensible Stylesheet Language for Transformations)

Este lenguaje define cómo transformar un documento XML a un documento HTML, texto o XML. Un documento (hoja) XSLT define un conjunto de reglas que son aplicadas a otros documentos XML. Es interpretado por un procesador (motor de plantillas)[7] que aplica las reglas definidas en él al documento que se desea transformar. Una hoja XSLT contiene tres tipos de elementos:

- Elementos de XSLT: Son el equivalente a las palabras claves del lenguaje de programación.
- Elementos LRE (Literal Result Elements): Son elementos que no pertenecen a XSLT, sino que se repiten en la salida (ejemplo <fecha>).
- Elementos de extensión: Elementos no estándares que son manejados por implementaciones concretas del procesador.

XSLT se apoya en XPath para navegar dentro del documento XML y para realizar las consultas necesarias dentro del mismo, por ejemplo si se quiere obtener un nuevo documento XML o HTML pero que contenga solamente parte de la información del documento original, se crea el documento XSLT y dentro utilizando XPath se selecciona que información se desea mostrar en el nuevo documento (Ver Anexos 10).

El Lenguaje Extensible de Marcado (XML) es utilizado como base a partir de ahora por la mayoría de los estándares que son expuestos a lo largo del capítulo.

2.3 Estándares de Descripción

En SOA los clientes se comunican con los servicios a través de la red, por tanto cada servicio debe presentar una descripción de sus características y su localización. Para esta descripción es utilizado como estándar el Lenguaje de Descripción de Servicios Web (WSDL).

[7] Es un software que permite la transformación de documentos XML. Recibe un documento plantilla y un documento de datos y aplica las reglas del documento plantilla al documento de datos y obtiene como resultado un tercer documento.

2.3.1 Lenguaje de Descripción de Servicios Web (WSDL o Web Services Description language)

El documento WSDL describe el servicio Web como una colección de elementos llamados “ports” o “puertos” capaces de intercambiar mensajes. Cada puerto presenta una definición abstracta (port type) y una definición concreta (binding). El “port type” es el conjunto de operaciones y mensajes que brinda un servicio Web, mientras el “binding” define la localización de red y los tipos de datos para un “port type” determinado.

Es posible definir un documento WSDL que contenga la definición de los “port type” sin el “binding” correspondiente. Esto posibilita la reutilización de los “port type”. Es decir, varios servicios Web pueden implementar la misma descripción WSDL, solamente agregando la localización y los tipos de datos (binding) (Ver Anexos 11).

Cuando el cliente descubre el documento WSDL está en condiciones de establecer la invocación al servicio pues ya conoce donde se localiza el servicio, qué funciones brinda y qué formato de mensajes acepta.

En resumen el documento WSDL se divide en cuatro partes fundamentalmente:

- Definición de puerto: Describe qué y cómo se deben hacer las cosas después de conectarse al servicio Web, es decir, las operaciones que pueden ser solicitadas y los mensajes que se pueden usar.
- Definición de Mensajes: Contiene los datos que acepta cada operación. Posibilita al cliente elaborar la petición correcta y entender la respuesta que recibirá.
- Definición de tipos: Define los tipos de datos que son usados por el servicio Web.
- Definición de binding: Define cómo puede conectarse al servicio.

Al desarrollar un servicio el programador no tiene que elaborar este documento manualmente pues existen un conjunto de herramientas que generan el mismo. Estas son algunas de estas herramientas:

- VisualStudio.Net.
- IBM Web Services Toolkit.
- Apache Web Services Toolkit.
- Microsoft Soap Toolkit V2 Rc 0.

2.4 Estándares de Descubrimiento

En el mundo de SOA implementar los servicios utilizando interfaces estándares y elaborar una descripción de los mismos, no es suficiente, hay que publicar esa descripción en algún lugar donde pueda ser encontrada por las demás aplicaciones. Ese lugar es el registro SOA y para lograr que el mismo sea accesible por todas las aplicaciones se crea utilizando un estándar.

2.4.1 Descripción, Descubrimiento e Integración Universal (UDDI o Universal Description, Discovery and Integration)

UDDI es un registro donde se publica y descubre información referente a los servicios, información técnica de las características de los servicios y sus interfaces de comunicación y las reglas de la empresa para el uso de los mismos.

Es un registro público para que las empresas puedan acceder a él en busca de los servicios que necesitan o publicar los suyos. Brinda dos interfaces una para el proveedor de servicios y otra para el consumidor de dichos servicios. Los proveedores utilizan la interfaz denominada “Publisher Interface” para publicar sus servicios y los consumidores utilizan “Inquiry Interface” para hacer búsquedas de servicios. Permite la interacción en tiempo de diseño y en tiempo de ejecución.

Está estrechamente relacionada con el estándar de descripción WSDL. Ella es quien posibilita que las empresas coloquen sus archivos WSDL en la red, así los usuarios pueden conocer cuáles son los servicios que brinda cada empresa y además encontrar en la descripción WSDL la forma de invocar esos servicios.

La publicación de los servicios en la UDDI no es un proceso muy complejo. Primero se define el archivo WSDL con todas las características de la empresa y de los servicios que se van a publicar, luego se procede a autenticarse como usuario de la UDDI, después se publica el documento en la UDDI ya sea mediante programación o utilizando alguna interfaz de usuario y por último se realiza una entrada para probar si se registró correctamente.

Cada compañía puede implementar su propio directorio UDDI. Actualmente existen herramientas que permiten desarrollar una UDDI para una intranet. “Microsoft UDDI Services” para Windows 2000 Server es una de esas herramientas.

Como se puede apreciar UDDI es la forma estándar de implementar un registro SOA, es dónde se va a encontrar la descripción de los servicios y la forma de conectarse a ellos. Puede implementarse a través de una página Web o utilizando algunos softwares que permitan crearla.

2.4.2 Lenguaje de Inspección de los Servicios Web (WSIL o Web Service Inspection Language)

Pero la UDDI a pesar de ser un estándar que es usado por la mayoría de las empresas que implementan SOA o desean utilizar o publicar un servicio presenta varios problemas que no puede corregir por sí misma sino que tiene que auxiliarse de otro estándar para lograrlo.

El Lenguaje de Inspección de los Servicios Web (WSIL) es el mecanismo adicional usado para corregir las deficiencias de UDDI que se muestran a continuación:

- Falta de moderación: La escalabilidad con que fue diseñada provoca que en ocasiones se publiquen WSDL inválidos o duplicados, e incluso se publiquen servicios Web no disponibles.
- Falta de calidad de servicio: No existe una garantía de que el servicio reúne la calidad requerida y cumple con las funcionalidades especificadas en el documento WSDL, porque UDDI es un directorio público en Internet.

Un documento WSIL es un puntero a una lista de servicios que permite a los clientes examinar los servicios publicados. Aunque al igual que UDDI su objetivo es garantizar la publicación y el descubrimiento de servicios a través de la red, WSIL tiene la ventaja de que propone un modelo descentralizado, la información puede ser distribuida hacia diferentes lugares utilizando XML.

Es como una tarjeta de presentación que emite el proveedor del servicio, es necesario conocer la URL[8] del documento WSDL para poder crearlo, por tanto, puede ser generado en cualquier momento del ciclo de desarrollo de un servicio Web siempre y cuando ya halla sido elaborado el documento WSDL.

2.5 Estándares de modelación de procesos de negocio

Después de que el cliente descubre en la UDDI los servicios que necesita, conoce las características del mismo mediante el documento WSDL y tiene definido los procesos de negocio que desea implementar, pues comienza modelar dichos procesos de negocio utilizando los sistemas de gestión de procesos de negocio (BPM). Estos sistemas trabajan utilizando estándares y a continuación se muestran los principales.

2.5.1 Notación de Modelado de Procesos de Negocio (BPMN Business Process Modeling Notation)

BPM utiliza el estándar BPMN para diseñar los procesos de negocio, su principal objetivo es obtener una notación gráfica entendible para los directivos, analistas y desarrolladores técnicos del negocio. Se considera como la parte abstracta en la modelación de un proceso de negocio.

Este estándar clasifica los procesos en tres tipos de submodelos:

- Procesos privados de negocio (internos).
- Procesos abstractos (públicos).
- Procesos de colaboración (globales).

Procesos privados de negocio (internos): Son aquellos procesos que satisfacen requisitos propios de una entidad específica y que no pueden exceder las fronteras de la entidad.

[8] Localizador Universal de Recursos

Procesos abstractos (públicos): Son los procesos que permiten a los procesos privados interactuar con otros procesos fuera de la entidad. Contienen las actividades utilizadas para comunicar los procesos privados con el exterior. Los procesos abstractos son quienes muestran al exterior la secuencia de mensajes necesarios para comunicarse con los procesos privados.

Procesos de colaboración (globales): Definen la interacción entre dos o más empresas como una secuencia de actividades que representan el patrón de intercambio de mensajes.

Los participantes en la interacción son representados como una senda (calle) dentro de la entidad. Cada senda puede contener dos participantes y la dirección del intercambio entre ellos. También pueden ser representados dos procesos abstractos que interactúan a través de un flujo de mensajes.

Para modelar los procesos de negocio, BPMN clasifica los elementos del proceso en cuatro categorías básicas (Ver Anexos 12):

- Objetos de Flujo.
    1. Eventos
    2. Actividades
    3. Nodos de decisión/unión
- Objetos de Conexión.
    1. Flujo de Secuencias
    2. Flujo de mensajes
    3. Asociación
- Roles (swimlane)
    1. Pool (entidad/rol)
    2. Lane (participantes dentro Pool)
- Artefactos
    1. Objeto de datos
    2. Anotación de texto
    3. Grupo

A continuación se explican los elementos fundamentales de un Diagrama de Proceso de Negocio (Ver Anexos 13).

- Eventos: Clasifica los tipos de eventos que pueden ocurrir durante un proceso de negocio como eventos de inicio, intermedios, de fin, de mensajes, de excepciones, de compensación, de tiempo y provee una representación para cada uno. Puede indicar también que ocurre cuando un proceso es iniciado.
- Tareas o actividades atómicas: Son las actividades o tareas que son ejecutadas durante el proceso de negocio.
- Subprocesos: Son secuencias de actividades que se ejecutan dentro de un proceso.
- Gateways o Nodos de decisión: Hace uso de las sentencias lógicas (AND, OR, XOR) para determinar que camino toma la ejecución del proceso.
- Flujo de secuencias: Es representado a través de flechas que van de un nodo a otro y pueden ser condicionales. Contiene varios tipos de flechas de flujo de secuencias

Para modelar un flujo de un procesos de negocio en BPMN (Notación de Modelado de Procesos de Negocio), se modelan los eventos que dan inicio al proceso (eventos de inicio), luego los procesos o tareas que se ejecutan durante el proceso y finalmente el resultado final obtenido.

Un proceso puede ser dividido en subprocesos para facilitar su entendimiento. Estos subprocesos se pueden modelar en diagramas individuales y luego relacionarlo con el proceso principal mediante hipervínculos. Esto evita la sobrecarga de los diagramas (Ver Anexos 14).

El BPMN permite especificar quién ejecuta cada acción mediante la inclusión de los Pools y Lanes (sendas). Los Lanes son divisiones dentro de los Pools para lograr una mayor especificación, por ejemplo un Pool representa una empresa y los Lanes dentro del mismo representan sus departamentos. BPMN posibilita la modelación de procesos de negocio muy complejos y mapear fácilmente estos modelos a un lenguaje ejecutable.

2.5.2 Lenguaje Unificado de Modelado (UML o Unified Modeling Language)

UML es un lenguaje gráfico para especificar, construir y documentar los artefactos que modelan un sistema. Fue diseñado para ser un lenguaje de modelado de propósito general, por lo que puede utilizarse para especificar la mayoría de los sistemas basados en objetos o en componentes, y para modelar aplicaciones de muy diversos dominios de aplicación (telecomunicaciones, comercio, sanidad, etc.) y plataformas de objetos distribuidos (como por ejemplo J2EE, .NET o CORBA) (PABLO ALVEZ ;PATRICIA FOTI ;MARCO SCALONE 2006).

Es un lenguaje que proporciona flexibilidad en el modelado de sistemas informáticos. UML es un lenguaje para la modelación de sistemas Orientado-Objeto (OO). Pero puede ser personalizado para ser usado también en la modelación de sistemas basados en componentes, modelación de procesos de negocio y diseño de otros sistemas.

Los estereotipos son una forma de caracterizar los elementos de un modelo. Las herramientas CASE[9] haciendo uso de la flexibilidad de UML posibilitan crear nuevos estereotipos o modificar los existentes para crear un nuevo perfil. Un perfil define un conjunto específico de extensiones para representar un dominio particular.

Haciendo uso de la versión 2.0 de UML se puede modelar procesos de negocio. Esta nueva versión incorpora un grupo de nuevos diagrama y estereotipos factibles para la modelación de procesos. UML como se ha dicho no fue diseñado para modelar procesos de negocio ni aplicaciones basadas en servicios. Pero se puede hacer uso de algunos de sus diagramas para modelar este tipo de aplicaciones.

- Diagrama de actividades.
- Diagrama de secuencia.
- Diagrama de transición de estados.
- Diagrama de componentes.
- Diagrama de Clases.

[9] Computer Aided Software Engineering o Ingeniería Software Asistida por Ordenador: Son herramientas informáticas que permiten el desarrollo de software.

La utilización de UML brinda los siguientes beneficios:

- Es un lenguaje conocido.
- Estándar
- Fácil de aprender.

También presenta algunas desventajas:

- No ha sido diseñado para modelar procesos de negocio.
- Implica un enfoque Orientado a Objeto.
- Solo lo conocen los expertos en IT[10].

Existen herramientas que permiten que los diagramas UML puedan ser convertidos a lenguaje ejecutable. Esto lo convierte en uno de los lenguajes candidatos a la hora de modelar una aplicación SOA.

Estos diagramas modelados con BPMN o UML 2.0 son mapeados a código ejecutable para que sean interpretados por motores de ejecución procesos. Existen dos lenguajes estándares que son los más utilizados en estos códigos ejecutables. Estos son descritos en los epígrafes siguientes.

2.5.3 Lenguaje de Ejecución de Procesos de Negocio (BPEL o Business Process Execution Language)

BPEL fue creado por IBM y Microsoft, pero luego en el 2003 fue aprobado por el comité de estándares de OASIS que se le llamara WS-BPEL (Web Service Business Process Execution Language) para que siguiera la nomenclatura de los estándares de WS (Web Service). En este documento se referencia como BPEL solamente por una cuestión de comodidad pero se refiere a WS-BPEL.

BPEL es un lenguaje ejecutable para la modelación de procesos de negocio basado en XML. Permite desarrollar aplicaciones más escalables y flexibles frente a los cambios de negocio. Modela dos tipos de procesos los procesos abstractos y los procesos ejecutables. Además establece cómo un proceso de negocio puede ser ejecutado utilizando servicios Web.

[10] Tecnologías de la Información

Un proceso abstracto de BPEL modela información que representa comportamientos dentro de un proceso (por ejemplo: cuándo esperar o enviar un mensaje o cuándo compensar una falla). Este tipo de modelación recibe el nombre de “programming in the large”.

Los procesos de corta vida o procesos ejecutables son modelados como una simple transición (por ejemplo: acceso a recursos como ficheros o bases de datos). Este tipo de modelación recibe el nombre de “Programming in the small”.

BPEL define un conjunto de tareas básicas:

- Tareas de Invocación (Invoke): Invocación de operaciones en un servicio web.
- Tareas de Recepción (Receive): Permite el bloqueo de un proceso a la espera de llegada de un mensaje.
- Tareas de Respuesta (Response): Permite enviar un mensaje en respuesta a un mensaje recibido previamente.
- Tareas de Espera (Wait): Permite la espera durante un tiempo del proceso.
- Tareas de Asignación (Assign): Permite copiar datos de un lugar a otro.
- Tareas de Lanzamiento (Throw): Permite indicar que ha ocurrido un error.
- Tareas de Finalización (End): Permite finalizar la orquestación de la instancia en curso.

También define tareas estructuradas, estas son la combinación de las básicas para formar tareas más complejas:

- Tareas secuenciales (sequence): Define un orden secuencial de tareas.
- Tareas de decisión (switch): Permite seleccionar un camino en particular en base a una condición.
- Tareas de elección*:* Permite bloquear y esperar la llegada de un mensaje o establecer un tiempo límite de espera (timeout). Cuando uno de los eventos ocurre, se ejecutan las tareas designadas.
- Tarea repetitiva (While): Permite repetir un grupo de tareas mientras se cumpla una determinada condición.
- Tareas paralelas: Permite ejecutar tareas en paralelo.

BPEL utiliza los documentos WSDL para implementar la orquestación entre los servicios. Durante la definición de un proceso utilizando BPEL se trata también con extensiones WSDL para soportar el intercambio de mensajes entre los participantes en el proceso. WSDL define tipos abstractos, los mensajes, las operaciones y los “port type”, también especifica cómo y dónde invocar una operación. Se puede decir que WSDL define las invocaciones y BPEL las orquesta. Para el acceso y manejo de datos BPEL utiliza los estándares XPath y XSTL.

2.5.4 Lenguaje de Descripción de Procesos (XPDL o XML Processing Description Language)

XPDL es un estándar desarrollado por el consorcio WFMC (Workflow Management Coalition). Permite crear código ejecutable aunque no fue diseñado con este fin. Su propósito es definir plantillas de procesos y luego desplegarlas en diferentes herramientas. Los flujos de trabajos definidos por XPDL pueden ser ejecutados por un motor de procesos**.**

Este lenguaje provee un fichero XML que pueda ser usado para el intercambio de modelos de procesos entre herramientas y puede ser usado también en conjunto con otras notaciones gráficas, no solo con BPMN.

Algunas herramientas permiten modelar en BPMN y exportar en formato XPDL. Estas herramientas toman la notación gráfica de BPMN y generan un código XPDL que puede ser interpretado por un motor de ejecución de procesos o por otra herramienta de diseño.

Pero la principal función de XPDL es actuar como intermediario, posibilita a una herramienta de diseño de un fabricante construir los diagramas y que estos puedan ser leídos o modificados en herramientas de otro fabricante. Aunque los ficheros XPDL pueden ser interpretados por los motores de ejecución no proveen una semántica de ejecución precisa y confiable (Ver Anexos 15).

2.5.5 WS-CDL (Choreography Description Language)

Se había visto con anterioridad que la orquestación se encarga de ordenar y controlar la ejecución de los diferentes servicios que intervienen en un proceso de negocio y que existe otro concepto que se encarga de la coordinación de los mensajes entre esos servicios que es la coreografía. Para desarrollar la coreografía en SOA se utilizan estándares también y el más usado es WS-CDL.

Lenguaje de Descripción de Coreografía (WS-CDL según sus siglas en ingles)

El WS-CDL es un estándar desarrollado por el consorcio W3C[11]. El concepto principal en WS-CDL es la interacción entre comprador (buyer), el vendedor de servicios (seller) y el fletador (shipper), pues se encarga del ordenamiento y la secuenciación de mensajes de esta interacción. Provee una vista integral de la interacción en términos de roles e intercambio de mensajes entre los participantes. Se encarga del intercambio entre las partes, no de los procesos individuales ejecutados por cada participante.

Los roles y sus intercambios están manejados por implementaciones que pudieran ser realizadas a través de BPEL o cualquier otro proceso tecnológico. Los procesos individuales son modelados por BPEL y ejecutados por un motor de procesos y WS-CDL los presenta a todos juntos en una coreografía que indica como pueden comunicarse a través de las actividades de interacción y mensajes especificados en el fichero WS-CDL.

Un archivo WS-CDL contiene la siguiente información:

- Los tipos de roles y el rol que desempeña cada participante: Cada participante va a jugar un rol.
- Relaciones: Define las relaciones dependiendo de rol.
- Coreografía, interacción y canales: Incluye las actividades intercambiadas por los participantes y las interacciones entre las partes. Los canales son el vínculo entre las partes y permiten el flujo de la información.
- Control de estructuras y actividades: Define el control y el orden de las actividades.

WS-CDL define un documento que puede verse como una guía o una política que rige el intercambio de mensajes durante una conversación entre cliente y proveedor de servicios.

2.6 Estándares de Transporte

Una vez orquestados los servicios que intervienen en el proceso de negocio, definida la coreografía en la interacción entre los mismos, entran a jugar su papel los mensajes que posibilitan el intercambio de información. Estos mensajes emplean protocolos estándares.

[11] Consorcio Word Wide Web: Es un consorcio reconocido a nivel internacional donde las organizaciones miembros trabajan conjuntamente para desarrollar estándares. Su misión es guiar la Web hacia su máximo potencial a través del desarrollo de protocolos y pautas.

2.6.1 Protocolo de Acceso a Servicios (SOAP o Services Object Access Protocol)

Este estándar es basado en XML, permite la transmisión de información. El elemento básico en esta transmisión es el mensaje SOAP (Ver Anexos 16). Un mensaje está compuesto por tres partes fundamentales y una cuarta que se omite en la mayoría de los mensajes:

- El sobre o envoltura: Encapsula el contenido del mensaje, identifica si es un mensaje SOAP o cualquier otro tipo de mensaje.
- El encabezado: Contiene las extensiones definidas para usar en SOAP. Puede invocar actividades adicionales como autenticación, también contiene información de seguridad como los tokens y puede que algunos mensajes no porten ningún encabezado.
- Cuerpo: Contiene el mensaje, ya sea la petición de servicio o la respuesta.
- Fallos: Es el responsable de los mensajes de error que genere el mensaje. En la mayoría de los mensajes no aparece.

Por ser basado en XML es independiente de sistemas operativos o plataformas y puede viajar a través de cualquier protocolo de comunicación (ejemplo: HTTP o TCP/IP). Aunque en SOA lo más común que el mensaje SOAP viaje utilizando HTTP como protocolo de comunicación porque trabaja bien y es rápido para documentos estáticos.

EL mensaje SOAP interviene en el descubrimiento y la publicación de los servicio en la UDDI y en el intercambio de peticiones/respuestas entre el consumidor y el proveedor de servicios. Los proveedores de servicio pueden enviar un mensaje SOAP a la UDDI para publicar sus servicios. Mediante mensajes SOAP los clientes pueden comunicarse con la UDDI en busca de determinados servicios y luego de encontrar el servicio el cliente se comunica con su proveedor a través de mensajes SOAP. Como se puede apreciar es un estándar muy usado en un entorno SOA.

Normalmente el programador no tiene que elaborar el mensaje SOAP, la aplicación cliente es desarrollada en un lenguaje orientado a objeto y hace la invocación al servicio a través de lenguajes de alto nivel (Java, C#, C++, PHP). Es una interfaz proxy quien se encarga de elaborar el mensaje SOAP e invocar el servicio remoto enviando el mensaje. La aplicación proveedor también está desarrollada en un lenguaje orientado a objeto, por tanto, presenta una interfaz proxy que traduce el mensaje. Esto permite mayor flexibilidad pues no importa el lenguaje de implementación de ninguno de los dos.

Por tanto SOAP puede verse como un elemento esencial que permite comunicar sistemas heterogéneos. Aporta mayor flexibilidad posibilitando desarrollar sistemas distribuidos con procedimientos semánticos remotos utilizando como medio de comunicación mensajes SOAP. Pero existe otra alternativa a utilizar que es mucho más sencilla que SOAP y esa es REST.

2.6.2 REST (Representation State Transfer)

REST no es un estándar es un estilo de arquitectura que ha ido ganando mucha popularidad. Aunque actualmente las aplicaciones que soportan REST escasean. REST provee una alternativa mucho más simple que SOAP. Utilizando este estilo las aplicaciones clientes se comunican directamente con el proveedor a través de su URL[12] (Ver Anexos 17). Utiliza una interfaz muy conocida como es URI.[13]

El mensaje de petición y el de respuesta son codificados en XML directamente, eliminando así el componente de procesamiento de SOAP. Utiliza HTTP como protocolo de transporte y hace uso de los comando HTTP (GET, POST, DELETE, PUT) para acceder a los datos. Utilizando una invocación directa es más simple y más rápido.

2.7 Estándares de Seguridad

La comunicación entre las aplicaciones en una SOA se realiza a través de mensajes, por tanto, es obligatorio garantizar que los mismos sean lo más seguro posibles. La seguridad de las aplicaciones en una SOA se garantiza a través de un grupo de políticas de seguridad definidas como estándares.

[12] Localizador Universal de Recurso
[13] Identificador Uniforme de Recurso

2.7.1 WS-Security Policy (Políticas de seguridad)

Este estándar contiene dos funciones fundamentales:

- Una es determinar qué tipo de requerimientos de seguridad deben imponerse para los mensajes que llegan al servicio. Especifica qué requisitos deben tener los mensajes para que sean atendidos por el servicio.
- La otra función es determinar los mecanismos de seguridad que deben ser adicionados antes de dar salida a los mensajes.

El archivo WS-Security Policy es un archivo XML que presenta dos secciones <wsSecurityIn> contiene los requerimientos de seguridad para la entrada de un mensaje y <wsSecurityOut> contiene los mecanismos de seguridad adicionados en la salida de un mensaje.

WS-Security Policy define una serie de estándares de seguridad dentro de los cuales están:

- WS-Security: Este se encarga de la seguridad de los mensajes SOAP, garantizando mayor integridad y confidencialidad de los mismos.

- WS-Trust: Permite publicar y divulgar tokens de seguridad y manejar relaciones confiables entre las aplicaciones. El objetivo de esta especificación es posibilitar a las aplicaciones construir un intercambio de mensajes confiables. Usualmente en el cliente y en el proveedor se implementan motores-trust que se encargan de controlar el intercambio de tokens y de comprobar la autenticidad de los mismos.

2.7.2 WS-Security

Abordado un poco sobre cómo se integra WS-Security se puede decir que este estándar garantiza mayor integridad y confidencialidad de los mensajes SOAP mediante la utilización de un token de seguridad. Este es un conjunto de información que elabora el remitente del mensaje (identidad del remitente).

Los tokens de seguridad son combinados con una firma digital para realizar la autenticación. La firma digital se utiliza para comprobar que el token realmente pertenece al remitente.

WS-Security brinda una proposición general de un mecanismo para asociar los token con los mensajes y describe cómo hacer una codificación binaria del token. El token viaja adjunto en el encabezado del mensaje.

Esta información se maneja mediante un interceptor que se implementa tanto en el cliente como en el proveedor de servicio, este intercepta el mensaje antes de salir del cliente y adiciona la información WS-Security en el encabezado del mismo, en el proveedor el mensaje es interceptado nuevamente y se comprueba la autenticidad de esta información y solo sí esta es correcta el mensaje es procesado por el proveedor de servicio. La idea de estos interceptores es implementar la seguridad de forma transparente para los desarrolladores.

WS-Security es un estándar bastante flexible y puede utilizarse con varios modelos de seguridad y tecnologías de codificación. Los tokens de seguridad que utiliza pueden implementarse de varias formas:

- Username (Nombre de Usuario).
- X.509 Certificates (Certificados X.509).
- Kerberos
- SAML (Security Assertion Markup Language) o Lenguaje de Marcado de Argumentos de Seguridad.

Username

Consiste en un usuario y una contraseña, es decir lo que se envía como token de seguridad es un nombre de usuario y una contraseña.

Certificados X.509

Este método de autenticación consiste en la obtención de un certificado emitido por una tercera entidad denominada Autoridad Certificante (CA, según sus siglas en inglés) este certificado es el token de seguridad que se adjunta o referencia en el encabezado del mensaje. Es decir, la CA emite el certificado y le otorga una clave privada y otra pública al cliente.

El certificado se puede obtener de tres formas:

- Se puede comprar el certificado a una Autoridad Certificante (CA).
- La empresa puede emitir sus propios certificados y contar con la autorización de la CA para firmarlos.
- La empresa puede emitir certificados firmados por ellos mismos (esto no es adecuado para propósitos de producción).

El Certificado X.509 reporta un grupo de beneficios que se beben tener en cuenta a la hora de otorgar seguridad y estos son:

- La autenticación se realiza a través de protocolos y puertos bien conocidos (http/https y los puertos 80/443).
- Los certificados pueden usarse para autenticar clientes y proteger mensajes a través de las fronteras organizacionales y los dominios de seguridad porque son basados en estándares ampliamente aceptados. Los Certificados X.509 mediante la Infraestructura de Claves Públicas (PKI)[14] tienen la capacidad de establecer una base común de confiabilidad más allá del ámbito de una empresa individual.
- La CA soporta la renovación y revocación de los certificados. La empresa puede solicitar la renovación de un certificado para extender su tiempo de vida, entonces la CA emite un nuevo certificado con nueva fecha de expiración junto a nuevas claves privada y pública. También el certificado puede ser revocado en caso de que la información de la empresa cambie o su clave privada se vea comprometida.
- La autenticación no requiere de una relación directa entre todos los clientes y el servicio.

Kerberos

Kerberos es un protocolo que representa un servicio de autenticación que emplea “tickets” como prueba de identificación. El ticket contiene la información de qué servicios puede acceder el usuario y qué privilegios tiene en cada uno de ellos. Presentando ventajas respecto a otras formas de autenticación donde el usuario tiene que autenticarse cada vez que accede a un recurso. El usuario sigue

[14] Es el conjunto de componentes, personas, políticas y procedimientos que provee la fundamentación para el manejo de claves y certificados usados por los servicios de seguridad basados en clave pública (LAROWE, R. *Public Key Infrastructure (PKI) and Its Application in the New Economy*, 2000.

autenticándose mediante un nombre de usuario y una contraseña, pero después que está autenticado se le emite un "tickets" que es usado para acceder a los recursos protegidos de la aplicación.

El **Key Distribution Center (KDC)**[15] se encarga de otorgar el ticket junto con una clave de sesión que se utiliza para encriptar el token de seguridad. El token de seguridad en Kerberos está compuesto por el ticket y una estructura de datos llamada "Autenticador". El token viaja adjunto en el encabezado del mensaje hasta el servicio deseado. Cuando el servicio recibe el mensaje extrae el ticket, lo desencripta y realiza la autenticación del usuario.

Para implementar la autenticación mediante Kerberos se necesita:

- Una base de datos de cuentas: Que almacena la información referente al cliente y es donde KDC verifica la credencial (usuario y contraseña) presentada por el usuario.
- Una Políticas Kerberos: Estas son un conjunto de reglas que definen el comportamiento para los dominios de Kerberos. Incluye las restricciones de autenticación de usuarios, y el tiempo de vida del ticket.

La utilización de Kerberos brinda beneficios como:

- Provee capacidades de SSO (single sign-on)[16], que permiten al cliente autenticarse solo una vez.
- Tiene una amplia aceptación y es usada en la mayoría de las empresas que tienen una infraestructura de autenticación centralizada.
- Está cerradamente integrado con los sistemas operativos Windows (2000 y superiores a este), permitiendo a los sistemas operativos proveer capacidades adicionales para la utilización de esta autenticación.
- Permite autenticación mutua cuando un servicio envía una respuesta que tiene datos encriptados con la clave de sesión compartida.

---

[15] Es el intermediario encargado de autenticar a los clientes y emitir el ticket, en la plataforma de Windows esta entidad es implementada en un directorio activo.

[16] Métodos de autenticación que permiten al usuario autenticarse una sola vez durante su interacción con un software sin importar a cuantas funcionalidades de dicho software accedan.

También presenta algunas desventajas porque es un protocolo muy ligado a Microsoft y a sus sistemas operativos:

- Su naturaleza centralizada hace que requiera de un KDC, para poder estar disponible todo el tiempo. Si el KDC falla el cliente y el servicio no podrán comunicarse. Aunque en este caso pueden utilizarse mecanismos alternativos como los Certificados X.509.
- Es provechoso solo para autenticación en la red y comunicación segura. No es factible para persistencias de un largo período por el tiempo de vida limitado de los tickets y las claves de sesión.
- No establece confirmación de autenticación para los clientes fuera del dominio de seguridad, a menos que se establezca una relación de confianza con otro dominio de seguridad.

Lenguaje de Marcado de Argumentos de Seguridad (Security Assertion Markup Language o SAML)

Otro de los estándares de seguridad que es reconocido y que soporta autenticación y autorización es el Lenguaje de Marcado de Argumentos de Seguridad (SAML). Es un lenguaje que proveen argumentos confiables que son usados por los servicios proveedores para conferir autoridad de consumo a los clientes. Este estándar se basa en el mismo principio que la autenticación mediante Certificados y Kerberos de que el usuario se autentique una sola vez durante su interacción con un sistema.

SAML especifica tres tipos de argumentos para diferentes propósitos:

- Argumentos de autenticación: Describen cómo y cuándo un sujeto se ha autenticado.
- Argumento de atributo: Especifica que un sujeto tiene un número de atributos (A1, A2) con valores (V1, V2).
- Argumentos de autorización: Hace valer la autorización de un sujeto para un recurso.

¿Cómo se emplean los tokens SAML?

- Una aplicación cliente, provee la información de autenticación (usuario y contraseña) a la autoridad de autenticación de SAML.
- Esta autoridad genera el argumento SAML (token SAML).
- La aplicación puede acceder a un servicio o recurso utilizando el token.
- En el lado del proveedor, un Punto de Ejecución de Políticas (PEP[17], siglas en ingles) intercepta la petición para determinar el acceso o la autorización, aquí otra vez es usado el token.

[17] Componente funcional que aplica y ejecuta políticas de seguridad de redes.

- Si la autorización es concebida, la autoridad de atributos genera el argumento de atributos y este argumento es adjuntado al token.
- Este token es adjuntado en el encabezado del mensaje SOA.

2.8 Estándar de calidad de los mensajes

Para garantizar la calidad de los servicios es necesario asegurar la calidad del intercambio de mensajes. Estos estándares que se muestran a continuación garantizan que los mensajes sean entregados a su destinatario bajo cualquier circunstancia.

2.8.1 WS-ReliableMessaging

Estándar desarrollado por OASIS que permite que los mensajes sean entregados confiablemente ante la presencia de fallos (software, sistemas o de la red). El cliente envía el mensaje con un encabezado WS-ReliableMessaging. El proveedor recibe el mensaje y envía un mensaje de reconocimiento o confirmación al cliente. Los mensajes serán reenviados una y otra vez hasta que el proveedor los reciba.

2.8.2 WS-Reliability

Este estándar también utiliza el encabezado del mensaje SOAP para garantizar la entrega del mensaje. Provee un diseño que puede ser usado para conformar mensajes en un bloque de encabezamiento de confiabilidad. También provee una especificación de comunicación HTTP y manejo de mensajes de fallo.

En el encabezamiento el mensaje enviado debe especificar el curso, destino, identificador del mensaje, tiempo del mensaje y una petición de confirmación. El mensaje de confirmación debe contener el identificador del mensaje, el tiempo del mensaje y el curso o la identificación de destino. La eliminación de la duplicación asegura que el mensaje es entregado a lo sumo una vez.

2.9 Conclusiones Parciales

En el presente capítulo se realizó un estudio de las características de los principales estándares utilizados para desarrollar una SOA. Se analizó el papel que juegan dentro de la arquitectura y cómo se relacionan entre sí. Permitiendo sentar las bases para seleccionar cuáles satisfacen en mayor medida las necesidades de la universidad.

CAPÍTULO 3 SOLUCIÓN PROPUESTA

3.1 Introducción

La Arquitectura Orientada a Servicios en un principio fue considerada un método alternativo mejorado de invocación de métodos pero poco a poco fue tomando auge y se convirtió en una nueva forma de ver la organización de las empresas y de trabajar el negocio de forma más eficiente.

Pero a pesar que hace algún tiempo es utilizada y puesta en práctica en la mayoría de las grandes empresas desarrolladoras de software es totalmente nueva para los desarrolladores cubanos y en particular para la Universidad de las Ciencias Informáticas (UCI).

Por tanto, fue preciso realizar una investigación que ayudara a entender cómo llevarla a cabo, de ahí que el presente capítulo expone una propuesta de desarrollo basada en estándares para la implementación de una SOA en la UCI.

3.2 ¿Cómo comenzar?

Cada empresa que desee poner en práctica una Arquitectura Orientada a Servicios (SOA) debe conocer primeramente qué es, cómo funciona, qué componente utiliza, qué beneficios reporta y si en realidad es necesario ponerla en práctica o no.

En caso que se desee utilizarla debe estudiar cuidadosamente las características de los procesos que se manejan, las condiciones tecnológicas en la organización y buscar qué estándares de SOA satisfacen en mayor medida sus necesidades.

Es importante saber que una SOA tiene sentido para empresas u organizaciones que manejen numerosas aplicaciones que necesitan integrarse. Es decir, es recomendable utilizarla cuando la empresa se encuentra en una situación difícil de manejar, como: enfrentarse a continuos cambios, tener las aplicaciones implementadas en distintos Sistemas Operativos, lenguajes y plataformas de programación como es el caso de la UCI.

Antes de que SOA se convirtiera en una alternativa de desarrollo, en las empresas se aplicaban esquemas clásico que solamente lo entendían los desarrolladores de software o aquel que tuviera plenos conocimientos informáticos. Los analistas especificaban los requisitos. A partir de ahí se elaboraban para el análisis y diseño diagramas UML. Se implementaban en un lenguaje de programación cualquiera y se procedía a realizar pruebas al software.

Con SOA es menos complejo y mucho más cortos los pasos a realizar, además la forma de llevar el negocio hace que sea posible un entendimiento común entre el informático y el cliente pues ambos hablan el mismo idioma técnico. De ahí que será mucho más fácil entender las necesidades del cliente y que este quede del todo satisfecho.

¿Cómo se llevaría a cabo en SOA lo que antes se hacía sin ella? Pues muy fácil, primeramente se analizan las aplicaciones ya existentes y se separan los componentes funcionales que son reusables, a partir de ahí se elaboran servicios reutilizables que serán utilizados por otros procesos de negocio. Para mostrar esta explicación un poco más clara fue elaborada una guía de pasos a seguir, donde se encuentra más organizada la forma de llevar a cabo una SOA. Es importante conocer que esta guía no explica cómo desarrollar cada componente o servicio que se utilice, sino más bien va dirigido a organizar la manera de trabajar.

3.3 Guía de pasos para implementar la SOA en la UCI

Como apoyo al personal encargado de poner en práctica la Arquitectura Orientada a Servicios se han propuesto un grupo de pasos a seguir que guiarán el camino de una manera sencilla y fácil de entender.

1. Desglosar todas las aplicaciones existentes en componentes funcionales reusables.
2. Implementar estos componentes como servicios Web XML.
3. Guardar la descripción de los servicios Web XML en el registro.
4. Definir correctamente los procesos de negocio de la UCI ya sean informáticos o no.
5. Modelar los procesos de negocio.
6. Determinar si los procesos de negocio definidos necesitan de algún servicio que no este implementado y en caso afirmativo implementar este servicio.
7. Ejecutar los procesos de negocio mediante la orquestación de los servicios.

8. Desarrollar las interfaces que van a utilizar los procesos de negocio modelados.

Para llevar a cabo cada uno de estos pasos es necesario utilizar estándares que permitan desarrollar cada componente que utiliza SOA para funcionar. La mejor forma de ver qué estándares son los más adecuados y en qué momento entran en escena es viendo la arquitectura desglosada en capas como se muestra a continuación.

3.4 Vista de las capas de SOA

Varias empresas desarrolladoras de software ya han puesto en práctica esta arquitectura y para aplicarlas la han desglosado en capas de abstracción obteniendo resultados favorables. Por tanto, se propone utilizar este mismo método en la UCI para tener una visión más clara respecto a cómo quedará distribuida cada parte de la SOA (Ver Anexos 18).

- Capa 1: Sistemas Operacionales. Contiene las aplicaciones existentes. Se conoce que SOA no desecha lo que está hecho en la empresa, sino que lo transforma y utiliza. Por tanto, las aplicaciones que están en funcionamiento en la UCI quedan ubicadas en esta capa y pueden ser integradas utilizando técnicas orientadas a servicio.
- Capa 2: Componentes Empresariales. Contiene los componentes funcionales de las aplicaciones existentes y los componentes que se agreguen posteriormente.
- Capa 3: Servicios. Es la capa que contiene los servicios Web creados utilizando los componentes de la capa anterior, los servicios que podrán ser invocados por las aplicaciones de negocio o por otros servicios.
- Capa 4: Composición de procesos de negocio. Contiene la orquestación y la coreografía de los servicios Web que intervienen en los procesos de negocio. Sobre esta capa trabaja el sistema de Gestión de Procesos de Negocio (BPM).
- Capa 5: Presentación o acceso. Es la capa donde se encuentran la interfaz de los procesos de negocio. Serían las aplicaciones que ejecutan los procesos de negocio.
- Capa 6: Integración de servicios. Posibilita la integración de servicios. Brinda funcionalidades de envío de mensajes y enrutamiento inteligente. Aquí entra a jugar su papel el Bus de Servicio de Empresa (ESB).

- Capa 7: Calidad de los servicios. En esta capa radica la parte de seguridad, monitoreo y administración de los servicios y los procesos de negocio.

Las dos primeras capas de esta arquitectura no necesitan estándares, pues son las aplicaciones y componentes que cada proyecto o departamento de la UCI desarrolla en el lenguaje y la plataforma que estime conveniente. En las capas restantes sí es necesario trabajar utilizando estándares para lograr el objetivo final que es la interoperabilidad de los sistemas dentro de la UCI. A continuación se expondrá cuáles serán utilizados en cada capa.

3.5 Descripción y descubrimiento de los servicios Web

Los servicios Web son ubicados dentro de la tercera capa. Cuando son creados es obligatorio generar un documento descriptivo de los mismos y proceder a su publicación para que puedan ser encontrados por otros servicios. A continuación se proponen dos estándares para describirlos y publicarlos.

El Lenguaje de Descripción de Servicios Web (WSDL) es un estándar de descripción, este provee en un documento bien detallado y fácil de comprender cómo se puede interactuar con un servicio determinado. La descripción contiene los tipos de datos que manipulan el servicio, el formato de mensajes que acepta, quién lo hizo y las políticas de seguridad.

El programador no tiene que crear manualmente este documento porque la mayoría de las herramientas de desarrollo de servicios Web lo generan. Este documento WSDL es publicado en el registro SOA. Para la creación en la UCI del registro SOA se propone la implementación de una UDDI en la red local.

La Descripción, Descubrimiento e Integración Universal (UDDI) es el estándar definido por OASIS, presenta la capacidad de interactuar con las aplicaciones a través de mensajes para posibilitar la publicación de la descripción de los servicios. Está UDDI puede radicar en uno de los nodos de comunicación de la universidad.

Todas las aplicaciones se comunican con este registro para publicar o buscar nuevos servicios disponibles en la red, por tanto, debe estar situado en una computadora o servidor potente para que pueda procesar de forma rápida las peticiones.

3.6 Modelación de los procesos de negocio

La cuarta capa contiene la modelación de los procesos de negocio. Los procesos de negocio en SOA se desarrollan orquestando un grupo de servicios que se encuentran publicados en la UDDI y descritos a través de un documento WSDL. Para ello se proponen los siguientes estándares de modelación de negocio.

3.6.1 Estándar para la modelación gráfica

Para el modelado gráfico de los procesos de negocio se recomienda la Notación de Modelado de Procesos de Negocio (BPMN). Este provee una serie de diagramas y estereotipos para representar entidades de negocio y la interacción entre las mismas.

No es el único estándar que existe para la modelación gráfica de procesos de negocio, pero si es el único hasta ahora que fue diseñado con ese fin. Existe un lenguaje que también es utilizado en el modelado de los procesos de negocio y es el Lenguaje Unificado de Modelado (UML) sobre todo en su versión 2.0.

UML es un lenguaje flexible que permite modificaciones en sus estereotipos para ser usados en diferentes tipos de software pero no se puede perder de vista que fue concebido para sistemas orientados a objeto no para sistemas orientados a servicios o procesos de negocio. UML 2.0 incluye nuevos diagramas y estereotipos que permiten la modelación de procesos de negocio.

Para fundamentar la elección de BPMN como el más apropiado para el modelado de procesos, a continuación se hace una comparación de BPMN con UML, enfocando la atención en un grupo de aspectos significativos en la modelación de un proceso de negocio.

- **Utilización de estereotipos y diagramas**

UML es un lenguaje de modelado de sistemas orientado a objetos, por tanto no provee estereotipos para modelar procesos pero como es un lenguaje flexible se puede agregar. En cuanto a los diagramas tampoco provee diagramas para procesos, pero son los diagramas de comportamiento (diagrama de actividades y diagrama de secuencia) y otros adicionados en su versión 2.0 (diagrama de composición de estructura y diagrama general de interacción) los que se utilizan para modelar los procesos de negocio en UML, por su similitud a los que se utiliza BPMN.

BPMN fue diseñado para modelar procesos de negocio, por tanto contiene estereotipos para los elementos que intervienen en un proceso y provee diagramas para modelar los procesos de negocio.

- **Métodos de modelación de procesos de negocio**

UML es más enfocado en objetos y define los objetos a través de los diagramas estáticos (diagrama de clases) y luego define la relación entre los mismos a través de los diagramas de comportamiento.

BPMN es enfocado a procesos por tanto define y modela procesos de negocio desde el comienzo del ciclo de desarrollo. El modelo de objetos del negocio es definido implícitamente. También puede modelar explícitamente los objetos de negocio que pueden ser expuestos a través de los servicios de negocio.

- **Mapeo de los diagramas a lenguaje ejecutable**

UML no posee ninguna definición de metadato que le permita mapear sus diagramas directamente a un lenguaje ejecutable. Estos diagramas son exportados como un fichero de Metadato de Intercambio XML (XMI). Dicho fichero es importado por otras herramientas (por ejemplo Eclipse[18]) para generar el fichero con el código ejecutable y la descripción del proceso (WSDL).

BPMN es basado en meta-modelos de ejecución de procesos y no necesita ningún paso o herramienta adicional para mapear a lenguaje de ejecución de procesos, lo hace directamente.

- **Utilización a lo largo del ciclo de desarrollo**

UML se centra más en la arquitectura y la ingeniería del software puede verse con mayor eficiencia desde el diseño hasta la implementación del software. Por esta razón en muchos casos es ajeno a los analistas del negocio.

BPMN es más enfocado al análisis del negocio, la arquitectura de software y la ingeniería de software. Puede verse como una forma de lograr eficiencia en todo el ciclo de vida de desarrollo del software.

[18] Herramienta de compilación para desarrollar aplicaciones en lenguaje JAVA.

Además BPMN permite realizar simulaciones a los procesos de negocio antes de ser ejecutados. Esta simulación del proceso permite a los analistas de negocio medir la eficiencia y calidad del mismo, saber hasta que punto es óptimo una actividad y determinar dónde se encuentra los cuellos de botella (lugares donde la eficiencia es mínima) del proceso. Permitiendo así corregir a tiempo los errores sin que ocasionen ninguna pérdida (Ver Anexos 19).

Por todo lo antes planteado se recomienda utilizar BPMN como notación para el modelado de procesos de negocio.

3.6.2 Lenguaje ejecutable para la orquestación

Los procesos modelados con UML o BPMN son descritos en un lenguaje ejecutable para que sean interpretados por los motores de ejecución. Como lenguaje ejecutable se recomienda el Lenguaje de Ejecución de Procesos de Negocio (BPEL). Este es un lenguaje altamente utilizado y aceptado en la orquestación de procesos de negocio. Presenta una relación muy estrecha con BPMN posibilitando una notación para todos los diagramas y estereotipos que presenta dicha notación.

BPEL interactúa con la descripción WSDL de los servicios para llevar a cabo la orquestación de los mismos. También utiliza el Lenguaje de Caminos de XML (Xpath) y el Lenguaje de Plantilla Extensible para Transformación (XSTL) para el acceso y manejo de los datos. Es un lenguaje que aporta flexibilidad y escalabilidad a los sistemas.

Además las herramientas de modelación de diagramas BPMN permiten generar automáticamente el código BPEL. Y este código es aceptado por todos los motores de ejecución.

3.6.3 Lenguaje para la coreografía

Los procesos de negocio necesitan ordenar y secuenciar los mensajes que son intercambiados entre los servicios orquestados. El Lenguaje de Descripción de Coreografía (WS-CDL, siglas en inglés) es el más recomendable para la descripción de la coreografía.

WS-CDL se encarga del ordenamiento y la secuenciación de los mensajes que se intercambian durante la interacción, es decir puede verse como una política que gobierna el intercambio de mensajes. Define tres roles fundamentales dentro de los cuales ubica a cada una de las parte involucradas en la interacción. Los roles son: comprador (buyer), el vendedor de servicios (seller) y el fletador (shipper). Cada rol define un comportamiento.

3.6.4 Otro estándar recomendado

Se propone utilizar el Lenguaje de Descripción de Procesos (XPDL, siglas en inglés) como lenguaje de intercambio, cuyo objetivo principal, es provee un formato de fichero que pueda ser ejecutado en cualquier herramienta de diseño.

El código ejecutable generado por la herramienta de diseño de un fabricante no puede ser ejecutado por el motor de ejecución de otro fabricante. Además los ficheros BPEL una vez que son generados no permiten revertir el proceso, es decir, a través de un código BPEL no se pueden obtener los diagramas BPMN que lo generaron.

Por lo anteriormente expuesto se recomienda generar un fichero XPDL como fichero de intercambio para cuando se necesite emigrar de herramienta de diseño o de motor de ejecución, solamente exportar los diagramas en este formato e importarlos en la nueva herramienta de diseño, recuperar los diagramas BPMN y generar el nuevo código ejecutable. Los ficheros de formatos XPDL pueden ser interpretados en un motor de ejecución pero no brinda una semántica de ejecución consistente y además no todos los motores de ejecución lo aceptan (Ver Anexos 15).

3.7 Protocolos de comunicación y mensajería

Para poder comunicar los servicios Web orquestados se utilizan los mensajes y en el caso de la UCI se propone como protocolo de mensajería el Protocolo de Acceso a Servicios (SOAP, siglas en inglés). Este protocolo es utilizado en la capa de servicios donde se define el tipo de mensaje que él utiliza para comunicarse, es basado en XML y permite la transmisión de información a través de mensajes SOAP.

Los programadores han empleado este protocolo como estándar de mensajería porque además de la información principal del mensaje, permite adjuntar un tokens de seguridad para la autenticación del remitente.

Los mensajes SOAP pueden viajar a través de varios protocolos de transporte (HTTP, SMTP[19]) pero HTTP es el más utilizado porque es un protocolo abierto en cualquier cortafuego (firewall) y casi todos los sistemas operativos admiten este protocolo por ser utilizado para la Web. Además la seguridad de HTTP aumenta la seguridad de los mensajes. En la UCI el protocolo HTTP es muy utilizado y existen las herramientas y el personal calificado para manejarlo.

Además de la seguridad que brinda, SOAP es un protocolo sencillo y los programadores no tienen que escribir el mensaje en formato XML, realizan la llamada en lenguaje orientado a objeto y las herramientas de compilación se encargan de construir el mensaje. Este protocolo es muy efectivo para invocar servicios complejos o que manejen información sensible y requieren una autenticación rigurosa.

Para la invocación de servicios sencillos (búsquedas en el directorio de personas, consulta del menú del comedor) se puede utilizar REST (Representation State Transfer). Este estilo arquitectónico permite invocar los servicios de manera más rápida y sencilla utilizando solamente la dirección URL del servicio Web y pasando los parámetros. La sencillez de este método radica en que realiza la invocación utilizando los comandos de HTTP (GET, POST, DELETE, UPDATE). No es tan seguro como SOAP, pues se apoya solamente en la seguridad que brinda HTTP, por tanto, no es recomendable utilizarlo para la invocación de servicios que requieran mucha seguridad. REST solamente puede utilizarse a través del protocolo HTTP (Ver Anexos 17).

3.8 Componente de integración

Una vez creados los componentes, formado los servicios y orquestados para obtener los procesos de negocio requeridos y realizadas las aplicaciones como modo de interfaz gráfica para darle al cliente una determinada prestación. Se necesita una infraestructura para integrar dichos elementos y se ubica en la sexta capa.

[19] Protocolo Simple de Transferencia de Correo.

Para la integración de los elementos (aplicaciones, servicios, componentes) de SOA en la UCI se propone la utilización de un Bus de Servicio de Empresa (ESB). El ESB es una infraestructura compuesta por diversos servicios dentro de los cuales se encuentra el servicio de mensajería, enrutamiento inteligente, auditorias, autorización, transporte de datos y conversión de mensajes. Incorpora las funcionalidades de algunos componentes de SOA (Servicio Intermediario (Broker) y Supervisor de SOA), por tanto la implementación de los últimos pasa a ser opcional si se utiliza el ESB.

El ESB elimina la necesidad de programar el enrutamiento en las aplicaciones o establecer relaciones rígidas entre dispositivos, pues automatiza el enrutamiento utilizando contenido de documentos y reglas del negocio, es decir, la dirección de destino se coloca dentro del mensaje y el ESB se encarga de buscarla y enviarlo a esa dirección.

Permite realizar auditorias pues guarda un historial de todos los mensajes que viajan a través de él, esto posibilita en caso de que ocurra alguna violación conocer quién envió el mensaje que ocasionó la misma.

Actúa como intermediario durante el proceso de autorización de servicios pues verifica que el remitente del mensaje esté autorizado a acceder al servicio invocado y tenga permiso para utilizar las funcionalidades que solicita.

Presenta un grupo de adaptadores (JDBC[20], FTP[21], RMI[22]/IIOP[23], HTTP[24]) que permiten conectar servicios con interfaces heterogéneas a través de un servicio de conversión de mensajes que es totalmente ajeno a las aplicaciones que están interactuando. Esto es muy útil porque en la UCI existen un grupo de aplicaciones desarrolladas en otras arquitecturas que no son basadas en servicios Web y que serán integradas a SOA.

[20] Java Database Connectivity: Interfaz estándar de conexión a base de datos a través del lenguaje Java.
[21] File transfer Protocol : Protocolo para la transferencia de fichero.
[22] Remote Method Invocation: Protocolo para la invocación remota de métodos.
[23] Internet Inter-ORB Protocol: Protocolo utilizado por Common Object Request Broker Architecture (CORBA).
[24] HyperText Transfer Protocol: Protocolo de transferencia de textos.

3.9 Seguridad de las aplicaciones

La séptima capa es la de seguridad en las aplicaciones SOA y es de gran importancia. Como los mensajes son el medio utilizado para el intercambio de información, son el objeto principal de atención en la seguridad.

Para implementar la seguridad de los mensajes en la UCI se propone la utilización del estándar de Seguridad de los Servicios Web (WS-Security). Este propone un mecanismo para asociar los tokens de seguridad con los mensajes SOAP, el token viaja encriptado en el encabezado del mensaje. El token es un conjunto de información que elabora el remitente para que el receptor del mensaje pueda autenticarlo.

En la universidad se recomienda implementar los tokens de seguridad utilizando Kerberos. Este es un protocolo de seguridad para redes que provee una autenticación robusta basada en la encriptación utilizando claves secretas. Kerberos elimina el envío de contraseñas no encriptadas a través de la red durante el proceso de autenticación, proponiendo la asignación de un ticket que el usuario podrá utilizar durante un período de tiempo determinado para autenticarse dentro del dominio de la red.

La autenticación basada en kerberos consiste en la instalación de un servidor Kerberos. Luego se instalan las versiones clientes en todas las computadoras que lo requieran. El servidor Kerberos es auxiliado por el Servidor de Dominio de Red (DNS, en inglés). Cuando el cliente se comunica con el servidor Kerberos para solicitar un ticket, este último interactúa con el DNS para comprobar que es un usuario del dominio.

Es importante aclarar que este protocolo es completamente seguro solo si todas las computadoras del dominio lo implementan. Además aclarar que aunque es un protocolo libre (es compatible con varias tecnologías) su nivel más alto de eficiencia se obtiene si se trabaja vinculado a tecnologías Microsoft.

Autorización

Con WS-Security se garantiza la autenticación de los usuarios, pero eso en SOA no es suficiente es necesario definir los permisos que tiene una aplicación sobre los servicios de otra. Estos permisos se establecen en la etapa de contrato del servicio y se almacenan en el registro SOA. Cuando el cliente invoca un servicio determinado el Servicio Intermediario (Broker) verifica en el registro que el remitente del mensaje tenga permiso para utilizar las funcionalidades del servicio que solicita.

3.10 Conclusiones Parciales

Este capítulo brinda una propuesta de desarrollo para implementar una SOA en la UCI. Propone un modelo de arquitectura dividido en capas que específica qué estándares puede utilizarse en cada una de ellas y se formula una guía de pasos para encaminar el trabajo de los desarrolladores.

CONCLUSIONES

- La Arquitectura Orientada a Servicios (SOA) es una vía eficiente para integrar y organizar los procesos de negocio en la Universidad de las Ciencias Informáticas.
- La propuesta de desarrollo favorece la puesta en práctica por parte de los programadores de la Arquitectura Orientada a Servicios en la Universidad de las Ciencias Informáticas, garantizando que se utilicen los estándares más apropiados, logrando alcanzar homogeneidad de comunicación entre las aplicaciones existentes, permitiendo la reutilización de código de forma factible y cómoda para cuando sea necesario utilizarla en otros procesos de negocio, así como un mayor grado de seguridad de la información que fluye en la universidad a través de los medios informáticos.
- El objetivo trazado en la investigación fue cumplido a través de la idea a defender, que culminó con una propuesta de desarrollo basada en estándares para aplicar una Arquitectura Orientada a Servicios en la Universidad de las Ciencias Informáticas.

RECOMENDACIONES

- Investigar cómo funcionan y cuáles son las herramientas que utiliza SOA para el desarrollo de todos los componentes que ella necesita para trabajar de forma eficiente.
- Investigar cómo se implementan cada uno de estos componentes y cada servicio que se ponga en práctica.
- Tratar de poner en práctica cuanto antes la SOA en la UCI o por lo menos comenzar a sentar las bases para en un futuro no muy lejano obtener resultados satisfactorios.
- Profundizar en el estudio de cómo se implementan los estándares propuestos.

REFERENCIAS BIBLIOGRÁFICAS

- ADOLFO R. DE SOTO; EVA CUERVO FERNÁNDEZ. *Nuevas Tendencias en Sistemas de Información: Procesos y Servicios*, 2006.
- ALEJANDRO GUINEAS DE SALAS; SERGIO JORRIN Arquitectura SOA para la interacción entre software libre y software propietario en entorno mixto.
- BILLY REYNOSO Arquitecutra Orientada a Servicios.
- CARLOS BILLY REYNOSO. *Introducción a la Arquitectura de Software*.
- Diccionario Terra.
- ED ORT. *Service-Oriented Architecture and Web Services: Concepts,*
- *Technologies, and Tools*, 2005. 32.
- HUILBERT AALBERS Introducción a SOA (II).
- JOHN EVDEMON. *Principios de diseño de servicios: patrones y antipatrones de servicios*, 2005.
- JUDITH HURWITZ, R. B., CAROL BAROUDI, MARCIA KAUFMAN. *Services Oriented Arquitecture for Dummies*, 2007. 387.
- LAROWE, R. *Public Key Infrastructure (PKI) and Its Application in the New Economy*, 2000.
- PABLO ALVEZ ;PATRICIA FOTI ;MARCO SCALONE. *Proyecto Batuta*, 2006.
- Wikipedia, 2007.

BIBLIOGRAFÍA

1- "2_UML.pdf" en: http://www.ia.uned.es/ia/asignaturas/adms/2_UML.pdf (16/05/07).

2- "6AD5D16960.BPMN_and_BPM.pdf" en: http://www.bpmn.org/Documents/6AD5D16960.BPMN_and_BPM.pdf (20/05/07).

3- "Arquitectura Orientada a Servicios (SOA)" en: http://www.microsoft.com/spanish/msdn/latam/mediacenter/webcast/architect.aspx (29/03/07).

4- "Arquitectura SOA para la interacción entre software libre y software propietario en entorno mixto" en: http://www.sigte.udg.es/JornadasSIGLibre/comun/1pdf/13.pdf (3/04/07).

5- "BPEL « Espacio SOA.mht" en: http://espaciosoa.wordpress.com/tag/bpel/ (21/05/07).

6- "Brokered Authentication Kerberos.mht" en: http://msdn2.microsoft.com/en-us/library/aa480562.aspx (16/05/07).

7- "Cómo trabajar con el certificado.mht" en: http://msdn2.microsoft.com/en-us/library/aa480565.aspx (24/04/07).

8- "Descripción y descubrimiento de servicios Web con UDDI, primera parte.mht" en: http://www.microsoft.com/spanish/msdn/articulos/archivo/281201/voices/service10032001.asp (25/04/07).

9- "diferencias entreSOAyWS.mht" en: http://www.help400.es/asp/scripts/nwart.asp?Num=167&Pag=10&Tip=T (13/4/2007).

10- "Enterprise Service Bus" en: http://rapidshare.com/files/28622205/xyz123.rar.html.

11- "From UML to BPEL.mht" en: http://www-128.ibm.com/developerworks/webservices/library/ws-uml2bpel/ (17/05/07).

12- "Historia de las arquitecturas" en: http://www3.unileon.es/pecvnia/pecvnia02/02_129_158.pdf (14/05/07).

13- "Introducción a la Arquitectura de Software" en: http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/intro.asp (29/03/07).

14- "Introducción a la arquitectura orientada a servicios (SOA)" en: http://www.microsoft.com/spanish/msdn/articulos/archivo/121205/voices/SOADesign.mspx (26/03/07).

15- "Introduccion a SOA(II)" en : http://huibert-aalbers.com/IT_Insight/Spanish/PDF/ITI005Sp-SOA(II).pdf(2/4/2007) (2/4/2007).

16- "Kerberos ppt" en: http://support.microsoft.com/default.aspx?scid=kb;en-us;822248 (23/4/07).

17- "Kerberos Technical Supplement for Windows.mht" en: http://msdn2.microsoft.com/en-us/library/aa480609.aspx (23/04/07).

18- "Nuevas Tendencias en Sistemas de Información: Procesos y Servicios" en: http://www3.unileon.es/pecvnia/pecvnia02/02_129_158.pdf. historia de la arquitectura.pdf (5/5/2007).

19- "Orquestación en procesos End-to-End" en: http://www.oracle.com/technology/tech/soa/mastering-soa-series/part3.html (12/4/2007).

20- Patterns: Service-Oriented Architecture and Web Services" en: http://publibb.boulder.ibm.com/Redbooks.nsf/RedpieceAbstracts/sg246303.html?Open (19/4/2007).

21- "Proyecto Batuta" en: http://www.willydev.net/descargas/WillyDev_ToolOrquesta.pdf (16/4/2007).

22- "Service Oriented Enterprises.pdf" en: http://rapidshare.com/files/28622205/xyz123.rar.html.

23- "UML 2.0.pdf" en: http://www.dte.upct.es/investigacion/is/UML2.0.pdf (16/05/07).

24- "UML 2_0 Profile for Software Services.mht" en: http://www-128.ibm.com/developerworks/rational/library/05/419_soa/ (14/05/07).

25- "UML2.pdf" en: http://alarcos.inf-cr.uclm.es/per/fruiz/conf/tbpm/tbpm.pdf (16/05/07).

26- "Web Service Definition Language (WSDL).mht" en: http://www.w3.org/TR/wsdl (29/03/07).

27- "WSDL%20and%20UDDI.pdf"

28- "WS-Policy.pdf" en: http://www.redbooks.ibm.com/redpapers/pdfs/redp3678.pdf (26/04/07).

29- "WS-Security Authentication.mht" en: http://www.oracle.com/technology/tech/java/newsletter/articles/wsaudit/ws_audit.html (23/04/07).

30- "XML Path Language (XPath).mht" en: http://www.sidar.org/recur/desdi/traduc/es/xml/xpath.html (9/05/07).

31- "XSLTXPath.pdf" en: http://www.di.uniovi.es/~labra/cursos/ext05/pres/XSLTXPath.pdf (05/05/07).

32- http://diccionario.terra.com.pe/cgi-bin/b.pl.

33- http://www.bpmn.org/.

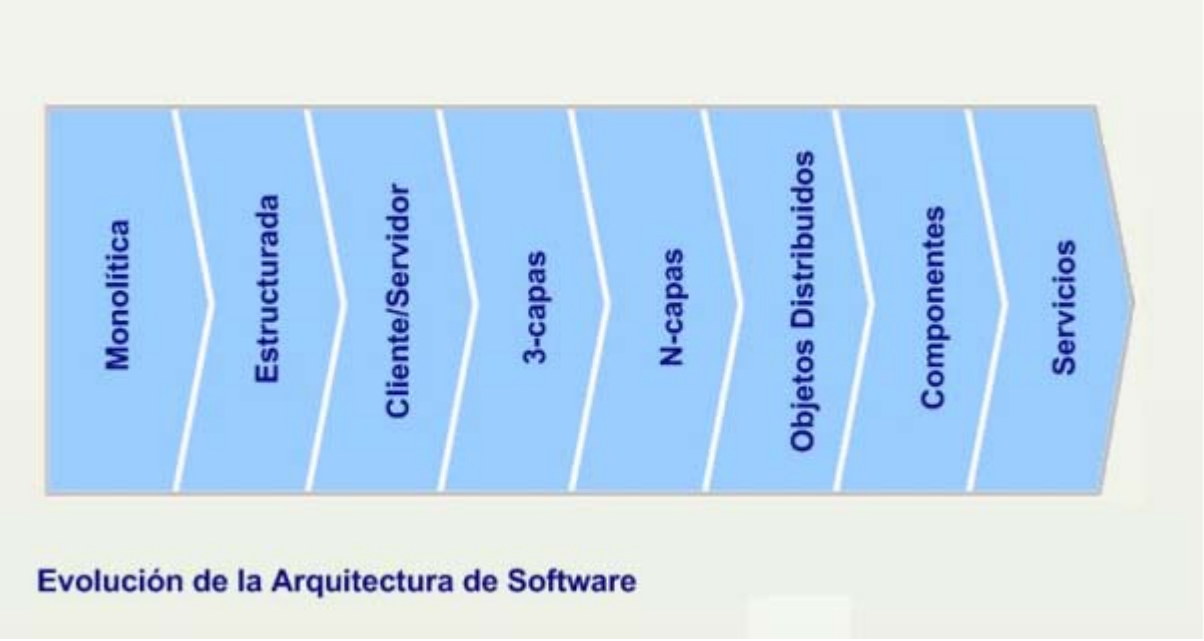
34- http://www.conexiones.eafit.edu.co/sobreConexiones/publicaciones/libroPdfs/capitulo12.pdf (17/5/2007).

35- http://www.microsoft.com/spanish/msdn/arquitectura/das/guias/AppArchCh2.mspx (17/5/2007).

36- Libro SOA for Dummies.pdf

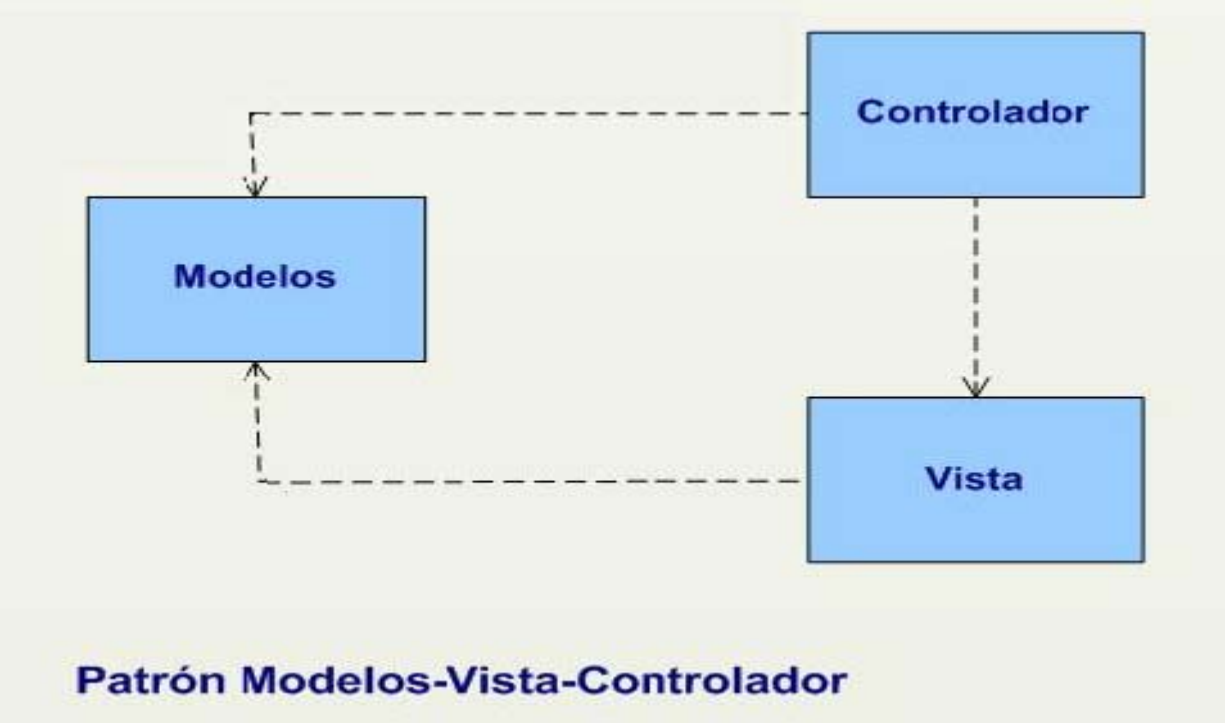
37- Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools en: http://java.sun.com/developer/technicalArticles/WebServices/soa2/soa2.pdf (27/03/07).

Anexos 1 Evolución de la Arquitectura

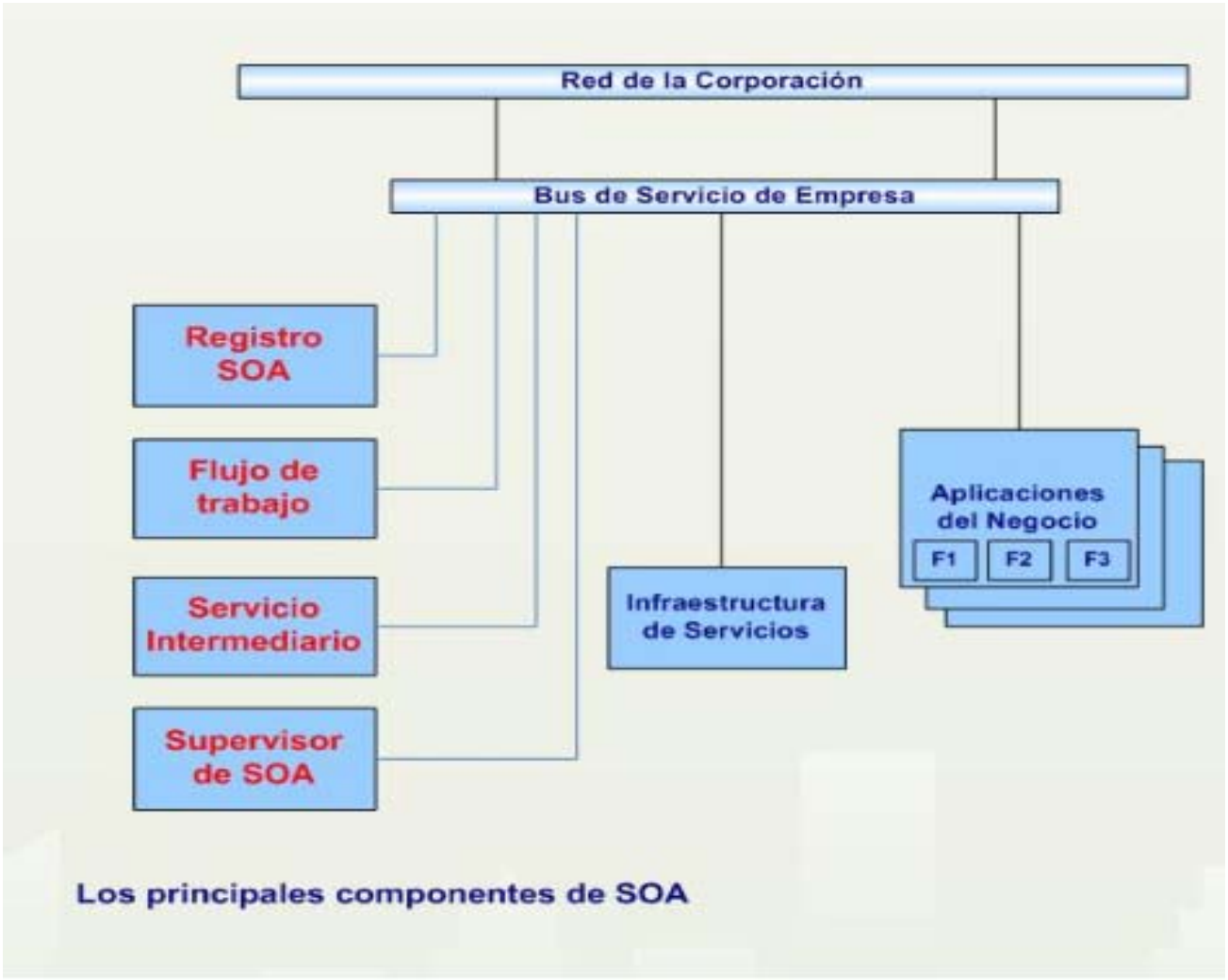


Evolución de la Arquitectura de Software

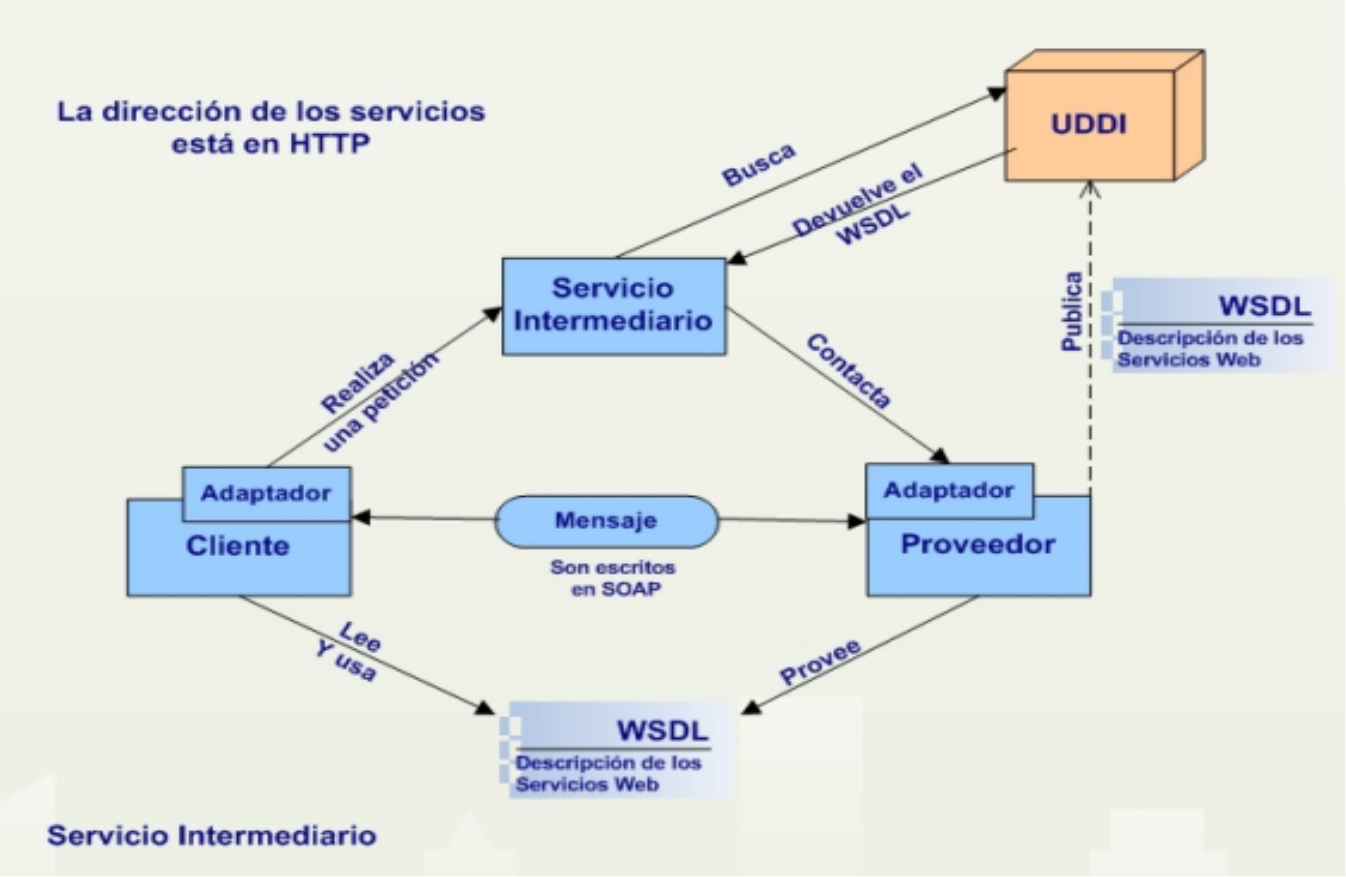
Anexos 2 Patrón arquitectónico Modelo-Vista-Controlador



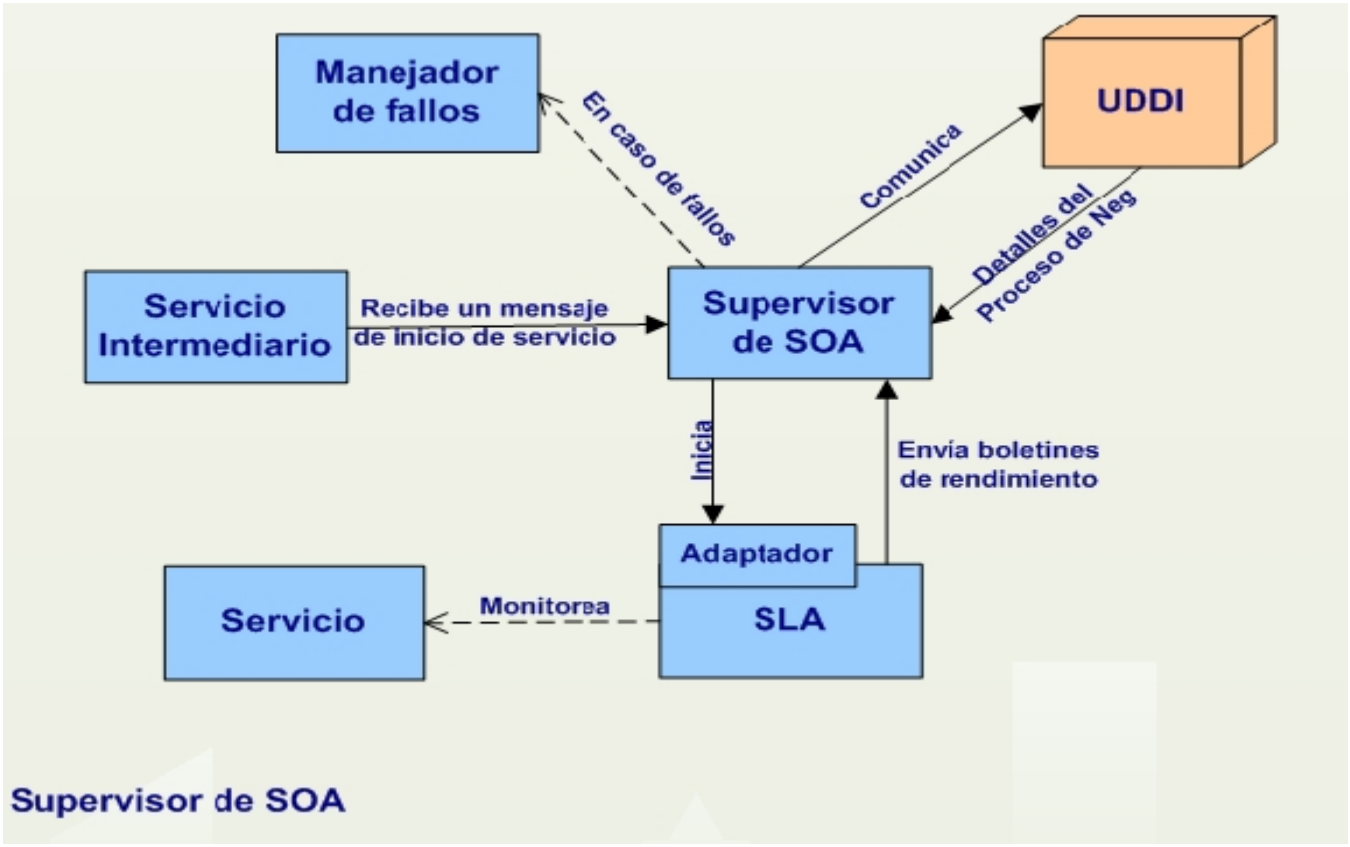
Anexos 3 Componentes de una SOA



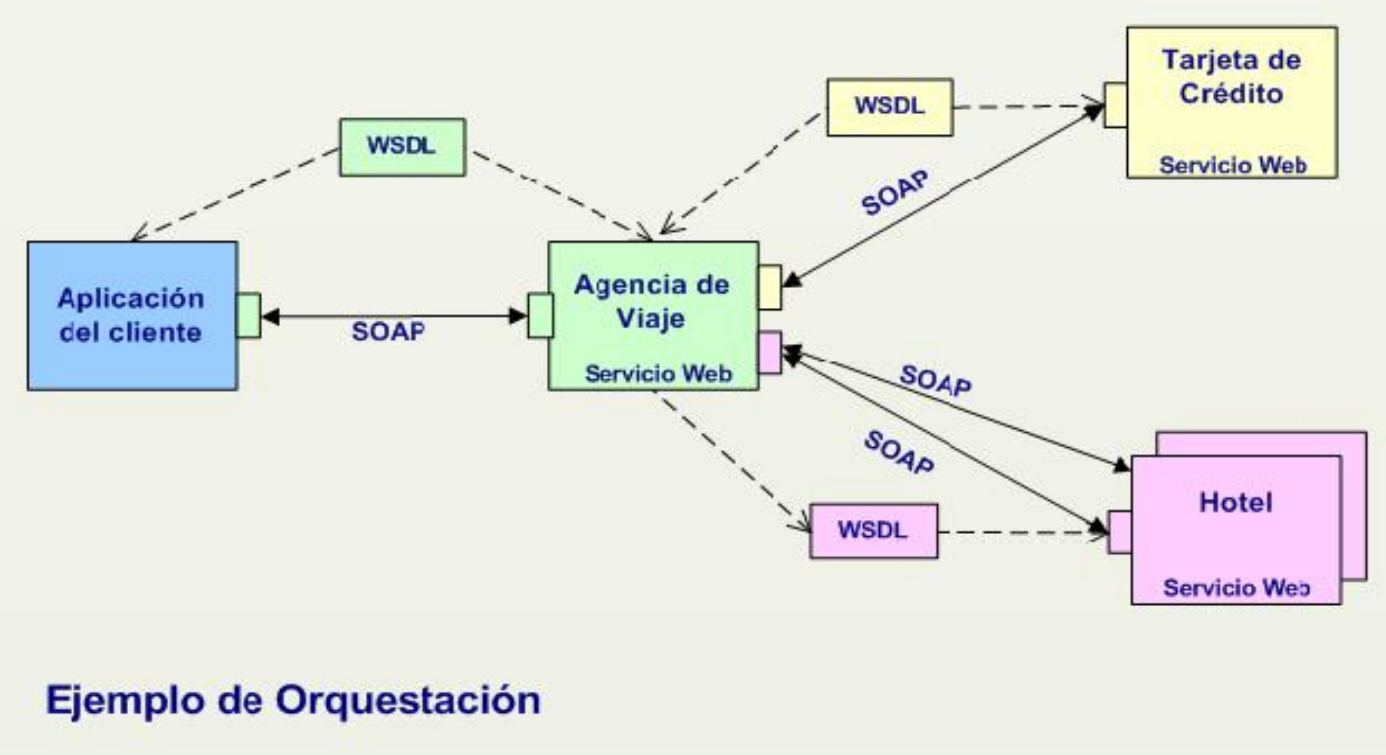
Anexos 4 Servicio Intermediario



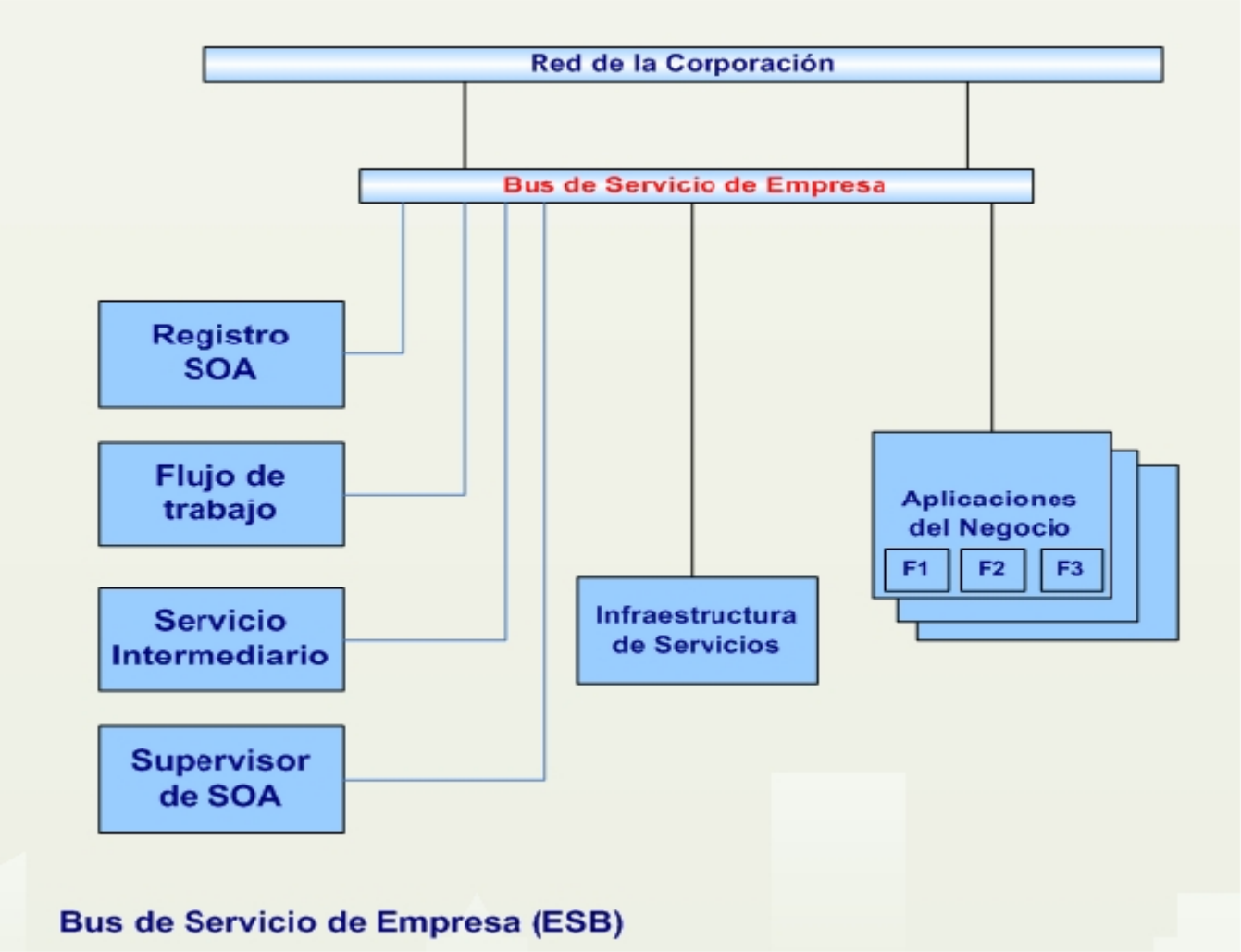
Anexos 5 Supervisor de SOA



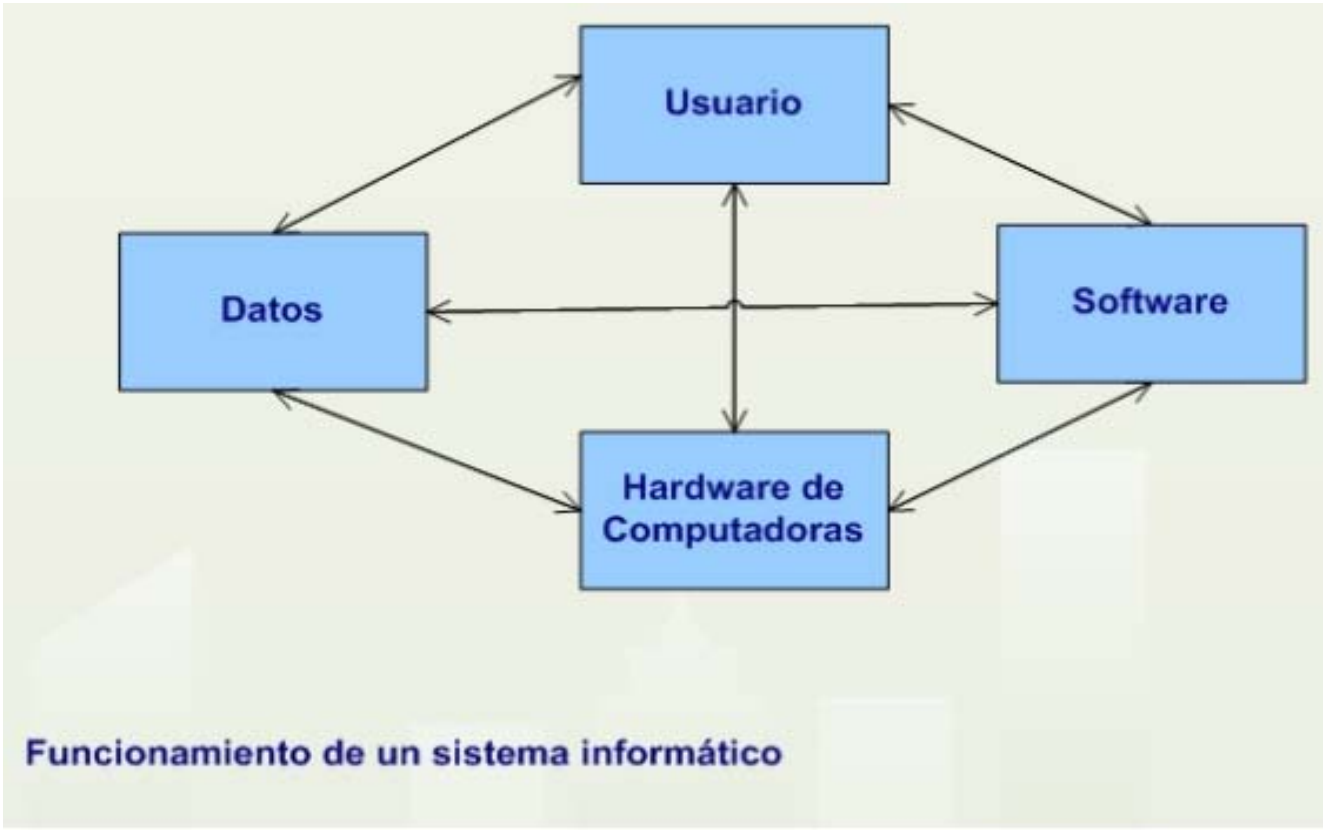
Anexos 6 Ejemplo de orquestación



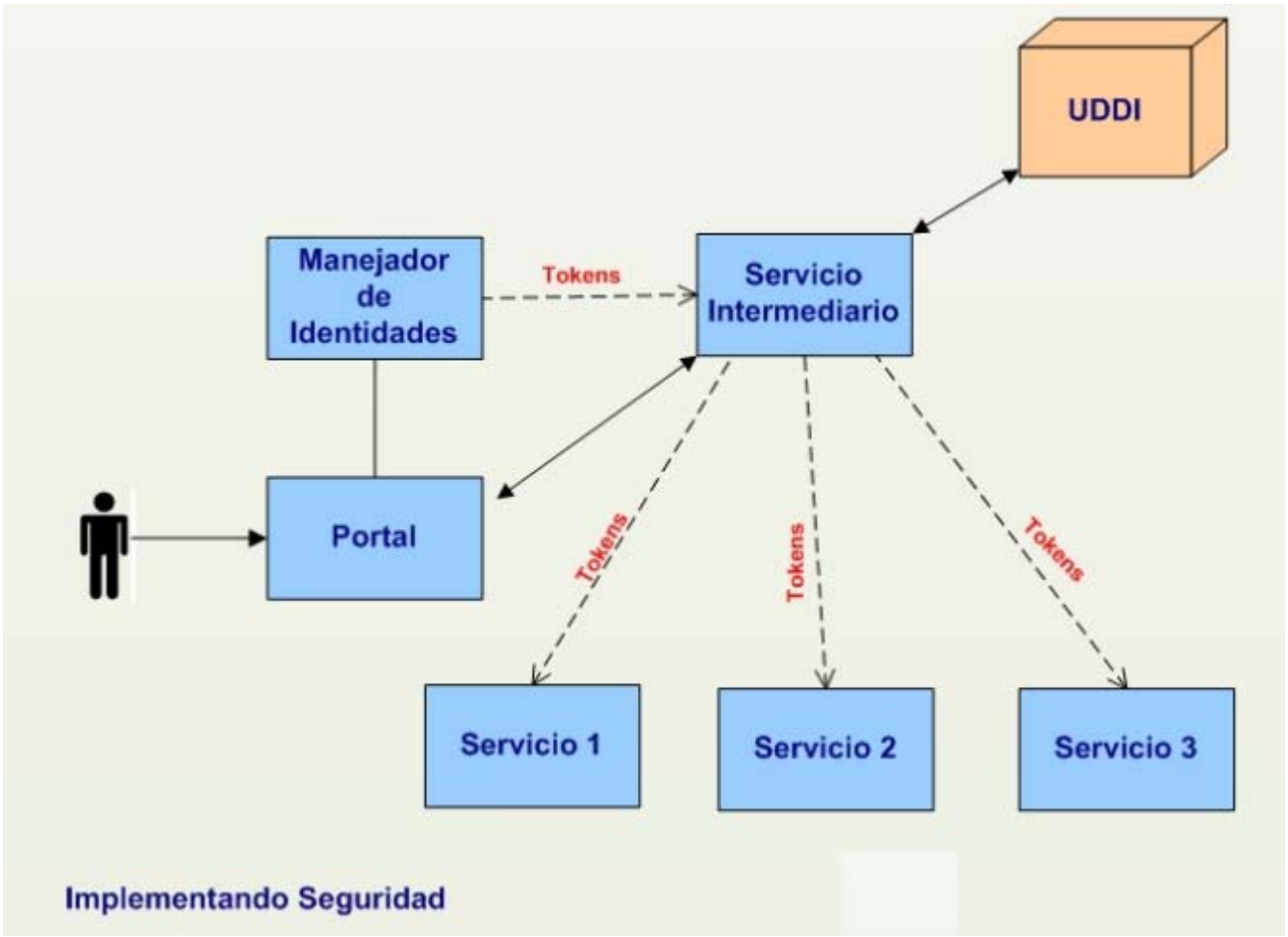
Anexos 7 Bus de Servicio de Empresa (ESB)



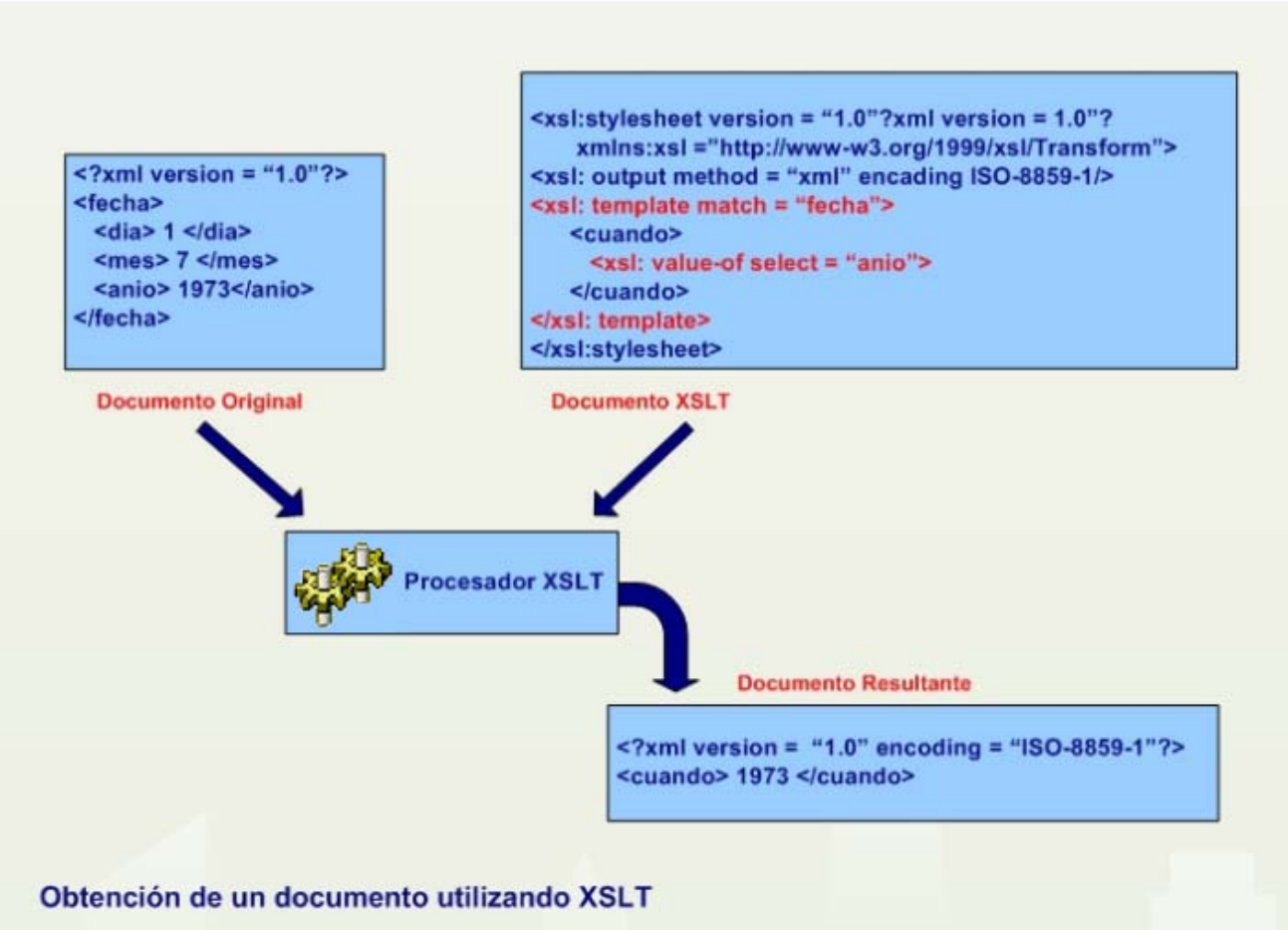
Anexos 8 Funcionamiento de un sistema informático



Anexos 9 Manejador de Identidades



Anexos 10 Obtención de un documento utilizando XSLT



Anexos 11 Partes del WSDL

Ejemplo del PortType

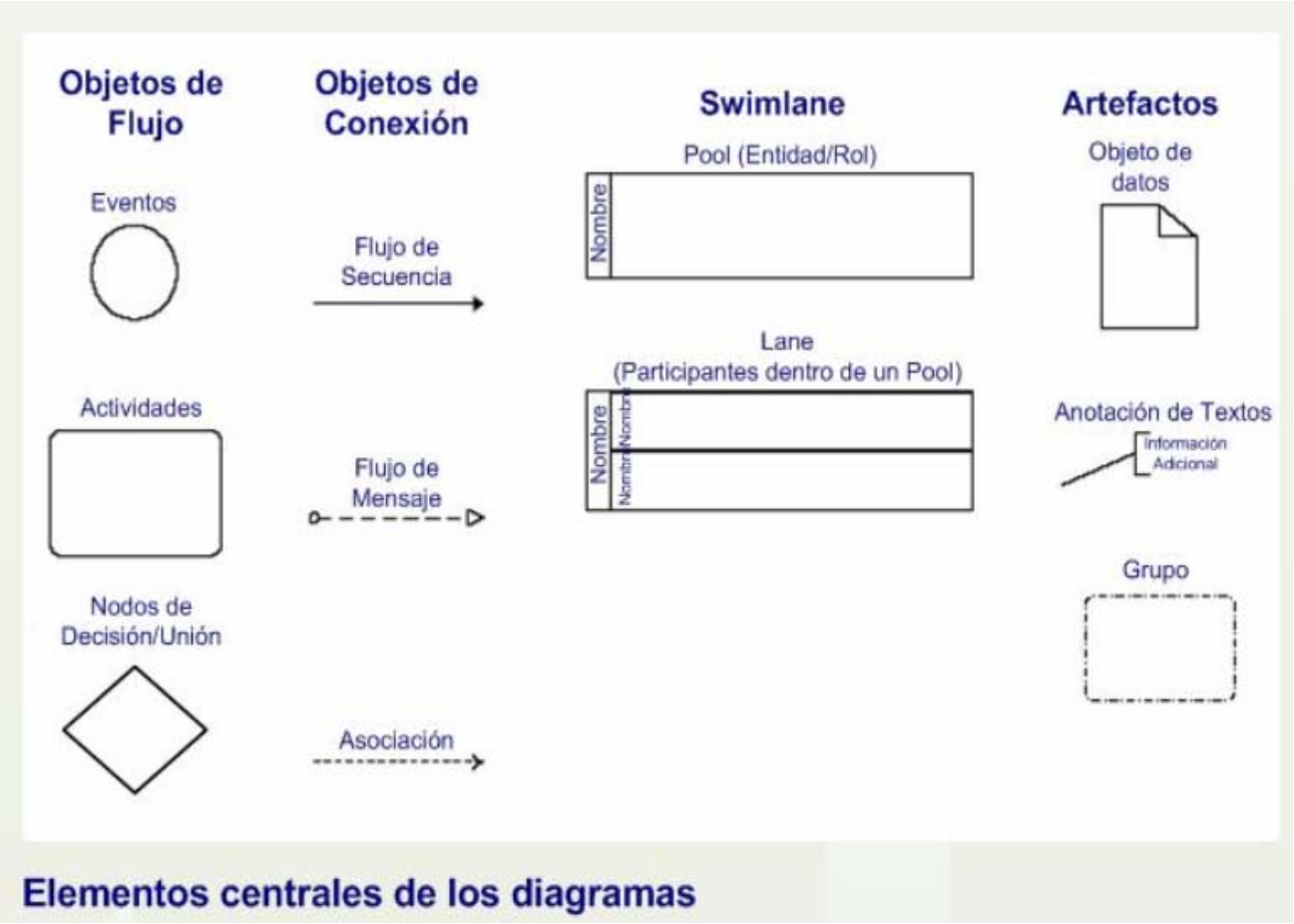
```
<portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
            <input message="tns:GetLastTradePriceInput"/>
            <output message="tns:GetLastTradePriceOutput"/>
        </operation>
</portType>
```

Ejemplo del binding

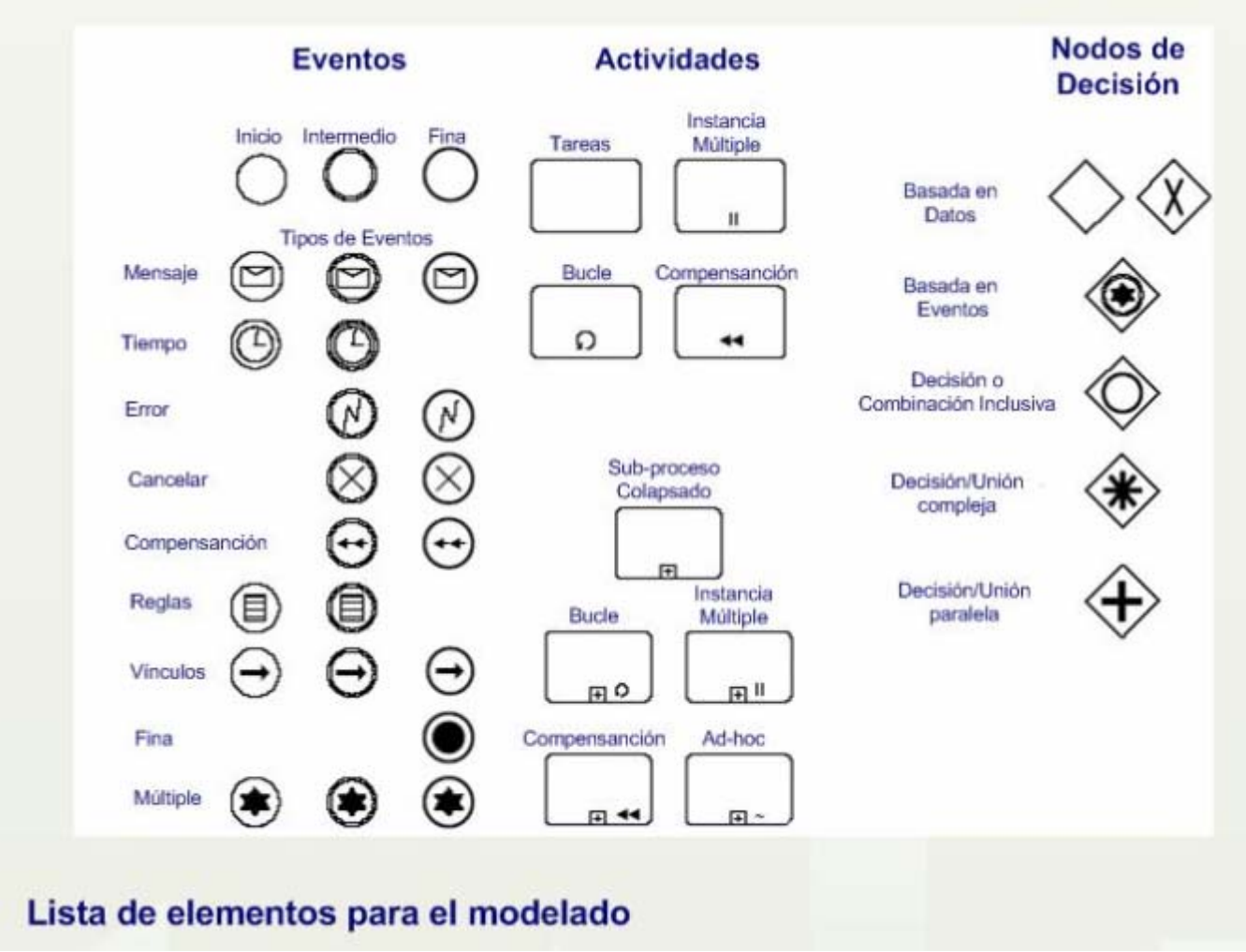
```
<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetLastTradePrice">
        <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
        <input>
            <soap:body use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
        </output>
    </operation>
</binding>
```

Partes del WSDL

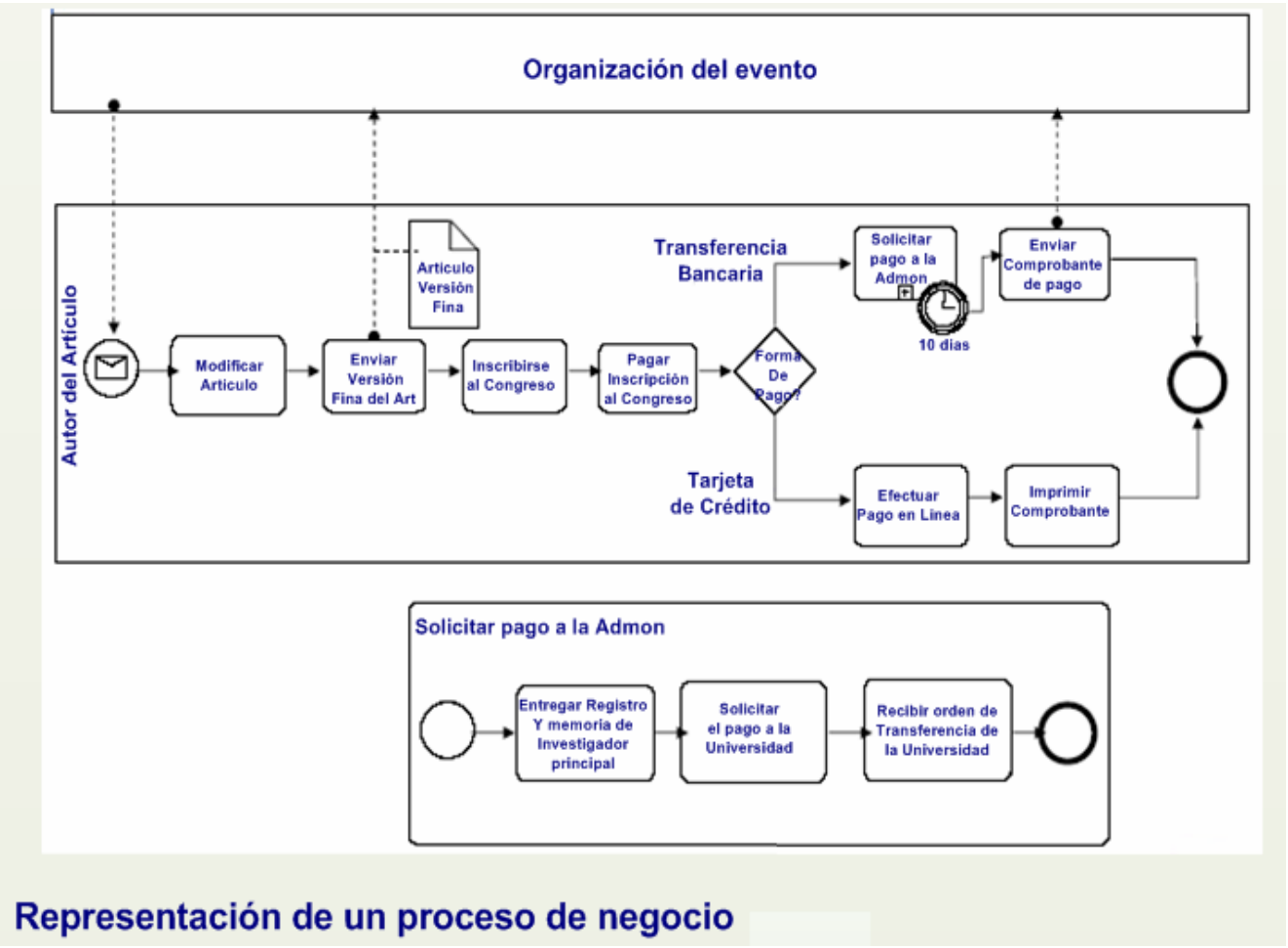
Anexos 12 Lista de elementos centrales de BPMN para los diagramas



Anexos 13 Lista de elementos de BPMN para el modelado



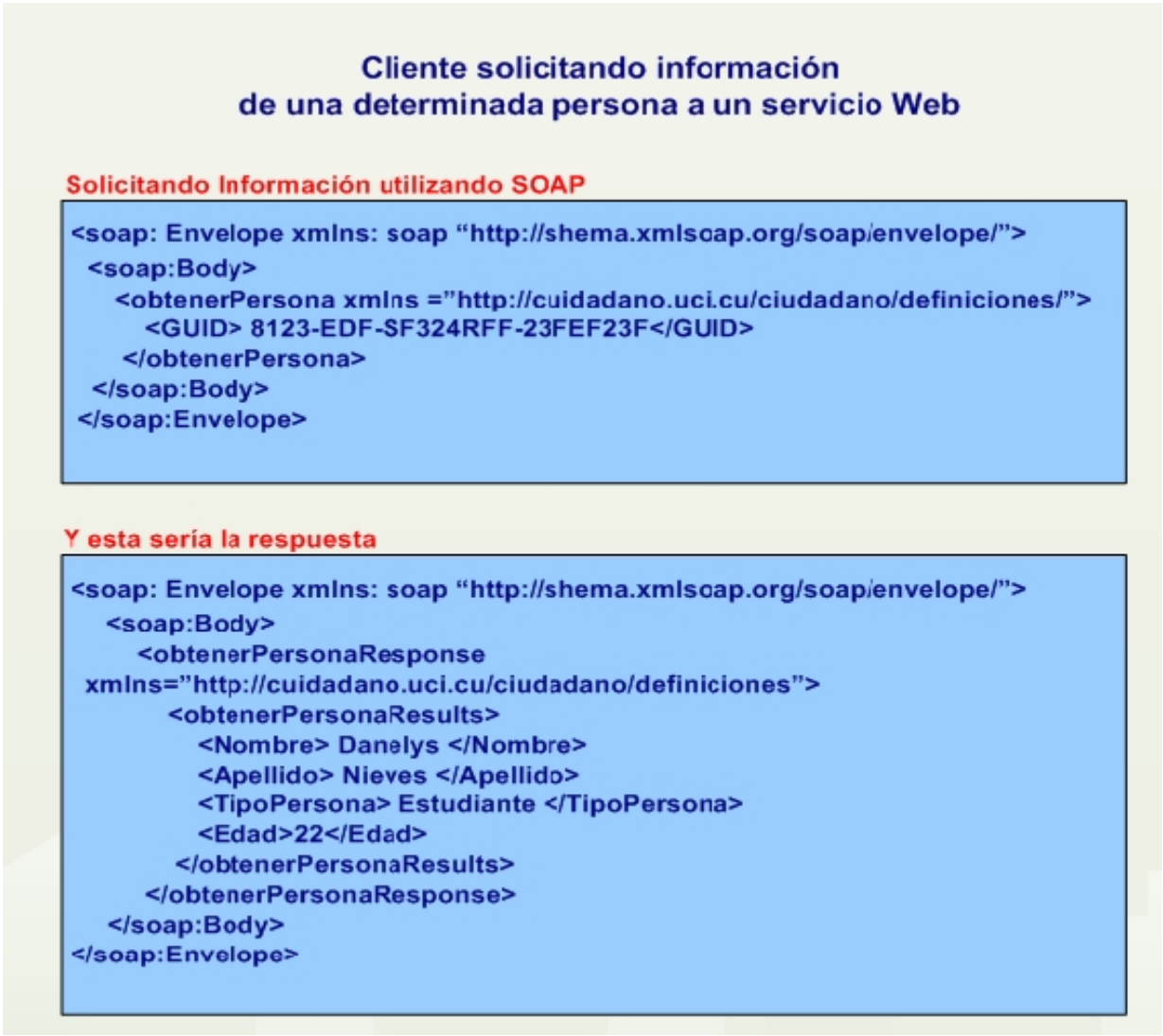
Anexos 14 Representación de un proceso de negocio



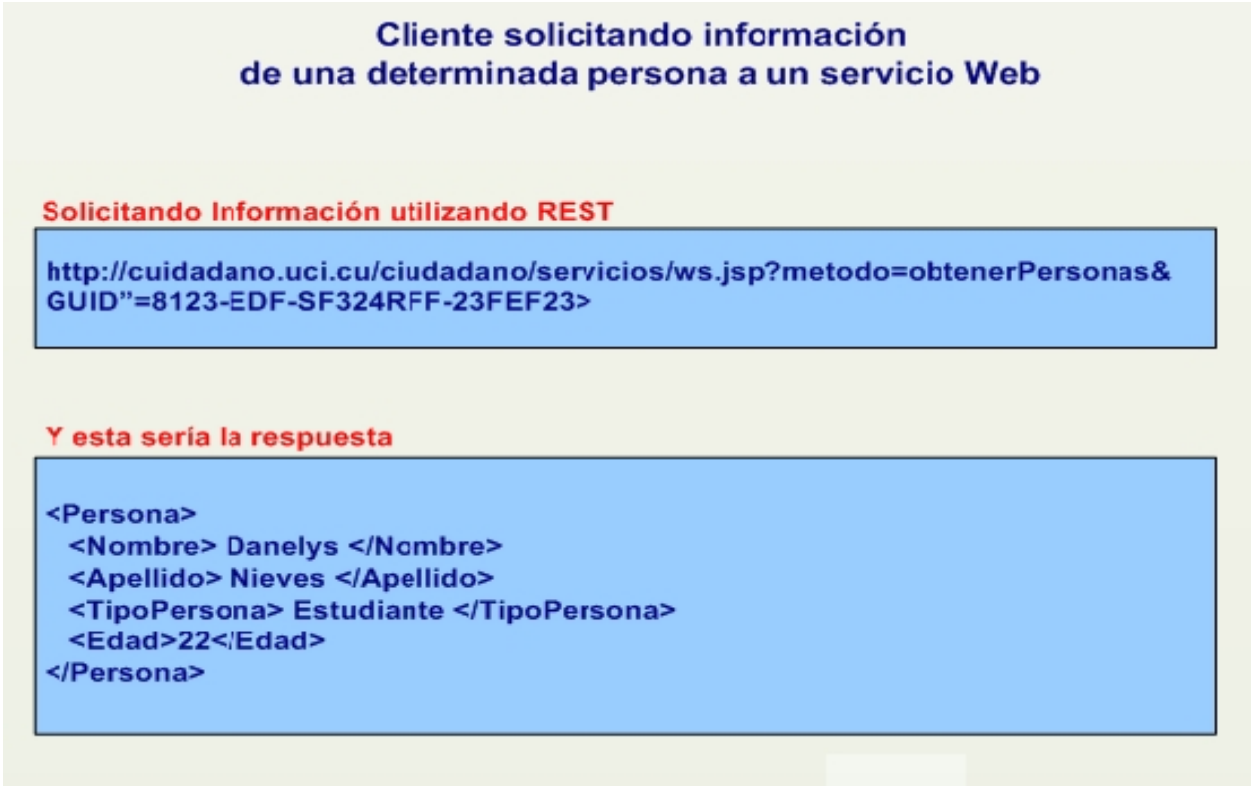
Representación de un proceso de negocio

Anexos 15 Cambio y diseño de soporte utilizando XPDL

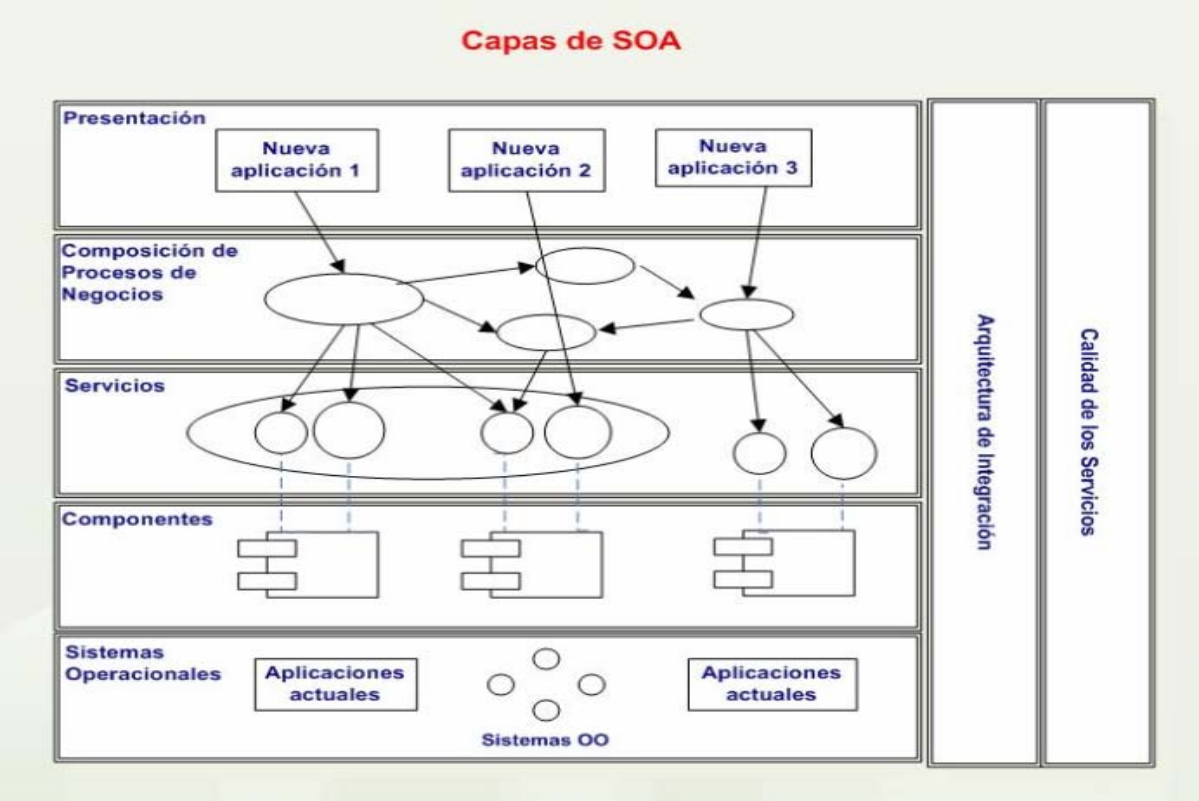
Anexos 16 Cliente solicitando información a un servicio Web utilizando SOAP



Anexos 17 Cliente solicitando información a un servicio Web utilizando REST



Anexos 18 Capas de la Arquitectura Orientada a Servicios (ADOLFO R. DE SOTO; EVA CUERVO FERNÁNDEZ 2006)



Anexos 19 Cambio y diseño de soporte