Facultad de Ingeniería
Escuela de Ingeniería Informática

Arquitectura Orientada a Servicio (SOA): la nueva ingeniería para los sistemas de información

Service Oriented Architecture: the new engineering for the information System

Josué Alvarado Gamboa[1]
Cédula #: 602990641

Universidad Latinoamericana de Ciencia y Tecnología

Tutor: Guillermo Oviedo Blanco

[1] Bachiller en Ingeniería de Sistemas. Candidato a Licenciatura en Ingeniería de Sistemas con énfasis en Gestión de Recursos Tecnológicos, ULACIT. Correo electrónico: jalvaradog@poder-judicial.go.cr

Índice de Contenidos

Resumen ..... *iii*

Abstract ..... *iii*

Introducción: *1*

Definición de la Arquitectura Orientada a Servicios (SOA) *2*
Estándares del SOA 7
Integración en una SOA 9
Arquitectura "Middle-Tier" 13
Requerimientos de integración en una SOA 15
Los componentes y operaciones del SOA: 17
Componentes 17
Operaciones 18
Modelo de una SOA 18
Aplicación del CMM a un proyecto SOA 19
Metodologías en una SOA 20
"Rational Unified Process" (RUP) 21
"Extreme Programming" (XP) 22
Beneficios del SOA 23
Requerimientos de SOA 26

Caso de Negocio *29*

Aspectos acerca de Migrar a una SOA *34*
Los Servicios y su Naturaleza: 36
Servicios como interfaces Compuestas 37
Orientación a los Problemas de conexiones: 37

Service Oriented Development of Application (SODA) *41*
Requisitos para diseñar bajo SODA 42
Requerimientos para el "run-time" de SODA: 43

El Panorama Actual y futuro del SOA *47*

Resumen

Muchas organizaciones ansían contar con opciones que le permitan vincular de alguna forma los actuales sistemas de información, no obstante años atrás era muy difícil lograrlo, ya que existían gran cantidad de fabricantes con ideas y tecnologías propietarias. Ahora, gracias a la Arquitectura Orientada a Servicios, se pueden crear estándares abiertos, que combinados con soluciones de plataforma independiente e Internet, empieza a generar muchas facilidades, culminando con ese acople entre los sistemas de una forma rápida, flexible y económica. Es importante mencionar que aspectos tecnológicos y de técnicas en la programación de los actuales aplicativos, podrían generar complejidad para la puesta en marcha de un sistema basado en servicios. Solo el tiempo dirá si esta arquitectura predominará en los próximos años, aunque todo apunta afirmativamente, según proyecciones realizadas por consultoras reconocidas tales como: “Gartner Group”, “M7 Corporation”, entre otras.

Abstract

Many organizations want to have options that allow to link in any form the current information systems, although the way to do it was something difficult in previous years, due to the existence of many providers with ideas and proprietary technologies. Now, with the Service Oriented Architecture, is possible to make open standard that combined with independent platform technologic and internet, begin to generate more facilities, culminating of a way quickly, flexible and economic this connections between the applications. It is important to mention that technological aspects and of techniques in the programming of the current systems could generate complexity for the implementation of a service based applications. Only the time will say if this architecture will predominate in the next years, although everything aims affirmatively, according to projections made by recognized consultants such as: “Gartner Group”, “M7 Corporation”, etc.

Introducción

En épocas recientes, gran cantidad de empresas están implementando sistemas de tecnologías de información (TI), para aquellos procesos críticos, que la empresa maneja, tales como Recursos Humanos, Contabilidad, Servicio al cliente, por decir algunos en condición de ejemplo; no obstante es común que cada uno de estos, sean independientes y poco o nada acoplables entre si con sistemas propios de la empresa o de terceros.

Si se considera en reemplazar los sistemas actuales por nuevos, además de ser muy costoso, podría resultar algo riesgoso. Por lo que el reto es encontrar una solución que sea extensible, flexible y que permita de alguna forma aprovechar lo que se tiene, sin olvidar también el acoplamiento para con los demás sistemas.

Estos desafíos son lo que pretende alcanzar la Arquitectura Orientada a Servicios (SOA), la cual tiene como objetivo proporcionar una solución relativamente rentable para tratar estos retos. SOA acoge la atención por parte de los analistas de TI, gracias a la aparición de sistemas de plataforma independientes y modelos de datos de plataforma neutral.

En este artículo se examinará la anatomía del SOA, sus ventajas y desventajas, las consideraciones para migrar las aplicaciones actuales bajo esta arquitectura, a la vez se describirá un caso de negocio que guiará desde su modelo conceptual hasta llegar a un diseño orientado a servicio.

Definición de la Arquitectura Orientada a Servicios (SOA)

W3C ("World Wide Web Consortium") define SOA como: "Conjunto de componentes que pueden ser invocados, cuyas descripciones de interfaces se pueden publicar y descubrir".

CBDI ("Computer By Design Incorporated") lo define: "Estilo resultante de políticas, prácticas y "frameworks" que permiten que la funcionalidad de una aplicación se pueda proveer y consumir como conjuntos de servicios, con una granularidad relevante para el consumidor. Los servicios pueden invocarse, publicarse y descubrirse y están abstraídos de su implementación utilizando una sola forma estándar de interfaces".

IBM ("International Business Machines Corporation") se refiere a SOA como: "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 negocios"

Gartner Group (proporciona "benchmarking", mediciones y evaluaciones de los sistemas y funciones de TI) explica: "SOA es una arquitectura de software que comienza con una definición de interfaces y construye todo el estudio de la aplicación como una topología de interfaces, implementaciones e invocaciones. Sería mejor llamada arquitectura orientada a interfaces, SOA es una relación de servicios y consumidores de servicios, ambos suficientemente amplios para representar una función de negocios completa".

El SOA comienza a tener más auge debido a que las tradicionales arquitecturas cada vez se vuelven más complejas a la hora de satisfacer las necesidades de realizar un sistema, de responder de una forma rápida y de reducir los costos que

estos manejan, al mismo tiempo la habilidad de absorber e integrar con sistemas nuevos o con las aplicaciones de negocios aliados.

La SOA dentro del ambiente de negocio, provoca que un sistema concrete todas sus funcionalidades como servicios independientes, con interfaces invocables bien definidas, que se pueden llamar en secuencias precisas para formar procesos de negocio. Si se fracciona este concepto, se tiene:

1. Todas las funciones son definidas como servicios: esto incluye funciones puras del negocio, transacciones de negocio integradas por funciones sencillas.

2. Todos los servicios son independientes: operando como cajas negras, los componentes externos no tienen conocimiento de cómo se realiza, simplemente se tiene el resultado que se espera.

3. En el sentido más general, las interfaces son invocables: se refiere a que es irrelevante si las llamadas de los servicios son locales (dentro del sistema) o remotos (sistemas externos), que interconectan esquemas o protocolos que son usados para los efectos de solicitud.

De todo lo anterior, es importante recalcar a la interfaz o conexión, ya que definen los parámetros requeridos y la naturaleza del servicio, que es totalmente diferente a la tecnología usada para implementarla; es responsabilidad del sistema efectuar y administrar la invocación de los servicios, dando lugar a dos características críticas:

1. Los servicios son realmente independientes.

2. Los servicios pueden ser administrados y estos incluyen algunas funciones como las siguientes:
    a. Seguridad: autorización de la solicitud, encriptación, desencriptación y validación.
    b. Instalado: este permitirá administrar la instalación de servicios en múltiples ubicaciones en donde va estar presente, ya sea por asuntos de rendimiento, disponibilidad, entre otras razones.
    c. “Login”: para aspectos de auditoría, mediciones, monitores, etc.
    d. Reruteo dinámico: para el soporte de balanceo de carga y manejo de fallas.

Ubicar este concepto en ocasiones es algo confuso; una explicación con una analogía facilitará el entenderlo. Se empezará con la necesidad de conectar componentes de audio y video (en donde los servicios en una SOA, serían estos componentes que tendrán opciones de entrada y salida, para dejar a las interfaces como encargadas de todas las conexiones que llevará a cabo la comunicación entre estos componentes).

El conjunto de componentes que se incluyen, son los siguientes: una televisión, un reproductor de VHS, una caja decodificadora de cable, un tocador de CD; donde todos se encuentran debidamente comunicados por un tipo de interfaz, en este caso un cable RCA, importante mencionar que cada componente es de fabricante distinto.

A simple vista ya se tiene una idea de esta arquitectura, sin embargo falta un pequeño detalle; los equipos tienen tres años de antigüedad y se decide adquirir un moderno sistema de teatro en casa, encontrándose gran cantidad de conectores nuevos, que mejoran radicalmente la calidad visual y auditiva, no obstante el actual televisor solo cuenta con antiguos conectores RCA para recibir

imágenes, mientras el teatro en casa maneja salida de S-video (súper video) y HDTV (TV de alta definición).

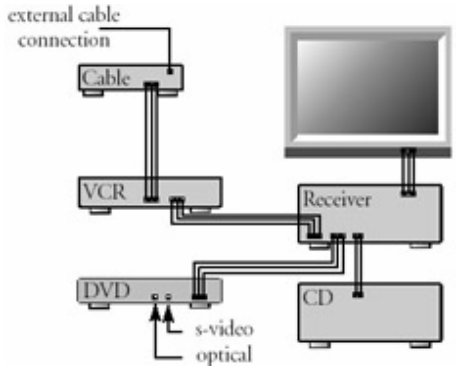

Figura 1: Componentes de audio y video. Fuente: Douglas Barry (2003)

En este corto relato se deja ver en esencia el concepto del SOA y el problema que este soluciona. En efecto lo que se pretende es tener un conector que sin importar la tecnología usada pueda acoplarse a cualquier característica o nivel de complejidad y de funcionalidad que posea. A continuación se dará una breve explicación de conceptos claves que ayudarán a entender más claramente lo que conlleva una SOA.

Servicios

En pocas palabras, un servicio es simplemente una función, la cual tiene que estar clara en su definición, la misma debe auto-contenerse y no depender del contexto o estado de otros servicios. ¿Y qué significado tiene esto para el desarrollo de Software? poca gente que trabajan en la escritura de software y más organizaciones que compran en lugar de construirlos.

“Extensible Markup Language” (XML)

Es un lenguaje diseñado específicamente para documentos Web, este permite que los diseñadores establezcan sus propios “Tags” (comandos insertados en un documento que especifica que parte del mismo deberá ser ajustado a un formato), habilitando la transmisión, definición, validación y la interpretación de datos entre aplicaciones.

Interfaces

La tecnología de “Web Services” ha sido la forma más común a la hora de acoplar los servicios. La manera en que el “Web Service” crea conexiones robustas, es debido al uso de XML; este lenguaje permite realizar comunicaciones fijas, que anteriormente fallaban si manejaban formatos no adecuados, además este lenguaje permitirá enviar mucho más datos de lo que quizás antes no se contaba.

Esto no quiere decir que anteriormente no existían tecnologías que pudieran permitir este tipo de comportamiento, de hecho los primeros pasos que dio la arquitectura orientada a servicios fue el DCOM (“Distributed Component Object Model”) y el CORBA (“Common Object Request Broker Architecture”), que permitían establecer sus objetos como servicios para otras aplicaciones construidas sobre estas tecnologías; las mismas se detallarán ampliamente más adelante.

Llegando al concepto de “Web Service”, este es definido como una aplicación identificada por un URI (“Uniform Resource Identifier”), cuyas interfaces y vinculaciones se pueden definir, describir y descubrir mediante artefactos XML, que soporta interacciones usando mensajes basados en XML vía protocolos de web (W3C, 2005).

Estándares del SOA

El conjunto de los principales estándares que integran SOA y que a su vez forman parte de los roles importantes del “Web Service” son los siguientes:

1. “Web Services Description Language” (WSDL): es el lenguaje con el que un proveedor de servicio describirá ese servicio.

2. “Universal Description, Discovery, and Integration” (UDDI): está pensado eventualmente para servir como medio descubridor de servicios. Es como un diccionario, en donde los distintos servicios pueden ser buscados y obtener la información del contacto y el “Web Service” disponible.

3. “Business Process Execution Language” (BPEL): un lenguaje basado en XML para la estandarización de los procesos del negocio en un ambiente distribuido, que permitirá la interconexión entre aplicaciones y tener los datos compartidos, ya sean en un ambiente local o con terceros. BPEL es conocido también como BPEL4WS.

4. “Simple Object access Protocol” (SOAP): provee el mecanismo para enviar mensajes “Web Services”. La mayoría de estos envíos se llevan a cabo vía “http” (Internet), no obstante se podrían usar otros tipos de conexiones.

5. “Web Service Security” (WS-Security): describe mejoras a la mensajería del SOAP, para proporcionar una verificación en la calidad de la integridad, confidencialidad y autentificación del mensaje.

Integrar metodologías basadas en los anteriores estándares abiertos, dará como resultado el conjunto de servicios vinculados, que facilitará todo el proceso de enlace con otras tecnologías.

Las interfaces que utilizan los servicios para exponer su funcionalidad son gobernadas por agentes propios de un proveedor seleccionado, los cuales definen claramente el conjunto de mensajes soportados, su contenido y las políticas aplicables.

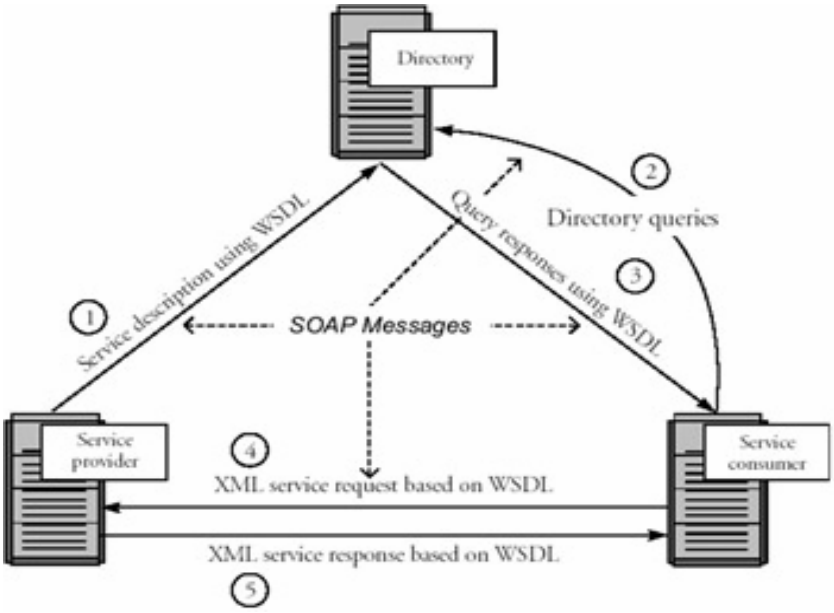

Figura 2: Arquitectura orientada a Servicios básica. Fuente: Douglas Barry (2003)

La figura 2 es una forma de ilustrar la arquitectura orientada a servicios, esta muestra como un consumidor de servicio, envía un mensaje al proveedor, solicitándole la utilización de un servicio. El proveedor retorna un mensaje de respuesta al consumidor; la solicitud y las subsecuentes conexiones de respuesta son definidas de una manera que es comprendida tanto por el solicitador del servicio como por el proveedor, se destaca que un proveedor de servicio puede asumir el papel de un solicitante.

Estas conexiones van a hacer realizadas por “Web Services” y los componentes que este integra; continuando con la figura 2 se tiene:

1. Un proveedor de servicio que se auto describe, usando el WSDL, la definición es publicada al directorio de servicios, en este caso será el UDDI.

2. El consumidor de servicios realiza una o más consultas al directorio para localizar un servicio y determinar como se irá a comunicar con el mismo.

3. Parte del WSDL provisto por el proveedor del servicio es pasado al consumidor, esto le dice al primero, cuáles son las peticiones y las respuestas para él mismo.

4. El consumidor de servicio usa WSDL para enviar una petición al proveedor del servicio.

5. Este proveedor suministrará la respuesta esperada por el consumidor del servicio.

Integración en una SOA

Para poder llevar a cabo las vinculaciones entre los sistemas, utilizando los estándares vistos párrafos atrás, es necesario analizar en detalles las siguientes técnicas:

1. “MiddleWare”: este término puede traducirse como intermediario, ya que su objetivo es suministrar soporte para que dos o más aplicaciones o servicios puedan conectarse, ocultando la complejidad al momento de realizar la comunicación, simplificando radicalmente el desarrollo. Las diferentes aplicaciones o servicios pueden ejecutarse sobre la misma

plataforma o en una diferente. Seguidamente se describirán tres "MiddleWare":

a. "Common Object Request Broker Architecture" (CORBA): fue desarrollado gracias al auspicio del "Object Management Group"(OMG). Todos los sistemas basados en este "MiddleWare" son totalmente interoperables, sin importar el fabricante, el lenguaje de programación, sistema operativo y tipo de red.
b. "Distributed Component Object Model" (DCOM): fue introducido en 1996, siendo una extensión del "Component Object Model" (COM) y se diseña para usarse entre múltiples transportes de red, incluyendo protocolos de Internet como "http". Este "MiddleWare" trabaja solamente bajo sistemas operativos de Microsoft Windows.
c. "Web Service": este viene a ser como un complemento a todas las formas de "MiddleWare", absorbiendo sus funciones y permitiendo, claro está, la creación de sistemas y servicios interoperables más estandarizados. Pero al adoptar el "Web Service", en el caso de empresas que han basado sus sistemas en CORBA o DCOM por ejemplo, no significa que deban desecharlo, ya que no existe ningún inconveniente en que las aplicaciones existentes participen en un "Web Service" alterando la manera en que un "MiddleWare" trabaja, así como se aprecia en la figura 3.

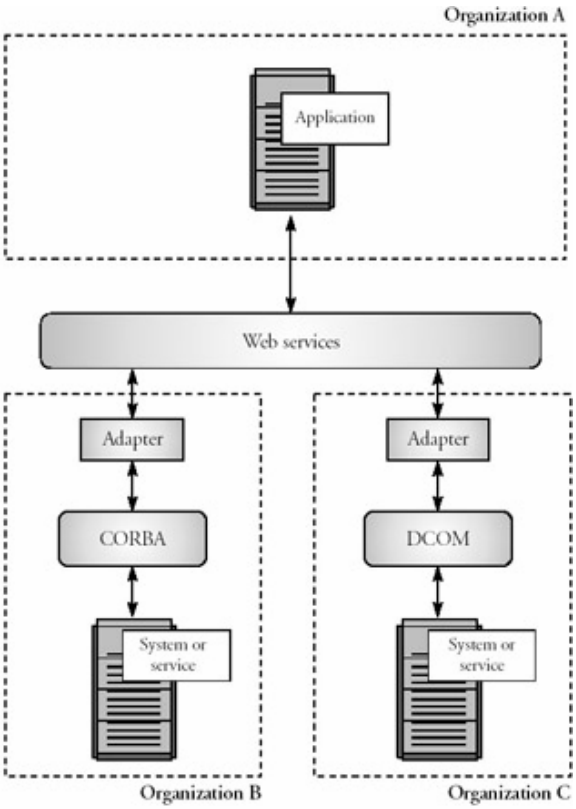

Figura 3: “Web Service” interoperando con CORBA y DCOM. Fuente: Douglas Barry (2003)

2. “Data Warehousing”: esta expresión se refiere a la combinación de datos, de muchas diversas fuentes a través de una empresa. Extraer estos datos que acceden los sistemas actuales y cargarlos en un solo lugar centralizado llegan a formar lo que se conoce como un “Enterprise Data Warehouse” (EDW), la cual ha sido una antigua y acertada práctica para integrar datos entre las aplicaciones. No obstante surge la necesidad de aplicar técnicas que permitan generar un EDW, sin importar el lugar donde se encuentren alojados los datos; para ello es necesario contar con software de “Extract, Transform, and Load” (ETL), que proporcionará todo el proceso de extraer, transformar y cargar los datos al EDW. Una vez completada esta acción, el acceso rápido y fácil a estos datos permitirá el uso de software especializados a lo que es llamado “Business Intelligence”

(BI). Este permitirá buscar patrones o nuevas oportunidades de negocio basados en la abundancia de datos contenidos en el EDW.

3. “Message Router”: Frecuentemente cuando se están integrando los sistemas en la empresa, es necesario propagar los datos entre todos ellos. Por ejemplo, si la dirección de un cliente ha sido cambiada en uno de estos sistemas, se desearía que ese movimiento fuera realizado en los otros sistemas. Si cada software interno mantuvieran enlaces entre si, se podría obtener una excesiva cantidad de conexiones. Así como se muestra en la figura 4.

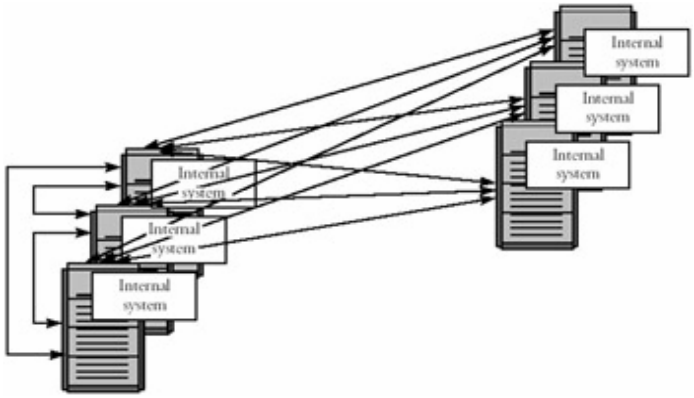

Figura 4: Posibles interconexiones en sistemas internos. Fuente: Douglas Barry (2003)

Arquitectónicamente, la mejor solución sería agregar un “Message Router” en los sistemas internos, así como se observa en la figura 5, es decir un ruteador de aplicaciones, el cual sería basado en “Web Service”. Este encaminador sabría a cuáles sistemas internos les corresponderá recibir un cierto tipo de actualización. Para que pueda llevar a cabo esta característica, necesitará transformar y estandarizar la información en formato XML, logrando de esta manera, que el dato esperado coincida con el sistema recibiente.

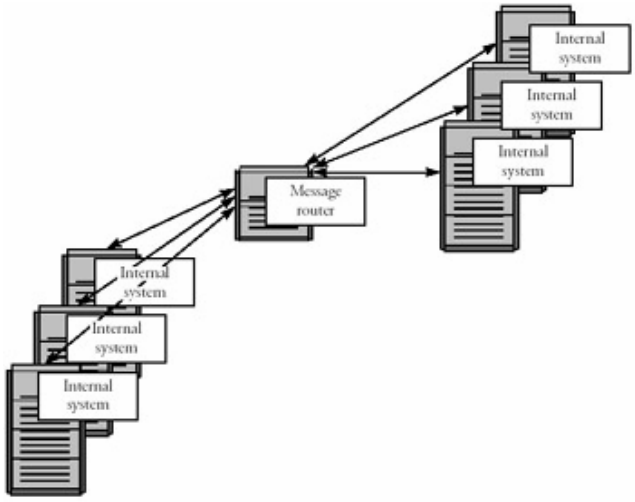

Figura 5: Interconexión cuando se usa un “Message Router”. Fuente: Douglas Barry (2003)

Ahora, si colocamos todas las anteriores técnicas de integración a trabajar en conjunto, estas formarían parte de los componentes, al establecerse en arquitectura orientada a servicios, así como se muestra en la figura 6.

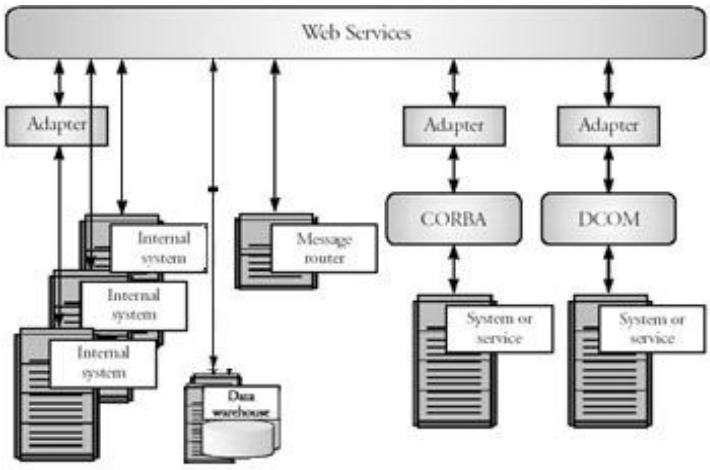

Figura 6: Técnicas de integración en una SOA. Fuente: Douglas Barry (2003)

Arquitectura “Middle-Tier”

Conocida como la capa intermedia, es la forma común de construir “Web Services” usando las bases de datos y sistemas existentes. “Middle-Tier” cambia

donde la integración pueda ocurrir, en ella residen frecuentemente los objetos de negocio. El anterior es la manera de representar la lógica de negocio, incluyendo atributos, comportamientos, relaciones y restricciones. Moviendo las técnicas de integración a una capa intermedia y usando objetos de negocio, se podría evitar futuros conflictos de acceso a datos entre sistemas operacionales y analíticos.

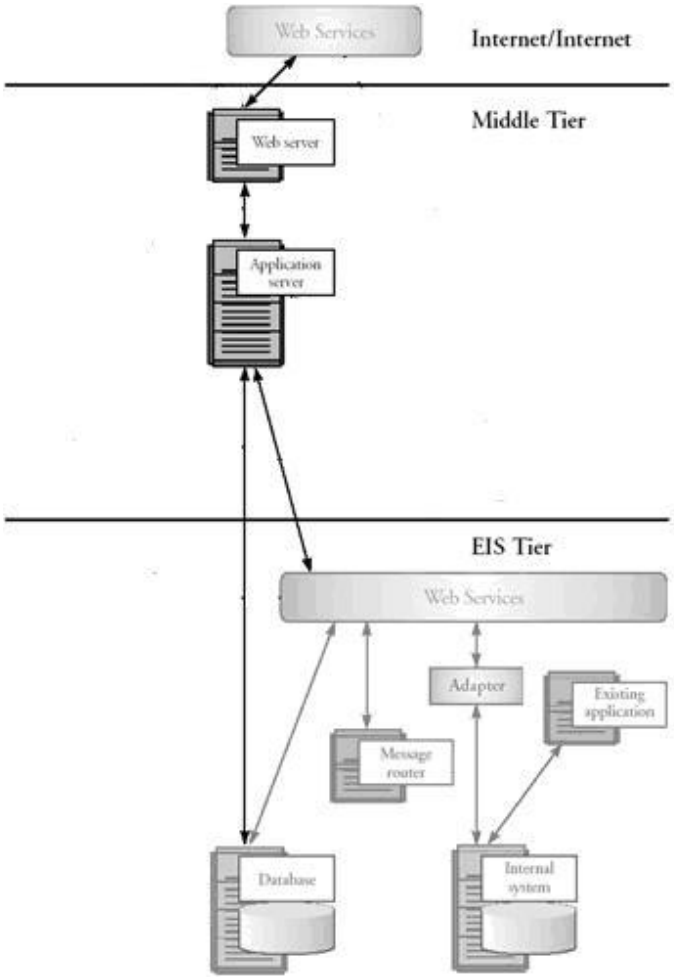

Figura 7: Arquitectura “Middle-tier”. Fuente: Douglas Barry (2003)

De la figura anterior, se rescata como los sistemas internos, “MiddleWare”, EDW y “Message Router” están en una capa llamada “Enterprice Information System” (EIS Tier), ubicada por debajo del “Middle-Tier”. Esta última tiene un “Web Server”

que se conectará con la Internet o Intranet. Las conexiones se dispondrán vía "Web Services" tanto para los servicios internos o externos de la organización. Más abajo del "Web Server" se encuentra un servidor de aplicaciones, cuya función será de crear una infraestructura para la administración e instalación de componentes de aplicaciones ya sea basado en JAVA o .NET. Los componentes dentro del servidor de aplicaciones es el modo usual de realizar abstracciones de alto nivel en los procesos del negocio. Esencialmente la "Middle-Tier" permite ver tanto los sistemas o servicios internos como los externos, de una forma más consistente. Un servidor de aplicaciones puede tener:

1. Acceso a "Web Services" externos

2. Acceso a recursos de Internet.

3. Acceso a "Web Service" internos.

4. Acceso directo a los sistemas internos, saltándose al "Web Service".

Requerimientos de integración en una SOA

La integración se ha destacado en la unificación de aplicaciones, pero mantiene algunas restricciones, que se deben considerar:

1. Integración con la interfaz del usuario final: se refiere a como el conjunto completo de aplicaciones y servicios que se solicitan, son unificados para proveer un estable, eficiente y consistente interfaz. Es un tema de mejora y del nuevo desarrollo que se tendrá a corto plazo, el cual será dominado por avances en el uso de portales en Internet (un sitio que integra múltiples procesos organizacionales en un solo lugar).

2. La conectividad de aplicación: se refiere a todos los tipos de conectividad que la arquitectura debe de soportar, tales como comunicaciones síncronas y asíncronas, enrutamiento, transformación, distribución de alta velocidad de datos, convertidores de protocolos, a la vez relaciona las entradas y salidas de información entre distintas plataformas.

3. La integración de la información: es el proceso de proveer un acceso consistente a todos los datos empresariales, para todas las aplicaciones que la necesiten, en cualquier forma que la ocupen, sin importar el formato, la fuente o localización de los datos. A la hora que se llegue a ocupar este requerimiento, será necesario contar con adaptadores y motores de transformaciones que permitirán homogenizar a un mismo idioma sin importar la plataforma que se utilice. Lo que más llama la atención es el proceso que puede implicar el desarrollo de un bus de datos en donde estos son solicitados a partir de servicios o interfaces estándares por todas las aplicaciones de la empresa. Por lo tanto los datos pueden ser presentados a la aplicación sin importar la proveniencia, ya sea una hoja electrónica, un archivo plano, una base de datos relacional o una estructura de datos en memoria, el formato de cómo esta hecho ese dato no necesariamente debe de ser conocido por la aplicación para que este la acceda; otro aspecto importante es que tampoco ocuparía saber la ubicación del archivo, ya que para eso existirá un servicio general de sistema, que tendrá bajo su responsabilidad la extracción del dato, ya sea que esté local o remoto y además de mostrarlo en el formato solicitado.

Uno de los requerimientos para el ambiente de desarrollo de aplicaciones, es que se considere cada una de las técnicas de integración que pudieran ser

implementados en la organización y prediga su desarrollo y evolución. Para ser verdaderamente robusto, el ambiente de desarrollo debería de incluir una metodología que prescriba claramente como los servicios y componentes van a ser diseñados y construidos en el orden que se quiera facilitar la reutilización, la eliminación de redundancia y la simplificación en los procesos de pruebas.

Los componentes y operaciones del SOA:

Dentro de la gran gama de protagonistas, que conlleva la arquitectura orientada a servicios, se le han definido componentes y operaciones.

Componentes:

1. “Service Provider”: es la entidad que efectúa una determinación de la proveniencia del servicio.

2. “Service consumer”: es también conocido como el “Requestor”, el cliente; este es la entidad que invoca a un “Service provider”, que puede ser una aplicación final u otro servicio.

3. “Service locator": viene a ser un tipo de “Service Provider” que actúa como un buscador de interfaces de servicios y sus respectivas ubicaciones.

4. “Service broker”: es otro tipo de “Service Provider”, que tiene como objetivo el pasar requerimientos de servicios a otros proveedores de servicios.

Operaciones

Son los comportamientos que se tienen a la hora de que una aplicación se base en una SOA:

1. “Publish”: para ser accesible, una descripción del servicio necesita ser publicada de modo que el solicitante del servicio pueda encontrarla.

2. “Find”: en esta operación, el “Service Consumer” recupera una descripción del servicio directamente o consulta al “Service locator” para el tipo de servicio requerido.

3. “Bind”: eventualmente, un servicio necesita ser llamado, en esta operación el “Service Consumer” invoca o inicia una interacción con el servicio en tiempo de ejecución, utilizando los detalles vinculados en la descripción del servicio para localizar, contactar e invocar el servicio.

Modelo de una SOA

En 1991 el Instituto de Ingeniería de Software en Estados Unidos introdujo la versión 1.0 del “Capability Maturity Model” (CMM). Este modelo es usado para describir los principios y prácticas logrando la madurez en el proceso del software. El objetivo del modelo es aumentar la eficacia de la tecnología de información (TI) en las organizaciones a la hora de entregar proyectos informáticos, haciendo el proceso del software más fiable y repetible.

El CMM define un modelo que las organizaciones pueden utilizar para determinar su madurez en el proceso del software, sin embargo también define un modelo que se puede utilizar para avanzar a partir de un nivel a otro. El CMM describe

cinco niveles de madurez, que se pueden caracterizar por los procesos realizados en cada uno de ellos:

1. Inicial: a este nivel, el avance del software es algo confuso. El éxito se basará individualmente por que pocos procesos de negocio estarán definidos.

2. Repetible: los procesos son definidos y repetidos constantemente por un equipo individual de proyecto, para usos similares.

3. Definido: el proceso es bien definido, documentado y estandarizado. Todos los proyectos dentro de la organización usan el mismo proceso de software, adaptándolo a sus necesidades específicas.

4. Administrado: el proceso de software es administrado en términos de aplicabilidad y calidad, a su vez es examinado cuantitativamente.

5. Optimización: la mejora continua del proceso es permitida por la administración cuantitativa; nuevas tecnologías y procesos son incorporados para responder al versátil mercado de la tecnología y de los negocios.

Aplicación del CMM a un proyecto SOA

La utilización de este modelo para el desarrollo de software, es el de ayudar a identificar las necesidades para una SOA en la organización, el hacerlo también puede facilitar la cuantificación de los costos y beneficios de SOA, con el propósito de establecer un sólido retorno de la inversión (ROI) para el proyecto.

Es por eso que algunos consultores especializados en este tema, tales como Kunal Mittal, Jon Bachman, entre otros han coincidido en una adaptación del actual CMM, generando entonces el "SOA Maturity Model" (SOA MM), este por lo tanto permitirá aplicar el CMM a una arquitectura de TI en una organización. Igualmente usando este modelo, se podrá rápidamente construir una visión, un alcance y el plan para un proyecto de SOA, así como determinar los principales indicadores de rendimiento.

Compañías dedicadas al desarrollo de software y firmas consultoras pueden influenciarse del SOA MM para alinear sus productos o servicios, de modo que las organizaciones puedan conseguir aumentar el nivel en su madurez arquitectónico. Después de obtener un proceso y se haya determinado la necesidad de SOA, el próximo paso será identificar la metodología a usar en la construcción del mismo. También será necesaria una metodología para el mantenimiento en curso del proyecto, después de que se haya enrolado en la organización.

Metodologías en una SOA

Una metodología de software es un acercamiento y un control sistemático de cómo desarrollar un software. Difiere totalmente a lo que es un CMM, en donde la finalidad del anterior es velar por la calidad de la metodología y del software realizado bajo esa metodología.
Dos metodologías distintas serán analizadas en el uso de SOA: "Rational Unified Process" (RUP) y "Extreme Programming" (XP). La primera cuenta con una robusta tecnología que es particularmente apropiada para los grandes y complejos proyectos; la segunda es una metodología ágil, que satisface particularmente el desarrollo basado en Internet.

Ambas llegan a cumplir dos fases de alto nivel de SOA. La primera llevada a cabo por RUP, que es la construcción de un sistema basado en SOA y la segunda todo lo referente al mantenimiento, en donde los nuevos proyectos se construyen a partir del SOA inicial, utilizando en este caso la metodología XP. A continuación se detallan las dos metodologías mencionadas.

"Rational Unified Process" (RUP)

Es un proceso de desarrollo de software interactivo, creado inicialmente por la corporación "Rational Software", la cual fue adquirida por IBM. Las primeras interacciones del RUP fueron enfocadas más en la obtención de requerimientos y menos en el desarrollo del software. Pero ahora, el RUP con el uso del "Unified Modeling Language" (UML) en el diseño de sistemas, fortalece la creación de un prototipo completo, durante las fases iniciales de un proyecto, para que incluya todas las capas funcionales y técnicas dentro del sistema final.

Aunque RUP es adaptable dentro de la metodología de cualquier organización, esta es muy preceptiva, pesada y a la vez diseñada alrededor en los procesos de implementación del software. Esta metodología divide el desarrollo del software en cuatro fases distintas:

1. Inicio: esta fase es generalmente donde se define la visión y el alcance para el proyecto.

2. Elaboración: aquí se completa la obtención de requerimientos y los procesos que tendrá el diseño.

3. Construcción: es la fase donde más tiempo se consume, la cual sería la etapa tradicional del desarrollo de un proyecto.

4. Transición: este coloca el sistema en producción y administra todo lo referente a soporte.

IBM mantiene dos consolidadas suites de herramientas para el RUP: “IBM Rational Software Development Platform” e “IBM Rational Suite”, que a la vez provee un total soporte para el desarrollo bajo SOA; eso no quiere decir que sean las únicas suites, ya que IBM implementó un acuerdo de código abierto para mejorar y aprovechar el desarrollo de proyectos informáticos para con otros fabricantes, un caso es “Eclipse Foundation”.

“IBM Rational Software Architect” y “Rational Software Modeler” (parte del “IBM Rational Software Development Platform”) permitirán modelar rápidamente una SOA y con las herramientas de desarrollo del “IBM Rational Suite” permitirán construir e instalar el aplicativo bajo SOA.

“Extreme Programming” (XP)

Es una metodología de desarrollo de software relativamente nueva, que tomó mucha fuerza durante el auge de Internet; XP abarca conceptos tales como: comprobar antes de programar, interacciones cortas, “pair programming” (dos programadores que trabajan lado a lado, colaborando en el mismo diseño, algoritmo, código). Esta metodología se está usando en importantes compañías como: “Ford Motor Company”, “Credit Swiss Life”, teniendo buena aceptación.

XP permite responder de una forma transparente a los cambios en los requerimientos, introduciendo el concepto de “Class-Responsibility-Collaborator”, este último presenta de manera conceptual una colección de tarjetas que se dividen en tres secciones:

1. Clase: representa un conjunto de objetos similares.

2. Responsabilidad: es lo que sabe o hace la clase.

3. Colaborador: viene a ser un comportamiento que interactúa con la clase, para satisfacer sus responsabilidades.

Basados en el concepto anterior el XP facilita una rápida y efectiva forma de capturar requerimientos, para luego organizarlos en estructuras interactivas lógicas, asignando prioridades y desarrollándolos. La debilidad más significativa de esta metodología es que no cubre la validación de requerimientos, aunque esto lo haga más ágil y optimo, se puede correr el riesgo de obtener resultados no satisfactorios.

Beneficios del SOA

Las características técnicas del SOA ha llamado la atención a los analistas e investigadores de TI, debido a los posibles beneficios que pudieran adquirir las organizaciones al implementar proyectos bajo esta arquitectura.

El SOA puede ser desarrollado basado en las aplicaciones existentes evitando requerir de una reescritura completa de los sistemas. El enfoque de las organizaciones en el desarrollo, se debería basar en lo que se refiere a la creación de servicios, usando las tecnologías actuales, combinando con los componentes ya creados, generando esto importantes beneficios:

1. Una mejor infraestructura: el desarrollo de la infraestructura y el “deployment”, llegarían a ser más consistentes entre todas las diferentes aplicaciones empresariales. La palabra “deploy” en este contexto, se refiere

a instalar y probar algo que se tiene listo y que se quiere dejar disponible, para que pueda ser utilizado, es decir que se cuenta con un componente preparado para su lanzamiento y utilización dentro de una aplicación o para formar parte de un servicio; ya sean estos componentes actuales, nuevos o adquiridos por terceros, para que estén consolidados dentro de una estructura de soporte definido en la cual otro proyecto de software puede ser organizado y desarrollado, lo anterior se le conoce como “Framework”. De esta manera cada uno de estos componentes podrían hacer “deploy” como servicios sobre la infraestructura actual.

2. Arquitectura de ensamblaje:
    a. Incrementa la agilidad organizacional, le permite a las compañías acoplar y modificar procesos del negocio en respuesta a los requerimientos del mercado.
    b. Suministra una ventaja competitiva, ofreciendo gran flexibilidad en la utilización de los sistemas informáticos.

3. Basado en estándar:
    a. La independencia a la plataforma, le permite a las compañías usar el software y hardware de su preferencia.
    b. Permite que las compañías se acojan a una estrategia de múltiples fuentes, reduciendo la dependencia a un solo proveedor.
    c. Proporciona economía en escala; la misma tecnología puede ser aplicada para tratar una amplia gama de problemas de negocio.
    d. Reduce la complejidad y la fragmentación resultante a partir del uso de tecnologías propietarias.
    e. Bajos requisitos de entrenamiento para el personal respectivo.

4. Un rápido “Time-to-Market”: se define como la duración del ciclo entre el concepto del producto y su cumplimiento, las librerías “Web Service” organizacionales llegarán a ser el activo base para las empresas que adapten un “Framework” basado en SOA (Bridgefield Group, 2005). Construir y hacer “deploy” en servicios con estas librerías reducirían dramáticamente el “Time-to-Market”, así como las nuevas iniciativas de reutilizar servicios y componentes existentes.

5. La reducción de costos: debido a que las demandas del negocio se desarrollan y se introducen de forma constante nuevos requerimientos, al introducir SOA será mucho más cómodo adaptar y crear nuevos servicios bajo un marco estandarizado, que promoverá la reutilización; de esta manera los servicios podrán ser fácilmente compartidos entre múltiples aplicaciones actuales o nuevas, provocando esta la alta reducción del costo.

6. Mitigación del riesgo: reutilizar los componentes existentes reduce el riesgo de introducir nuevas fallas al proceso de adaptar o crear nuevos servicios.

7. Mejoramiento continúo de los procesos del negocio: SOA permite una representación clara del flujo del proceso, identificados por el orden de componentes usados en un servicio particular de negocio. Esto provee a los usuarios del negocio un ambiente óptimo para supervisar las operaciones empresariales.

Requerimientos de SOA

Las iniciativas de SOA proveerán un gran valor empresarial cuando estas instalen sus aplicativos basados en esta arquitectura, sin embargo es necesario tener claro los siguientes requerimientos:

1. Servicios: comprender de forma clara lo que es un servicio en una SOA y como una empresa podría emplearla.

2. Influencia de tecnologías existentes: tomar la decisión de desechar los sistemas, son raramente considerados, debido a que estos en la mayoría de los casos, contienen información de gran valor para la empresa. Esencialmente lo ideal sería contar con una nueva arquitectura que cumpla con todo el valor esperado para una empresa, pero que estratégicamente, los sistemas existentes puedan ser integrados tales que, a corto plazo puedan ser reemplazados en proyectos incrementales y manejables.

3. Dar soporte a las técnicas de integración mencionados anteriormente.

4. Registro: este requerimiento especifica que las compañías que empleen el uso de SOA necesitarán compartir, usar y buscar servicios. Por lo tanto deberán de publicar sus servicios en un registro o catálogo el cual corresponde al UDDI.

5. Mensajería: las empresas necesitan un bus de mensaje que permitan a las partes comunicarse entre si confiablemente a través de una infraestructura base de mensajería.

6. Manejable: las empresas requieren una infraestructura o lógica de negocio compartida y sólida que proporcione una clase de robustez y de

confiabilidad que va a ser necesaria para incorporar en un sistema basado en servicios. No basta con tener servicios, es necesario agregar seguridad, monitoreo, manejabilidad, un control de versiones y características básicas que existen o que son esperadas en cualquier sistema empresarial. Y es importante que esta infraestructura compartida trabaje uniformemente sin importar en que plataforma esté contenido el servicio.

7. Orquestación: una capa de orquestación en el "Business Processes Manager" (BPM), es necesaria para facilitar las relaciones entre los servicios, ya que la capacidad de cambiar dinámicamente procesos del negocio es una de las promesas más grandes de SOA. EL BPM consiste con la ayuda de software especializado, en describir actividades y eventos que son realizados para optimizar procesos de negocio, permitiendo a las organizaciones modelar y orquestar gráficamente sus servicios.

8. Permitir la puesta en práctica de nuevos modelos computacionales:
   a. Modelamiento "on-demand computing": este permite que los recursos o procesos, estén disponible para el usuario según como este lo necesite. El "Web Service" es una tecnología que envuelve SOA y esta a su vez es una arquitectura que permite aplicaciones "on-demand".
   b. Modelamiento "Grid computing": este modelamiento es mucho más que el uso de un gran número de MIPS (million instructions per second) para efectuar una solución a los problemas complejos. Este considera la virtualización de todos los recursos del sistema incluyendo el hardware y los datos para poderlos utilizar en cualquier lugar.

9. Incluir un ambiente de desarrollo que permita construir bajo “framework” estándares.

10. Interfaz de usuario: proporciona el componente interactivo a una aplicación compuesta. Esto se refiere a que las empresas desearán exponer sus servicios y procesos como una aplicación combinada a través de un portal personalizado basado en Web.

Caso de Negocio

Cambiando un poco el contexto y ubicando esta tecnología desde el punto de vista conceptual, el hecho de establecer una arquitectura orientada a servicios, puede ser descrito como una visión, por parte de alguna empresa.

En el siguiente párrafo se relata una forma conceptual los distintos requerimientos tecnológicos que se pudieran involucrar en una arquitectura orientada a servicios:

La Organización C.R. como parte de su estrategia requiere que sus ejecutivos de venta puedan contactar y visitar a clientes que cumplan con una serie de características en varias zonas geográficas, para eso se ocupa interactuar con el software CRM (Customer Relationship Manager) accediendo a las bases de datos internas, una vez localizados, este aplicativo se enlazará entonces con los sistemas de agencias de viajes y estás a su vez dispondrán los itinerarios de vuelos de las aerolíneas comerciales, para la adecuada calendarización de las visitas a los clientes en la zona identificada previamente, asimismo el aplicativo local mantendrá informado de cualquier cambio en algún itinerario de vuelo a los ejecutivos de venta a través de distintos medios portables, por ejemplo correo electrónico, mensajes cortos de textos a celular, etc.; además, estos tendrán una pequeña aplicación ubicada en dispositivos PDA, que suministrará información correspondiente a sus labores, manteniendo una conexión inalámbrica (donde tenga cobertura) con la red de la empresa para obtener siempre datos actualizados de sus actividades y cuando no haya cobertura, este utilizará una versión ajustada de la base de datos corporativo pero en el ambiente local del PDA. Del mismo modo, el sistema local mantendrá interacción con las aplicaciones de algunas firmas hoteleras usando como puente a la agencia de viajes con el fin de poder programar las reservaciones de forma automatizada.

Finalmente, se tendrá un módulo que se enlaza con sistemas de información de diferentes empresas de rentas de vehículos, con el objetivo de ir escogiendo el automotor a utilizar, programando así la mejor ruta de acceso para situar al cliente por medio de sistemas GPS.

En la figura 8 se resume la tecnología que se ha utilizado en primera instancia en el relato del párrafo anterior; es muy claro que hay un gran despliegue de tecnología en este relato tanto a nivel de software como de hardware.

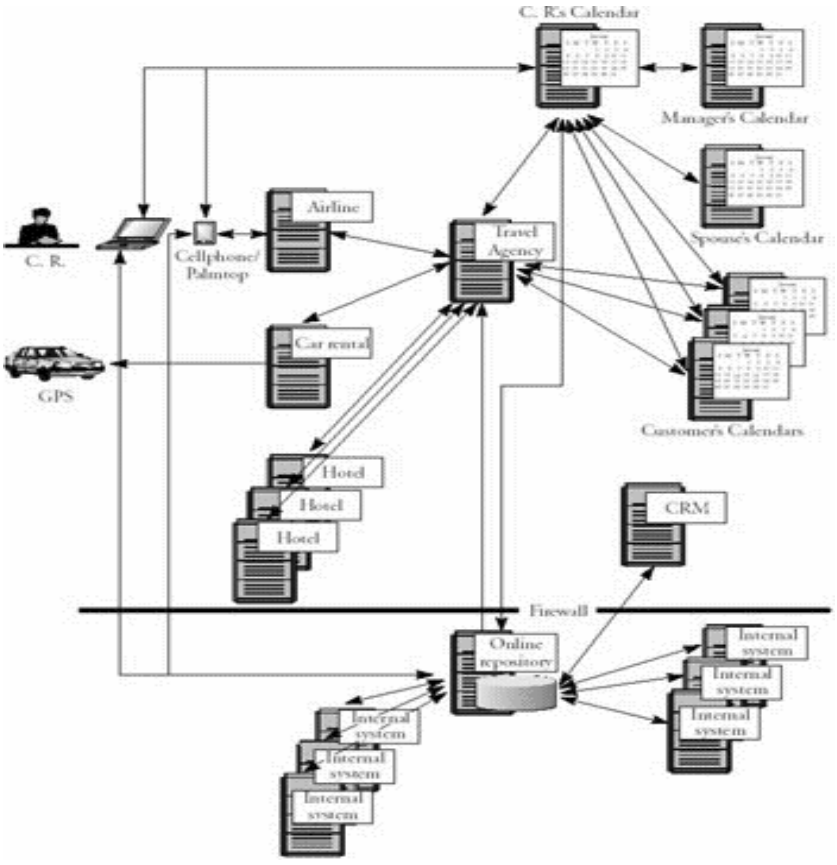

Figura 8: Servicios e intercambios de datos. Fuente: Brian Randell, Rockford Lhotka (2004).

El proceso de llevar a cabo la SOA dependerá con lo que cuente la empresa en relación a los sistemas de información. En la actualidad existen fabricantes que

ofrecen una suite de paquetes que se enfocan en esta arquitectura, tales como Oracle, IBM, Sybase, BEA Systems y Microsoft.

Muchos de los fabricantes indicados generan sus paquetes para la mayoría de los sistemas operativos existentes en el mercado, bajo el hardware en donde estos son instalados, en la mayoría de los casos, con el equipo que se tiene basta para colocar las herramientas que el fabricante requiere para trabajar bajo SOA, teniendo que considerar la curva de aprendizaje de los analistas, el costo de los paquetes, capacitaciones, el nuevo hardware si es necesario, consultorías, etc.

Cuando se da el caso de que es un nuevo proyecto o más bien lo que se tiene a nivel de software no se adapta en la arquitectura orientada a servicios, habrá que considerar una mayor gama de aspectos, tales como cual proveedor se seleccionará, que equipos y que sistema operativo se recomendará para instalar los paquetes, que tipo de interfaz o conexión se usará para la comunicación entre los servicios, etc. En una organización grande, se pueden encontrar diversidad de tecnologías por lo que armar un diseño orientado a servicio podría ser algo confuso cuando en principio no se pensó en enlazar nada.

Siguiendo con el caso de la organización C.R, el reto es de llevar el proyecto a un programa de transformación de negocio electrónico, con el objetivo de integrar los sistemas y procesos que conectan a esta compañía con los involucrados en el caso de estudio. Cuya meta será de ejecutar accesos en tiempo real, mejorar el rendimiento de todos los procesos que se envuelvan.

Esto requerirá conectarse con los sistemas de los aliados, considerando que alguno de ellos esté construido en una tecnología propietaria. Es necesario en este punto una solución que permita basarse en estándares que puedan abrir aquella lógica propietaria o cerrada.

Con la ayuda de proveedores que han asumido lo que conlleva desarrollar proyectos basados en SOA, C.R ha integrado sus sistemas, siendo capaz de recibir procesos de negocio de sus aliados en forma de servicios y a la vez convirtiendo los suyos mismos en servicios, es decir moviéndose a un ambiente basado en estándares, en este caso los de SOA. Con la puesta en marcha de este proyecto se podría incrementar la funcionalidad para todo lo referente a la coordinación para ubicar a clientes estratégicos, despreocupándose de todos los trámites que genera la movilización de los agentes de venta hasta donde se encuentra el muy apreciado cliente.

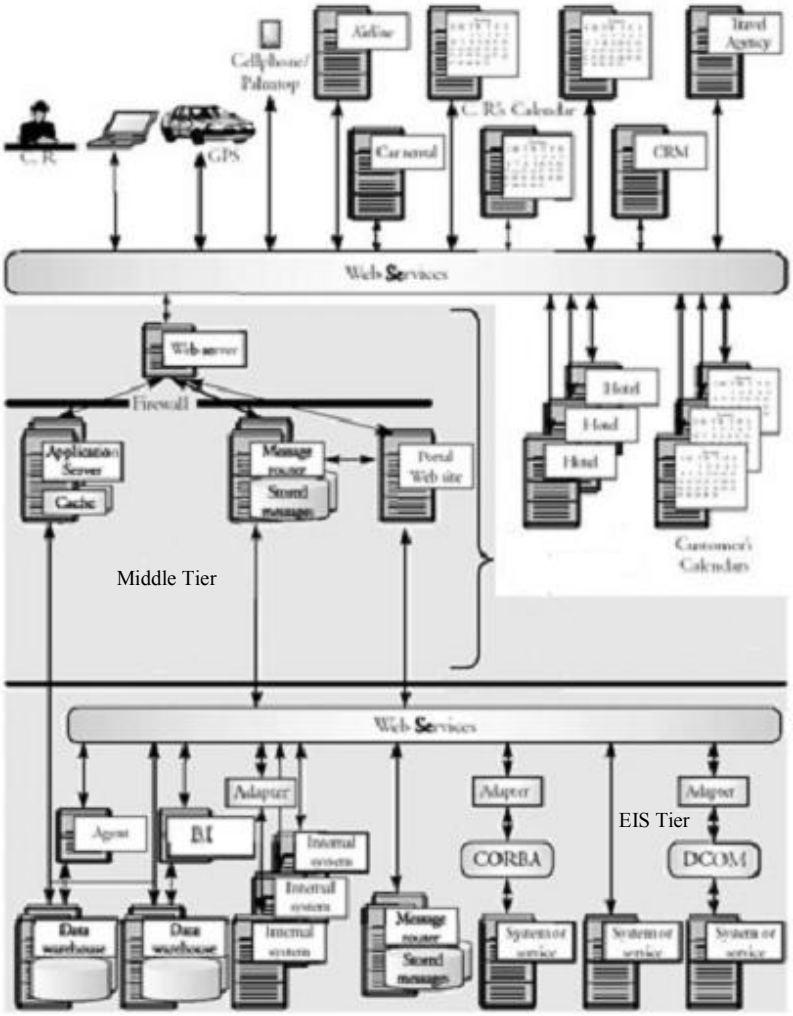

Figura 9: Servicios e intercambios de datos en la organización C.R. Fuente: Brian Randell, Rockford Lhotka (2004)

Como se nota en la figura anterior, la integración empieza en la capa EIS. Luego, subiendo a la capa intermedia, un portal de un sitio en Internet mediante “Web Services”, será el encargado de acceder a la información tanto de los servicios internos como externos de la organización C.R. Es este Portal, con la ayuda de las herramientas de BI, quien permitirá acceder por aquellos clientes seleccionados bajo un criterio de búsqueda, con el fin de localizarlos y monitorearlos. Los mensajes que emiten tanto el portal como el software de BI son controlados por el “Message Router” en la capa intermedia. Y en esta misma capa, el portal interactúa con el servicio de la agencia de viaje, enviándole la información requerida para calendarizar itinerarios de vuelos y hospedajes para la estancia del ejecutivo de venta. Una vez que este proceso termina, por medio de mensajes “Web Services” la información es enviada al portal y este a su vez descargará debidamente formateada esa información en dispositivos móviles (mensaje de texto al celular, o al PDA). Al mismo tiempo estos dispositivos móviles notificarán a cuales clientes se tendrán que contactar para planear esa visita. La aplicación residente en el PDA, interactúa de forma en línea con el portal mediante mensajes “Web Services”, así el ejecutivo de venta, siempre tendrá a la mano toda la información referente al cliente que le corresponda contactar.

De esta forma la organización C.R ha entrelazado una SOA eficaz, interoperando entre los sistemas existentes, el nuevo “Data Warehouse”, “Middle-Tier”, un portal en Internet y una serie de servicios externos que interactúan usando “Web Service”.

Aspectos acerca de Migrar a una SOA

A continuación se considerará algunos de los problemas fundamentales que se podrían esperar a la hora de migrar a SOA, ya que estos de acuerdo a como se traten, determinará el éxito o fracaso del cambio:

1. Complejidad*:* definitivamente la tendencia en arquitectura de sistemas esta sufriendo cambios, ya que se cuentan con ambientes con un alto nivel de complejidad; los sistemas requieren ser reutilizados en lugar de sustituidos, debido al alto costo que se tiene que considerar si se piensa en una solución como esta; no obstante con la llegada de Internet y lo barato que es accederlo, este ha creado la posibilidad de crear nuevos modelos de negocio. Los sistemas deben de ser desarrollados considerando fundamentalmente la heterogeneidad, por que deben de adecuar a una gran variedad de hardware, sistemas operativos, lenguajes, entre otros, generando claro está la complejidad.

2. Programación Redundante y no reutilizable*:* en muchos ambientes de trabajo se puede dar el caso de tener sistemas autónomos, que son independiente de otros, pudiendo mantener un diseño óptimo; no obstante cada uno de esos sistemas fueron creados en una línea diferente de negocio dentro de la misma organización, totalmente aislados. Es muy probable que en estos sistemas existan semejanzas en algunos de sus procesos, accediendo quizás a los mismos datos. ¿Y que sucederá entonces a la hora de que se requiera un sitio en Internet en línea y este requiere integrar módulos de estos sistemas aislados? Lamentablemente un nuevo proyecto más, una inversión más, más redundancia a menos que se pueda reutilizar algo de lo que se tiene hecho.

3. Multiplicidad de interfaces*:* en la mayoría de los casos en que una organización tiene la necesidad de unir sistemas, ya sea por efectuar una alianza de negocio o simplemente por interconectar aplicaciones propias; tendrán que hacerle frente a problemas que conlleva el realizar estos vínculos. Podemos resumir estos problemas mencionando que la cantidad de conexiones o interfaces será igual al producto del total de aplicaciones, con el total de aplicaciones menos una, generando la siguiente fórmula: Z=Y (Y-1), en donde "Z" es la cantidad de conexiones y "Y" el total de aplicaciones a conectar.

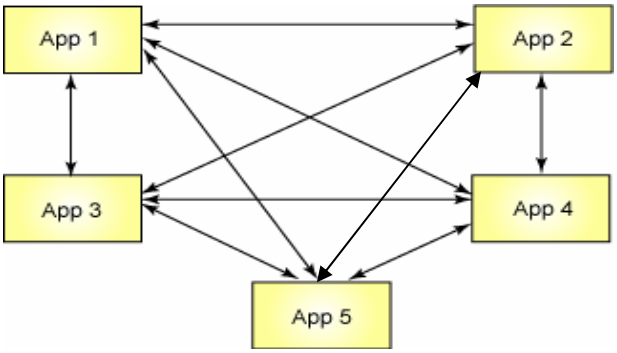

Figura 10: Ejemplo de interfaces en una integración directa de aplicaciones

Como se mira en la gráfica anterior, cinco aplicaciones requerirán 20 interfaces, ahora que pasa si se quiere integrar una sexta, bueno siguiendo la fórmula, esta aumentaría en diez la cantidad de interfaces; peor aún, el código en cada una de estas aplicaciones se debería de modificar para poder agregar a este nuevo integrante, elevando el costo a la hora de poner en marcha, si lo ubicamos a la realidad, pues todo el tiempo que se invierte en el cambio en la programación, las pruebas pertinentes, etc. Esta muy claro que en este tipo de análisis hay que optimizar para tener la menor cantidad de interfaces posibles, aspecto que se tratará más adelante.

Los Servicios y su Naturaleza:

Se involucra ahora un nuevo concepto llamado granularidad, este se refiere al alcance de funcionalidad que un servicio expone y se puede clasificar en dos niveles:

1. Servicios de granularidad fina: son aquellos que proporcionan una pequeña cantidad de útiles procesos de negocios, por ejemplo el acceso básico a datos.

2. Servicios de granularidad gruesa: suministran operaciones rudimentarias de mucho valor para la empresa, ya que ayudan a satisfacer necesidades específicas de negocio.

Como la figura 11 ilustra, el nivel de la granularidad dependerá generalmente del propósito del software. En los servicios la granularidad tenderá a ser más gruesa comparado con los objetos y componentes, ya que el servicio expone típicamente un solo y discreto proceso de negocio.

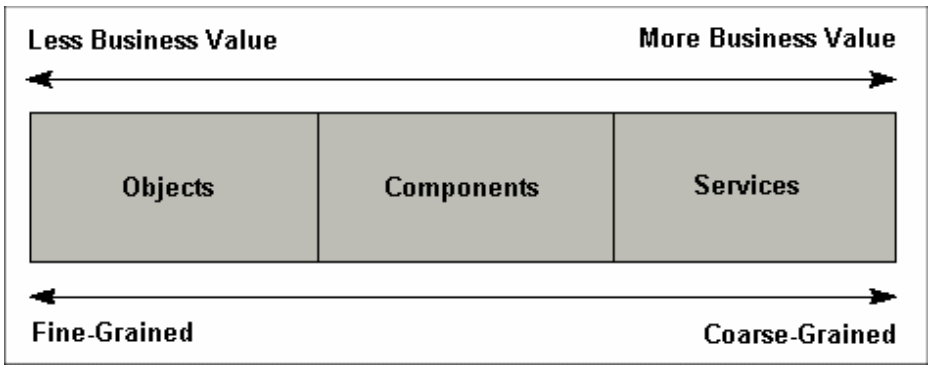

Figura 11: Niveles de la granularidad. Fuente: Jeff Hanson (2003).

Servicios como interfaces Compuestas

Usando interfaces de granularidad gruesa, un sistema basado en SOA puede controlar el acceso a los objetos referidos en cada servicio. Por lo tanto cada servicio puede ser llevado a cabo agrupando objetos, componentes y servicios de granularidad fina, exponiéndolos como una sola unidad a través de una interfaz, como es mostrado en la figura 12.

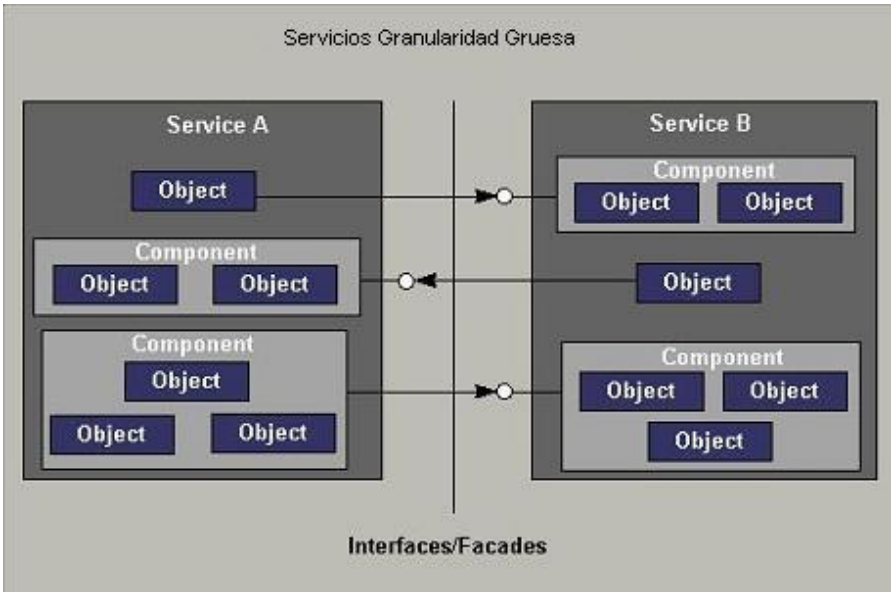

Figura 12: Servicios de granularidad gruesa. Fuente: Jeff Hanson (2003).

Orientación a los Problemas de conexiones:

Anteriormente se había mencionado que uno de los grandes problemas se presenta a la hora de integrar aplicaciones, estas conexiones o interfaces generarían mucho trabajo y por ende costo; no obstante también se indicó, que la forma de corregirlo era disminuyendo la cantidad de interfaces, en la siguiente figura se muestra interfaces minimizadas:

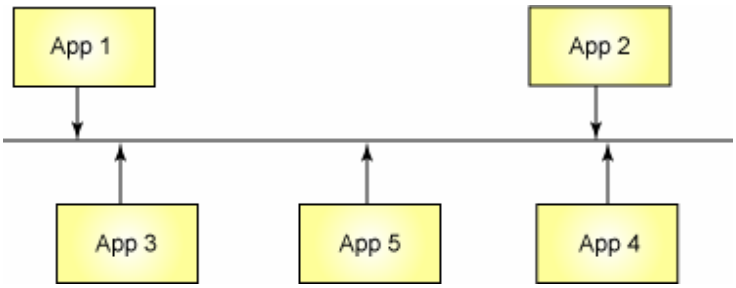

Figura 13: Minimización de conexiones

Esto pudiera verse como algo muy simple, pero hay que considerar que dentro de un "framework", esta figura muestra el punto de partida para direccionar el problema de tener muchas interfaces. Ahora si agregamos el concepto de la arquitectura de bus a lo que son servicios, obtendríamos una línea central y un tipo de administrador de flujo ("flow Manager"), que conectará los servicios y proveerá una ruta para las solicitudes que estas lleven a cabo.

Este administrador tiene definido un lenguaje de secuencia de ejecución, que invocará los servicios requeridos en el orden apropiado para generar el resultado final. Técnicamente a este proceso se le conoce como el BPEL ("Business Process Execution Language"), el cual es un ejemplo, de que tal lenguaje pueda definir un proceso como un sistema de invocaciones de servicios. En la siguiente figura se muestra en resumen lo que se ha explicado anteriormente:

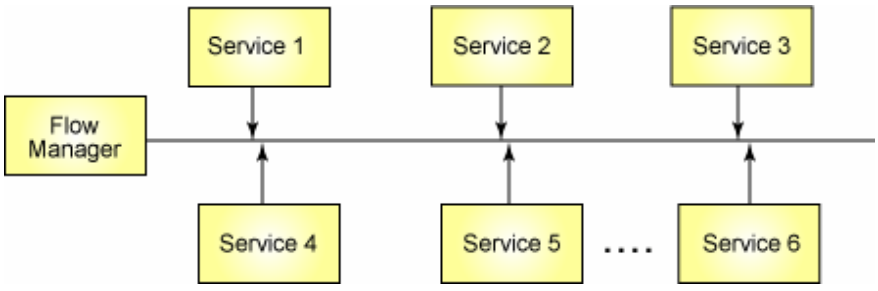

Figura 14: Agregación del bus y del administrador de flujo.

A partir de aquí, viene la necesidad de cómo llamar estos servicios para añadírselo a la configuración de la aplicación, para luego contener las entradas y las salidas, para finalmente proporcionar la conectividad con los procesos de "backend" (soporte a sistemas propietarios para que compartan sus bases de datos o servicios externos, con el fin de ayudar a la interacción con las interfaces disponibles a los usuarios); generándose una estructura ya completa así, como se puede notar de manera gráfica en la siguiente figura:

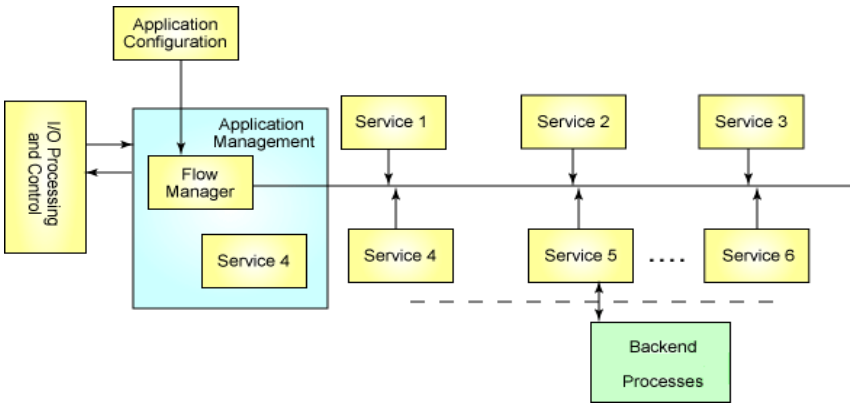

Figura 15: Estructura básica terminada

A un alto nivel, cualquier "Framework" de aplicación robusta debería de proveer los puntos anteriores, para poder llegar a construir componentes basados en servicios. Es por esta razón que muchos expertos de tecnologías de información toman la decisión de usar "Framework" estandarizados, realizados por terceros, con el fin de mantener siempre la compatibilidad que esta arquitectura de servicio ofrece.

El proceso de descomponer la lógica de aplicaciones que se tiene actualmente a componentes basados en un "Framework", es un trabajo arduo, fuera de la reinvención del resto de componentes de propósito general y de sistemas, aún sabiendo que estos se necesitan. Sin embargo se puede aprovechar, el hecho de que se logra poner en práctica la arquitectura, usando tecnologías y

“Framework” que actualmente están disponibles en el mercado, estando al corriente que de nueva cuenta se volverá al inicio, en donde el proceso empieza, con un análisis de los problemas del negocio, que deben de ser solventados, pero manteniendo ahora un acoplamiento para con los otros sistemas ya sean propios o de terceros, siempre y cuando estos estén bajo la misma arquitectura.

“Service Oriented Development of Application” (SODA)

Entendiendo que el SOA es todo lo referente a la creación de servicios, interacción, presentación e integración de las capacidades en los procesos del negocio, es necesario ahora en contar con infraestructuras que faciliten el desarrollo, para eso han implementado un alineamiento para que los fabricantes elaboren herramientas de desarrollo de aplicaciones orientadas a servicios (SODA por sus siglas en ingles), este usualmente se refiere al uso del modelamiento, diseño e implementación para la construcción de aplicaciones. SODA ofrece los siguientes beneficios:

1. Permite desarrollar progresivamente, creando componentes como servicios y combinando también con los servicios existentes en el orden de que se quiera crear aplicaciones compuestas. SODA resalta la reutilización de los actuales componentes, mejorando el rendimiento y calidad de las entregas.

2. SODA permite que los desarrolladores modifiquen servicios activos. Cualquier servicio debidamente diseñado puede ser cambiado, sin necesitar alterar a otros servicios que interactúan con el servicio original.

3. SODA preserva acoplamiento para futuras características; las aplicaciones pueden ser independientemente diseñadas, actualizadas o modificadas. Esto limita el riesgo asociado a integración, cambios y a migraciones costosas.

4. Con el uso de SODA, se provoca que los desarrolladores estén más concentrados en funciones propias del negocio, sin tener que preocuparse por la infraestructura de los sistemas y de los “middleware”.

5. Los procesos de aplicaciones externas y propias se benefician gracias a la reutilidad.

6. La independencia de las aplicaciones ayuda a maximizar su flexibilidad, la facilidad de integración y operabilidad.

Requisitos para diseñar bajo SODA

La implementación exitosa del SODA depende de la efectividad de las herramientas de diseño, en facilitar el desarrollo en servicios. Si el desarrollo es igual a codificación o si los cambios a los estándares significan volver a rehacer más que convertir o actualizar, entonces el tiempo de diseño podría inhibir el uso de herramientas basadas en SODA. Existen algunos requerimientos que SODA debe de seguir en el tiempo-diseño:

1. Ser una herramienta ágil: una de las primordiales necesidades del SODA es contar con un rápido ROI ("Return on Investment"), contar con aplicaciones enfocadas al negocio. El conjunto de herramientas en tiempo diseño debe de ser ágil, para entregar o facilitar lo siguiente:

    a. Descubrir los componentes existentes de la aplicación, por ejemplo los EJB ("Enterprice Java Beans") o los procedimientos almacenados en una base de datos, con el fin de convertirlos a servicios.
    b. Abstracción de código a partir de infraestructuras de sistemas heterogéneos y métodos de comunicación.

    c. Minimización de código requerido para vincular sistemas distintos, manteniendo una variedad de protocolos, interactuando con base de datos, enviando y recibiendo mensajes entre sistemas de

transportes y trabajando con un número de interfaces de programación de aplicaciones.

2. Modelamiento: SODA depende de la habilidad del set de herramientas para realizar modelos y remodelar el flujo de información entre la empresa y los aliados estratégicos que este tenga. Estos modelos dependen de los procesos del negocio y la reutilización de los modelos existentes para definir nuevas y complejas interacciones entre ellas.

3. Desarrollo Orientado a Procesos: SODA se basa sobre los procesos del negocio para conectar servicios y orquestar el flujo de información. Algunos de estos procesos son disponibles como servicios, habilitando todo el proceso de negocio para que sea reutilizado por otros procesos.

Requerimientos para el ”run-time” de SODA:

Es necesario para SODA contar con soporte para la ejecución de servicios, la transformación y enrutamiento de datos, la administración de los procesos del negocio, mensajería y la construcción de aplicaciones compuestas a través de servicios. Estos requisitos se pueden detallar ampliamente con las siguientes características:

1. Ejecución de Servicio: el servicio requiere soporte para descubrir, vincular, publicar e interactuar con componentes de aplicaciones “Back End” para transportar entradas y salidas de mensajes. El contenedor de servicios sigue una interacción apropiada del modelo y de los protocolos, para conectar y ejecutar componentes de aplicación (“deployed”) en distintas infraestructuras.

2. Monitoreo de las actividades del negocio (BAM por sus siglas en ingles): la infraestructura del “run-time” podría alimentar datos BAM con el fin de contar con una herramienta que permita monitorear la administración para el proceso del negocio. Con esta habilidad se facilita la generación de rastros y alertas.

3. Escalabilidad: SODA se construye reutilizando y convirtiendo las aplicaciones existentes en servicios y por supuesto en la creación de nuevos servicios siguiendo diseños compuestos. Es por eso que se requiere que el “run-time” del SODA sea escalable y capaz de ejecutar sobre múltiples plataformas.

4. Seguridad: soluciones basadas en SOA se comunicarán inevitablemente a través de Internet o Intranet, llevando consigo el hecho de que estará detrás de un “Firewall”. El “run-time” debe de soportar todas las especificaciones de seguridad empresariales, asegurando la administración apropiada de la identidad y tomando las peticiones de cualquier acceso hechos por los usuarios o los mismos servicios.

Existen en el mercado fabricantes que han creado herramientas implementando los alineamientos de SODA, a continuación se muestra un listado de los fabricantes con sus respectivas herramientas y los costos que se tienen (datos obtenidos directamente del sitio en Internet de cada fabricante):

1. Microsoft:
    a. “Framework .Net”: es gratuita.

    b. “Visual Studio .NET 2003 Enterprise Architect”: toda la suite cuesta $2,499 la licencia por usuario.

2. IBM:
    a. Rational® : Los productos que se incluyen en esta suite son los siguientes:
        I. “Rational Application Developer for WebSphere Software 6.0”: esta herramienta tiene un costo de $4,000.00 la licencia por usuario.
        II. “Rational Software Architect”: este paquete tiene un costo de $5,500 por usuario.
        III. “Rational Software Modeler”: el precio de esta opción es de $1,795.00 por usuario.
        IV. “Rational Unified Process®”: un “framework” que es utilizado e integrado en el “Rational Software Architect”

    b. “WebSphere Studio Application Developer Integration 6.0”: el costo reportado por IBM para este producto es de $7,279.00 por usuario.

3. Oracle:
    a. “SOA Development Kit”: esta suite incluye los siguientes componentes:
        i. “Internet Developer Suite”: este producto tiene un valor aproximado de $5,000 por usuario.
        ii. “Oracle BPEL Process Manager 10.1.2”: el costo de la licencia por usuario es de aproximadamente $1,000.
        iii. “Oracle Application Server Enterprice”: el valor de este paquete es de $600 la licencia por estación.
        iv. “Oracle Toplink”: un “framework” cuyo costo por usuario es de $100.

v. “Oracle Web Services Manager”: esta opción está incluida dentro del paquete Oracle Application Server Enterprice.

4. Sybase:
    a. Sybase Workspace TX Enterprice: esta suite tiene un valor de $7,995 por usuario e incluye los siguientes productos:
        I. Adaptive Server Anywhere
        II. EAServer 5.2
        III. Unwired Orchestrator 5.0
        IV. Unwired Accelerator 7.0
        V. Adaptive Server Enterprise 12.5.3

5. BEA Systems:
    a. WebLogic Server: PCMag.com publica el precio de este producto en $10,000 por CPU.
    b. BEA Tuxedo y AquaLogic Service Bus: este producto tiene un valor en el mercado de $45,000 por CPU.

El Panorama Actual y futuro del SOA

Muchos consideran que esta arquitectura traerá numerosos beneficios para las organizaciones que conciban y migren hacia este adelanto. Esta tecnología de computación distribuida llegará a ser lo suficientemente flexible y óptima para responder a las necesidades del negocio, proporcionando la agilidad que las compañías han anhelado por tanto tiempo ("The Rational Edge", 2004).

Este auge de SOA ha tenido éxito ya que es la forma de hacer más con menos recursos, reutilizando e integrando de una manera más sencilla, permitiendo disminuir la duración de desarrollo y abaratando los costos.

No es de asombrar que el 80% de las organizaciones de TI están realizando proyectos basados en SOA con "Web Services" ("M7 Corporation", 2005). Además proyecciones hacia el 2008 revelan que más del 75% de las aplicaciones de entonces serán originariamente SOA o expondrán interfaces SOA a través de una capa de cobertura de interfaces, siendo la forma predominante de ingeniería de software, terminando con cuatro décadas de supremacía de las arquitecturas monolíticas ("Gartner Group", 2003).

Las afirmaciones y proyecciones anteriores dan a entender que la dificultad que existe en acoplar los distintos procesos de negocio esta empezando a superarse en los proyectos de desarrollo de aplicaciones; que el gran impulso fue provocado por el auge de Internet, induciendo con el asentamiento de SOA y proyectándose éste como la arquitectura dominante en los próximos años.

Referencias Bibliográficas

Brian Randell, Rockford Lhotka (2004). *Bridge the gap between development and operation with Whitehorse*. MSDN Magazine, 16, 24-45.

Bridgefield Group. (2005). *Time to Market: Business Planning+Executionsm.* Recuperado el 12 de setiembre de 2005, de http://www.bridgefieldgroup.com/glos9.htm.

CBDI. (2005). "*WS Roadmap Report Section 5 - Moving to SOA*". Recuperado el 12 de setiembre de 2005, de http://www.cbdiforum.com/report_summary.php3?page=/bronze/moving_soa.php3&area=bronze.

Dirk Krafzig, Karl Banke, Dirk Slama (2005). *Entreprise SOA, Service oriented Architecture Best Practices.* Estados Unidos: Pearson Education.

Douglas Barry (2003). *"Web Services and Service-Oriented Architectures".* Estados Unidos: Morgan Kaufmann.

Gartner, Inc. (2005). *Service-Oriented Architecture: Mainstream Straight Ahead*. Recuperado el 13 de setiembre de 2005, de http://www.gartner.com/pages/story.php.id.3586.s.8.jsp

Hugo Haas, W3C. (2005). *Web Services Architecture*. Recuperado el 23 de octubre de 2005, de http://www.w3.org/TR/2003/WD-ws-arch-20030808/#service_oriented_model.

IBM Corporation. (2005). *SOA expands the vision of Web services.* Recuperado el 13 de setiembre de 2005, de http://www-128.ibm.com/developerworks/library/ws-soaintro.html

IBM Corporation. (2005). Software A to Z. Recuperado el 28 de noviembre de 2005, de http://www-306.ibm.com/software/info/ecatalog/en_US/products/C816729Z84706A83.jsp?OC=Y&CC=Costa%20Rica&VP=NO&EO=CRI&S_TACT=105AGX38&S_CMP=eCtlgES

Jeff Hanson. (2003). Coarse-grained interfaces enable service composition in SOA. Recuperado el 19 de octubre de 2005, de http://builder.com.com/5173-6386-0-2.html?tag=sc

Kruchten, P. Boston. (2004). *The Rational Unified Process: An Introduction*". Estados Unidos:
Addison-Wesley.

Kunal Mittal. (2005). *The buzz about Beehive - Programming techniques for open source SOA tools*. Recuperado el 16 de setiembre de 2005, de http://www-128.ibm.com/developerworks/opensource/library/os-beehive/

Marc Brooks (MITRE). (2005). *Service Oriented Architecture and Grid Computing*. Recuperado el 12 de setiembre de 2005, de http://web-services.gov/Brooks32404.ppt.

Network World. (2005). *BEA unveils service software family*. Recuperado el 28 de noviembre de 2005, de http://www.networkworld.com/news/2005/062005-bea.html.

Oracle.com. (2005). *Oracle Store*. Recuperado el 28 de noviembre de 2005, de http://oraclestore.oracle.com/OA_HTML/ibeCCtpSctDspRte.jsp?a=b

PCMag.com. (2005). *BEA WebLogic Server*. Recuperado el 28 de noviembre de 2005, de http://www.pcmag.com/article2/0,1759,1215929,00.asp.

Prensa Microsoft. (2005). *Precio y disponibilidad Visual Studio .Net.* Recuperado el 28 de noviembre de 2005, de http://www.microsoft.com/latam/prensa/2003/abr/Lanzamiento_vsnet.asp

Rushton Prince*. (2005). Using RUP/UP: 10 Easy Steps.* Recuperado el 15 de octubre de 2005, de http://www.x-tier.com

Software Engineering Institute*. (2005). Capability Maturity Model® Integration.* Recuperado el 28 de setiembre del 2005, de http://www.sei.cmu.edu/cmmi.

Werner Vogels. (2005). *Web services are not distributed objects*. Recuperado el 16 de setiembre de 2005, de http://weblogs.cs.cornell.edu/AllThingsDistributed/archives/000119.html - 2003